Making Pre-trained Language Models Better Learn Few-Shot Spoken Language Understanding in More Practical Scenarios

Yufan Wang[1,3*], **Mei Jie**[3†], **Bowei Zou**[2], **Rui Fan**[1,3], **Tingting He**[3‡], **Ai Ti Aw**[2]
[1]National Engineering Research Center for E-Learning, Central China Normal University, China
[2]Institute for Infocomm Research (I^{2}R), A*STAR, Singapore
[3]Hubei Provincial Key Laboratory of Artificial Intelligence and Smart Learning,
National Language Resources Monitoring and Research Center for Network Media,
School of Computer, Central China Normal University, China
{yufan_wang,meijie,fanrui}@mails.ccnu.edu.cn
{zou_bowei,aaiti}@i2r.a-star.edu.sg
tthe@mail.ccnu.edu.cn

Abstract

Most previous few-shot Spoken Language Understanding (SLU) models typically need to be trained on a set of data-rich source domains and adapt to the target domain with a few examples. In this paper, we explore a more practical scenario for few-shot SLU, in which we only assume access to a pre-trained language model and a few labeled examples without any other source domain data. We concentrate on understanding how far the few-shot SLU could be pushed in this setting. To this end, we develop a prompt-based intent detection model in few-shot settings, which leverages the BERT original pre-training next sentence prediction task and the prompt template to detect the user's intent. For slot filling, we propose an approach of reconstructing slot labels, which reduces the training complexity by reducing the number of slot labels in few-shot settings. To evaluate the few-shot SLU for a more practical scenario, we present two benchmarks, FewShotATIS and FewShotSNIPS. And a dynamic sampling strategy is designed to construct the two datasets according to the learning difficulty of each intent and slot. Experiments on FewShotATIS and FewShotSNIPS demonstrate that our proposed model achieves state-of-the-art performance.

1 Introduction

Spoken Language Understanding (SLU) is one of the fundamental modules for task-oriented dialogue systems, which mainly includes two sub-tasks, *intent detection* and *slot filling*. The remarkable success of most neural SLU models typically relies on a large quantity of training data (Chen et al., 2019; Qin et al., 2020, 2021; Wang et al., 2022). However, acquiring large amounts of annotated data for domain-specific is arduous and expensive in practical applications. Situations like few or even no training data may happen in a brand-new application, which motivates us to address the challenge of the SLU module in few-shot settings.

The previous few-shot SLU studies mainly focused on semi-supervised learning (Basu et al., 2021; Gaspers et al., 2021; Kumar et al., 2022) and metric learning (Hou et al., 2020a; Krone et al., 2020a; Yang and Katiyar, 2020; Hou et al., 2021a; Yu et al., 2021; Yang et al., 2022; Gao et al., 2022; Hou et al., 2022; Yang et al., 2022). The models need to be trained on source domains with abundant data and then adapt to the data-scarce target domain. Straying from the pattern by making minimal assumptions about available resources, we explore a more practical scenario for few-shot SLU, in which we only use moderately sized pre-trained models, such as BERT (Devlin et al., 2018) or RoBERTa (Liu et al., 2019), and only a tiny amount of annotated examples to fine-tune the pre-trained model. The settings are pretty attractive as (1) the few-shot settings conform more to real application scenarios, as it is straightforward to get a few annotated data (e.g., 10 examples); (2) such models are not demanding hardware resources for training; and (3) the development of prompt tuning has brought a novel paradigm for few-shot learning, and updating parameters of the pre-trained model typically leads to better performance (Nigam et al., 2019; Han et al., 2021, 2022; Zhu et al., 2022a). We focus on exploring what the performance of the few-shot SLU model can achieve without any other source domain data.

We revisit existing few-shot SLU benchmarks (Larson et al., 2019; Hou et al., 2020a,b; Yu et al., 2021) and find that the settings of them tend to deviate more from practical application scenarios. The main reasons are as follows. (1) These benchmarks

[*]Contribution during the internship at Institute for Infocomm Research.
[†]Yufan Wang and Mei Jie contributed equally this work.
[‡]Corresponding author.

Findings of the Association for Computational Linguistics: ACL 2023, pages 13508–13523

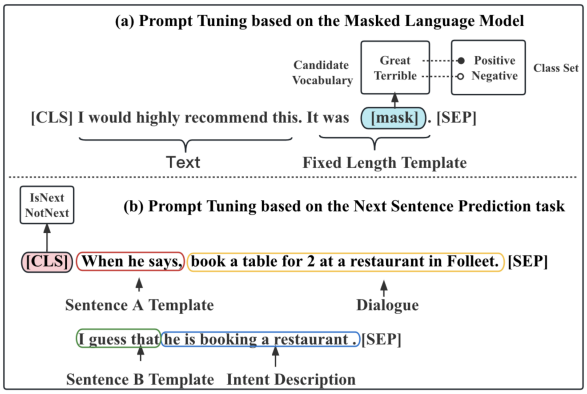

Figure 1: Prompt tuning for different pre-training tasks.

provide data-rich source domains for training, such as SNIPS (Hou et al., 2020a) and CLINC (Larson et al., 2019). However, in practical applications, there might be little or even no training data available. (2) Some benchmarks typically uniformly random sample k examples for each intent and slot. However, in practical applications, the learning complexity varies with different intents and slots and may necessitate different amounts of training data. To fill these gaps, we construct two multi-domain few-shot SLU benchmarks, FewShotATIS and FewShotSNIPS, without any source domain data, which are based on ATIS (Price, 1990) and SNIPS (Coucke et al., 2018), respectively. In addition, we design a dynamic sampling strategy that allocates the number of samples depending on the evaluation metrics of each intent and slot. The sampling strategy is more reasonable and adheres to practical scenarios since slight variations in the training data can have a major impact on the model's performance in few-shot settings. The proposed benchmarks provide a fair comparison between various methods on common ground and measuring progress in practical scenarios.

Recently, prompt tuning achieved competitive results with only a limited amounts of training data, such as prompt tuning based on BERT with the masked language model (as shown in Figure 1(a)), which has been proven to be effective for text classification (Gao et al., 2020; Hu et al., 2021; Zhu et al., 2022b). However, it suffers from the following challenges when employed for intent detection. (1) The candidate vocabulary for each intent must be manually constructed based on natural language templates. (2) The semantics of intent are complex and difficult to represent with fixed-length tokens.

Moreover, there are some challenges for the slot filling task in few-shot settings. (1) The BIO labeling format (Huang et al., 2015) is typically utilized

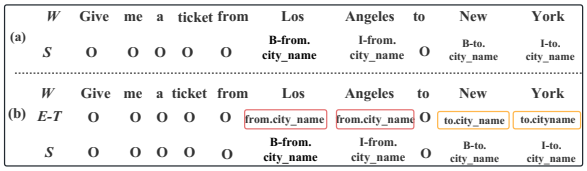

Figure 2: The BIO labeling format and reconstructing slot labels format. *W* is token of the dialogue. *S* and *E-T* denote slot and entity type.

in standard datasets for slot filling, as shown in Figure 2(a). In the case of sufficient data, the beginning labeling 'B-' of the entity provides supervised information to the model to detect the entity boundaries. Nevertheless, in few-shot settings, the excessive amount of slot labels in the labeling format exacerbates the difficulty of the model training. (2) As the training sample decreases, it becomes increasingly difficult for the model to differentiate between similar slot labels, such as "from.city_name" vs. "to.city_name" in Figure 2.

