IDPG: An Instance-Dependent Prompt Generation Method

Anonymous ACL submission

Abstract

Prompt tuning is a new, efficient NLP transfer learning paradigm that adds a task-specific prompt in each input instance during the model training stage. It freezes the pre-trained language model and only optimizes a few task-specific prompts. In this paper, we propose a conditional prompt generation method to generate prompts for each input instance, referred to as the Instance-Dependent Prompt Generation (IDPG). Unlike traditional prompt tuning methods that use a fixed prompt, IDPG introduces a lightweight and trainable component to generate prompts based on each input sentence. Empirical experiments on ten natural language understanding (NLU) tasks show that our proposed method consistently outperforms various prompt tuning methods and other efficient transfer learning methods such as Compacter while tuning far fewer model parameters.

1 Introduction

Recently, pre-training a transformer model on a large corpus with language modeling tasks and fine-tuning it on different downstream tasks has become the main transfer learning paradigm in natural language processing (Devlin et al., 2019). Notably, this paradigm requires updating and storing all the model parameters for every downstream task. As the model size proliferates (e.g., 330M parameters for BERT (Devlin et al., 2019) and 175B for GPT-3 (Brown et al., 2020)), it becomes computationally expensive and challenging to fine-tune the entire pre-trained language model (LM). Thus, it is natural to ask the question of whether we can transfer the knowledge of a pre-trained LM into downstream tasks by tuning only a small portion of its parameters with most of them freezing.

Studies have attempted to address this question from different perspectives. One line of research (Li and Liang, 2021) suggests to augment the model with a few small trainable modules and freeze the original transformer weight.

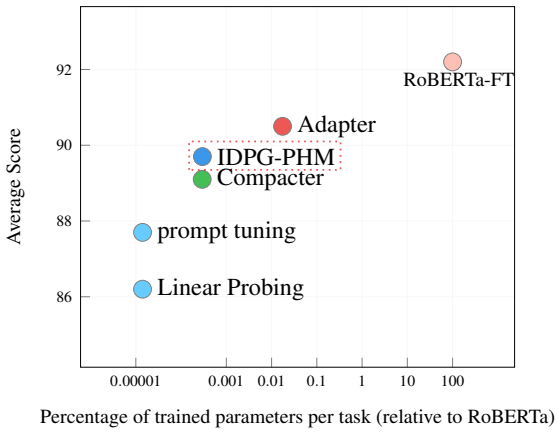


Figure 1: Overall evaluation on 10 NLU tasks. Note that we do not include parameters from classification heads.

Take Adapter (Houlsby et al., 2019; Pfeiffer et al., 2020a,b) and Compacter (Mahabadi et al., 2021) for example, both of them insert a small set of additional modules between each transformer layer. During fine-tuning, only these additional and task-specific modules are trained, reducing the trainable parameters to 1–3% of the original transformer model per task.

Another line of works focus on prompting. The GPT-3 models (Brown et al., 2020; Schick and Schütze, 2020) find that with proper manual prompts, a pre-trained LM can successfully match the fine-tuning performance of BERT models. LM-BFF (Gao et al., 2020), EFL (Wang et al., 2021), and AutoPrompt (Shin et al., 2020) further this direction by insert prompts in the input embedding layer. However, these methods rely on grid-search for a natural language-based prompt from a large search space, resulting in difficulties to optimize.

To tackle this issue, prompt tuning (Lester et al., 2021), prefix tuning (Li and Liang, 2021), and P-tuning (Liu et al., 2021b) are proposed to prepend trainable prefix tokens to the input layer and train these soft prompts only during the fine-tuning stage. In doing so, the problem of searching discrete

prompts are converted into an continuous optimization task, which can be solved by a variety of optimization techniques such as SGD and thus significantly reduced the number of trainable parameters to only a few thousand. However, all existing prompt-tuning methods have thus far focused on task-specific prompts, making them incompatible with the traditional LM objective. For example, it is unlikely to see many different sentences with the same prefix in the pre-training corpus. In light of these limitations, we instead ask the following question: *Can we generate input-dependent prompts to smooth the domain difference?*

