TransPrompt: Towards an Automatic Transferable Prompting Framework for Few-shot Text Classification

Chengyu Wang[1]*, **Jianing Wang**[2]*, **Minghui Qiu**[1], **Jun Huang**[1], **Ming Gao**[2]†
[1] Alibaba Group
[2] School of Data Science and Engineering, East China Normal University
{chengyu.wcy,minghui.qmh,huangjun.hj}@alibaba-inc.com,
lygwjn@gmail.com,mgao@dase.ecnu.edu.cn

Abstract

Recent studies have shown that prompts improve the performance of large pre-trained language models for few-shot text classification. Yet, it is unclear how the prompting knowledge can be transferred across similar NLP tasks for the purpose of mutual reinforcement. Based on continuous prompt embeddings, we propose *TransPrompt*, a transferable prompting framework for few-shot learning across similar tasks. In *TransPrompt*, we employ a multi-task meta-knowledge acquisition procedure to train a meta-learner that captures cross-task transferable knowledge. Two de-biasing techniques are further designed to make it more task-agnostic and unbiased towards any tasks. After that, the meta-learner can be adapted to target tasks with high accuracy. Extensive experiments show that *TransPrompt* outperforms single-task and cross-task strong baselines over multiple NLP tasks and datasets. We further show that the meta-learner can effectively improve the performance on previously unseen tasks. *TransPrompt* also outperforms strong fine-tuning baselines when learning with full training sets.

1 Introduction

Fine-tuning Pre-trained Language Models (PLMs) has become the standard practice to train models for a majority of NLP tasks (Devlin et al., 2019; Liu et al., 2019b; Qiu et al., 2020). To ensure high accuracy, it is necessary to obtain a sufficient amount of training data for downstream tasks, which is often the bottleneck in low-resource scenarios.

The application of ultra-large PLMs such as GPT-3 (Brown et al., 2020) proves that such PLMs can learn to solve a task with very few training samples. Inspired by these works, Gao et al. (2020) propose a prompt-based approach to fine-tune BERT-style PLMs in a few-shot learning setting, which adapts PLMs into producing specific tokens corresponding to each class, instead of learning the prediction head. The effectiveness of prompts has also been shown in Schick and Schütze (2020); Scao and Rush (2021); Schick and Schütze (2021) and others. However, designing high-performing prompts is challenging and requires a very large validation set. To alleviate this problem, Liu et al. (2021) propose *continuous prompt embeddings* with fully differentiable parameters, avoiding the cumbersome manual prompt engineering process.

Despite the remarkable success, we notice that current prompt-based approaches may have a few limitations. For few-shot learning, the performance of downstream tasks is still constrained by the number of training instances. It would be highly desirable if the model can acquire the transferable knowledge from similar NLP tasks before it is adapted to specific tasks with few samples. However, it is unclear how the knowledge in prompt encoders and PLMs with prompting techniques is transferred across tasks. A natural question arises: *how can we design a prompting framework for BERT-style models that captures transferable knowledge across similar NLP tasks to improve the performance of few-shot learning*?

A straightforward solution to the above question is to adopt multi-task fine-tuning across these similar NLP tasks. When the training data is scarce, the fine-tuned PLM would easily be over-fitted to specific instances (Nakamura and Harada, 2019). In machine learning, the meta-learning paradigm is extensively studied, which produces models that are capable of being adapted to a group of similar tasks quickly with few learning steps (Wang et al., 2020c; Huisman et al., 2020). For PLMs, Wang et al. (2020a) discover that training a meta-learner for PLMs is effective to capture the transferable knowledge across different domains. Yet, this method is not designed for prompts for few-shot learning and lacks the mechanism to learn *unbiased*

* C. Wang and J. Wang contributed equally to this work.
† Corresponding author.

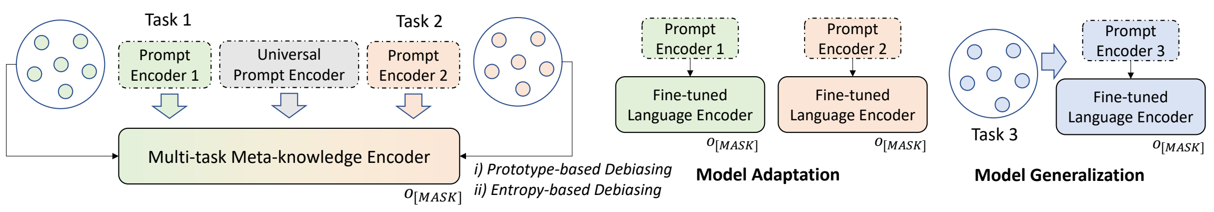

Task 1
Task 2
Prompt Encoder 1
Universal Prompt Encoder
Prompt Encoder 2
Multi-task Meta-knowledge Encoder
$o_{[MASK]}$
i) Prototype-based Debiasing
ii) Entropy-based Debiasing
a) Multi-task Meta-knowledge Acquisition
Prompt Encoder 1
Prompt Encoder 2
Fine-tuned Language Encoder
Fine-tuned Language Encoder
$o_{[MASK]}$
$o_{[MASK]}$
Model Adaptation
Task 3
Prompt Encoder 3
Fine-tuned Language Encoder
$o_{[MASK]}$
Model Generalization
b) Task-aware Model Specification

Figure 1: The high-level architecture of the *TransPrompt* framework. In the toy example, Task 1 and Task 2 are existing tasks, while Task 3 is a new task for the meta-learner to generalize. (Best viewed in color.)

representations for all the tasks.

In this paper, we present *TransPrompt*, a prompting framework that allows PLMs to capture *cross-task transferable knowledge* for few-shot text classification, with the high-level architecture shown in Figure 1. *TransPrompt* firstly employs a *Multi-task Meta-knowledge Acquisition* (MMA) procedure to learn the transferable representations of prompt encoders and PLMs jointly across similar NLP tasks. To reduce *over-fitting* and make the underlying PLM *more task-agnostic* and *less unbiased* towards any specific tasks, we propose two *de-biasing* techniques, namely *prototype-based de-biasing* and *entropy-based de-biasing*. The learned model can be viewed as the *meta-learner* for a group of similar NLP tasks.

