FewNLU: Benchmarking State-of-the-Art Methods for Few-Shot Natural Language Understanding

Yanan Zheng[*12], **Jing Zhou**[*1], **Yujie Qian**[3], **Ming Ding**[1], **Chonghua Liao**[1]
Jian Li[1], **Ruslan Salakhutdinov**[4], **Jie Tang**[†12], **Sebastian Ruder**[† 5], **Zhilin Yang**[†16]
[1]Tsinghua University, [2]BAAI, [3]MIT CSAIL,
[4]Carnegie Mellon University, [5]Google Research, [6]Shanghai Qi Zhi Institute
{zyanan, jietang, zhiliny}@tsinghua.edu.cn,
zhouj18@mails.tsinghua.edu.cn, ruder@google.com

Abstract

The few-shot natural language understanding (NLU) task has attracted much recent attention. However, prior methods have been evaluated under a disparate set of protocols, which hinders fair comparison and measuring progress of the field. To address this issue, we introduce an evaluation framework that improves previous evaluation procedures in three key aspects, i.e., test performance, dev-test correlation, and stability. Under this new evaluation framework, we re-evaluate several state-of-the-art few-shot methods for NLU tasks. Our framework reveals new insights: (1) both the absolute performance and relative gap of the methods were not accurately estimated in prior literature; (2) no single method dominates most tasks with consistent performance; (3) improvements of some methods diminish with a larger pretrained model; and (4) gains from different methods are often complementary and the best combined model performs close to a strong fully-supervised baseline. We open-source our toolkit, FewNLU, that implements our evaluation framework along with a number of state-of-the-art methods. [1] [2]

1 Introduction

Few-shot learning for natural language understanding (NLU) has been significantly advanced by pretrained language models (PLMs; Brown et al., 2020; Schick and Schütze, 2021a,b). With the goal of learning a new task with very few (usually less than a hundred) samples, few-shot learning benefits from the prior knowledge stored in PLMs. Various few-shot methods based on PLMs and prompting have been proposed (Liu et al., 2021b; Menon et al., 2021; Gao et al., 2020).

Although the research of few-shot NLU is developing rapidly, the lack of a standard evaluation protocol has become an obstacle hindering fair comparison between various methods on a common ground and measuring progress of the field. While some works (Schick and Schütze, 2021b; Menon et al., 2021) experimented with a fixed set of hyper-parameters, prior work (Perez et al., 2021; Zhang et al., 2020) noted that such a setting might be exposed to the risk of overestimation .[3] Other works (Liu et al., 2021b; Gao et al., 2020; Perez et al., 2021) proposed to use a small development set to select hyper-parameters, but their evaluation protocols vary in a few key aspects (e.g., how to construct data splits), which in fact lead to large differences as we will show (in Section 4.2). The above phenomena highlight the need for a common protocol for the evaluation of few-shot NLU methods. However, the fact that few-shot learning is extremely sensitive to subtle variations of many factors (Dodge et al., 2020; Gao et al., 2020) poses challenges for designing a solid evaluation protocol.

In this work, aiming at addressing the aforementioned challenge, we propose an evaluation framework for few-shot NLU. The evaluation framework consists of a repeated procedure—selecting a hyper-parameter, selecting a data split, training and evaluating the model. To set up a solid evaluation framework, it is crucial to specify a key design choice—how to construct data splits for model selection. We conduct a comprehensive set of experiments to answer the question. Specifically, we propose a "Multi-Splits" strategy, which randomly splits the available labeled samples into training and development sets multiple times, followed by aggregating the results from each data split. We show that this simple strategy outperforms several

[*] The authors have contributed equally to this work.
[†] Corresponding Authors.
[1] Leaderboard: https://fewnlu.github.io
[2] Code available at https://github.com/THUDM/FewNLU
[3] This is because the fixed hyper-parameters are selected according to practical considerations, which are informed by the test set performance from previous evaluations.

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 501 - 516
May 22-27, 2022 ©2022 Association for Computational Linguistics

baseline strategies in three dimensions: (1) the test set performance of the selected hyper-parameters; (2) correlation between development set and true test set performance; and (3) robustness to hyper-parameter settings.

We then take a step further to re-evaluate recent state-of-the-art few-shot NLU methods under this common evaluation framework. Our re-evaluation leads to several key findings summarized in Section 2.

To aid reproducing our results and benchmarking few-shot NLU methods, we open-source FewNLU, a toolkit that contains implementations of a number of state-of-the-art methods, data processing utilities, as well as our proposed evaluation framework.

To sum up, our contributions are as follows.

1. We introduce a new evaluation framework of few-shot NLU. We propose three desiderata of few-shot evaluation and show that our framework outperforms previous ones in these aspects. Thus our framework allows for more reliable comparison of few-shot NLU methods.
2. Under the new evaluation framework, we benchmark the performance of recent methods individually as well as the best performance with a combined approach. These benchmarks reflect the current state of the art and will serve as important baselines for future research.
3. Throughout our exploration, we arrive at several key findings summarized in Section 2.
4. We open-source a toolkit, FewNLU, to facilitate future research with our framework.

2 Summary of Findings

For reference, we collect our key findings here and discuss each of them throughout the paper.

Finding 1. Our proposed Multi-Splits is a more reliable data-split strategy than several baselines with improvements in (1) test performance, (2) correlation between development and test sets, and (3) stability w.r.t. the number of runs.

Finding 2. The absolute performance and the relative gap of few-shot methods were in general not accurately estimated in prior literature. It highlights the importance of evaluation for obtaining reliable conclusions. Moreover, the benefits of some few-shot methods (e.g., ADAPET) decrease on larger pretrained models.

Finding 3. Gains of different methods are largely complementary. A combination of methods largely outperforms individual ones, performing close to a strong fully-supervised baseline with RoBERTa.

Finding 4. No single few-shot method dominates most NLU tasks. This highlights the need for the development of few-shot methods with more consistent and robust performance across tasks.

3 Related Work

The pretraining-finetuning paradigm (Howard and Ruder, 2018) shows tremendous success in few-shot NLU tasks. Various methods have been developed such as [CLS] classification (Devlin et al., 2018), prompting-based methods with discrete prompts (Schick and Schütze, 2021b; Gao et al., 2020) or continuous prompts (Liu et al., 2021b; Shin et al., 2020; Li and Liang, 2021; Lester et al., 2021), and methods that calibrate the output distribution (Yang et al., 2021; Zhao et al., 2021).

The fact that few-shot learning is sensitive to many factors and thus is extremely unstable (Liu et al., 2021a; Lu et al., 2021; Zhang et al., 2020; Dodge et al., 2020) increases the difficulty of few-shot evaluation. Several works address evaluation protocols to mitigate the effects of instability: Gao et al. (2020) and Liu et al. (2021b) adopt a held-out set to select models. Perez et al. (2021) proposed K-fold cross-validation and minimum description length evaluation strategies. Our work differs from these works on few-shot evaluation in several aspects: (1) we propose three metrics to evaluate data split strategies; (2) while most prior work proposed evaluation protocols without justification, we conduct comprehensive experiments to support our key design choice; (3) we formulate a general evaluation framework; (4) our re-evaluation under the proposed framework leads to several key findings.

Though there have been a few existing few-shot NLP benchmarks, our work is quite different in terms of the key issues addressed. FLEX (Bragg et al., 2021) and CrossFit (Ye et al., 2021) studied principles of designing tasks, datasets, and metrics. FewGLUE (Schick and Schütze, 2021b) is a dataset proposed for benchmarking few-shot NLU. CLUES (Mukherjee et al., 2021) pays attention to the unified format, metric, and the gap between human and machine performance. While the aforementioned benchmarks focus on "what data to use" and "how to define the task", our work discussed "how to evaluate" which aims at establishing a proper evaluation protocol for few-shot NLU methods. Since FewNLU is orthogonal to the aforementioned prior work, it can also be employed on the data and tasks proposed in previous work.

4 Evaluation Framework

Formally, for a few-shot NLU task, we have a small labeled set $\mathcal{D}_{\text{label}} = \{(x_i, y_i)\}_i^N$ and a large test set $\mathcal{D}_{\text{test}} = \{(x_i^{\text{test}}, y_i^{\text{test}})\}_i$ where N is the number of labeled samples, x_i is a text input (consisting of one or multiple pieces), and $y_i \in \mathcal{Y}$ is a label. The goal is to finetune a pretrained model with $\mathcal{D}_{\text{label}}$ to obtain the best performance on $\mathcal{D}_{\text{test}}$. An unlabeled set $\mathcal{D}_{\text{unlab}} = \{x_i^{\text{unlab}}\}_i$ may additionally be used by semi-supervised few-shot methods (§5.1).