In this paper, we present the instance-dependent prompt generation (IDPG) strategy for efficiently tuning large-scale LMs. Different from the traditional prompt-tuning methods that rely on a fixed prompt for each task, IDPG instead develops a conditional prompt generation model to generate prompts for each instance. Formally, the IDPG generator can be denoted as $f(x;\mathbf{W})$, where x is the instance representation and $\mathbf{W}$ represents the trainable parameters. In the extreme case, setting $\mathbf{W}$ to a zero matrix and only training the bias would degenerate it to the traditional prompt tuning (Lester et al., 2021). To further reduce the number of parameters in the generator $f(x;\mathbf{W})$, we first apply a lightweight bottleneck architecture (a two-layer perceptron), and then decompose it by a parameterized hypercomplex multiplication (PHM) layer (Zhang et al., 2021). In summary, our contributions are three-fold:

- We proposed an input-dependent prompt generation method, IDPG, which only requires training 105K parameters per task (roughly equal to 0.03% of a pre-trained LM like RoBERTa-Large (Liu et al., 2019)).
- A systematic evaluation on ten natural language understanding (NLU) tasks shows that our proposed method consistently outperforms the traditional task-specific prompt tuning methods by 0.4–1.9 points. Our method also has comparable performance to Adapter-based methods while using much fewer parameters (105K vs. 6.2M).
- This method provides a new research angle for prompt-tuning, which can be combined with other efficient transfer learning paradigm such as multi-layer prompt tuning.

2 Preliminary

2.1 Manual Prompt

Manual prompt learning (Brown et al., 2020; Schick and Schütze, 2020) insert a pre-defined label words in each input sentence. For example, it reformulates a sentence sentiment classification task with an input sentence S_1 as

$$x_{in} = \texttt{[CLS]}\, P\, \texttt{[SEP]}\, S_1\, \texttt{[EOS]},$$

where P is the prompt such as "indicating the positive user sentiment". Using pre-trained language model $\mathbf{M}$, we can obtain the sentence representation $\mathbf{h}_{\texttt{[CLS]}} = \mathbf{M}(x_{in})$, and train a task-specific head $\texttt{softmax}(\mathbf{W}\mathbf{h}_{\texttt{[CLS]}})$ to maximize the log-probability of the correct label. LM-BFF (Gao et al., 2020) showed that adding a specifically designed prompt during fine-tuning can benefit the few-shot scenario. EFL (Wang et al., 2021) further showed that reformulating the task as entailment can further improve the performance in both low-resource and high-resource scenarios.

2.2 Prompt Tuning

Prompt tuning (Lester et al., 2021), prefix tuning (Li and Liang, 2021), and P-tuning (Liu et al., 2021b) methods propose to insert a trainable prefix in front of the input sequence. Specifically, they reformulate the input for single sentence tasks as

$$x_{in} = \texttt{concat}[\mathbf{W}_p, \mathbf{E}(\texttt{[SEP]}\, S_2\, \texttt{[EOS]})]$$