After MMA, *TransPrompt* takes the *Task-aware Model Specification* (TMS) step, which can be further divided into two cases. i) When the model is adapted to existing tasks during MMA, a variation of P-tuning (Liu et al., 2021) can be applied for effective adaptation. ii) When it is required to fit a previously unseen task, a *model generalization* strategy is employed, specifically considering the universal prompting knowledge in the model. This is often the case where re-training of the *meta-leaner* across all the tasks is infeasible due to data privacy or computation efficiency issues. [1]

For evaluation, we test the *TransPrompt* framework on three sets of few-shot NLP tasks (including seven public datasets in total): i) sentiment analysis; ii) Natural Language Inference (NLI); and iii) paraphrase. Experimental results show that *TransPrompt* consistently outperforms both single-task and cross-task strong baselines. We further show that i) the meta-learner trained by *TransPrompt* is effective to generalize to unseen tasks; and ii) *TransPrompt* also outperforms popular fine-tuning algorithms when learning with the full training sets. In summary, we make the following major contributions in this work:

- We introduce the novel *TransPrompt* framework to learn cross-task transferable knowledge for few-shot text classification.
- A prompt-based meta-learner training algorithm with two de-biasing techniques is presented to capture transferable knowledge.
- Experiments on multiple types of NLP tasks show that *TransPrompt* consistently outperforms strong baselines for both few-shot learning and standard fine-tuning.

2 Related Work

We summarize the related work on PLMs, PLM prompting, transfer learning and meta-learning.

2.1 Pre-trained Language Models

With large-scale pre-training, PLMs have achieved significant improvements on various NLP tasks (Qiu et al., 2020). BERT (Devlin et al., 2019) learns contextual representations by transformer encoders. Other transformer encoder-based PLMs include Transformer-XL (Dai et al., 2019), XLNet (Yang et al., 2019), StructBERT (Wang et al., 2020b), Big Bird (Zaheer et al., 2020) and many others. The encoder-decoder architectures are used in T5 (Raffel et al., 2020) and GPT-3 (Brown et al., 2020). As the neural architecture design of PLMs is not our major focus, we do not further elaborate.

2.2 Prompt Learning for PLMs

The huge GPT-3 model (Brown et al., 2020) enables few-shot learning without fine-tuning, which

[1]Note that our settings are slightly different from existing works on few-shot learning for PLMs (Gao et al., 2020; Liu et al., 2021) in that we focus on few-shot learning over a series of similar NLP tasks. When tasks during TMS are the same as MMA, our approach can be viewed as a *transfer learning* algorithm that learns how knowledge can be transferred better. When *TransPrompt* is required to fit new tasks during TMS, it is placed in a *meta-learning* setting.

relies on handcraft prompts. To facilitate automatic prompt construction, Gao et al. (2020) generates prompts from the T5 model (Raffel et al., 2020). Jiang et al. (2020) mines prompts from the training corpus. AutoPrompt (Shin et al., 2020) employs token-based gradient searching to detect prompts. However, these approaches focus on discrete prompts only. P-tuning (Liu et al., 2021) is a pioneer work to learn continuous prompt embeddings with differentiable parameters. Our work further extends P-tuning (Liu et al., 2021) that allows PLMs to learn from similar tasks to improve few-shot text classification.

2.3 Transfer Learning and Meta-learning

Transfer learning aims to transfer knowledge or resources from source domains to target domains (Zhuang et al., 2021). For deep neural networks, it is common practice to learn similar tasks by multi-task learning (Liu et al., 2019a). With the popularity of PLMs, fine-tuning has become the standard practice by learning from PLMs for similar tasks (Sun et al., 2019; Arase and Tsujii, 2021). In contrast, meta-learning aims to learn models that can quickly adapt to different tasks with little training data available (Wang et al., 2020c; Huisman et al., 2020), typically formulated as a *K-way N-shot* problem. Meta-learning algorithms have been applied in few-shot NLP tasks, such as text classification (Geng et al., 2020), relation extraction (Gao et al., 2019), question answering (Hua et al., 2020) and knowledge base completion (Sheng et al., 2020). Similar to Wang et al. (2020a); Pan et al. (2021); Wang et al. (2021), the proposed *TransPrompt* framework can be viewed as a combination of transfer learning and meta-learning, which learns transferable knowledge from similar tasks to improve the performance of few-shot text classification, either for existing tasks or new tasks.

3 The *TransPrompt* Framework

We formally present our task and the techniques of the proposed *TransPrompt* framework in detail.

3.1 Overview

We begin with a brief summary of our task. Let $\mathcal{T}_1, \cdots, \mathcal{T}_M$ be M similar few-shot text classification tasks. The m-th task can be formulated as: $\mathcal{T}_m : x \to y$, where x and $y \in \mathcal{Y}$ represent the input text and the classification label, respectively. $\mathcal{Y}$ is the pre-defined label set with $|\mathcal{Y}| = N$, where N is a pre-defined constant. In our setting, we assume that there are K training samples associated with each class $y \in \mathcal{Y}$ in each task $\mathcal{T}_m$. Hence, we have a training set $\mathcal{D}_m$ for each task $\mathcal{T}_m$, each containing $N \times K$ samples. The total number of training instances of M tasks is $N \times K \times M$.[2]

In *TransPrompt*, we train a *meta-learner* $\mathcal{F}_{meta}$ with parameters initialized from any PLMs, based on the M few-shot training sets $\mathcal{D}_1, \cdots, \mathcal{D}_M$. After that, $\mathcal{F}_{meta}$ is adapted to each task $\mathcal{T}_m$ based on its own training set $\mathcal{D}_m$. The task-specific model is denoted as $\mathcal{F}_m$. As $\mathcal{F}_{meta}$ is designed to digest the *transferable knowledge* across tasks, rather than simple multi-task learning, $\mathcal{F}_{meta}$ can also be adapted to previously unseen tasks. Due to the data privacy or computation efficiency issues, when the few-shot training set $\tilde{\mathcal{D}}$ of a similar task $\tilde{\mathcal{T}}$ is not available during the training process of $\mathcal{F}_{meta}$, we explore how *TransPrompt* can be used to generate an accurate model $\tilde{\mathcal{F}}$ based on $\mathcal{F}_{meta}$ and $\tilde{\mathcal{D}}$. In this case, $\mathcal{F}_{meta}$ does not have any knowledge of the new task $\tilde{\mathcal{T}}$ when it is trained during MMA.