4.1 Formulation of Evaluation Framework

Our preliminary results (in Appendix §A.1) show that using a fixed set of hyper-parameters (Schick and Schütze, 2021a,b) is sub-optimal, and model selection is required. It motivates us to study a more robust evaluation framework for few-shot NLU. The goal of an evaluation framework is twofold: (1) benchmarking few-shot methods for NLU tasks such that they can be fairly compared and evaluated; and (2) obtaining the best few-shot performance in practice. In light of the two aspects, we propose the few-shot evaluation framework in Algorithm 1.

The framework searches over a hyper-parameter space $\mathcal{H}$ to evaluate a given few-shot method M, obtaining the best hyper-parameter setting $h^\star$ and its test set results. [4] The measurement for each h is estimated by performing training and evaluation on multiple data splits (obtained by splitting $\mathcal{D}_{\text{label}}$ according to a strategy) and reporting their average dev set results. Finally, the method is evaluated on $\mathcal{D}_{\text{test}}$ using the checkpoints corresponding to $h^\star$. For benchmarking, we report the average and standard deviation over multiple test set results. Otherwise, that is, to achieve a model with the best practical performance, we re-run on the entire $\mathcal{D}_{\text{label}}$ with $h^\star$.

The framework requires specifying a key design choice—how to construct the data splits, which we will discuss in §4.2.

Algorithm 1: A Few-Shot Evaluation Framework

```
Data: $\mathcal{D}_{\text{label}}$, $\mathcal{D}_{\text{test}}$, a hyper-parameter space $\mathcal{H}$, a
      few-shot method $M$, the number of runs $K$.
Result: test performance; best hyper-parameter $h^\star$.
for $k \leftarrow 1 \cdots K$ do
    Divide $\mathcal{D}_{\text{label}}$ into $\mathcal{D}^k_{\text{train}}$ and $\mathcal{D}^k_{\text{dev}}$ according to a
      data-split strategy;
end
for $h \in \mathcal{H}$ do
    for $k \leftarrow 1 \cdots K$ do
        Run the method $M$ by training on $\mathcal{D}^k_{\text{train}}$ and
          evaluating on $\mathcal{D}^k_{\text{dev}}$;
        Report the dev-set performance $\mathcal{P}^{h,k}_{\text{dev}}$.
    end
    Compute the mean and standard deviation over
      $K$ dev-set results, $\mathcal{P}^h_{\text{dev}} \pm \mathcal{S}^h_{\text{dev}}$;
end
Select $h^\star$ with the best $\mathcal{P}^h_{\text{dev}}$.;
if the goal is to evaluate a method then
    Evaluate on the test set $\mathcal{D}_{\text{test}}$ with the $K$
      checkpoints that correspond to $h^\star$;
    Report the mean and standard deviation over the
      $K$ test results $\mathcal{P}^{h\star}_{\text{test}} \pm \mathcal{S}^{h\star}_{\text{test}}$.
else if the goal is to obtain the best performance then
    Re-run on the entire $\mathcal{D}_{\text{label}}$ using fixed $h^\star$ with $L$
      different random seeds;
    Evaluate on the test set with the $L$ checkpoints;
    Report the mean and stddev over $L$ test results.
end
```

4.2 How to Construct Data Splits

4.2.1 Desiderata: Performance, Correlation, and Stability

We first propose the following three key desiderata for the evaluation of different data split strategies.

1. **Performance of selected hyper-parameter.** A good data split strategy should select a hyper-parameter that can achieve a good test set performance. We use the same metrics as (Schick and Schütze, 2021b), along with standard deviations.
2. **Correlation between dev and test sets (over a hyper-parameter distribution)**. Since a small dev set is used for model selection, it is important for a good strategy to obtain a high correlation between the performances on the small dev set and test set over a distribution of hyper-parameters. We report the Spearman's rank correlation coefficient for measurement.
3. **Stability w.r.t. number of runs K.** The choice of the hyper-parameter K should have small impacts on the above two metrics (i.e., performance and correlation). To analyze the stability w.r.t K, we report the standard deviation over multiple different values of K. Besides, it is desirable to have reduced variance when K increases. Thus we report the above two metrics with different

[4] For simplicity and ease of use, we use grid search for searching the hyper-parameter space $\mathcal{H}$ and identify critical hyper-parameters to limit its size. More complex search methods such as Bayesian Optimization (Snoek et al., 2012) could be used to search over larger hyper-parameter spaces.

values of K and the standard deviation of test scores over K runs.

4.2.2 Data Split Strategies

We consider several data split strategies. Some are proposed by previous work, including K-fold cross validation (CV) (Perez et al., 2021), minimum description length (MDL) (Perez et al., 2021), and bagging (BAG) (Breiman, 1996). We also consider two simple strategies worth exploring, including random sampling (RAND) and model-informed splitting (MI). And we propose a new data split strategy, Multi-Splits (MS). Besides, we also experiment a special case of CV when K equals the number of labeled sample, which is leave-of-out cross validation (LOOCV). Since LOOCV takes much longer time and suffers from efficiency problem, we only experimented on several tasks and left the results in Appendix A.2.4. They all fit into the pipeline of the proposed framework in §4.1:

1. **K-fold CV** equally partitions $\mathcal{D}_{\text{label}}$ into K folds. Each time, it uses the k^{th} fold for validation and the other $K-1$ folds for training.
2. **MDL** assigns half of $\mathcal{D}_{\text{label}}$ as the joint training data and equally partitions the other half into K folds. Each time, it uses the k^{th} fold for validation, and all its previous $k-1$ folds together with the joint training data for training.
3. **Bagging** samples $N \times r$ ($r \in (0, 1]$ is a fixed ratio) examples with replacement from the labeled sample as the training set, leaving samples that do not appear in the train set for validation.
4. **Random Sampling** performs random sampling without replacement from $\mathcal{D}_{\text{label}}$ twice, respectively sampling $N \times r$ and $N \times (1-r)$ data as the training and development sets.
5. **Model-Informed Splitting** computes representations of each labeled example using a model, and clusters them into two distinct sets, respectively as the training and development sets. [5]
6. **Multi-Splits** randomly splits $\mathcal{D}_{\text{label}}$ into training and development sets using a fixed split ratio r.

Essentially, these data split strategies differ in several key aspects.

	#Train	#Dev
CV	$(K-1) \times N/K$	N/K
MDL	$N/2 + N(k-1)/(2K)$	$N/(2K)$
BAG	$N \times r$	$> (N \times (1-r))$
RAND	$N \times r$	$N \times (1-r)$
Multi-Splits	$N \times r$	$N \times (1-r)$

Table 1: Number of examples of training and development sets for different strategies. N: number of labeled data, K: number of runs, k: the k^{th} split for MDL; r: split ratio. MI is omitted since its number of examples depends on the dataset.

1. For CV and MDL, K controls the number of runs and the split ratio. For Multi-Splits, BAG and RAND, the split ratio is decoupled from K and is controlled by r. For MI, the split ratio and number of runs depend on $\mathcal{D}_{\text{label}}$.
2. They use a different amount of data for training and development sets as Table 1 shows.
3. There are cases when CV and MS share the same split ratio. The difference is that MS allows overlap between splits while CV does not.
4. BAG allows duplicated training data, while RAND and Multi-Splits do not. The training and development sets do not overlap for BAG and Multi-Splits but overlap for RAND.

In the limit, our Multi-Splits is similar to leave-P-out cross-validation (LPOCV; Celisse, 2014)[6] where LPOCV runs $\binom{N}{P}$ times (P is the number of dev set examples) while Multi-Splits runs K times. As K increases, Multi-Splits gradually approaches LPOCV. Since it is impossible to enumerate the large number of possible splits in practice, Multi-Splits can be viewed as a practical version of LPOCV. Compared to the strategy of (Gao et al., 2020) that uses multiple datasets, our Multi-Splits uses multiple data splits for a single dataset. It is thus more practical as in real-world scenarios, it is hard to obtain multiple labeled datasets for a true few-shot problem; otherwise, it could be formulated as a fully-supervised learning problem. The strategy in (Liu et al., 2021b) is a special case of Multi-Splits when $K = 1$, which suffers from higher variance.