To address the aforementioned challenges, we proposed a BERT-NSP-Prompt model for intent detection and a reconstructing slot labels approach for slot filling. In particular, BERT-NSP-Prompt leverages the next sentence prediction (NSP) task of BERT and a constructed prompt template, to complete intent detection in the few-shot or even zero-shot settings, as shown in Figure 1(b). It is used to assess which intent description text is the most *"fluent"* following sentence of the user's dialogue. Moreover, an approach of reconstructing slot labels is proposed to reduce the model training complexity by reducing the number of slot labels in few-shot settings, as shown in Figure 2(b). We convert the BIO slot labeling format to slot entity labeling format, which resulted in the slot category cutting in half. Furthermore, we introduce the focal loss function (Lin et al., 2017) to distinguish between similar slot labels.

To sum up, our key contributions are as follows.

(1) We investigate more practical scenarios for few-shot SLU and propose the BERT-NSP-Prompt model and a reconstructing slot labels approach.

(2) We construct two multi-domain few-shot SLU benchmarks, FewShotATIS and FewShotSNIPS, to encourage the research community to create algorithms that can demonstrate generalization capabilities with minimal data.

(3) We conduct extensive experiments to demonstrate the effectiveness of our model, and the experimental results reflect that our models achieve state-of-the-art performance on the two datasets.

2 Related work

Most works use semi-supervised learning or metric learning to implement few-shot SLU tasks (Hou et al., 2020a; Zhu et al., 2020; Krone et al., 2020b; Zhang et al., 2020; Hou et al., 2021b; Yu et al., 2021). Hou et al. (2020a) adopted the TapNet and label dependency transferring to complete the slot-filling task. Krone et al. (2020b) used a prototype network to complete a few-shot SLU task. Han et al. (2021) further extend the prototypical network to achieve joint learning, which guarantees that two tasks can mutually enhance each other. Yu et al. (2021) adopted the retrieval framework to match the token spans in the input with the most similar labeled spans in the retrieval index. Cui et al. (2021) leverages prompts for few-shot NER. However, these models still require a set of data-rich source domains. Recently, prompt tuning has taken advantage of the powerful generalization ability of pre-trained language models and dramatically reduces the reliance on supervised data for downstream tasks (Liu et al., 2021; Lester et al., 2021; Gu et al., 2021). The paradigm narrows the gap between the pre-trained model and downstream tasks, which provides a novel insight into few-shot learning. We propose a simple and effective method based on pre-trained models to accomplish few-shot SLU tasks. The proposed model does not have a transfer from the source domain to the target domain, nor does it use any additional data resources. It will serve as an essential baseline for future exploration. In addition, we believe the constructed FewShotATIS and FewShotSNIPS datasets can comprehensively evaluate the models and inspire the research of the few-shot SLU.

3 Datasets: FewShotATIS and FewShotSNIPS

In few-shot SLU, existing benchmarks usually uniformly random sample k examples for each intent and slot, such as SNIPS (Hou et al., 2020a), and CLINC (Larson et al., 2019). However, a reasonable and conform to practical scenarios sampling strategy is: suppose the total number of samples is limited; for easy-to-classify intents or slots, it will sample less than k samples; in contrast, it will sample more than k samples for hard-to-classify intents and slots. Thus, we design a dynamic sampling strategy to simulate the "sampling-iterative" process, which assigns the number of samples according to the evaluation metrics of each intent and slot. Since small changes in the training data may significantly affect the model's performance in few-shot settings, the advantages of the strategy will be magnified further. Finally, we adopt the dynamic sampling strategy to build two new multi-domain few-shot SLU benchmarks, FewShotATIS and FewShotSNIPS[1], respectively.

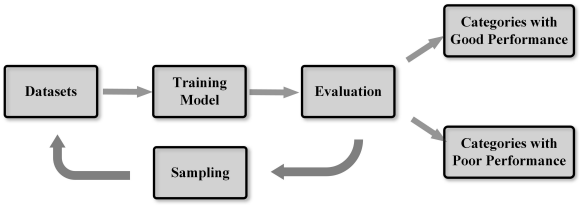

Figure 3: Architecture of the dynamic sampling strategy.

3.1 Dynamic Sampling Strategy

In the few-shot settings, the models are susceptible to subtle variations of training data. Small changes in the training set may significantly affect the model's performance. More importantly, the pre-experimental results show that some intents and slots are easy to predict correctly only by keywords. In contrast, some intents and slots are difficult to predict correctly, and they need to predict based on context and syntactic information. Experiments reflect that different types of intents and slots have different learning difficulties and may require different training numbers of samples.

Based on the above discussion and better simulating practical application, we propose a dynamic sampling strategy when constructing a few-shot dataset, as shown in Figure 3. Specifically, suppose the number of samples is fixed in the iteration process (i.e., limiting the total sum of samples for each iteration). In that case, the dynamic sampling strategy assigns the number of samples for each intent and slot based on the evaluation metrics at each iteration. The dataset is built through a "sampling-iterative" process. Compared to random sampling, the dynamic sampling strategy can more reasonably sample data with different intents and slots.

More specifically, in the experiment of few-shot learning, multiple sets of comparative experiments usually sample K samples of each category uniformly, called $K-$shot experiment (Dong and Xing, 2018). $K \in \{k_1, k_2, \cdots, k_n\}$ is a parameter. k_1-shot, k_2-shot, $\cdots$, k_n-shot experiments are independent of each other. However, in the dynamic sampling strategy, for a $k_n$shot experiment,

[1]https://github.com/wyf401/Few-shot-SLU-Dataset

Statistic	FewShotATIS					FewShotSNIPS				
# Domian	1					7				
# Intent	18					7				
# Slot	128					76				
# K-shot for intent	2-shot	4-shot	6-shot	8-shot	10-shot	2-shot	4-shot	6-shot	8-shot	10-shot
# Training instances for intents	33	67	101	130	161	14	27	40	54	67
# K-shot for slot	5-shot	10-shot	20-shot	30-shot	40-shot	5-shot	10-shot	20-shot	30-shot	40-shot
# Training instances for slots	336	639	1,212	1,675	1,998	195	391	787	1,185	1,584
# Testing instances	878					700				

Table 1: Detail Statistics of FewShotATIS and FewShotSNIPS.

Algorithm 1 Dynamic Sampling Strategy

Setting: The sampling process can be regarded as an iterative process. The sum of the samples is limited to $M * N$ at each iteration, where $\mathrm{M} \in \{m_0, m_1, \cdots, m_{K_{\max}}\}$. N is the number of label categories. M denotes number of samples taken per iteration and $K_{\max}$ is the iteration times, maximize the models performance of the $m_{K_{\max}}$ time.
Parameter: Optional list of parameters $M, N, K_{\max}$.
Output: Dynamic sampling results.