In the following, we introduce the detailed techniques of the *TransPrompt* framework, which consists of two major stages, i.e., *Multi-task Meta-knowledge Acquisition* (MMA) and *Task-aware Model Specification* (TMS). Finally, we discuss how to apply *TransPrompt* to standard fine-tuning scenarios where we have relatively large training sets, instead of solving the *N-way K-shot* problem.

3.2 Multi-task Meta-knowledge Acquisition

For clarity, we illustrate the general architecture of the meta-learner for MMA in Figure 2.

3.2.1 Prompt Encoding

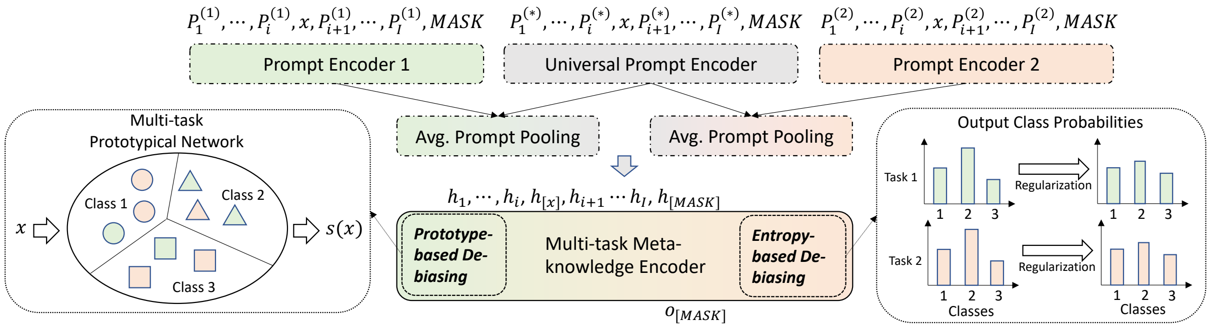
As the *TransPrompt* framework is placed in the multi-task setting, for each task $\mathcal{T}_m$, we have a task-specific prompt template $t^{(m)}(x)$ as follows:

$$P_1^{(m)}, \cdots, P_i^{(m)}, x, P_{i+1}^{(m)}, \cdots, P_I^{(m)}, MASK$$

where $P_i^{(m)}$ is a prompt pseudo token (as proposed in Liu et al. (2021)), I is the total number of pseudo tokens, and $MASK$ is a special token as the placeholder for model output. We also define a universal prompt template $t^{(*)}(x)$ for all the tasks:

$$P_1^{(*)}, \cdots, P_i^{(*)}, x, P_{i+1}^{(*)}, \cdots, P_I^{(*)}, MASK$$

[2]Note that the input x can be either a single sentence or a sentence pair (which has the same setting as that of BERT (Devlin et al., 2019)). For simplicity, we uniformly denote the input as x throughout this paper.



$P_1^{(1)}, \cdots, P_i^{(1)}, x, P_{i+1}^{(1)}, \cdots, P_I^{(1)}, MASK$
$P_1^{(*)}, \cdots, P_i^{(*)}, x, P_{i+1}^{(*)}, \cdots, P_I^{(*)}, MASK$
$P_1^{(2)}, \cdots, P_i^{(2)}, x, P_{i+1}^{(2)}, \cdots, P_I^{(2)}, MASK$
Prompt Encoder 1
Universal Prompt Encoder
Prompt Encoder 2
Avg. Prompt Pooling
Avg. Prompt Pooling
Multi-task Prototypical Network
Class 1
Class 2
Class 3
x
$s(x)$
$h_1, \cdots, h_i, h_{[x]}, h_{i+1} \cdots h_I, h_{[MASK]}$
Prototype-based De-biasing
Multi-task Meta-knowledge Encoder
Entropy-based De-biasing
$o_{[MASK]}$
Output Class Probabilities
Task 1
Task 2
Regularization
Regularization
Classes
Classes

Figure 2: The model architecture of the meta-learner training process during MMA. For simplicity, we assume there are two tasks and three classes for few-shot text classification. (Best viewed in color.)

For an instance $(x, y) \in \mathcal{D}_m$, the prompt embedding $\mathrm{PE}^{(m)}(x)$ can be computed as follows:

$$\mathrm{PE}^{(m)}(x) = \mathrm{AvgPool}\big(\mathrm{MLP}(\mathrm{BiLSTM}(t^{(m)}(x))), \\ \mathrm{MLP}(\mathrm{BiLSTM}(t^{(*)}(x)))\big),$$

where we use bidirectional LSTM networks with multi-layer perceptrons as prompt encoders (Liu et al., 2021). The average pooled results from both task-specific and universal prompt encoders are treated as the prompt embedding. The prompt embedding $\mathrm{PE}^{(m)}(x)$ is a sequence as the input of the PLM:

$$h_1, \cdots, h_i, h_{[x]}, h_{i+1}, \cdots, h_I, h_{[MASK]}$$

where $h_{[x]}$ is the sequence embedding of input x, and $h_{[MASK]}$ is the masked output token embedding. [3] As prompt parameters are fully differentiable, during back propagation, they effectively capture the task-specific and universal knowledge.