4.2.3 Experimental Setup

To evaluate different data split strategies, we experiment on the FewGLUE benchmark (Schick and Schütze, 2021b). We evaluate strategies based on the widely used prompt-based few-shot method PET (Schick and Schütze, 2021b) with DeBERTa as the base model.[7] We run experiments on the same tasks with the same hyper-parameter space

[5] Specifically, we used a BERT-Base model to encode data and take the [CLS] representations.

[6] Leave-P-out cross-validation uses P data examples as the development set and the remaining data examples as the training set. This is repeated on all ways to cut the labeled dataset in a development set and a training set.

[7] We fixed the parameters of DeBERTa's bottom one-third layers due to GPU memory limitations, which did not affect the performance much in our preliminary experiments.

	BoolQ	RTE	WiC	CB		MultiRC		WSC	COPA	Avg.
	Acc.	Acc.	Acc.	Acc.	F1	F1a	EM.	Acc.	Acc	
CV	82.71 ±1.29	77.80 ±2.25	64.42 ±1.63	90.18 ±2.31	87.52 ±2.20	80.08 ±1.15	45.02 ±1.46	82.45 ±3.71	92.25 ±1.71	78.72
MDL	76.43 ±7.12	83.94 ±1.49	63.68 ±3.38	84.38 ±5.13	82.03 ±5.69	77.63 ±1.20	43.81 ±1.32	81.49 ±3.95	89.50 ±3.32	77.00
BAG	81.77 ±1.48	77.98 ±1.56	65.56 ±3.26	87.50 ±6.90	77.15 ±13.76	79.62 ±1.26	43.60 ±1.98	85.34 ±2.87	88.75 ±3.10	77.62
RAND	78.79 ±5.40	82.13 ±0.91	59.60 ±3.89	86.16 ±3.05	74.04 ±12.94	80.14 ±2.20	44.88 ±4.45	84.38 ±2.99	90.75 ±3.59	76.89
MI	78.25 ±1.59	77.35 ±4.06	64.66 ±1.48	88.84 ±1.71	84.75 ±4.32	76.75 ±0.44	40.95 ±0.10	83.41 ±6.00	78.75 ±8.06	75.44
MS	82.67 ±0.78	79.42 ±2.41	67.20 ±1.34	91.96 ±3.72	88.63 ±4.91	78.20 ±1.86	42.42 ±3.04	84.13 ±4.87	89.00 ±2.94	79.00

Table 2: Test performance of different data-split strategies with PET on FewGLUE (K=4).Larger scores means the strategy effectively selects a model that achieves better test set performance.

	BoolQ	RTE	WiC	CB	MultiRC	WSC	COPA	Avg.
CV	-0.0497	0.8561	0.8184	0.5286	0.2283	0.1507	0.5668	0.4427
MDL	-0.1143	0.7806	0.6326	0.3274	0.1910	0.1278	0.6342	0.3685
BAG	0.5533	0.8714	0.9572	0.6809	0.6340	0.2550	0.7491	0.6716
RAND	0.7453	0.7602	0.8048	0.6764	0.3253	0.0795	0.9004	0.6131
MI	0.5651	0.6832	0.7780	0.6618	0.6651	0.0200	0.5902	0.5662
MS	0.7079	0.8266	0.9464	0.7558	0.4983	0.3986	0.8997	0.7190

Table 3: Correlation results of different data-split strategies with PET on FewGLUE (K=4). Larger values means the strategy is better at selecting the best test results using dev sets.

to ensure a fair comparison; in this experiment we search learning rate, evaluation ratio, prompt pattern and maximum training step. More experimental details are in Appendix A.2.

4.2.4 Main Results and Analysis

Table 2, Table 3 and Figure 1 show the main results with 64 labeled samples.

It is noteworthy that we also experimented with 32 labeled samples and have observed that varying the number of labeled examples does not affect the following conclusion (see Appendix A.2).

Test Performance and Correlation. From both Table 2 and Table 3, we find that Multi-Splits achieves the best average test set performance as well as the best average correlation among all strategies. We analyze them as follows:[8]

1. Multi-Splits uses fewer labeled samples for training (i.e., 128) while CV and MDL use more (i.e., 192 and 176). Despite using more training data, both CV and MDL do not perform better. This indicates few-shot performance is limited by not being able to select the best model rather than not having sufficient training data. Both CV and MDL use fewer data for validation (i.e., 64 and 32) than Multi-Splits (i.e., 128), thus leading to poor correlation.
2. Although Multi-Splits and BAG use the same number of training data (i.e., 128), there could be duplication in the training set of BAG, making it poor in diversity and further leading to lower test performance, compared to Multi-Splits. This indicates diversity of training sets is crucial when constructing few-shot data splits.

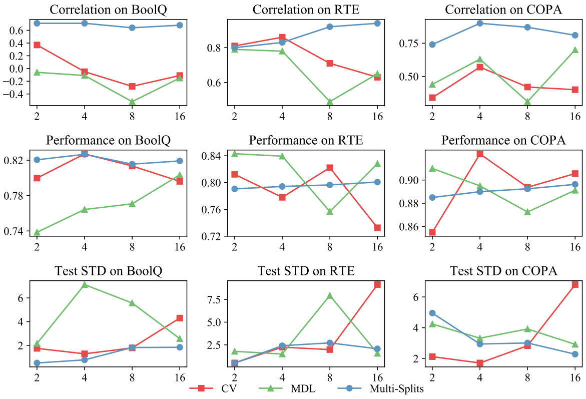

Figure 1: Test performance, correlation and standard deviation along with different K on BoolQ, RTE, and COPA tasks under different strategies. A smooth and stable dot-line indicates the setting is insensitive to the choice of K.

3. RAND uses similar-sized dev and train sets to BAG and MS but performs worse in test performance. Since there could be overlap between train and dev sets, the model may have memorized data, leading to poor test performance.
4. MI constructs very different train and dev sets. Overfitting on one of them and validating on the other pose more challenges for the few-shot method on out-of-distribution tasks.

Stability w.r.t. the number of runs K. Figure 1 shows the results on stability. In light of limited computation resources, we only experiment with

[8] In the following explanation, the numbers refer to the total training/development data covering K=4 runs.

some representative strategies. Both CV and MDL represent strategies whose number of runs are coupled with the size of data split, while Multi-Splits represents strategies that have a fixed ratio and independent K. We observe: (1) Multi-Splits (blue lines) is the most stable in correlation and performance, while other strategies CV and MDL are more sensitive to the choice of K. (2) Multi-Splits shows the smallest variance over multiple runs on both BoolQ and RTE. For COPA, though Multi-Splits shows high variance when $K = 2$, the variance becomes smaller with larger K, while CV and MDL suffer from increasing or unstable variance.

A possible explanation is that increasing K does not affect the number of training and development examples for Multi-Splits; instead, it increases the confidence of results. An important practical benefit of Multi-Splits is that one can always choose to increase K for lower variance. However, for CV and MDL, the sizes of training and development sets are affected by K, where extremely large K value leads to a failure mode and extremely small K leads to unstable results. In practice, it is hard to know which value of K to use a priori.

To sum up, based on the aforementioned results and analysis, we arrive at the following finding.

Finding 1. Our proposed Multi-Splits is a more reliable data-split strategy than several baselines with improvements in (1) test performance, (2) correlation between development and test sets, and (3) stability w.r.t. number of runs.

Remark Our evaluation framework is better in terms of test performance, dev-test correlation, and stability, which proves it can achieve possible peak performance, reliably select the corresponding hyperparameters according to dev results without overfitting, and mitigate the effects of randomness to the maximum extent. Therefore, the estimation of our evaluation framework for model performance is more reliable than previous evaluations.

5 Re-Evaluation of State-of-the-Art Methods

5.1 Few-Shot Methods

We now proceed to re-evaluate state-of-the-art few-shot methods under our evaluation framework with the Multi-Splits strategy. We consider two types: *minimal few-shot methods*, which only assume access to a small labeled dataset, including Classification (CLS; Devlin et al., 2018), PET (Schick and Schütze, 2021b), ADAPET (Menon et al., 2021), P-tuning (Liu et al., 2021b) and FlipDA (Zhou et al., 2021); and *semi-supervised few-shot methods*, which allow accessing an additional unlabeled dataset, including PET+MLM (Schick and Schütze, 2021a), iPET (Schick and Schütze, 2021b) and Noisy Student (Xie et al., 2020).