1: //Initialize array.
2: **for** $i = 0$ to N **do**
3: //num_samp is a 1-d array storing number of data need to be sampled for each class.
4: num_samp[i]← 0.
5: //F1 is a 2-d array storing F1 score for each class at each iteration.
6: F1[0][i]←1.
7: **end for**
8: **for** $k = 0$ to $K_{\max} - 1$ **do**
9: //Total number of samples taken per iteration.
10: **if** $k = 0$ **then**
11: $\Delta \leftarrow m_0 *$ N
12: **else**
13: $\Delta \leftarrow (m_{k+1} - m_k) *$ N
14: **end if**
15: **for** $j = 0$ to N **do**
16: //Sampling for each category.
17: temp $\leftarrow \left(1 - \frac{k}{K_{\max}}\right) * (m_{k+1} - m_k) + \frac{k}{K_{\max}} * \Delta * \frac{(1-\mathrm{F1}[k][j])}{\sum_i^{i=N}(1-\mathrm{F1}[k][j])}$
18: num_sampled[j]← num_samp[j]+ $temp$;
19: **end for**
20: Sample data based on num_sampled$[0, \cdots, N]$ and training $model_k$;
21: Evaluate $model_k$ on F1 score metric.
22: Update F1$[k][0, \cdots, N]$ based on evaluation result
23: **end for**

we treat it as an n-iteration experiment. For the $(i+1)$-th iteration, the total number of samples is increased by $(k_{i+1} - k_i) * N$, where N is the number of categories.

For easy-to-classify intents or slots, it will sample less than $(k_{i+1} - k_i)$ samples, while it will sample more than $(k_{i+1} - k_i)$ samples for hard-to-classify intents and slots. The pseudo-code is shown in Algorithm 1. It is worth noting that the number of samples per sampling is stored in the variable $temp$ (line 17), and we use BERT to calculate the F1 score (line 21).

3.2 Datasets

Table 1 reports the details of FewShotATIS and FewShotSNIPS. k samples are sampled on each intent and slot to form the k-shot dataset. Due to the differences between the intent detection and slot filling tasks, we take different sampling spans in constructing the datasets. During the experiment, we adopt finer-grained and larger-range sampling. Experimental results show that the intent detection task in 10-shot and the slot filling task in 40-shot achieve decent performance, as shown in Appendix A. Therefore, we construct datasets ranging from 2-shot to 10-shot for the intent detection task and ranging from 5-shot to 40-shot for the slot-filling task. It is worth noting that the test set in the two datasets is consistent with the standard dataset. Thus, the two datasets can measure the model's performance more comprehensively and solidly. The datasets are constructed to simulate real applications and encourage the community to build algorithms capable of generalizing with only a few intents and slots.

4 Methodology

4.1 Task Definition

Task-oriented dialogue systems are usually oriented to a specific domain, in which the intents and slots are typically predefined and limited in number. Thus, intent detection is considered a text classification task, and the slot filling task is considered a sequence labeling task. In few-shot settings, the number of samples that can be used for training is much smaller than the number of samples in a standard dataset. Learning both tasks simultaneously maybe increases the complexity of model training. Furthermore, it is difficult to unify the few-shot sampling method of two tasks in K-shot sampling (Dong and Xing, 2018). For a given dialogue $X = \{x_1, x_2, x_3, \cdots, x_m\}$, the intent detection task is to learn a model M_I to get the intent I; the slot filling task is to learn a model M_S to get

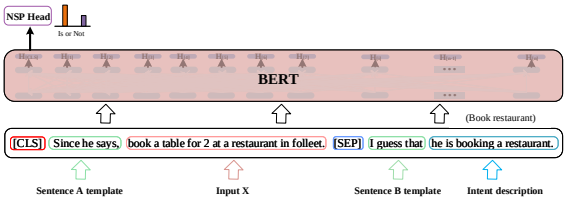

Figure 4: Illustration of the BERT-NSP-prompt model. For each dialogue, the input with the prompt template is constructed according to each intent in the dataset.

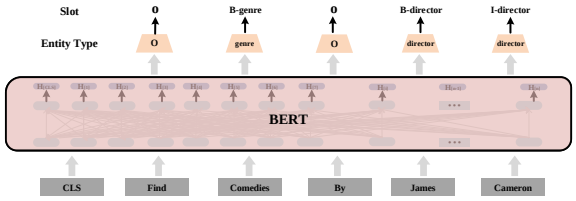

Figure 5: Illustration of slot filling model based on reconstructing slot label.

the slots $S = \{s_1, s_2, s_3, \cdots, s_m\}$.

4.2 Prompt-based Intent Detection Model

In few-shot settings, we proposed a BERT-NSP-prompt model, which utilizes the next sentence prediction (NSP) task of BERT pre-training to detect the intent of the dialogue. No new parameters are introduced to the whole model. The architecture of the BERT-NSP-prompt is shown in Figure 4. The NSP task aims to predict whether the relationship between two sentences is contextual. Therefore, the NSP task can evaluate whether two sentences are related to the same topic and contain similar semantics. We transform intent detection into the NSP task by building a prompt to unify the BERT pre-training and intent detection tasks.

The input of the BERT-NSP-prompt consists of four parts: the sentence template T_A, the original dialogue A, the sentence template T_B, and the natural language text description of the intent label B. T_A and T_B are manually designed prompt templates. The format of the input sequence is as follows:

$$\mathrm{X}' = ([\mathrm{CLS}], T_A, X_A, [\mathrm{SEP}], T_B, X_B, [\mathrm{EOS}]), \quad (1)$$

where X_A is sentence A, X_B is sentence B. T_A and T_B denote the templates. `[CLS]` is the special identifier used to complete the next prediction task, and `[SEP]` and `[EOS]` are the special identifiers for sentence segmentation and sentence ending.

For example, there are a total of seven intents in the SNIPS dataset. For an original dialogue x, the final intent is predicted by constructing seven inputs X' with prompt. Appendix B shows the description text of each intent in the SNIPS dataset. After the construction of input X', the representation of each input token $E = (e_0, \cdots, e_n)$ consists of Token Embedding, Segment Embedding, and Position Embedding. n is the length of the inputs X'. And then, the input sequence E is encoded by BERT to obtain the hidden state $(h_0, \cdots, h_n)$ of the final layer output:

$$(h_0, \cdots, h_n) = BERT(\mathrm{E}), \quad (2)$$

where h_0 is the hidden state vector of `[CLS]`.

The output layer uses the BERT pre-trained NSP task classifier to determine the relationship between sentence *A* and sentence *B*. Then, the most relevant text descriptions of intent to the user's dialogue are determined by comparing the "IsNext" tags scores of the NSP task. Finally, the intent is predicted. The probability distribution of the intent is shown in Equation (3).

$$P^I = \mathrm{Softmax}\left(W_{nsp} h_0 + b_{nsp}\right), \quad (3)$$

where W_{nsp} and b_{nsp} are learnable parameters.