3.2.2 Training the Meta-learner

A naive approach for obtaining the meta-learner is applying the P-tuning process (Staudemeyer and Morris, 2019) across the M tasks with the $M + 1$ prompt encoders. However, in practice, it does not guarantee satisfactory results. As large PLMs can easily suffer from over-fitting during few-shot learning (Gao et al., 2020), in cross-task scenarios, the meta-learner would unfortunately memorize the *non-transferable knowledge* from non-target tasks. To alleviate this problem, we propose two de-biasing techniques to obtain a more *unbiased* meta-learner encoded with *transferable knowledge*, namely i) *prototype-based de-biasing* and ii) *entropy-based de-biasing*.

Prototype-based De-biasing. This technique aims to give more importance to *prototypical instances across tasks* during the training process of the meta-learner. Here, we extend Snell et al. (2017) to construct a lite *Multi-task Prototypical Network* $\mathcal{G}$. In the network $\mathcal{G}$, the class centroid embedding $\mathbf{c}_m(y)$ $(y \in \mathcal{Y})$ for each task $\mathcal{T}_m$ is computed and stored as:

$$\mathbf{c}_m(y) = \frac{1}{|\mathcal{D}_{m,y}|} \sum_{(x,y)\in\mathcal{D}_{m,y}} \mathcal{E}(x)$$

where $\mathcal{D}_{m,y}$ is the subset of $\mathcal{D}_m$ such that each instance in $\mathcal{D}_{m,y}$ has the label y, and $\mathcal{E}(x)$ is the representation of x generated by the meta-learner described previously [4]. For each instance $(x, y) \in \mathcal{D}_m$, we pass the text x through the network to generate the *cross-task prototype score*, denoted as $s(x)$:

$$s(x) = \zeta \cdot \frac{\mathrm{sim}(\mathcal{E}(x), \mathbf{c}_m(y))}{\sum_{\tilde{y}\in\mathcal{Y}} \mathrm{sim}(\mathcal{E}(x), \mathbf{c}_m(\tilde{y}))} \\ + \frac{1-\zeta}{M-1} \sum_{\tilde{m}=1(m\neq\tilde{m})}^{M} \frac{\mathrm{sim}(\mathcal{E}(x), \mathbf{c}_{\tilde{m}}(y))}{\sum_{\tilde{y}\in\mathcal{Y}} \mathrm{sim}(\mathcal{E}(x), \mathbf{c}_{\tilde{m}}(\tilde{y}))},$$

where $0 < \zeta < 1$ is a pre-defined balancing factor, and $\mathrm{sim}(\cdot,\ of Knowcdot)$ is the similarity function between two embeddings. We can see that an instance receives a higher score if it is semantically related to the centroids from both the task $\mathcal{T}_m$ itself and other tasks, hence is *more transferable across tasks*. By treating $s(x)$ as the optimization weight, the overall loss function $\mathcal{L}(\Theta)$ of $\mathcal{F}_{meta}$ can be given by:

$$\mathcal{L}(\Theta) = \sum_{m=1}^{M} \sum_{(x,y)\in\mathcal{D}_m} s(x) l(x, y; \Theta) + \lambda_1 \|\Theta\|,$$

[3] Note that the length of $h_{[x]}$ is varied, depending on the tokenization result of x.

[4] In this work, we use the pooled results of the last layer of the PLM encoder as $\mathcal{E}(x)$.

where Θ is the collection of all model parameters, $l(x, y; \Theta)$ is the *sample-wise cross-entropy loss*, and λ_1 is the regularization hyper-parameter.

Entropy-based De-biasing. One potential risk of applying the *prototype-based de-biasing* technique only is obtaining a *non task-agnostic* meta-learner. Consider three tasks $\mathcal{T}_1$, $\mathcal{T}_2$ and $\mathcal{T}_3$. If $\mathcal{T}_1$ and $\mathcal{T}_2$ are highly similar, and $\mathcal{T}_3$ is more dis-similar. Instances in $\mathcal{D}_1$ and $\mathcal{D}_2$ would naturally receive high prototype scores, making the meta-learner *biased* towards $\mathcal{T}_1$ and $\mathcal{T}_2$, and pays little attention to $\mathcal{T}_3$. Hence, when the meta-learner is required to fit $\mathcal{T}_3$, it may have poor parameter initialization settings. To make it *more task-agnostic*, inspired by Jamal and Qi (2019), we consider the model prediction entropy $\mathcal{H}(\mathcal{D}_m)$ over $\mathcal{D}_m$:

$$\mathcal{H}(\mathcal{D}_m) = -\frac{1}{|\mathcal{D}_m|} \sum_{(x,y)\in\mathcal{D}_m} \sum_{\hat{y}\in\mathcal{Y}} \hat{y}(x) \log \hat{y}(x),$$

where $\hat{y}(x)$ is the predicted probability of x being assigned to the class $\hat{y} \in \mathcal{Y}$. When $\mathcal{H}(\mathcal{D}_m)$ is used as a part of the model regularizers, the meta-learner will be *less over-trained* on any specific tasks.

By plugging the term $\mathcal{H}(\mathcal{D}_m)$ into the loss function $\mathcal{L}(\Theta)$, we have the new loss function $\mathcal{L}'(\Theta)$:

$$\begin{aligned}\mathcal{L}'(\Theta) = &\sum_{m=1}^{M} \sum_{(x,y)\in\mathcal{D}_m} (s(x) l(x, y; \Theta) \\ &- \frac{\lambda_2}{|\mathcal{D}_m|} \sum_{\hat{y}\in\mathcal{Y}} \hat{y}(x) \log \hat{y}(x)) + \lambda_1 \|\Theta\|,\end{aligned}$$

where λ_2 is the regularization hyper-parameter.

Optimization Procedure. Despite its simple formula, minimizing $\mathcal{L}'(\Theta)$ is a non-trivial problem. This is because when we calculate $s(x)$, we must obtain model parameters of the PLM beforehand, which is not available before the training process. On the other hand, the optimization of $\mathcal{L}'(\Theta)$ requires the values of $s(x)$ for all training samples, which poses the *"chicken-and-egg"* problem.