5.2 Experimental Setup

The same benchmark datasets, metrics, and hyperparameter space as in §4.2.3 are used. We use 32 labeled samples for training. We consider two labeling strategies to obtain the pseudo-labels on unlabeled samples used by the semi-supervised methods for self-training, including *single-split labeling* and *cross-split labeling*. In the single-split setting (Schick and Schütze, 2021b), pseudo-labels are generated by the models trained on the same data split. In the cross-split setting in our evaluation framework, the pseudo-labels are generated by the models trained on multiple different data splits. More configuration details are in Appendix A.4.

5.3 Main Results and Analysis

Re-Evaluation Results Table 4 shows our re-evaluation results. The prompt-based fine-tuning paradigm significantly outperforms the classification fine-tuning on all tasks and on both pretrained models (with an advantage of more than 15 points on average). DeBERTa outperforms ALBERT consistently. We observe significant differences in performance between different prompt-based minimal few-shot methods with ALBERT (e.g., ADAPET and FlipDA outperform PET respectively by about 4 points and 2 points on average) while differences with DeBERTa are slight (e.g., PET, ADAPET, P-tuning, and FlipDA have a performance gap of only about 1.0 points on average). In contrast, semi-supervised few-shot methods (i.e., iPET and Noisy) generally improve 1–2 points on average compared to minimal few-shot methods on both models.

Comparison to Prior Evaluations Since we have proved that our evaluation framework is more reliable in estimating method performance as shown in Section 4.2.4, we conduct experiments to compare the estimates by our evaluation framework and prior evaluations to study whether model performance was accurately estimated in prior work.

Table 6 lists the absolute performance from prior evaluations and our evaluation. Results show the absolute performance of few-shot methods in prior evaluations was generally overestimated on RTE

Base Models	Few-Shot Methods	BoolQ	RTE	WiC	CB		MultiRC		WSC	COPA	Avg.
		Acc.	Acc.	Acc.	Acc.	F1	F1a	EM.	Acc.	Acc	
ALBERT	CLS	55.01 ±2.95	53.97 ±5.49	50.82 ±3.02	67.97 ±18.29	52.18 ±10.30	59.95 ±10.69	18.86 ±9.80	52.64 ±10.25	64.25 ±9.36	53.74
	PET	76.70 ±1.85	72.83 ±1.30	53.87 ±4.47	84.38 ±4.47	62.56 ±7.66	76.51 ±1.52	36.46 ±2.13	80.05 ±2.53	81.75 ±4.03	70.74
	ADAPET	79.24 ±1.42	74.28 ±3.57	58.07 ±2.96	92.86 ±1.46	89.99 ±3.91	77.24 ±1.99	37.17 ±2.64	78.85 ±4.51	81.75 ±3.95	74.40
	P-tuning	76.55 ±2.68	63.27 ±3.63	55.49 ±1.21	88.39 ±3.72	84.24 ±5.15	75.91 ±1.74	38.01 ±0.78	78.85 ±1.76	85.25 ±3.30	71.81
	FlipDA	77.95 ±2.60	70.85 ±2.71	57.17 ±2.59	83.93 ±4.37	74.30 ±13.23	76.05 ±1.33	35.68 ±1.44	79.57 ±1.82	87.50 ±3.70	72.57
	PET+MLM[3]	76.83 ±1.18	71.48 ±1.64	52.39 ±1.44	83.93 ±5.05	67.37 ±8.31	75.15 ±0.34	35.68 ±1.10	81.97 ±1.82	85.75 ±3.40	71.36
	iPET(single)[3,4]	74.29 ±4.10	72.35 ±3.71	54.78 ±3.93	84.67 ±3.18	76.92 ±5.44	76.33 ±1.18	37.72 ±2.58	77.80 ±2.79	84.00 ±6.02	71.58
	Noisy(single)[3,4]	76.11 ±2.16	72.62 ±2.80	54.11 ±1.98	84.38 ±5.60	72.57 ±11.84	76.59 ±1.40	37.00 ±2.34	79.17 ±3.31	83.50 ±3.34	71.54
	iPET(cross)[3,4]	76.83 ±1.39	74.28 ±4.31	58.35 ±2.42	83.48 ±2.68	73.86 ±2.48	75.71 ±2.14	37.30 ±2.71	76.44 ±2.78	83.25 ±4.19	72.05
	Noisy(cross)[3,4]	75.64 ±1.82	75.27 ±1.97	56.43 ±2.67	84.82 ±4.49	77.79 ±8.46	77.11 ±1.49	38.25 ±0.92	80.53 ±7.17	83.00 ±4.76	72.84
DeBERTa	CLS	59.49 ±1.74	49.55 ±2.23	54.08 ±2.15	68.30 ±3.96	60.10 ±10.14	75.42 ±2.39	34.23 ±5.02	53.13 ±5.17	85.25 ±2.22	60.07
	PET	82.67 ±0.78	79.42 ±2.41	67.20 ±1.34	91.96 ±3.72	88.63 ±4.91	78.20 ±1.86	42.42 ±3.04	84.13 ±4.87	89.00 ±2.94	79.00
	ADAPET	81.28 ±1.26	82.58 ±2.44	66.50 ±2.11	89.73 ±6.08	86.63 ±7.29	77.88 ±2.55	43.05 ±3.60	85.34 ±2.13	88.75 ±4.43	79.01
	P-tuning	82.25 ±0.85	82.22 ±1.23	66.22 ±1.18	94.20 ±2.25	91.76 ±3.30	78.45 ±1.46	43.78 ±3.93	85.10 ±4.87	86.50 ±3.70	79.48
	FlipDA	83.52 ±0.35	80.14 ±1.93	65.28 ±1.56	95.09 ±2.68	93.57 ±2.62	80.21 ±1.35	46.67 ±0.82	85.34 ±3.27	90.50 ±1.00	80.37
	PET+MLM[3]	82.80 ±0.97	83.30 ±2.40	58.23 ±4.98	90.18 ±3.09	87.18 ±6.17	77.05 ±1.80	40.63 ±1.64	81.73 ±5.77	85.75 ±3.40	77.05
	iPET(single)[3,4]	81.27 ±1.61	81.11 ±1.89	64.75 ±4.27	89.88 ±5.01	87.70 ±6.52	79.99 ±1.94	45.23 ±2.19	82.93 ±3.76	90.83 ±2.79	78.90
	Noisy(single)[3,4]	81.60 ±1.54	81.95 ±2.01	65.97 ±2.44	91.67 ±2.33	89.17 ±2.95	79.85 ±1.22	45.10 ±2.58	84.46 ±2.49	90.67 ±2.53	79.65
	iPET(cross)[3,4]	83.45 ±0.90	83.12 ±1.04	69.63 ±2.15	91.52 ±3.05	90.72 ±2.68	79.92 ±1.11	44.96 ±3.13	86.30 ±1.64	93.75 ±2.99	81.40
	Noisy(cross)[3,4]	82.19 ±0.65	81.95 ±0.51	68.26 ±1.12	90.18 ±2.31	86.74 ±3.00	79.48 ±2.53	44.20 ±4.14	83.41 ±4.18	93.75 ±3.30	79.98
DeBERTa	Our Best[3,4] (few-shot)	**84.0** ±0.55	**85.7** ±0.63	**69.6** ±2.15	**95.1** ±2.68	**93.6** ±2.62	**81.5** ±0.76	**48.0** ±0.99	**88.4** ±2.82	**93.8** ±2.99	**85.44**[1]
RoBERTa	RoBERTa [5] (fully sup.)	86.9	86.6	75.6	98.2	-	85.7	-	91.3	94.0	88.33
DeBERTa	DeBERTa [2] (fully sup.)	88.3	93.5	-	-	-	87.8	63.6	-	97.0	-

[1] For comparison with RoBERTa (fully sup.), the average of Our Best (few-shot) 85.17 excludes MultiRC-EM and CB-F1.

[2] The fully-supervised results on DeBERTa are reported in `https://github.com/THUDM/GLM`.

[3] Unlabeled data are used.

[4] The ensemble technique is used.

[5] The RoBERTa (fully-sup.) results by (Liu et al., 2019). RoBERTa-large has less parameters than DeBERTa-xxlarge-v2.

Table 4: Re-evaluation of few-shot methods on ALBERT and DeBERTa under our evaluation framework with Multi-Splits strategy on test set of our setup. For iPET and Noisy Student, (cross) and (single) respectively means cross-split labeling and single-split labeling strategies as introduced in §5.2. "Our Best (few-shot)" is the results achieved by a combination method as introduced in §5.4. **Globally best results** for each task are in bold. Best results for minimal few-shot methods are underlined. Best results for semi-supervised few-shot methods are marked with wavelines.