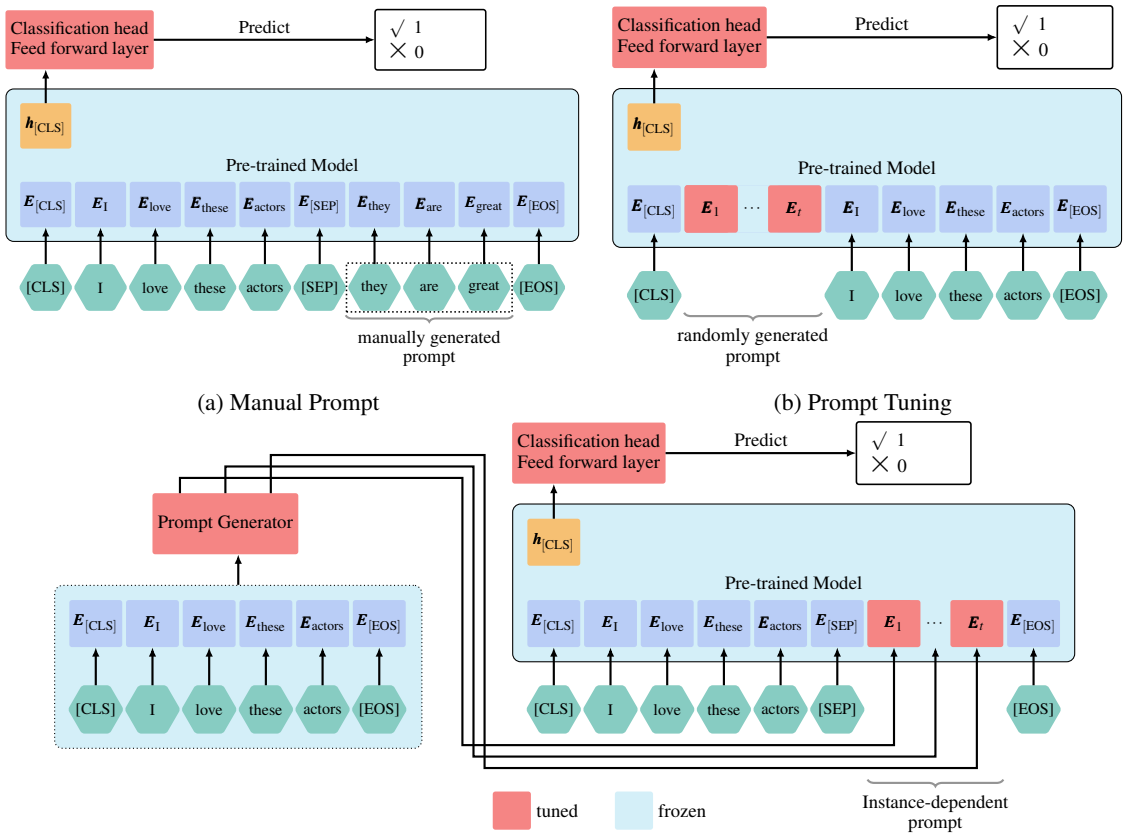
and for sentence pair tasks as

$$x_{in} = \texttt{concat}[\mathbf{W}_p, \mathbf{E}(\texttt{[SEP]}\, S_2\, \texttt{[SEP]}\, S_3\, \texttt{[EOS]})],$$

where $\mathbf{W}_p$ is the embedding table of the inserted prompt, S_2 and S_3 are input sentences, and $\mathbf{E}$ denotes the operation of tokenization and extraction of embedding. Apart from LM-BFF and EFL, there is no corresponding real text for the prompt as $\mathbf{W}_p$ is a set of random-initialized tensors to represent the soft prompt.

3 Instance-Dependent Prompt Generation (IDPG)

We now introduce our proposed method, IDPG, along with various model optimizations. The main procedure is illustrated in Figure 2.



(c) Instance-Dependent Prompt Generation

Figure 2: An illustration of (a) manual prompt; (b) prompt-tuning method; (c) our proposed method. The red block refers to the trainable module, while the blue block refers to the frozen module.

3.1 Instance-Dependent Generation

Let us assume a task T with training data $D_{train} = \{(x_i, y_i)\}_{i=1}^{N}$. Following prompt tuning, we define the input $x_i = \mathbf{E}(\texttt{[SEP]}\, S_1\, \texttt{[SEP]}\, S_2\, \texttt{[EOS]})$ for sentence-pair task or $x_i = \mathbf{E}(\texttt{[SEP]}\, S_1\, \texttt{[EOS]})$ for single-sentence task, where $\mathbf{E}(\cdot)$ is the token embedding for input sentences. Different from all previous works that only define a task-specific prompt $\mathbf{W}_p(T) \in \mathbb{R}^{d \times t}$, where t is the number of tokens in prompt representation and d is the hidden dimension, we propose a instance-dependent prompt generation method. Specifically, we suppose that the generation of prompt should not only depend on the task T, but also be affected by input sequence x_i. If $\mathbf{M}(x_i) \in \mathbb{R}^d$ is a representation of the input sequence x_i from the same pre-trained LM $\mathbf{M}$, we design a lightweight model $\mathbf{G}$ to generate the prompt,

$$\mathbf{W}_p(T, x_i) = \mathbf{G}(\mathbf{M}(x_i), T),\ x_i \in D_{train} \quad (1)$$