We employ a *dual optimization* process to solve the problem of $\mathcal{L}'(\Theta)$. In the initial stage, all $s(x)$s are uniformly initialized. Next, we fix $s(x)$s as constants to minimize $l(x, y; \Theta)$ in $\mathcal{L}'(\Theta)$. An inference procedure on the PLM can be applied to obtain all $s(x)$s. This process iterates for a certain number of epochs. Readers can also refer to Algorithm 1 for an algorithmic overview.

3.3 Task-aware Model Specification

After MMA, the meta-learner can be adapted to specific tasks with ease. For a task $\mathcal{T}_m$ that has already "seen" by the meta-learner, we fine-tune the corresponding prompt encoder and the PLM by minimizing the loss function $\mathcal{L}^{(m)}(\Theta)$:

$$\mathcal{L}^{(m)}(\Theta) = \sum_{(x,y)\in\mathcal{D}_m} l(x, y; \Theta) + \lambda_1 \|\Theta\|,$$

which is a variant of P-tuning (Liu et al., 2021) with better parameter initialization.

For a previously unseen task $\tilde{\mathcal{T}}$, the *model generalization* strategy is employed. Here, we use the universal prompt encoder to initialize its prompt encoder. The entire model is trained over the dataset $\tilde{\mathcal{D}}$, with the loss function $\tilde{\mathcal{L}}(\Theta)$ as follows:

$$\tilde{\mathcal{L}}(\Theta) = \sum_{(x,y)\in\tilde{\mathcal{D}}} l(x, y; \Theta) + \lambda_1 \|\Theta\|,$$

As the meta-learner is highly generalized, it can provide good initialization for the few-shot learning task $\tilde{\mathcal{T}}$.

3.4 Learning with Full Training Sets

TransPrompt can also be applied for standard fine-tuning when we have relatively large training sets with few modifications. During MMA, we notice that when it is not a *N-way K-shot* problem, the sizes of $\mathcal{D}_1, \cdots, \mathcal{D}_M$ can be significantly different. Optimizing $\mathcal{L}'(\Theta)$ directly on these datasets would make the meta-learner *biased towards large datasets*. To address this problem, when we sample a batch from $\mathcal{D}_1, \cdots, \mathcal{D}_M$, instead of randomly selection, we employ stratified sampling where training instances are selected with the probability proportional to the dataset distribution $\Pr(\mathcal{D}_m)$:

$$\Pr(\mathcal{D}_m) = \frac{\log|\mathcal{D}_m| + \gamma}{\sum_{\tilde{m}=1}^{M} \log|\mathcal{D}_{\tilde{m}}| + \gamma},$$

where $\gamma > 0$ is a smoothing factor. This results in the *over-sampling* of small datasets and the *under-sampling* of large datasets.

4 Experiments

In this section, we conduct extensive experiments to evaluate the *TransPrompt* framework and compare it against strong baselines.

4.1 Datasets and Experimental Settings

Following Gao et al. (2020), we select seven public datasets to evaluate *TransPrompt*, divided into three sets of NLP tasks: sentiment analysis (SST-2 (Socher et al., 2013), MR (Hu and Liu, 2004) and

Algorithm 1 Meta-learner Training Algorithm

```
1: for each instance $(x, y) \in \bigcup_{m=1}^{M} \mathcal{D}_m$ do
2:     Uniformly set $s(x) = 1$;
3: end for
4: while number of training epochs does not reach a limit do
5:     while current training epoch is not finished do
6:         Sample a batch $\mathcal{B} = \{(x, y)\}$ from $\bigcup_{m=1}^{M} \mathcal{D}_m$;
7:         Use $\mathcal{B}$ to update $\mathcal{F}_{meta}$ by minimizing $\mathcal{L}'(\Theta)$;
8:     end while
9:     for each instance $(x, y) \in \bigcup_{m=1}^{M} \mathcal{D}_m$ do
10:        Compute $s(x)$ based on the updated model;
11:    end for
12: end while
13: return the meta-learner $\mathcal{F}_{meta}$ (i.e., parameters of the PLM and $M + 1$ prompt encoders).
```

Task Type	Task Name	#Train	#Test
Sentiment	SST-2	6,920	872
	MR	8,662	2,000
	CR	1,775	2,000
NLI	MNLI	392,702	9,815
	SNLI	549,367	9,842
Paraphrase	MRPC	3,668	408
	QQP	363,846	40,431

Table 1: Dataset statistics.

CR (Pang and Lee, 2005)), NLI (MNLI (Williams et al., 2018) and SNLI (Bowman et al., 2015)) and paraphrase (MRPC (Dolan and Brockett, 2005) and QQP[5]).[6] The statistics of these datasets are reported in Table 1. The training/development/testing splits are the same as Gao et al. (2020).

For few-shot learning, the evaluation protocols are the same as Gao et al. (2020). The underlying PLM is the RoBERTa large model (with 335M parameters) (Liu et al., 2019b) and we set $K = 16$. We measure the average performance in terms of accuracy across 5 different randomly sampled training and development splits. Refer to Gao et al. (2020) for more experimental settings. We employ standard BERT fine-tuning (Devlin et al., 2019) [7], the LM-BFF prompting model (Gao et al., 2020) (with both manually-compiled and automatically-mined prompts) [8] and P-tuning (Liu et al., 2021) [9] (which produces state-of-the-art performance for PLM-based few-shot learning) as single-task baselines. Because we focus on learning knowledge across tasks, we also use the multi-task versions of BERT fine-tuning (Sun et al., 2019), LM-BFF (Gao et al., 2020) and P-tuning (Liu et al., 2021), and Meta Fine-tuning (Wang et al., 2020a) [10] as cross-task baselines. Specifically, we employ separate prompts (either discrete prompts or continuous prompt embeddings) for different tasks in the multi-task versions of LM-BFF and P-tuning. As we consider three sets of NLP tasks, we constrain that the knowledge is transferred across the same set of NLP tasks (for example, the cross-task models for sentiment analysis are jointly trained over the training sets of SST-2, MR and CR). Besides, we are interested in how *TransPrompt* can be applied when learning with full training sets. We follow the base-scale experimental settings in Liu et al. (2021), with the RoBERTa base model (with 109M parameters) as the underlying PLM.