	BoolQ	RTE	WiC	CB	MultiRC	WSC	COPA
Minimal Few-Shot Methods	PET	ADAPET	PET	FlipDA	ADAPET	ADAPET	PET
Training Paradigm	iPET(cross)	Noisy(cross)	iPET(cross)	single	Noisy(cross)	Noisy(single)	iPET(cross)
+ MLM	✓	-	-	-	-	-	-

Table 5: The combination of methods that achieves the best few-shot performance for each task. There are five minimal few-shot methods and five training paradigms as combined options, as §5.4 illustrates. "+MLM" means adding an additional MLM loss.

Methods	RTE		WiC		COPA	
	Prev.	Ours	Prev.	Ours	Prev.	Ours
PET	69.80	72.83	52.40	53.87	95.00	81.75
ADAPET	76.50	74.28	54.40	58.07	89.00	81.75
P-tuning	76.50	63.27	56.30	55.49	87.00	85.25
FlipDA	70.67	70.85	54.08	57.17	89.17	87.50
+MLM	62.20	71.48	51.30	52.39	86.70	85.75
iPET	74.00	72.35	52.20	54.78	95.00	84.00

Table 6: Comparison of prior evaluations and our evaluation. We report the absolute performance of different methods respectively from previous evaluation (Prev.) and our evaluation framework (Ours.) on RTE, WiC and COPA tasks. The results are based on ALBERT. Results of previous evaluation are taken from the original papers, including ADAPET (Menon et al., 2021), P-tuning (Liu et al., 2021b), FlipDA (Zhou et al., 2021) and iPET (Schick and Schütze, 2021b). Since (Schick and Schütze, 2021a) reported results of PET+MLM on different tasks, we re-experimented on the same tasks under the same setting as (Schick and Schütze, 2021a). Wave lines and underlines indicate examples of inaccurate estimates of relative gaps in prior works (see text for details).

and COPA. Similar findings have been highlighted in prior works (Perez et al., 2021; Zhang et al., 2020), and our evaluation framework confirms the findings under a more reliable setup. This results from a more reliable evaluation procedure that emphasizes dev-test correlation to prevent overfitting (discussed in Section 4.2).

Besides, the relative gaps between different methods were not accurately estimated by the prior reported numbers. For example, according to the reported results in prior works, ADAPET outperforms P-Tuning on COPA and P-Tuning beats ADAPET on WiC, while our evaluation reveals the opposite. On one hand, this is because prior results were obtained under a less reliable evaluation procedure (discussed in Section 4.2). Deviation in the estimates of absolute performance contributes to inaccuracy in the estimates of relative performance. On the other, prior experiments were not conducted under a shared evaluation procedure. These two factors are corrected by our re-evaluation under the more reliable proposed framework.

To sum up, our re-evaluation compares all methods on a common ground, revealing the following:

Finding 2. The absolute performance and the relative gap of few-shot methods were in general not accurately estimated in prior literature. This is corrected by our new evaluation framework with improved reliability. It highlights the importance of evaluation for obtaining reliable conclusions. Moreover, the benefits of some few-shot methods (e.g., ADAPET) decrease on larger pretrained models like DeBERTa.

5.4 What is the Best Performance Few-Shot Learning can Achieve?

We further explore the best few-shot performance by combining various methods, and evaluating under our evaluation framework. For combined options, we consider five minimal few-shot methods (i.e., CLS, PET, ADAPET, P-tuning, and FlipDA), five training paradigms (i.e., single-run, iPET (single/cross), and Noisy Student (single/cross)), and the addition of a regularized loss (+MLM). We experiment with all possible combinations and report the best for each task.

"Best (few-shot)" in Table 4 achieves the best results on all tasks among all methods. Existing few-shot methods can be practically used in combination. Compared to RoBERTa (fully-sup) (Liu et al., 2019), the performance gap has been further narrowed to 2.89 points on average.[9] Compared to DeBERTa (fully-sup), there is still a sizeable gap between few-shot and fully-supervised systems.

We list the best-performing combination for each task in Table 5. The best combinations are very different across tasks, and there is no single method that dominates most tasks. PET and ADAPET as well as iPET and Noisy Student are about equally preferred while cross-split labeling and no regularization term perform better. We thus recommend future work to focus on the development of methods that achieve consistent and robust performance across tasks. We summarize the following findings:

Finding 3. Gains of different methods are largely complementary. A combination of methods largely outperforms individual methods, performing close to a strong fully-supervised baseline on RoBERTa. However, there is still a sizeable gap between the best few-shot and the fully-supervised system.

Finding 4. No single few-shot method dominates most NLU tasks. This highlights the need for the development of few-shot methods with more consistent and robust performance across tasks.

6 FewNLU Toolkit

We open-source FewNLU, an integrated toolkit designed for few-shot NLU. It contains implemen-

[9] Note that the gap could be larger since RoBERTa-Large has a smaller number of parameters than DeBERTa, and RoBERTa (fully-sup) does not incorporate additional beneficial techniques such as ensembling or self-training.

tations of state-of-the-art methods, data processing utilities, a standardized few-shot training framework, and most importantly, our proposed evaluation framework. Figure 2 shows the architecture. We hope FewNLU could facilitate benchmarking few-shot learning methods for NLU tasks and expendit the research in this field.

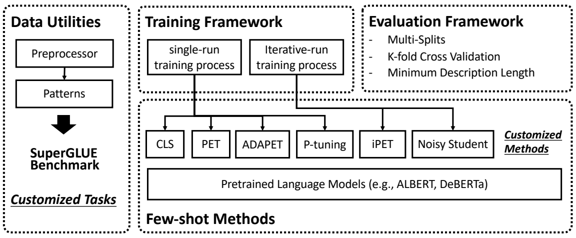

Figure 2: Architecture of FewNLU.

7 Conclusions

We introduce an evaluation framework, re-evaluate a number of few-shot learning methods under the evaluation framework with a novel Multi-Splits strategy, and release a few-shot toolkit. Apart from this, we also aim at advancing the development of few-shot learning by sharing several new experimental findings. We identify several new directions for future work: (1) In practice, how to define the hyper-parameter search space a priori is a challenge. (2) It is critical for the community to iterate and converge on a common evaluation framework. (3) Few-shot natural language generation might also be studied in a similar framework.

Acknowledgements

We thank Dani Yogatama for valuable feedback on a draft of this paper. Tang is funded by NSFC for Distinguished Young Scholar (61825602). Zheng, Ding, Tang, and Yang are funded by the National Key R&D Program of China (2020AAA0105200) and supported by Beijing Academy of Artificial Intelligence (BAAI). Zheng is Funded by China Postdoctoral Science Foundation (2021M690471). Zhou and Li are supported in part by the National Natural Science Foundation of China Grant 62161146004, Turing AI Institute of Nanjing and Xi'an Institute for Interdisciplinary Information Core Technology.

References

Jonathan Bragg, Arman Cohan, Kyle Lo, and Iz Beltagy. 2021. FLEX: unifying evaluation for few-shot NLP. CoRR, abs/2107.07170.

Leo Breiman. 1996. Bagging predictors. Mach. Learn., 24(2):123–140.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. CoRR, abs/2005.14165.

Alain Celisse. 2014. Optimal cross-validation in density estimation with the l^2-loss. The Annals of Statistics, 42(5):1879–1910.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 2924–2936.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In Machine Learning Challenges Workshop, pages 177–190. Springer.

Marie-Catherine De Marneffe, Mandy Simons, and Judith Tonhauser. 2019. The commitmentbank: Investigating projection in naturally occurring discourse. In proceedings of Sinn und Bedeutung, volume 23, pages 107–124.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. CoRR, abs/1810.04805.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah A. Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. CoRR, abs/2002.06305.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2020. Making pre-trained language models better few-shot learners. CoRR, abs/2012.15723.

Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. In Proceedings of ACL 2018.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In Proceedings of the 2018 Conference of the North American Chapter of the Association for

Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 252–262.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. CoRR, abs/2104.08691.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning. Citeseer.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. CoRR, abs/2101.00190.

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

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021b. GPT understands, too. CoRR, abs/2103.10385.

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

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2021. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. CoRR, abs/2104.08786.