Then, we insert a prompt $\mathbf{W}_p(T)$ together with input sequence x_i to infer y_i during fine-tuning. In this way, we have a unified template

$$\texttt{softmax}(\mathbf{W}\mathbf{h}_{\texttt{[CLS]}}) \quad (2)$$

$$\mathbf{h}_{\texttt{[CLS]}} = \mathbf{M}(\texttt{concat}[x_i, \mathbf{W}_p(T, x_i)]) \quad (3)$$

where $\mathbf{W}$ is the trainable LM classification head.

To reduce the number of trainable parameters in $\mathbf{G}$, we apply a lightweight bottleneck architecture (i.e., a two-layer perceptron) for generation. As illustrated in Figure 3, the generator $\mathbf{G}$ first projects the original d-dimensional sentence representation $\mathbf{h}_i$ into m dimensions. After passing through a nonlinear function, generator $\mathbf{G}$ projects the hidden representation back to a d dimensions with t timestamps. The total number of parameters for generator $\mathbf{G}$ is $m(d+1)+td(m+1)$ (bias term included). This model can be regarded as the general version of prompt tuning: the bias term $t \times d$ in the second layer of $\mathbf{G}$ is a task-specific prompt, with preceding parts generating an instance-dependent prompt. The final prompt our method generated is a combination of both. We can control the added number of trainable parameters by setting $m \ll d$, but it is still expensive since hidden dimension d is usually large (1024 in BERT/RoBERTa-Large). In the sequel, we will introduce a parameter squeezing method to further reduce trainable parameters

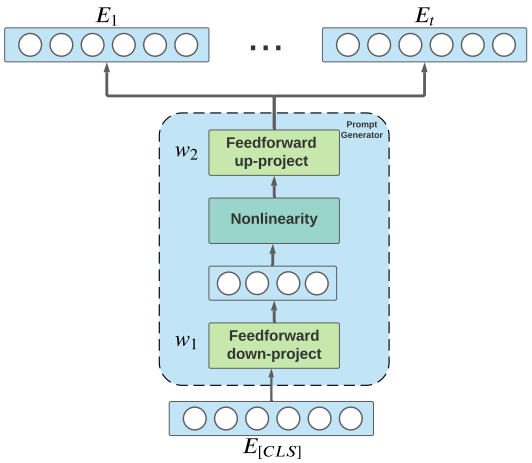


Figure 3: An illustration of prompt generator. In practice, PHM decomposes $\mathbf{W}_1$ and $\mathbf{W}_2$ as sum of Kronecker products.

without sacrificing performance.

Note that our proposed method relies on the input sentence representation $\mathbf{M}(x_i)$ to generate prompts. One caveat is that this method will have two forward passes of the pre-trained LM during inference time – first to generate $\mathbf{M}(x_i)$ and then to generate classification results. However, the sentence representation $\mathbf{M}(x_i)$ used in our method is task-agnostic. In practice, we can cache the prediction $\mathbf{M}(x_i)$ and use it in various downstream tasks or rely on a lightweight sentence representation such as Glove (Pennington et al., 2014).

3.2 Optimization

We propose two optimization techniques to further improve our proposed method.