For fair comparison, we re-produce all baselines based on their open-source codes under the same settings. Our own *TransPrompt* algorithm is implemented in PyTorch and run with NVIDIA V100 GPUs. In default, we set $\zeta = 0.5$, $\gamma = 0.001$ and $\lambda_2 = 0.01$. The parameter regularizers are the same as in Liu et al. (2021). The model is trained with the Adam optimizer (Kingma and Ba, 2015) and a batch size of 16. The model architecture of prompt encoders is the same as Liu et al. (2021). Therefore, the increased number of parameters of *TransPrompt* remains minimal. We further tune the learning rates and epochs, with results reported in the following experiments.

4.2 General Experimental Results

The results of *TransPrompt* and all baselines on all seven testing sets for few-shot learning are shown in Table 2. From the experimental results, we have the following conclusions. i) Prompting baselines (such as LM-BFF (Gao et al., 2020) and P-tuning (Liu et al., 2021)) outperform standard fine-tuning by a large margin. This shows prompts are useful for few-shot learning. Based on our reproduction results, LM-BFF and P-tuning have similar performance. Automatically-mined prompts are sightly better than manually-compiled prompts for LM-BFF. ii) As for cross-task baselines, the

[5] https://www.quora.com/q/quoradata/.

[6] Note that we do not use other benchmark datasets (such as CoLA) because datasets on similar tasks to CoLA are not available to us. Hence, it is impossible to train the meta-learner in *TransPrompt*.

[7] https://github.com/huggingface/transformers.

[8] https://github.com/princeton-nlp/LM-BFF.

[9] https://github.com/THUDM/P-tuning.

[10] https://github.com/alibaba/EasyTransfer.

Method	Task: Sentiment Analysis			Task: NLI		Task: Paraphrase		Avg.
	SST-2	**MR**	**CR**	**MNLI**	**SNLI**	**MRPC**	**QQP**	
Single-task Baselines								
Fine-tuning (Devlin et al., 2019)	81.42	76.15	84.50	54.17	44.45	73.28	59.64	67.66
LM-BFF (man) (Gao et al., 2020)	90.75	86.60	90.50	63.62	70.77	74.05	60.27	76.65
LM-BFF (auto) (Gao et al., 2020)	91.62	87.25	91.80	64.25	71.21	74.23	60.59	77.28
P-tuning (Liu et al., 2021)	91.85	86.60	91.75	62.41	70.28	66.42	60.57	75.70
Cross-task Baselines								
Fine-tuning (mtl) (Sun et al., 2019)	83.37	79.30	84.75	41.32	48.14	53.12	59.31	64.19
Meta Fine-tuing (Wang et al., 2020a)	86.32	83.85	88.42	48.52	58.20	71.56	67.12	72.00
LM-BFF (mtl) (Gao et al., 2020)*	91.97	87.45	90.70	69.09	75.90	50.00	67.40	76.07
P-tuning (mtl) (Liu et al., 2021)*	93.12	87.75	91.35	68.83	74.24	70.83	69.99	79.44
***TransPrompt* (Proposed Approach)**	**93.58**	**88.80**	**92.00**	**71.90**	**76.99**	**75.98**	**75.80**	**82.15**

Table 2: The few-shot testing results of *TransPrompt* and baselines in terms of accuracy (%). "man", "auto" and "mtl" refer to manually-compiled prompts, automatically-mined prompts and multi-task learning, respectively. * refers to the multi-task variants of the original approaches. Hereinafter the same.

multi-task version of P-tuning is more effective than that of LM-BFF, which shows that continuous prompt embeddings are more suitable for multi-task learning than discrete prompts. iii) The performance gains of *TransPrompt* over all three sets of tasks and seven datasets are consistent. Overall, the average improvement is around 3% in terms of accuracy, compared to the strongest baseline (i.e., the multi-task version of P-tuning). We also conduct *paired t-tests* over the results produced on all tasks. The results show that the improvement of *TransPrompt* is statically significant (with the p-value $p < 0.01$).

4.3 Detailed Model Analysis

In the following, we study how *TransPrompt* improves the performance in various aspects.

Ablation Study. In the *TransPrompt* framework, we propose two de-biasing techniques to improve the effectiveness of the meta-learner, i.e, *prototype-based* and *entropy-based*. Here, we remove each one and all two de-biasing techniques, and implement three variants of *TransPrompt*. The results of the ablation study are in Table 3. As seen, both de-biasing techniques are proved effective for *TransPrompt*. Particularly, *prototype-based de-biasing* plays a slightly more important role than *entropy-based de-biasing* in 6 out of 7 tasks. We conclude that de-biasing the meta-learner is crucial for obtaining the cross-task knowledge.

Parameter Tuning. We further tune the learning epoch and the learning rate during the training process of *TransPrompt*, and report the performance over the development sets. Due to space limitation, we illustrate the results over SST-2, MR and CR, shown in Figure 3. We fix the learning rate to be 1e-5 and tune the learning epochs. Figure 3(a) shows that the performance of the meta-learner becomes stable over 20 epochs (which is tested on the combination of three developments sets for SST-2, MR and CR). Figure 3(b) gives the results of the three tasks during TMS. We fix the learning epoch to be 20, and tune the learning rate from 1e-5 to 5e-4. As seen in Figure 3(c), the learning rate should be set in the range of 1e-5 to 5e-5.