The loss function for intent detection is calculated by:

$$\mathcal{L}_I = -l_I \log P^I, \quad (4)$$

where l_I is the intent label.

4.3 Slot Filling Via Reconstructing Slot Labels

The slot filling task is generally solved as a sequence labeling task. The slot filling uses the BIO labeling format in standard datasets. The beginning of the entity with 'B-' and the interior of the entity with 'I-'. In the case of sufficient data, the beginning labeling 'B-' of the entity provides supervised information for the model to detect the entity boundaries. However, in few-shot settings, the excessive number of slot labels increases the difficulty of model training.

To address this issue, we adopt a two-step approach to reconstructing slot labels. First, the sequence labeling task in BIO labeling format is transformed into a predicting slot entity task. The model predicts the entity types of all tokens in dialogue. Second, the prediction results are reconstructed into BIO format by rules for evaluation according to the order of natural language from left to right. The reconstructing slot labels approach is shown in Figure 5. Applying the approach in the slot filling task can reduce the number of slots by half. Thus, half of the classifier parameters are reduced, ultimately reducing the difficulty of training models.

Model	FewShotATIS						FewShotSNIPS					
	Zero-shot	2-shot	4-shot	6-shot	8-shot	10-shot	Zero-shot	2-shot	4-shot	6-shot	8-shot	10-shot
BERT-ID	0	20.29±2.56	30.58±1.91	59.90±2.41	71.73±1.99	85.75±1.68	0	59.02±2.24	76.36±2.21	79.87±2.31	85.44±2.22	87.30±1.42
RoBERTa-ID	0	5.33±2.1	7.45±2.46	74.47±4.11	77.58±2.17	85.37±1.33	0	22.14±2.51	75.28±1.49	77.64±1.55	85.77±1.52	86.07±1.81
BERT-NSP-Prompt	75.42	77.55±±2.04	81.94±1.90	87.30±1.76	88.77±1.71	91.12±1.66	79.57	80.21±2.08	82.64±1.70	86.77±1.25	89.14±1.51	90.78±1.41

Table 2: Intent accuracy on FewShotATIS and FewShotSNIPS. "BERT-ID" and "RoBERTa-ID" are fine-tuned on BERT and RoBERTa for intent detection.

Model	FewShotATIS					FewShotSNIPS				
	5-shot	10-shot	20-shot	30-shot	40-shot	5-shot	10-shot	20-shot	30-shot	40-shot
BERT-SF(BIO)	22.76±2.17	41.79±1.88	66.64±1.84	76.02±1.46	82.86±1.43	2.86±0.80	40.14±1.73	69.42±1.66	74.72±1.75	81.39±1.84
BERT-SF(ET)	27.17±1.88	49.34±1.63	78.43±1.45	86.85±1.53	90.22±1.20	12.50±2.40	57.33±2.00	76.16±2.11	80.62±2.07	83.48±1.02
BERT-SF(ET+FL)	29.05±1.57	57.44±4.41	83.84±1.44	89.43±1.12	90.50±1.23	15.02±1.80	59.61±1.34	78.51±1.44	82.15±2.10	84.94±1.14
RoBERTa-SF(BIO)	28.12±2.47	64.23±2.71	78.21±1.67	83.27±3.02	88.65±1.05	14.51±1.56	41.09±1.38	66.21±1.13	75.25±1.16	76.71±1.31
RoBERTa-SF(ET)	28.96±1.75	70.85±3.65	82.99±2.00	86.26±1.99	89.71±0.95	22.61±2.25	46.23±2.44	68.08±1.27	77.13±1.72	78.77±1.03
RoBERTa-SF(ET+FT)	31.10±1.76	72.27±3.39	83.59±1.49	87.05±1.84	90.58±1.24	22.54±2.21	49.99±3.49	71.02±2.05	79.91±2.35	80.75±2.47

Table 3: Performance of slot filling on FewShotATIS and FewShotSNIPS (F1-score). "BIO" is the model labeled with BIO format, "ET" denotes the model with reconstructed slot labels, and "ET+FL" represents the model with reconstructing slot labels and the focal loss function.

In addition to the problems mentioned above, in low-resource scenarios, semantically similar slots are often more difficult to distinguish, e.g., from.city_name $vs.$ to.city_name slots in the ATIS dataset. The reasons are as follows. (1) Due to a small number of samples, it is difficult for the model to learn the features of the text structure of the dialogue, such as from location A to location B. (2) Both are slots of location type entities, and the encoded representations are closer in the semantic space. To address the problem that similar labels are difficult to distinguish, we introduce the focal loss function to replace the typically used cross-entropy loss function.

$$\mathrm{FL}\left(p_t\right) = -\left(1-p_t\right)^{\gamma} \log\left(p_t\right), \tag{5}$$

where p_t is the probability that the model predicts the correct, γ is a non-negative hyperparameter that regulates the balance of the loss values of the easy-to-classify and hard-to-classify samples, and when γ is 0, the focal loss is consistent with the calculated result of cross entropy.

The focal loss function introduces the coefficient term $(1-p_t)^{\gamma}$ to the cross entropy for reducing the relative loss of easily-to-classify samples ($p_t > 0.5$) so that the model focuses more on the hard-to-classify samples. When $p_t \to 1$, the loss value of easily-to-classify samples is toned down; when $p_t \to 0$, the coefficient term $(1-p_t)^{\gamma} \to 1$, which is not much different compared to the cross entropy loss. As γ increases, the term $(1-p_t)^{\gamma}$ decreases the contribution of loss values for easy-to-classify samples, and the weights of hard-to-classify samples are relatively elevated, increasing the importance of hard-to-classify samples.

5 Experiments

As stated in the paper, the few-shot SLU setting assumes access to a moderately sized pre-trained language model and a few labeled data, without the data from any other source domain. As such, moderately sized pre-trained models, namely BERT and RoBERTa, are employed as the baseline models. The details of implementation and evaluation metrics can be found in Appendix C. The main reasons for not considering metric learning models as baseline models are as follows. (1) These models necessitate training on source domains with a voluminous dataset to adapt to the data-scarce target domain, which is not feasible in our few-shot SLU setting. (2) The two presented benchmarks are with limited annotation and without source domain data, making it difficult for the previous metric learning models to be trained efficiently. As such, it is impossible to fairly compare the performances of the proposed and previous models on proposed two datasets. To further compare with the previous few-shot SLU models, we attempt to conduct comparative experiments following the dataset setting of Hou et al. (2020a). The results are presented in Appendix D.

5.1 Main Results

We compare the proposed models with previous pre-trained BERT and RoBERTa models on a range of training data amounts in few-shot settings. The results concerning intent detection and slot filling on FewShotATIS and FewShotSNIPS are presented in Table 2 and Table 3 [2], respectively. To mitigate

[2]https://github.com/wyf401/Few-ID-BERT-Prompt

the potential impact of randomness on the experiment, we conducted it five times with different random seeds and reported the average performance with standard deviation. It is evident that the proposed BERT-NSP-Prompt and BERT-SF(ET+FL) outperform the baseline models. It is noteworthy that the proposed model's performance is significantly improved in a more practical scenario.