3.2.1 Parameterized Hypercomplex Multiplication (PHM) Layers

Inspired by the recent application of parameterized hypercomplex multiplication (PHM) layers (Zhang et al., 2021) in Compacter (Mahabadi et al., 2021), we can leverage PHM layers to optimize our prompt generator, $\mathbf{G}$. Generally, the PHM layer is a fully-connected layer with form $y = \mathbf{W}x + b$, where $x \in \mathbb{R}^d$ is the input feature, $y \in \mathbb{R}^m$ is the output feature, and $\mathbf{W} \in \mathbb{R}^{m \times d}$ and $b \in \mathbb{R}^m$ are the trainable parameters. When m and d are large, the cost of learning $\mathbf{W}$ becomes the main bottleneck. PHM replaces the matrix $\mathbf{W}$ by a sum of Kronecker products of several small matrices. Given a user-defined hyperparameter $n \in \mathbb{Z}^+$ that divides m and d, $\mathbf{W}$ can be calculated as follows:

$$\mathbf{W} = \sum_{i=1}^{n} \mathbf{A}_i \bigotimes \mathbf{B}_i \quad (4)$$

where $\mathbf{A}_i \in \mathbb{R}^{n \times n}$, $\mathbf{B}_i \in \mathbb{R}^{\frac{m}{n} \times \frac{d}{n}}$, and $\bigotimes$ is Kronecker product. In this way, the number of trainable parameter is reduced to $n \times (n \times n + \frac{m}{n} \times \frac{d}{n}) = n^3 + \frac{m \times d}{n}$. Considering n is usually much smaller than m and d, PHM reduces the amount of parameters by a factor of n.

Suppose that we have a two layer perceptron with down-sample projection $\mathbf{W}_1 \in \mathbb{R}^{m \times d}$ and up-sample projection $\mathbf{W}_2 \in \mathbb{R}^{t \times d \times m}$, where d is the input embedding dimension, m is the hidden layer dimension, and t is the number of tokens we generate. For example, we use RoBERTa-Large with hidden size $d = 1024$, generator hidden size $m = 256$, $n = 16$, prompt length $t = 5$. By substituting the $\mathbf{W}_1$ and $\mathbf{W}_2$ by two PHM layers and letting A_i shared by both layers, we can reduce the number of parameters from 1.5M to 105K.

3.2.2 Supplementary Training

According to previous works (Phang et al., 2018; Wang et al., 2021), supplementing pre-trained LMs with rich data helps tasks with limited labels and stabilizes downstream fine-tuning. Following this idea, we conduct intermediate training for our proposed method.

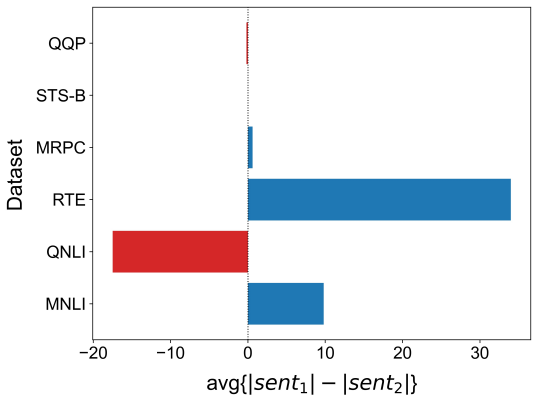


Figure 4: Length difference of GLUE sentence pair datasets.

However, a drawback of supplementary training is that if the data distribution of the downstream tasks is quite different from the supplementary training task, i.e., MRPC vs. MNLI (Wang et al., 2018), it may harm the downstream performance. Figure 4 provides a comprehensive statistic among all sentence pair tasks in GLUE benchmark. For example, the length of the first sentence in MNLI is 9.8 longer than the second sentence on average, while this length difference in MRPC is only 0.6. One natural solution to smooth the length distribution difference between tasks is to insert prompt in

both supplementary training and downstream fine-tuning stage. For example, assuming that we are adding a prompt with a length $t = 5$ after the second sentence in the supplementary training stage on MNLI. Then, when fine-tuning downstream tasks such as MRPC, we concatenate the prompt after the first sentence. In this way, the length difference in MNLI and MRPC becomes more balanced: 4.8 vs. $0.6 + 5 = 5.6$. As shown in Figure 5, we test five different insertion positions (Pos 0–4) for sentence pair tasks and three different positions (Pos 0, 1, 4) for single sentence tasks. We further reduce the distribution difference by reconstructing the supplementary training data. We double the MNLI dataset by reordering the two sentences on one shard, and use the doubled dataset during intermediate training.