4.4 Model Generalization to New Tasks

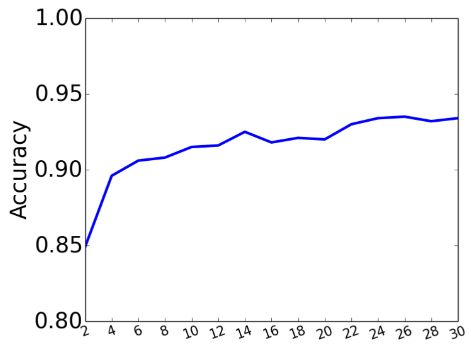
One advantage of *TransPrompt* is that it can train a meta-learner with *cross-task transferable knowledge* encoded. In this set of experiments, we consider three tasks of sentiment analysis: SST-2, MR and CR. Each time, we train the meta-leaner over two out of the three datsets (the MMA step), and then make the model generalized to each of the other tasks (the TMS step). For example, we train the meta-learner over the few-shot SST-2 and MR datasets and then take the TMS step over the few-shot CR dataset. Here, the meta-learner has no knowledge of CR before the TMS step. We test whether the usage of the meta-learner is better than simply applying LM-BFF (Gao et al., 2020) or P-tuning (Liu et al., 2021) initialized from PLMs. From Table 4, it is clearly reflected that the meta-learner brings improvement in all three cases, hence generalizes to new tasks accurately.

4.5 Learning with Full Datasets

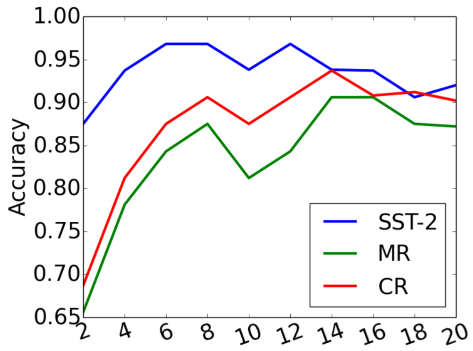
Apart from few-shot learning, we also investigate how *TransPrompt* performs when the full training sets are available, compared to other approaches. The results are presented in Table 5. On average, *TransPrompt* outperforms all single-task baselines by around 1% to 5% in terms of accuracy. This shows that our proposed paradigm can be of help in

Task Name	**w/o. Prototype**	**w/o. Entropy**	**w/o. Both**	**Full Implementation**
SST-2	92.90	**93.18**	92.67	93.58
MR	87.97	**88.14**	87.75	88.80
CR	91.50	**91.70**	91.07	92.00
MNLI	67.72	**69.74**	67.08	71.90
SNLI	**76.76**	76.66	76.08	76.99
MRPC	75.98	**76.72**	68.38	75.98
QQP	73.44	**74.00**	73.02	75.80
Avg.	80.90	**81.45**	79.44	82.14

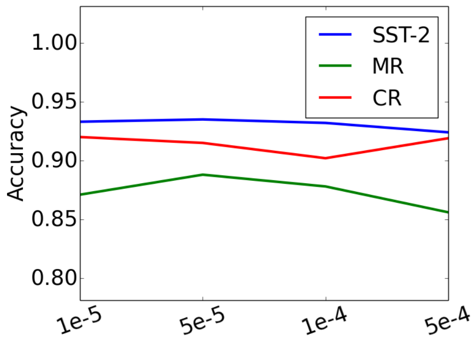
Table 3: Ablation study of *TransPrompt* for few-shot learning. Experimental results are reported on the testing sets in terms of accuracy (%).



(a) Tuning the learning epoch of the meta-learner during MMA.



(b) Tuning the learning epoch of the fine-tuned PLM during TMS.



(c) Tuning the learning rate.

Figure 3: Parameter analysis over the three development sets of SST-2, MR and CR for few-shot learning.

Task	**SST-2**	**MR**	**CR**	**Avg.**
LM-BFF (m)	90.75	86.60	90.50	89.28
LM-BFF (a)	91.62	87.25	91.80	90.22
P-tuning	91.85	86.60	91.75	90.05
TransPrompt	**93.35**	**88.25**	**91.85**	**91.15**

Table 4: Model generalization results in terms of accuracy (%). LM-BFF and P-tuning are strong baselines w/o. the usage of the meta-learner.

non few-shot learning scenarios by learning from a group of similar NLP tasks.

Another interesting finding is that when it comes to multi-task learning, the performance of LM-BFF (Gao et al., 2020) and P-tuning (Liu et al., 2021) drops, compared to the single-task setting. A most possible cause is that with a large amount of training data from other tasks, existing prompt-based approaches may capture non-transferable knowledge that is harmful to the target task. In contrast, the two-step paradigm of *TransPrompt* learns different types of knowledge at different steps (i.e., the universal knowledge in MMA, and the task-specific knowledge in TMS), hence produces better results. Overall, *TransPrompt* is competitive in standard fine-tuning scenarios with datasets from similar NLP tasks available.

4.6 Case Studies

For a more initiative understanding of which instances are *more transferable across tasks*, in Table 6, several review texts from SST-2, MR and CR with high and low prototype scores are presented. Although these texts come from different tasks, our *TransPrompt* algorithm is able to find texts that express general polarities instead of specific points. For instance, “time waster”, “remarkable” and “5 stars” are strong indicators of polarities, which receive high scores generated by *TransPrompt*. In

Method	Task: Sentiment Analysis			Task: NLI		Task: Paraphrase		Avg.
	SST-2	MR	CR	MNLI	SNLI	MRPC	QQP	
Single-task Baselines								
Fine-tuning (Devlin et al., 2019)	93.00	90.15	90.90	82.87	87.87	72.28	89.53	86.65
LM-BFF (man) (Gao et al., 2020)	93.65	88.50	90.98	87.23	91.10	88.75	85.12	89.33
LM-BFF (auto) (Gao et al., 2020)	93.81	88.75	91.25	87.01	91.51	88.97	83.12	89.20
P-tuning (Liu et al., 2021)	93.69	90.10	90.25	87.17	91.67	88.97	90.87	90.38
Cross-task Baselines								
Fine-tuning (mtl) (Sun et al., 2019)	94.72	90.65	91.05	87.10	91.80	69.85	90.20	87.91
Meta Fine-tuing (Wang et al., 2020a)	95.70	91.25	91.42	83.67	89.48	78.92	89.72	88.59
LM-BFF (mtl) (Gao et al., 2020)*	95.41	90.45	91.50	86.76	88.25	69.36	90.32	87.43
P-tuning (mtl) (Liu et al., 2021)*	95.30	90.40	90.08	86.97	91.48	68.87	90.59	87.67
***TransPrompt* (Proposed Approach)**	**96.05**	**91.78**	**91.59**	**88.70**	**91.88**	**86.87**	**91.27**	**91.16**

Table 5: The testing results of *TransPrompt* and baselines with full training sets in terms of accuracy (%).