In particular, BERT-NSP-Prompt has revealed exceptionally competitive results in zero-shot settings. The results verify that BERT-NSP-Prompt can well motivate the knowledge related to the intent detection task in the pretrained model. Even in the absence of training data, intent detection can still leverage the knowledge contained in the pre-trained model to achieve satisfactory performance. BERT-SF (ET+FL) has also demonstrated remarkable performance. Notably, reconstructing slot labels has yielded a remarkable improvement in the model. We attribute the improvement to the method that reduced the complexity of the training process for the slot filling task. In contrast, intent detection performs better and shows a marked improvement in the few-shot settings. The main reasons are outlined below. (1) The intent detection task is relatively straightforward. (2) The features and patterns of intent are easier to capture. It indicates that reducing the classification difficulty of the model is a beneficial idea for few-sample learning. The results demonstrate that the proposed model is stable in terms of performance and brings values of practical applications.

5.2 Effects of Intent Detection Model

Table 2 shows that BERT-NSP-Prompt achieves intent accuracy of 75.42% and 79.57% in zero-shot settings on FewShotATIS and FewShotSNIPS respectively, which indicates that BERT-NSP-Prompt is capable of accurately detecting the user's intent by using prompt and a priori information from the NSP task, even without domain-specific labeled data for training. It justifies that our designed prompt template reduces the gap between the BERT pre-training task and the intent detection task, so as to effectively utilize the knowledge acquired by BERT during the pre-training stage. As the number of samples increases, both BERT-NSP-Prompt and BERT-ID steadily increase in intent accuracy. In the 10-shot setting, BERT-NSP-Prompt still outperforms BERT-ID on FewShotATIS and Few-ShotSNIPS by 5.37% and 3.48%, respectively. Experimental results demonstrate the effectiveness of the proposed model in a variety of amounts of training data. Furthermore, FewShotATIS and FewShotSNIPS can effectively be used to assess the model's generalizability.

Table 2 reports that intent detection using BERT-ID yields an accuracy of only 20.29% on FewShotATIS but 59.02% on Few-ShotSNIPS in the 2-shot setting. An obvious reason is that FewShotATIS contains only samples of the flight domain with relatively high similarity between intent labels, making intent detection more challenging. In contrast, FewShotSNIPS includes more domains. Thus, thereby making the boundaries of different intents more conspicuous. Moreover, the performance of RoBERTa-ID is slightly lower than that of BERT-ID in the few-shot setting. The higher number of RoBERTa parameters may necessitate more training data to adequately fit the downstream task. As the quantity of training data increases, the size of the model plays a dominant factor in performance, and the performance of RoBERTa will continue to improve. Our preliminary analysis is due to the higher number of RoBERTa pre-trained model parameters, which require more training data to fit the downstream task. With more training data, however, the model size plays a dominant factor in model performance, and the performance of RoBERTa will keep improving. An additional advantage of the proposed model is that it can be effortlessly transferred to a multi-intent SLU without any modifications.

5.3 Effects of Reconstructing Slot Labels

Table 3 shows that slot label reconstructing can bring about 4.41% and 7.64% improvement on FewShotATIS in 5-shot and 40-shot settings, respectively. Similarly, it shows that the improvement yields 9.64% and 2.08% respectively in 5-shot and 40-shot settings on FewShotSNIPS. The most significant improvement is observed in the 10-shot setting. The experimental results of BERT-SF (ET+FL) are similar to RoBERTa (ET+FL). It illustrates the effectiveness of the proposed slot label reconstructing approach for slot filling across varying training data ranges. We believe that it is mainly due to the lack of training data in the few-shot setting and that reducing the training complexity of the model is beneficial to its training. Comparing the results of BERT-SF(ET) with BERT-SF(ET+FL) (in Table 3), it has been found that

Sample Strategy	ATIS K-shot				
	2-shot	4-shot	6-shot	8-shot	10-shot
Average Sample	74.60	77.33	85.19	89.64	90.21
Dynamic Sample	74.60	83.49	90.32	89.98	91.34
Sample Strategy	**Snips K-shot**				
	2-shot	4-shot	6-shot	8-shot	10-shot
Average Sample	81.29	87.86	86.29	89.43	90.71
Dynamic Sample	81.29	83.57	88.71	90.29	91.43

Table 4: Intent accuracy of BERT-NSP-Prompt under different sampling strategies.

Sample Strategy	ATIS K-shot				
	5-shot	10-shot	20-shot	30-shot	40-shot
Average Sample	29.26	49.36	82.01	89.93	91.90
Dynamic Sample	29.26	62.76	85.57	90.65	92.01
Sample Strategy	**Snips K-shot**				
	5-shot	10-shot	20-shot	30-shot	40-shot
Average Sample	15.54	60.81	76.69	79.93	83.06
Dynamic Sample	15.54	59.11	80.03	84.56	85.52

Table 5: Performance of BERT-SF(ET+FL) for slot filling under different sampling strategies.

replacing the cross-entropy function with the focal loss function can improve by 8.10% on FewShotATIS and 2.28% on FewShotSNIPS in 10-shot settings. It indicates that introducing focal loss can balance the "hard-to-classify" and "easy-to-classify" slots. In addition, the slot filling task requires more training data for higher performance in comparison to the sentence-level intent detection task. The primary reason is that the slot filling task can be regarded as a word-level classification task, which is more complex.

The results indicate that slot label reconstructing contributes more significantly to the improvement of the proposed model than introducing the focal loss function. It is an exciting exploration to reduce the complexity of slot labels, thereby necessitating a lesser number of samples for training. That means the approach can be extended to other sequence labeling tasks for use in scenarios with limited resources.

5.4 Effects of Dynamic Sampling Strategy

To further assess the effectiveness of the proposed dynamic sampling strategy, we compare the results of applying the dynamic sampling strategy with the average sampling strategy on two datasets in each iteration. Both sampling approaches ensure that the training set of the previous experiment is a subset of the training set of the latter experiment, e.g., the 2-shot training data is encompassed within the 4-shot training data.

The results of applying different sampling strategies on BERT-NSP-Prompt and BERT-SF(ET+FL) for the intent detection and slot filling tasks are shown in Table 4 and 5, where k-shot represents the K samples of each intent. The results of the Dynamic Sampling (DS) strategy are better than the Average Sampling (AS) strategy in the majority of settings on both FewShotATIS and FewShotSNIPS, which indicates that the Dynamic Sampling strategy can effectively sample according to the performance of different intents and slots.

Comparing BERT-NSP-Prompt(AS) with BERT-NSP-Prompt(DS) in the 4-shot and 6-shot settings on FewShotATIS, an improvement of 6.16% and 5.13% was observed, respectively. Additionally, comparing BERT-SF(ET+FL+AS) with BERT-SF(ET+FL+DS) in the 10-shot and 20-shot settings on FewShotATIS, an improvement of 13.4% and 3.56% in intent accuracy was observed, respectively. It verifies that the smaller the number of labeled data within a specific range, the more significant the model performance improvement with the dynamic sampling strategy. It reflects that the SLU model is susceptible to the amount of data due to the limited amounts of training data.