Rakesh R. Menon, Mohit Bansal, Shashank Srivastava, and Colin Raffel. 2021. Improving and simplifying pattern exploiting training. CoRR, abs/2103.11955.

Subhabrata Mukherjee, Xiaodong Liu, Guoqing Zheng, Saghar Hosseini, Hao Cheng, Greg Yang, Christopher Meek, Ahmed Hassan Awadallah, and Jianfeng Gao. 2021. CLUES: few-shot learning evaluation in natural language understanding. CoRR, abs/2111.02570.

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. 2021. True few-shot learning with language models. CoRR, abs/2105.11447.

Mohammad Taher Pilehvar and José Camacho-Collados. 2018. Wic: 10, 000 example pairs for evaluating context-sensitive representations. CoRR, abs/1808.09121.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning, pages 90–95.

Timo Schick and Hinrich Schütze. 2021a. Exploiting cloze-questions for few-shot text classification and natural language inference. In EACL, pages 255–269. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021b. It's not just size that matters: Small language models are also few-shot learners. pages 2339–2352.

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

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical bayesian optimization of machine learning algorithms. Advances in neural information processing systems, 25.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019a. Superglue: A stickier benchmark for general-purpose language understanding systems. In NeurIPS.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019b. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In ICLR (Poster). OpenReview.net.

Qizhe Xie, Minh-Thang Luong, Eduard H. Hovy, and Quoc V. Le. 2020. Self-training with noisy student improves imagenet classification. In CVPR, pages 10684–10695. IEEE.

Shuo Yang, Lu Liu, and Min Xu. 2021. Free lunch for few-shot learning: Distribution calibration. In ICLR. OpenReview.net.

Qinyuan Ye, Bill Yuchen Lin, and Xiang Ren. 2021. Crossfit: A few-shot learning challenge for cross-task generalization in nlp.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q. Weinberger, and Yoav Artzi. 2020. Revisiting few-sample BERT fine-tuning. CoRR, abs/2006.05987.

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

Jing Zhou, Yanan Zheng, Jie Tang, Jian Li, and Zhilin Yang. 2021. Flipda: Effective and robust data augmentation for few-shot learning.

A Appendix

A.1 Fixed Hyper-Parameters are not Optimal

Some prior works (Schick and Schütze, 2021a,b; Menon et al., 2021) perform few-shot learning with a fixed set of hyper-parameters (determined by practical considerations and experiences) without early stopping and any model selection.

	Hyper-Parameters				Test Acc.	Avg.
	P	LR	Step	WR		
Fixed	0	1e-5	250	0	69.31 ±4.39	67.36
	1				61.13 ±0.91	
	2				63.06 ±1.50	
	3				63.06 ±1.82	
	4				80.26 ±1.85	
Optimal	0	1e-5	300	0.05	72.44 ±1.85	70.42
	1	5e-6	300	0.05	63.78 ±1.37	
	2	5e-6	300	0	69.07 ±5.55	
	3	5e-6	300	0	65.70 ±1.25	
	4	5e-6	300	0	81.11 ±1.37	

Table 7: Performance of PET on RTE task with different hyper-parameters. The patterns and fixed hyper-parameters are reported by (Schick and Schütze, 2021b). Base model: DeBERTa-xxLarge, "P": pattern ID, "LR": learning rate, "Step": number of training steps, "WR": warmup ratio.

We first study how well fixed hyper-parameters transfer to a new scenario, e.g. switching to another base pretrained model. We perform preliminary experiments on FewGLUE with 64 labeled sample based on DeBERTa. Firstly, we experiment with the fixed hyper-parameters used for ALBERT in (Schick and Schütze, 2021b). Secondly, we manually try other hyper-parameters to find out whether there are better configurations. From Table 7, we observe:

1. Certain factors, especially the patterns, impact the performance a lot (best 80.26%, and worst 61.13%). However, we cannot differentiate between them without a development set.
2. There exists a hyper-parameter ("Optimal" in Table 7) that performs much better than the fixed one. A mechanism to identify the best hyper-parameter setting is thus necessary.
3. Results show a good hyper-parameter on ALBERT does not work well on DeBERTa. Fixed hyper-parameters are not optimal and we need to re-select them given new conditions.

A.2 Details of How to Construct Data Splits

A.2.1 Datasets

To justify the proposed evaluation framework, we perform experiments on the few-shot SuperGLUE benchmark, which was constructed to include some of the most difficult language understanding tasks for current NLP approaches (Wang et al., 2019a). Unlike other NLU benchmarks (e.g., GLUE (Wang et al., 2019b)) that contain single-sentence tasks, SuperGLUE consists of complicated ones that are sentence-pair or sentence-triple tasks, which demand advanced understanding capabilities. Seven SuperGLUE tasks are considered, including question answering (BoolQ (Clark et al., 2019) & MultiRC (Khashabi et al., 2018)), textual entailment (CB (De Marneffe et al., 2019) & RTE (Dagan et al., 2005)), word sense disambiguation (WiC (Pilehvar and Camacho-Collados, 2018)), causal reasoning (COPA (Roemmele et al., 2011)), and co-reference resolution (WSC (Levesque et al., 2012)).

A.2.2 Hyper-parameters

To quantitatively evaluate different data-split strategies, we perform extensive experiments with the following hyper-parameter search space. Data-split experiments are based on DeBERTa-xxLarge. The hyper-parameter search space is shown in Table 8. We use the same prompt patterns as in (Schick and Schütze, 2021b). To observe the changes of performance and correlation metrics w.r.t different K values, we also experimented with $K = \{2, 4, 8, 16\}$ over three tasks (i.e., BoolQ, RTE and COPA).

Hyper-parameter	Value
Learning Rate	$\{5e-6, 1e-5\}$
Maximum Training Step	$\{250, 500\}$
Evaluation Frequency	$\{0.02, 0.04\}$
Number of Runs K	4
Split Ratio r for Multi-Splits	1:1

Table 8: Hyper-parameter Search Space for Data-Split Strategy Evaluation

A.2.3 Evaluation Results with 32 Labeled Data

In the data-split strategy evaluation, in addition to the 64-data-setting results in the main text, we also experimented with 32 labeled data as (Schick and Schütze, 2021b,a; Menon et al., 2021). The 32-data-setting results are also provided in Table 10.

A.2.4 Leave-One-Out Cross Validation Results

We also experiment with another useful data split strategy, leave-one-out cross validation (LOOCV). In fact, LOOCV is a special case of K-fold cross validation when K equals the number of labeled data. Since LOOCV takes even longer time than any other data split strategies, we only experi-

		BoolQ	RTE	WiC
Multi-Splits	Perf.	82.67 ±0.78	79.42 ±2.41	67.20 ±1.34
	Corr.	0.7079	0.8266	0.9464
CV	Perf.	82.71 ±1.29	77.80 ±2.25	64.42 ±1.63
	Corr.	-0.0497	0.8561	0.8184
LOOCV	Perf.	80.20 ±5.63	63.91 ±5.37	62.40 ±4.70
	Corr.	-0.8001	-0.5070	0.1998

Table 9: Test performance and correlation results of leave-one-out cross validation on BoolQ, RTE and WiC tasks with 64 labeled examples.

mented on three tasks, including BoolQ, RTE and WiC tasks. Both performance and correlation results are shown in Table 9. Our results show that compared to other strategies, LOOCV achieved worse test performance as well as correlation. LOOCV only uses a single instance for validation each time, and thus leads to poor correlation and random model selection. As a result, the performance estimation is subject to much randomness.

A.3 How to Define the Hyper-parameter Search Space

Aside from how to construct the data splits, another important question for the evaluation framework is how to define the hyper-parameter search space. We left this question in the future work. However, we did several preliminary experiments that could reveal certain insights into the problem.

A.3.1 Should We Search Random Seeds?

We focus on two types of factors that affect few-shot evaluation, hyper-parameters and randomness. Randomness could cause different weight initialization, data splits, and data order during training. Empirically, how randomness is dealt with differs depending on the use case. In order to obtain the best possible performance, one could search over sensitive random factors such as random seeds. However, as we focus on benchmarking few-shot NLU methods, we report mean results (along with the standard deviation) in our experiments in order to rule out the effects of randomness and reflect the average performance of a method for fair comparison and measurement.

A.3.2 Experiments

Experimental Setup To examine how a certain factor affects few-shot performance, we assign multiple different values to a target factor while fixing other hyper-parameters. We report the standard deviation over the multiple results. Larger values indicate that a perturbation of the target factor would largely influence the few-shot performance and the factor thus is crucial for searching. We experiment on BoolQ, RTE, CB, and COPA tasks. Considered factors include: sample order during training, prompt pattern, training batch size, learning rate, evaluation frequency, and maximum train steps.