Score	Task	Review Text	Label
High	SST-2	There are many definitions of "time waster" but this movie must surely be one of them...	NEG
	MR	It is most remarkable not because of its epic scope, but because of the startling intimacy...	POS
	CR	5 stars all the way!	POS
Low	SST-2	It's a treat watching show, a British stage icon, melting under the heat of phocion's attentions.	POS
	MR	Humorous, artsy, and even cute, in an off-kilter, dark, vaguely disturbing way.	POS
	CR	However, the calls constantly drop in my area and I experience mega-static, to the point...	NEG

Table 6: Cases of review texts in SST-2, MR and CR with high and low cross-task prototype scores.

contrast, review texts with low scores are overly specific and hence are less transferable across tasks. Hence, our meta-learner truly captures transferable knowledge for effective knowledge transfer.

5 Conclusion and Future Work

In this paper, we present the *TransPrompt* framework for few-shot learning across similar NLP tasks based on continuous prompt embeddings. Experimental results show that *TransPrompt* consistently outperforms strong baselines in both few-shot learning and standard fine-tuning settings. Additionally, we find that the meta-learner trained by *TransPrompt* can be adapted to previously unseen tasks easily. In the future, we will explore how *TransPrompt* is applied to other PLMs apart from BERT-style models and other NLP tasks.

Acknowledgement

This work was partially supported by Alibaba Group through the Alibaba Innovative Research Program and the National Natural Science Foundation of China under Grant Nos. U1911203, 61877018, 61672234, and 61672384.

References

Yuki Arase and Junichi Tsujii. 2021. Transfer fine-tuning of BERT with phrasal paraphrases. *Comput. Speech Lang.*, 66.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*, pages 632–642.

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. In *NeurIPS*.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc Viet Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. In *ACL*, pages 2978–2988.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *IWP@IJCNLP*.

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

Tianyu Gao, Xu Han, Hao Zhu, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2019. Fewrel 2.0: Towards more challenging few-shot relation classification. In *EMNLP-IJCNLP*, pages 6249–6254.

Ruiying Geng, Binhua Li, Yongbin Li, Jian Sun, and Xiaodan Zhu. 2020. Dynamic memory induction networks for few-shot text classification. In *ACL*, pages 1087–1094.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *KDD*, pages 168–177.

Yuncheng Hua, Yuan-Fang Li, Gholamreza Haffari, Guilin Qi, and Tongtong Wu. 2020. Few-shot complex knowledge base question answering via meta reinforcement learning. In *EMNLP*, pages 5827–5837.

Mike Huisman, Jan N. van Rijn, and Aske Plaat. 2020. A survey of deep meta-learning. *CoRR*, abs/2010.03522.

Muhammad Abdullah Jamal and Guo-Jun Qi. 2019. Task agnostic meta-learning for few-shot learning. In *CVPR*, pages 11719–11727.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know. *Trans. Assoc. Comput. Linguistics*, 8:423–438.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

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

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019a. Multi-task deep neural networks for natural language understanding. In *ACL*, pages 4487–4496.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Akihiro Nakamura and Tatsuya Harada. 2019. Revisiting fine-tuning for few-shot learning. *CoRR*, abs/1910.00216.

Haojie Pan, Chengyu Wang, Minghui Qiu, Yichang Zhang, Yaliang Li, and Jun Huang. 2021. Meta-kd: A meta knowledge distillation framework for language model compression across domains. In *ACL/IJCNLP*, pages 3026–3036.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL*, pages 115–124.

Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *CoRR*, abs/2003.08271.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Teven Le Scao and Alexander M. Rush. 2021. How many data points is a prompt worth? *CoRR*, abs/2103.08493.

Timo Schick and Hinrich Schütze. 2020. It's not just size that matters: Small language models are also few-shot learners. *CoRR*, abs/2009.07118.

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

Jiawei Sheng, Shu Guo, Zhenyu Chen, Juwei Yue, Lihong Wang, Tingwen Liu, and Hongbo Xu. 2020. Adaptive attentional network for few-shot knowledge graph completion. In *EMNLP*, pages 1681–1691.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *EMNLP*, pages 4222–4235.

Jake Snell, Kevin Swersky, and Richard S. Zemel. 2017. Prototypical networks for few-shot learning. In *NIPS*, pages 4077–4087.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, pages 1631–1642.

Ralf C. Staudemeyer and Eric Rothstein Morris. 2019. Understanding LSTM - a tutorial into long short-term memory recurrent neural networks. *CoRR*, abs/1909.09586.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune BERT for text classification? In *CCL*, pages 194–206.

Chengyu Wang, Haojie Pan, Yuan Liu, Kehan Chen, Minghui Qiu, Wei Zhou, Jun Huang, Haiqing Chen, Wei Lin, and Deng Cai. 2021. Mell: Large-scale extensible user intent classification for dialogue systems with meta lifelong learning. In *KDD*, pages 3649–3659.

Chengyu Wang, Minghui Qiu, Jun Huang, and Xiaofeng He. 2020a. Meta fine-tuning neural language models for multi-domain text mining. In *EMNLP*, pages 3094–3104.

Wei Wang, Bin Bi, Ming Yan, Chen Wu, Jiangnan Xia, Zuyi Bao, Liwei Peng, and Luo Si. 2020b. Structbert: Incorporating language structures into pre-training for deep language understanding. In *ICLR*.

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

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL-HLT*, pages 1112–1122.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*, pages 5754–5764.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big bird: Transformers for longer sequences. In *NeurIPS*.

Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. 2021. A comprehensive survey on transfer learning. *Proc. IEEE*, 109(1):43–76.