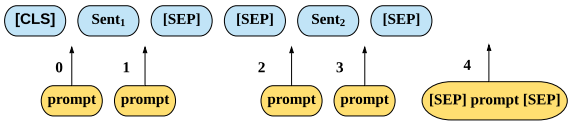


Figure 5: Insertion positions for sentence-pair tasks.

4 Experiment Results

4.1 Experimental Setup

We evaluate on ten standard natural language understanding (NLU) datasets – MPQA (Wiebe et al., 2005), Subj (Pang and Lee, 2004), CR (Hu and Liu, 2004), MR (Pang and Lee, 2005), and six tasks from GLUE (Wang et al., 2018) viz. SST-2, QNLI, RTE, MRPC, STS-B (Cer et al., 2017) and QQP. We compare our proposed method with a wide range of methods, as follows:

Transformer fine-tuning: We instantiated two versions – vanilla transformer fine-tuning (Liu et al., 2019) and the entailment-based fine-tuning (Wang et al., 2021).

Prompt tuning: We implemented two versions – standard prompt tuning (Lester et al., 2021) and prompt tuning with re-parameterization (Li and Liang, 2021). We also implemented the linear probing method that can be regarded as prompt tuning with length $t = 0$.

Adapter-based fine-tuning: This efficient transfer learning method inserts a adaptation module inside each transformer layer including Compactor (Mahabadi et al., 2021) and Adapter (Houlsby et al., 2019).

We compare these against two versions of instance-dependent generation methods: **IDPG-DNN** and **IDPG-PHM**. The first version is based on a 2-layer perceptron generator, which contains 1.5M parameters. The second one uses the PHM layer and only contains 105K parameters. For a fair comparison, all the pre-trained LM are 24-layer 16-head RoBERTa-Large models (Liu et al., 2019), and include supplementary training on MNLI dataset before any downstream fine-tuning.

Fine-tuning on small datasets is significantly affected by initialization (Wang et al., 2021). We follow the existing standard paradigm, where each model runs 5 times using 5 different random seeds. We report both average and voting results that takes the ensemble strategy across the 5 predictions. Using a majority vote can help us measure the limitation of each model after reducing the large variance problem. For intermediate training on MNLI, as introduced in Section 3.2, we run all the methods on a modified MNLI data (i.e., double the MNLI data and reverse sentence order on one copy) for 5 epochs. We adopt a default fine-tuning setting: training for 10 epochs on 8 GPUs, using a batch size of 16/32, a learning rate of $5e^{-4}$ and a prompt length $t = 5$. More training details can be found in Appendix A.

4.2 Performance in high-resource scenario

Table 1 shows the results of all the methods on full datasets across 10 NLU tasks. In all prompt-tuning methods, we report both average and voting results across 5 prompt implementations. We observe that: (i) Supplementary training on MNLI can increase the performance of prompt-tuning across most tasks, especially in the sentence-pair tasks. (ii) Our proposed method IDPG-PHM (DNN) consistently outperforms the prompt tuning method (with MNLI training) by average 0.4pt (0.5pt) or with voting by 0.6pt (0.5pt). The improvement is even bigger in single-sentence classification task. For example, our proposed method performs 1.9pt better than prompt-tuning in CR. (iii) PHM-based generator performs on par with the DNN-based generator while having significantly lower number of trainable parameters. Another interesting observation is that the re-parameterization scheme proposed in (Li and Liang, 2021) performs slightly worse compared to directly tuning the prompts, which is consistent with the recent observation in (Liu et al., 2021a).