Results and Analysis Results are in Table 11. We mark values larger than a threshold of 2.0 in bold. We can see that the prompt pattern is the most influential factor among all, indicating the design or selection of prompt patterns is crucial. Training example order also significantly affects the performance. The evaluation frequency affects the score on the small development but not on the test set. We speculate that a lower frequency selects a model with better performance on the small development set, but the gains do not transfer to the test set because of partial overfitting. To conclude:

Finding 5. We recommend to at least search over prompt patterns during hyper-parameter tuning, and it is also beneficial to search others. All comparison methods should be searched and compared under the same set of hyper-parameters.

A.3.3 Detailed Configuration

For a given task and a target factor, we fixed the hyper-parameters to be the best-performing ones obtained in Section 4.2, and assigned multiple values for the target factor. For the prompt pattern, we assigned it with the same values as (Schick and Schütze, 2021b). Possible values for other hyper-parameters are in Table 12.

A.4 Details of Re-Evaluation

A.4.1 Methods

The five considered minimal few-shot methods are introduced as follows.

1. **Classification** is a conventional finetuning algorithm, which uses the hidden states of a special [CLS] token for classification.
2. **PET** is a prompt-based finetuning algorithm. It transforms NLU problems into cloze problems with prompts, and then converts the cloze outputs into the predicted class.
3. **ADAPET** is based on PET and decouples the losses for the label tokens. It proposes a label-conditioned masked language modeling (MLM) objective as a regularization term.

(a) Results of test performance of the selected hyper-parameter.

	BoolQ Acc.	RTE Acc.	WiC Acc.	CB Acc.	 F1	MultiRC F1a	 EM.	WSC Acc.	COPA Acc	Avg.
CV	77.29 ±3.32	75.63 ±4.26	55.56 ±1.06	**89.29** **±3.86**	**80.66** ±14.87	**78.61** **±0.84**	**42.26** ±2.07	**78.37** ±4.26	**90.00** ±2.45	**74.61**
MDL	**79.29** ±6.01	75.87 ±5.19	53.53 **±0.58**	79.61 ±5.42	59.25 ±11.27	75.77 ±4.72	37.30 ±6.27	77.82 ±4.19	76.25 ±12.50	69.82
Multi-Splits	78.11 ±2.63	**79.42** **±1.79**	**61.72** ±3.10	83.04 ±6.66	70.93 ±13.40	78.23 ±1.24	41.45 **±1.74**	74.52 ±3.96	84.75 **±2.12**	73.62

(b) Results of correlation between the development and training sets.

	BoolQ	RTE	WiC	CB	MultiRC	WSC	COPA	Avg.
CV	0.4134	0.6759	0.4189	0.0938	0.1061	-0.1683	0.6567	0.3138
MDL	**0.6394**	0.5687	-0.0732	0.2127	0.1690	0.0741	0.1100	0.2429
Multi-Splits	0.5347	0.6911	**0.8448**	**0.7232**	**0.6280**	**0.0853**	0.4531	**0.5657**

Table 10: Evaluation results of different few-shot data-split strategies with PET on FewGLUE (K=4) under the same data setting as (Schick and Schütze, 2021b,a; Menon et al., 2021) with 32 labeled data. Larger scores indicate that a data-split strategy effectively selects a model that achieves better test-set performance. The best results for each task are denoted in bold.

	Hyper-params	BoolQ	RTE	COPA	CB
Dev Set	Train Order	**3.64**	**4.01**	**2.17**	**2.21/6.09**
	Prompt Pattern	**3.44**	**10.28**	**5.80**	**3.18/4.07**
	Train Batch	**3.34**	1.33	**2.64**	1.01/**5.87**
	Learning Rate	0.00	1.63	1.97	1.56/**4.56**
	Eval Freq	**2.39**	**2.96**	**2.73**	0.45/0.82
Test Set	Train Order	0.87	1.87	**2.17**	**3.01/4.73**
	Prompt Pattern	**2.85**	**10.03**	**2.65**	**6.45/7.08**
	Train Batch	**2.44**	1.09	0.72	0.89/1.32
	Learning Rate	0.17	0.65	0.52	**4.82/7.25**
	Eval Freq	0.84	0.53	1.18	0.77/**2.07**

Table 11: Analysis of different factors on BoolQ, RTE, CB and COPA using PET and DeBERTa. The metric is standard deviation. Hyper-parameters are set the best-performing ones obtained in §5 while the target factor is assigned with multiple values. "Train Order": training sample order; "Train Batch": total train batch size; "Eval Freq": evaluation frequency.

Hyper-parameter	Value
Learning Rate	$\{6e-6, 8e-6, 1e-5\}$
Evaluation Frequency	$\{0.02, 0.04, 0.08\}$
Training Batch Size	$\{8, 16, 32, 64\}$
Sample Order Seed	$\{10, 20, 30, 40, 50, 60, 70, 80\}$

Table 12: Hyper-parameter Search Space for Crucial Factor Evaluation

4. **P-tuning** is also based on PET and automatically learns continuous vectors as prompts via gradient update.
5. **FlipDA** is similar to PET but uses both labeled data and augmented data for training. The augmented data are automatically generated by taking labeled data as inputs. [10]

The three semi-supervised few-shot methods are introduced as follows.

1. **PET+MLM** is based on PET and additionally adds an auxiliary language modeling task performed on unlabeled dataset. It was first proposed by (Schick and Schütze, 2021a) to resolve catastrophic forgetting.
2. **iPET** is a self-training method. It iteratively performs PET for multiple generations. At the end of each generation, unlabeled data are assigned with pseudo-labels by the fully-trained model, and will be used for training along with train data in the next generation.
3. **Noisy Student** is similar to iPET with the difference that Noisy Student injects noises into the input embeddings of the model.

A.4.2 Hyper-parameter Search Space

The hyper-parameter search space for other few-shot methods are shown in Table 17.

A.4.3 The Searched Best Hyper-parameters

We list the searched best hyper-parameter configuration for different tasks and methods in Table 13, Table 14, Table 15, Table 16.

A.4.4 More Discussion on ADAPET

Since it is observed ADAPET shows less improvement on DeBERTa than it has achieved on ALBERT, we further discuss the phenomena by raising the question what other differences it has made. We respectively visualize the few-shot performance distribution over the same hyper-parameter space of PET and ADAPET in Figure 3. We observe that PET is more likely to obtain extremely bad

[10] In our experiments, we use the best checkpoints searched with PET as the classifier for data selection.

[11] As recommended in (Zhou et al., 2021), we fix one mask ratio for each dataset, i.e., 0.3 for BoolQ, MultiRC, and WSC, 0.5 for RTE and CB, and 0.8 for COPA and WiC. We fix one fill-in strategy for each dataset, i.e., "default" for BoolQ, RTE, WiC, CB, and WSC, "rand_iter_10" for MultiRC, and "rand_iter_1" for COPA.

	BoolQ	RTE	WiC	CB	MultiRC	WSC	COPA
Learning Rate	1e-5	5e-6	5e-6	1e-5	1e-5	1e-5	1e-5
Maximum Training Step	250	250	250	250	250	500	500
Evaluation Frequency	0.02	0.02	0.02	0.02	0.04	0.04	0.02
Prompt Pattern	1	5	2	5	1	2	0

Table 13: The best hyper-parameters searched for PET. We search each task with a learning rate of {1e-5,5e-6}, max steps of {250,500}, evaluation frequency ratio of {0.02,0.04}, and all the available prompt patterns. Therefore, each task has $8N$ hyper-parameter combinations, where N is the number of available prompt patterns, i.e., 6 for BoolQ and RTE, 3 for WiC, and 2 for COPA.

	BoolQ	RTE	WiC	CB	MultiRC	WSC	COPA
Learning Rate	1e-5	5e-6	5e-6	1e-5	5e-6	5e-6	5e-6
Maximum Training Step	250	500	500	500	500	250	500
Evaluation Frequency	0.04	0.04	0.02	0.02	0.02	0.04	0.04
Prompt Pattern	1	5	2	5	0	1	0

Table 14: The best hyper-parameters searched for ADAPET. We search each task with a learning rate of {1e-5,5e-6}, max steps of {250,500}, evaluation frequency ratio of {0.02,0.04}, and all the available prompt patterns. Therefore, each task has $8N$ hyper-parameter combinations, where N is the number of available prompt patterns, i.e., 6 for BoolQ and RTE, 3 for WiC, and 2 for COPA.