The experimental results for each intent on both datasets are in Appendix E. The experimental results report that the model with a dynamic sampling strategy has a smaller variance than the model with an average sampling strategy. It further demonstrates the effectiveness of the dynamic sampling strategy from fine-grained intent categories.

In addition, we compare the two sampling methods on the pre-trained model RoBERTa. More experiment results are in Appendix F. The experimental results are consistent with the results on BERT. It reflects that the BERT model can sample data according to the difficulty of each intent in the process of dynamic sampling rather than overfitting the BERT model. We believe the dynamic sampling strategy significantly enhances the model's performance in few-shot settings and effectively extends to other data sampling tasks.

5.5 Effects of γ in Focal Loss Function

The setting of γ in the focal loss function significantly affects the model's performance. Therefore, a sensitivity analysis of γ is conducted. We conduct experiments on Few-ShotATIS and FewShotSNIPS for BERT-SF(ET+FL) with the values of γ of 0.25, 0.5, 1, 2, 4. Figure 6 illustrates a sensitivity analysis of γ. In focal loss function, the larger the value of γ, the higher weight the loss function assigns

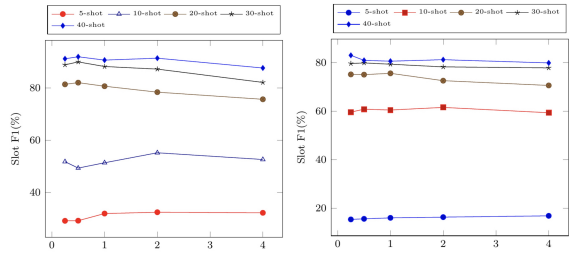

Figure 6: Hyperparameter analysis of γ on FewShotATIS and FewShotSNIPS for slot filling.

to the "hard-to-classify" samples and the model pays more attention to the "hard-to-classify" samples. On both datasets, the F1 scores of slot filling show a trend of increasing and then decreasing with increasing γ, indicating that when assigning too much weight to the "hard-to-classify" samples affects the overall performance of the model. Based on the results, we set γ as 2 in our experiments.

6 Conclusion

In this paper, we explore a more realistic scenario for the few-shot SLU. There are two main contributions toward developing a few-shot SLU. (1) We built two benchmarks, FewShotATIS and FewShotSNIPS, to simulate the few-shot SLU task in a more realistic scenario. (2) We develop BERT-NSP-Prompt, which utilizes BERT's NSP task with a prompt template to detect user intent. In addition, we proposed a reconstructing slot label approach to reduce the difficulty of training the classifier by reducing the number of labels. Experimental results indicate that the proposed model achieves the SOTA performance and endows the SLU module with solid generalization ability. There are two interesting directions for future work in the few-shot setting. (1) Fully explore the close relationship between intent and slot to improve the performance of SLU. (2) It is an idea worth exploring that reduces the complexity of training models.

Acknowledgement

This research is substantially supported by the Key Research and Development Program of Hubei Province (2020BAB017), and the Institute for Scientific Research Center Program of National Language Commission (ZDI135-135), and the Institute for Infocomm Research of A*STAR (CR-2021-001). This research is also supported by the China Scholarship Council (202106770034).

limitations

In this paper, we explore a more realistic scenario for a few-shot SLU and propose the BERT-NSP-Prompt model and a reconstructing slot labels approach. In fact, intent detection and slot filling are highly tied. This paper under-explores fully exploiting the connection between intents and slots to improve the performance of SLU in low-resource settings. In the future, we will explore how to capture the shared knowledge across the two tasks in low-resource settings, which improves the performance of intent detection and slot filling.

References

Samyadeep Basu, Karine lp Kiun Chong, Amr Sharaf, Alex Fischer, Vishal Rohra, Michael Amoake, Hazem El-Hammamy, Ehi Nosakhare, Vijay Ramani, and Benjamin Han. 2021. Semi-supervised few-shot intent classification and slot filling.

Qian Chen, Zhu Zhuo, and Wen Wang. 2019. Bert for joint intent classification and slot filling. *arXiv preprint arXiv:1902.10909*.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv: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. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Nanqing Dong and Eric P Xing. 2018. Few-shot semantic segmentation with prototype learning. In *BMVC*, volume 3.

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

Yingying Gao, Junlan Feng, Chao Deng, and Shilei Zhang. 2022. Meta auxiliary learning for low-resource spoken language understanding. *arXiv preprint arXiv:2206.12774*.

Judith Gaspers, Quynh Do, Daniil Sorokin, Patrick Lehnen, and AI Amazon Alexa. 2021. The impact of

intent distribution mismatch on semi-supervised spoken language understanding. In *Interspeech*, pages 4708–4712.

Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. 2021. Ppt: Pre-trained prompt tuning for few-shot learning.

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

Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. 2022. Ptr: Prompt tuning with rules for text classification. *AI Open*.

Yutai Hou, Wanxiang Che, Yongkui Lai, Zhihan Zhou, Yijia Liu, Han Liu, and Ting Liu. 2020a. Few-shot slot tagging with collapsed dependency transfer and label-enhanced task-adaptive projection network. *arXiv preprint arXiv:2006.05702*.

Yutai Hou, Cheng Chen, Xianzhen Luo, Bohan Li, and Wanxiang Che. 2022. Inverse is better! fast and accurate prompt for few-shot slot tagging. *arXiv preprint arXiv:2204.00885*.

Yutai Hou, Yongkui Lai, Cheng Chen, Wanxiang Che, and Ting Liu. 2021a. Learning to bridge metric spaces: Few-shot joint learning of intent detection and slot filling. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3190–3200, Online. Association for Computational Linguistics.

Yutai Hou, Yongkui Lai, Cheng Chen, Wanxiang Che, and Ting Liu. 2021b. Learning to bridge metric spaces: few-shot joint learning of intent detection and slot filling. *arXiv preprint arXiv:2106.07343*.

Yutai Hou, Jiafeng Mao, Yongkui Lai, Cheng Chen, Wanxiang Che, Zhigang Chen, and Ting Liu. 2020b. Fewjoint: A few-shot learning benchmark for joint language understanding.

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

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization.

Jason Krone, Yi Zhang, and Mona Diab. 2020a. Learning to classify intents and slot labels given a handful of examples.

Jason Krone, Yi Zhang, and Mona Diab. 2020b. Learning to classify intents and slot labels given a handful of examples. *arXiv preprint arXiv:2004.10793*.

Ayush Kumar, Rishabh Kumar Tripathi, and Jithendra Vepa. 2022. Low resource pipeline for spoken language understanding via weak supervision.

Stefan Larson, Anish Mahendran, Joseph J Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K Kummerfeld, Kevin Leach, Michael A Laurenzano, Lingjia Tang, et al. 2019. An evaluation dataset for intent classification and out-of-scope prediction. *arXiv preprint arXiv:1909.02027*.

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

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*.

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.

Zihan Liu, Genta Indra Winata, Peng Xu, and Pascale Fung. 2020. Coach: A coarse-to-fine approach for cross-domain slot filling. *arXiv preprint arXiv:2004.11727*.

Amber Nigam, Prashik Sahare, and Kushagra Pandya. 2019. Intent detection and slots prompt in a closed-domain chatbot. In *2019 IEEE 13th international conference on semantic computing (ICSC)*, pages 340–343. IEEE.

Patti Price. 1990. Evaluation of spoken language systems: The atis domain. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*.

Libo Qin, Tailu Liu, Wanxiang Che, Bingbing Kang, Sendong Zhao, and Ting Liu. 2021. A co-interactive transformer for joint slot filling and intent detection. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8193–8197. IEEE.

Libo Qin, Xiao Xu, Wanxiang Che, and Ting Liu. 2020. Agif: An adaptive graph-interactive framework for joint multiple intent detection and slot filling. *arXiv preprint arXiv:2004.10087*.

Chengyu Wang, Suyang Dai, Yipeng Wang, Fei Yang, Minghui Qiu, Kehan Chen, Wei Zhou, and Jun Huang. 2022. Arobert: An asr robust pre-trained language model for spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:1207–1218.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2019. Huggingface's transformers: State-of-the-art natural language processing.

Fengyi Yang, Xi Zhou, Yi Wang, Abibulla Atawulla, and Ran Bi. 2022. Diversity features enhanced prototypical network for few-shot intent detection.

Yi Yang and Arzoo Katiyar. 2020. Simple and effective few-shot named entity recognition with structured nearest neighbor learning.

Dian Yu, Luheng He, Yuan Zhang, Xinya Du, Panupong Pasupat, and Qi Li. 2021. Few-shot intent classification and slot filling with retrieved examples. *arXiv preprint arXiv:2104.05763*.

Jian-Guo Zhang, Kazuma Hashimoto, Wenhao Liu, Chien-Sheng Wu, Yao Wan, Philip S Yu, Richard Socher, and Caiming Xiong. 2020. Discriminative nearest neighbor few-shot intent detection by transferring natural language inference. *arXiv preprint arXiv:2010.13009*.

Su Zhu, Ruisheng Cao, Lu Chen, and Kai Yu. 2020. Vector projection network for few-shot slot tagging in natural language understanding. *arXiv preprint arXiv:2009.09568*.

Yi Zhu, Xinke Zhou, Jipeng Qiang, Yun Li, Yunhao Yuan, and Xindong Wu. 2022a. Prompt-learning for short text classification.

Yi Zhu, Xinke Zhou, Jipeng Qiang, Yun Li, Yunhao Yuan, and Xindong Wu. 2022b. Prompt-learning for short text classification.

A The Results of BERT

To simulate intent detection and slot-filling tasks in low-resource scenarios, we sample few-shot data from the ATIS and Snips datasets and conduct k-shot experiments. We set fine-grained k values in the experiment to observe how the model's performance changes as the training data increases. The k-shot settings for intent detection and slot filling are shown in Table 6 and Table 7. We can reasonably set the k value according to the experimental results when constructing the k-shot dataset. In this way, the model's performance can be thoroughly evaluated through only a few k-shot experiments.

B Description of each intent in the Snips dataset

Table 8 shows the description text of each intent in the SNIPS dataset, which contains seven intents.

Intent	Description Text
AddToPlaylist	He is adding something to playlist.
PlayMusic	He wants to play some music.
BookRestaurant	He is booking a restaurant.
SearchCreativeWork	He is searching for something like game, videos or TV shows.
RateBook	He is rating a book with a score.
GetWeather	He wants to know the weather information.
SearchScreeningEvent	He is searching for a movie.

Table 8: Description of each intent in the SNIPS dataset.

C Implementation Detail and Evaluation Metrics

We implement BERT (Devlin et al., 2018), and RoBERTa (Liu et al., 2019) models based on Hugging-face Pytorch Transformers (Wolf et al., 2019). The pre-trained model BERT is initialized with Bert-base-uncased. The pre-trained model RoBERTa is initialized with RoBERTa-base. It contains a 12-layer Transformer encoder, a multi-headed attention mechanism containing 12 attention heads on each layer, and a hidden layer size of 768. The initial learning rate of the model is 5e-5. The sentence length is 80. The model parameters are optimized using Adam (Kingma and Ba, 2014) with a weight decay optimizer. The dropout is 0.1. With the above experimental environment and parameter settings, the experiment requires 4,776M memory and spends 145s of each epoch training time. To overcome the randomness of the experiment, we ran it 5 times with different random seeds and reported average performance and standard deviation for the experiments. The intent accuracy is used as the evaluation metric of the intent detection task, while the slot F1 score is used as the evaluation metric of the slot filling task.

D The Result of SNIPS dataset

We evaluate our models following the k-shot dataset provided by Hou et al. (2020a) on SNIPS (Coucke et al., 2018). SNIPS contains seven intents: GetWeather (We), PlayMusic (Mu), AddToPlaylist (PI), RateBook (Bo), SearchScreeningEvent(Se), BookRestaurant (Re), and SearchCreativeWork (Cr). The previous few-shot SLU methods train models on five source domains, use the sixth one for development, and test on the seventh domain. And then, we complete the testing of all seven domains by cross-validation seven times. In comparison, our model is only fine-tuned in the testing domain. Table 9 summarizes the experiment results on the SNIPS dataset. TransferBERT and SimBERT results are reported in Hou et al. (2020a). Moreover, the ProtoToken result

Model	K-shot									
	1-shot	2-shot	3-shot	4-shot	5-shot	6-shot	7-shot	8-shot	9-shot	10-shot
ATIS	8.93	18.11	25.73	31.55	39.87	59.57	67.92	72.44	80.61	87.70
SNIPS	32.49	59.29	64.26	78.71	79.64	82.29	83.06	86.71	85.81	87.86

Table 6: Performance of BERT for intent detection.

Model	K-shot							
	5-shot	10-shot	15-shot	20-shot	25-shot	30-shot	35-shot	40-shot
ATIS	22.21	41.21	52.17	66.23	70.14	78.06	78.92	82.21
SNIPS	1.8	41.98	50.80	69.63	72.04	75.46	77.93	80.38

Table 7: Performance of BERT for slot filling.

Model	We	Mu	PI	Bo	Se	Re	Cr	Avg.
TransferBERT	59.41	42.00	46.07	20.74	28.20	67.75	58.61	46.11
SimBERT	53.46	54.13	42.81	75.54	57.10	55.30	32.38	52.96
ProtoToken	67.82	55.99	46.02	72.17	73.59	60.18	66.89	63.24
Coach (Liu et al., 2020)	73.56	45.85	47.23	61.61	65.82	69.99	57.28	60.19
TapNet (Hou et al., 2020a)	53.03	49.80	54.90	83.36	63.07	59.84	67.02	61.57
TapNet+CDT (Hou et al., 2020a)	66.48	66.36	68.23	85.76	73.60	64.20	**68.47**	70.44
L-WPZ+CDT (Hou et al., 2020a)	74.68	56.73	52.20	78.79	80.61	69.59	67.46	68.58
L-TapNet+CDT (Hou et al., 2020a)	71.64	67.16	75.88	84.38	82.58	70.05	73.41	75.01
Retriever (Yu et al., 2021)	**82.95**	61.74	71.75	81.65	73.10	79.54	51.35	71.72
Ours	79.70	**78.09**	**76.64**	**90.04**	**82.87**	**79.91**	63.88	**78.73**

Table 9: F1 Scores on 5-shot dialogue language understanding task on SNIPS dataset.

is reported in Yu et al. (2021). Experiments reflect that the proposed model achieves competitive performance compared to previous few-shot SLU models even though it does not use source domains.