	BoolQ	RTE	WiC	CB	MultiRC	WSC	COPA
Learning Rate	5e-6	5e-6	5e-6	1e-5	1e-5	5e-6	1e-5
Maximum Training Step	500	250	500	250	500	500	500
Warmup Ratio	0.0	0.0	0.0	0.1	0.1	0.1	0.1
Evaluation Frequency	0.02	0.02	0.02	0.04	0.02	0.02	0.04
Prompt Encoder Type	mlp	lstm	lstm	lstm	lstm	lstm	mlp

Table 15: The best hyper-parameters searched for P-tuning.We search each task with a learning rate of {1e-5,5e-6}, max steps of {250,500}, warmup ratio of {0.0,0.1}, evaluation frequency ratio of {0.02,0.04}, and prompt encoder implemented with {“mlp”, “lstm”}.

	BoolQ	RTE	WiC	CB	MultiRC	WSC	COPA
Learning Rate	5e-6	1e-5	5e-6	1e-5	1e-5	5e-6	1e-5
Maximum Training Step	250	500	250	250	500	250	500
Evaluation Frequency	0.04	0.04	0.04	0.04	0.02	0.02	0.04
Prompt Pattern	0	5	2	5	0	0	0
Generation Method	sample	greedy	sample	greedy	greedy	sample	greedy
Drop Inconsistant Data	-	✓	-	-	✓	-	✓

Table 16: The best hyper-parameters searched for FlipDA. We search three generation methods, try dropping inconsistant data or not. We search each task with a learning rate of {1e-5,5e-6}, max steps of {250,500}, evaluation frequency ratio of {0.02,0.04}, and all the available prompt patterns. Therefore, each task has $8N$ hyper-parameter combinations, where N is the number of available prompt patterns, i.e., 6 for BoolQ and RTE, 3 for WiC, and 2 for COPA.

Method	Hyper-Parameter	Value
CLS	Learning Rate (DeBERTa)	$\{1e-5, 5e-6\}$
	Learning Rate (ALBERT)	$\{1e-5, 2e-5\}$
	Maximum Training Step	$\{2500, 5000\}$
PET/ ADAPET	Learning Rate (DeBERTa)	$\{5e-6, 1e-5\}$
	Learning Rate (ALBERT)	$\{1e-5, 2e-5\}$
	Maximum Training Step	$\{250, 500\}$
	Evaluation Frequency	$\{0.02, 0.04\}$
P-tuning	Learning Rate (DeBERTa)	$\{5e-6, 1e-5\}$
	Learning Rate (ALBERT)	$\{1e-5, 2e-5\}$
	Maximum Training Step	$\{250, 500\}$
	Evaluation Frequency	$\{0.02, 0.04\}$
	Warmup Ratio	$\{0.0, 0.1\}$
	Prompt Encoder Type	{mlp, lstm}
FlipDA	Learning Rate (DeBERTa)	$\{5e-6, 1e-5\}$
	Learning Rate (ALBERT)	$\{1e-5, 2e-5\}$
	Maximum Training Step	$\{250, 500\}$
	Evaluation Frequency	$\{0.02, 0.04\}$
	DA Method	{greedy,sample,beam}
	Drop Inconsistant Data	{yes, no}
	Mask Ratio	Fixed [11]
	Fill-in Strategy	Fixed [9]
iPET/ Noisy	Unlabeled Data Number	500
	Increasing Factor	3.0
	Sample Ratio (single-split)	1.0
	Sample Ratio (cross-split)	2/3
	Dropout Rate for Noisy	0.05

Table 17: Hyper-parameter Space for Re-Evaluation

task	method	g1	g2	g3
WiC	Multi-Patterns	60.11 ±5.64	60.19 ±4.12	59.66 ±4.27
	Best-Pattern	64.21 ±2.58	64.18 ±4.61	63.37 ±6.29
RTE	Multi-Patterns	65.08 ±10.07	69.20 ±7.13	71.46 ±5.59
	Best-Pattern	79.39 ±2.75	81.95 ±1.04	83.12 ±1.42

Table 18: The performance results of iPET on both WiC and RTE at every generation (g1, g2, and g3). Each experiment uses either ensemble over all patterns (Multi-Patterns) or ensemble over the only best pattern (Best-Pattern). This experiment is conducted with 1000 unlabeled data and an increasing factor 5.

results on BoolQ and RTE, while ADAPET shows stable results. It suggests that ADAPET appears to be more robust to the hyper-parameters, and overall achieves good performance regardless of hyper-parameter selection. However, ADAPET is less inclined to produce better peak results. To sum up, we can conclude: Loss regularization (e.g., ADAPET (Menon et al., 2021)) enhances stability w.r.t. hyper-parameters.

A.4.5 More Discussion on Semi-supervised Few-shot Methods

We focus on semi-supervised methods that iteratively augment data (i.e., iPET and Noisy Student), which have demonstrated promising results on both models in Table 4. Several key points for their success are especially discussed.

1. For semi-supervised methods such as iPET and Noisy Student, it is time-consuming when searching over a large hyper-parameter space for each generation. We directly use the searched best hyper-parameters for PET in each generation. From Table 4, we can see that their results show advantages over PET (by more than 1 points). It suggests that the best hyper-parameters can be transferred to such methods, to reduce the cost of time and computational resources. If we search for each generation, results might be even better.
2. Comparing the single-split labeling strategy, the cross-split labeling strategy works better. As the results show, both iPET (cross) and Noisy (cross) outperform iPET (single) and Noisy (single) in most tasks on both models.
3. Another simple and effective technique is our proposed ensemble labeling strategies. (Schick and Schütze, 2021b) utilizes the ensemble results over all patterns to label unlabeled data, since it is hard to select patterns. Under the Multi-Splits strategy, self-training methods can recognize the best pattern, and only ensemble trained models for the best pattern when labeling unlabeled data. Table 18 shows the results of iPET on WiC and RTE tasks, respectively ensemble over multiple patterns or ensemble over the only best pattern. We can see that results of ensemble with the best pattern significantly outperform results of ensemble with all patterns at every generation.

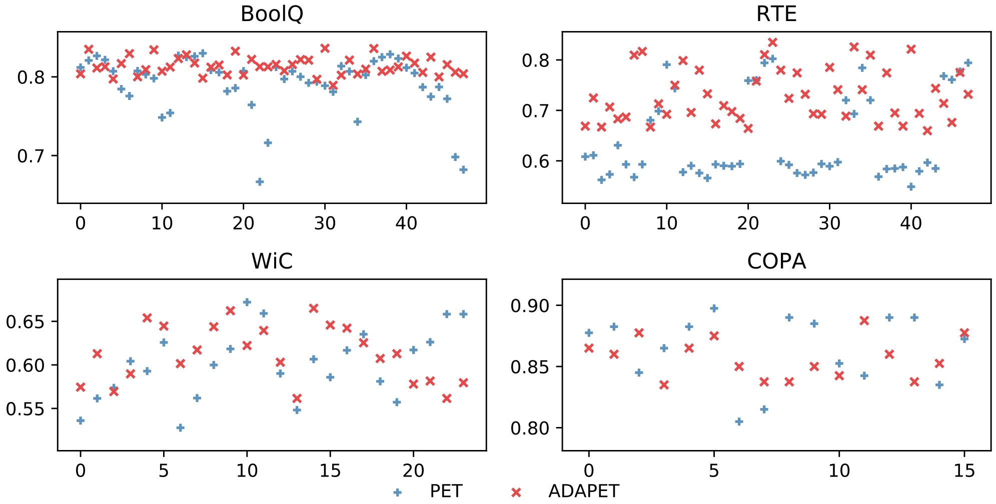

Figure 3: Visualization of few-shot performance over the same hyper-parameter space of ADAPET and PET based on DeBERTa and Multi-Splits. The x-axis is the index of the hyper-parameter combination. We search each task with a learning rate of 1e-5 or 5e-6, max steps of 250 or 500, evaluation ratio of 0.02 or 0.04, and all the available prompt patterns. Therefore, each task has $8N$ hyper-parameter combinations, where N is the number of available prompt patterns, i.e., 6 for BoolQ and RTE, 3 for WiC, and 2 for COPA. The y-axis is the score of each task given a certain hyper-parameter combination.