It is worth noting that our model is only fine-tuned on the testing domain without training on any source domains. The main reason for the model performance improvement is that the reconstructing slot label approach reduces the complexity of model training. Specifically, the test set has only one domain. Thus there are few categories of slots. Moreover, reconstructing slot labels reduces the number of slot types in half. Therefore, it extremely reduces the search space of the model. Experiments demonstrate that the reconstructing slot label approach is simple and effective in the few-shot SLU.

E Intent Accuracy on FewShotSNIPS and FewShotATIS

The results of BERT-NSP-Prompt on the FewShotSNIPS and FewShotATIS (6-shot) are reported in Table 10 and Table 11. The experimental results report that the model with a dynamic sampling strategy has a smaller variance than the model with average sampling. The main reason for the model performance improvement may be that the dynamic sampling strategy can assign a different number of samples according to the learning difficulty of each intent. It further demonstrates the effectiveness of the dynamic sampling strategy. This sampling method is more in line with the practical scenario and improves the overall performance of intent detection.

F The Results of RoBERTa in Different Sampling Strategies

The results of different sampling strategies on the RoBERTa models for the intent detection and slot filling task are shown in Table 12 and Table 13. With the increase in the total number of samples, the experimental results of the dynamic sampling strategy show a steady improvement trend. The experimental results are consistent with the results on BERT. It reflects that the BERT model can sample data according to the difficulty of each intent in the process of dynamic sampling rather than overfitting the BERT model. We believe the dynamic sampling strategy significantly enhances the model's performance in few-shot settings and effectively extends to other data sampling tasks.

Intent	6-shot Average Sample			6-shot Dynamic Sample			Samples
	Precision	Recall	F1	Precision	Recall	F1	
AddToPlaylist	97.00	78.23	86.61	99.10	88.71	93.62	124
PlayMusic	64.96	88.37	74.88	72.32	94.19	81.82	86
BookRestaurant	96.81	98.91	97.85	95.83	1.00	97.87	92
SearchCreativeWork	71.03	71.03	71.03	73.17	84.11	78.26	107
RateBook	98.75	98.75	98.75	98.59	87.50	92.72	80
GetWeather	97.09	96.15	96.62	96.19	97.12	96.65	104
SearchScreeningEvent	85.86	79.44	82.52	93.90	71.96	81.48	107

Table 10: The results of different intents in a 6-shot setting on FewShotSNIPS.

Intent	6-shot Average Sample			6-shot Dynamic Sample			Samples
	Precision	Recall	F1	Precision	Recall	F1	
atis_flight	99.07	84.34	91.11	98.11	90.51	94.16	632
atis_ground_fare	60.00	85.71	70.59	50.00	71.43	58.82	7
atis_flight_time	25.00	1.00	40.00	25.00	1.00	40.00	1
atis_quantity	0.00	0.00	0.00	0.00	0.00	0.00	3
atis_distance	81.82	90.00	85.71	1.0000	1.0000	1.0000	10
atis_city	19.35	100.00	32.43	66.67	66.67	66.67	6
atis_capacity	100.00	85.71	92.31	100.00	100.00	100.00	21
atis_aircraft	56.25	100.00	72.00	87.50	77.78	82.35	9
atis_flight_no	88.89	100.00	94.12	100.00	100.00	100.00	8
atis_cheapest	0.00	0.00	0.00	0.00	0.00	0.00	0
atis_airline	55.88	100.00	71.70	61.67	97.37	75.51	38
atis_abbreviation	100.00	78.79	88.14	91.18	93.94	92.54	33
atis_restriction	0.00	0.00	0.00	0.00	0.00	0.00	0
atis_airport	90.00	100.00	94.74	94.74	100.00	97.30	18
atis_meal	42.86	100.00	60.00	31.25	83.33	45.45	6
atis_airfare	74.00	77.08	75.51	83.67	85.42	84.54	48
atis_ground_service	100.00	86.11	92.54	100.00	86.11	92.54	36
atis_day_name	50.00	100.00	66.67	66.67	100.00	80.00	2

Table 11: The results of different intents in a 6-shot setting on FewShotATIS.

Model	ATIS K-shot					Snips K-shot				
	2-shot	4-shot	6-shot	8-shot	10-shot	2-shot	4-shot	6-shot	8-shot	10-shot
RoBERTa-ID(AS)	4.55	4.67	71.75	72.83	85.42	20.14	75.57	75.71	82.02	83.71
RoBERTa-ID(DS)	4.55	7.57	81.32	73.87	86.64	20.14	77.00	77.86	84.43	85.57

Table 12: Intent accuracy of different sampling strategies. "AS" is the Average Sample, and "DS" denotes the Dynamic Sample.

Model	ATIS K-shot					Snips K-shot				
	5-shot	10-shot	20-shot	30-shot	40-shot	5-shot	10-shot	20-shot	30-shot	40-shot
RoBERTa-SF(BIO+AS)	29.08	59.89	77.03	83.46	87.39	14.35	42.33	65.42	73.63	77.28
RoBERTa-SF(BIO+DS)	29.08	67.96	80.16	87.26	90.35	14.35	42.48	67.52	76.28	77.32
RoBERTa-SF(ET+AS)	29.11	65.00	80.59	88.61	88.65	22.42	49.58	67.59	76.61	77.61
RoBERTa-SF(ET+DS)	29.11	77.03	85.94	86.09	90.48	22.42	43.51	67.63	74.67	78.12
RoBERTa-SF(ET+FL+AS)	31.51	65.24	81.07	86.78	89.53	24.10	54.39	68.25	75.25	78.16
RoBERTa-SF(ET+FT+DS)	31.51	77.09	83.77	88.24	91.02	24.10	45.99	70.97	78.12	78.42

Table 13: The performance of different sampling strategies for slot filling (F1). "BIO" is the model labeled with BIO format, "ET" denotes the model with reconstructed slot labels, and "ET+FL" represents the model with reconstructing slot labels and the focal loss function. "AS" is the Average Sample, and "DS" denotes the Dynamic Sample.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
section 7

☐ A2. Did you discuss any potential risks of your work?
Not applicable. Left blank.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
section 1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

section 3

☑ B1. Did you cite the creators of artifacts you used?
section 3

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Not applicable. Left blank.

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
section 3

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Not applicable. Left blank.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
section 3

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
section 3

C ☑ Did you run computational experiments?

section 5

☐ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
No response.

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Appendix C

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
section 5

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Appendix C

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Not applicable. Left blank.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Not applicable. Left blank.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Not applicable. Left blank.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Not applicable. Left blank.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Not applicable. Left blank.