Compared with other efficient transfer learning methods, IDPG performs slightly better than the

Table 1: Main results of different transfer learning method. Each methods are evaluated on full test sets. We report both average results (top) and the voting result (bottom) across 5 runs. **Bold** and underline mark the best result in average and voting results, respectively. We report the average of accuracy and F1 for both MRPC and QQP, and average of Pearson and Spearman correlation coefficients for STS-B. For all the other tasks, we report accuracy.

Method	MPQA	Subj	CR	MR	SST-2	QNLI	RTE	MRPC	STS-B	QQP	Avg
Transformer Fine-tuning											
RoBERTa	90.3	97.2	92.4	91.8	95.8	94.8	86.2	91.2	92.0	90.6	92.2
EFL	90.7	97.3	92.6	92.3	96.3	94.3	84.8	90.9	92.0	90.7	92.2
Adapter											
Compacter	90.5	96.9	85.8	84.1	93.0	92.9	77.7	90.8	90.5	88.6	89.1
Adapter	90.4	97.4	86.1	85.6	96.1	94.4	89.5	87.6	89.2	89.0	90.5
Prompting											
Linear Probing	$82.9_{\pm 0.4}$	$92.2_{\pm 0.3}$	$87.9_{\pm 0.7}$	$86.1_{\pm 0.2}$	$90.4_{\pm 0.8}$	$81.4_{\pm 0.2}$	$85.5_{\pm 0.4}$	$83.8_{\pm 0.4}$	$85.2_{\pm 0.3}$	$81.0_{\pm 0.2}$	85.6
	83.7	92.6	88.6	86.3	91.1	81.7	85.9	84.2	86.3	81.3	86.2
Prompt tuning	$84.9_{\pm 3.3}$	$93.4_{\pm 0.4}$	$86.5_{\pm 1.5}$	$87.2_{\pm 0.5}$	$93.9_{\pm 0.6}$	$86.0_{\pm 0.4}$	$87.0_{\pm 0.2}$	$82.8_{\pm 0.7}$	$\mathbf{86.6}_{\pm 0.3}$	$80.8_{\pm 0.2}$	86.9
	86.2	94.1	86.7	88.0	94.8	86.6	88.1	84.1	87.4	81.4	87.7
+ MNLI	$88.8_{\pm 0.2}$	$94.3_{\pm 0.5}$	$88.4_{\pm 2.8}$	$89.1_{\pm 0.7}$	$94.4_{\pm 0.3}$	$\mathbf{90.8}_{\pm 0.1}$	$89.2_{\pm 0.1}$	$83.8_{\pm 0.5}$	$85.1_{\pm 0.6}$	$82.0_{\pm 0.4}$	88.6
	89.3	94.7	90.5	89.8	94.7	91.2	89.5	83.7	86.1	82.3	89.2
+ Re-param	$89.0_{\pm 1.0}$	$93.7_{\pm 0.3}$	$87.4_{\pm 1.7}$	$88.7_{\pm 0.7}$	$94.2_{\pm 0.5}$	$90.7_{\pm 0.3}$	$87.9_{\pm 0.9}$	$\mathbf{85.2}_{\pm 1.1}$	$84.5_{\pm 0.8}$	$80.9_{\pm 0.9}$	88.2
	89.5	94.0	89.7	89.5	94.5	90.9	88.4	86.5	86.3	81.6	89.1
IDPG-PHM	$\mathbf{89.6}_{\pm 0.3}$	$94.4_{\pm 0.3}$	$\mathbf{90.3}_{\pm 0.2}$	$89.3_{\pm 0.4}$	$94.7_{\pm 0.2}$	$90.7_{\pm 0.3}$	$89.2_{\pm 0.2}$	$84.3_{\pm 0.8}$	$84.7_{\pm 0.9}$	$82.5_{\pm 0.2}$	89.0
	90.1	95.3	91.3	90.1	95.3	91.3	89.5	85.4	86.5	83.0	89.8
IDPG-DNN	$89.5_{\pm 0.7}$	$\mathbf{94.9}_{\pm 0.4}$	$89.9_{\pm 1.5}$	$\mathbf{90.2}_{\pm 0.6}$	$\mathbf{95.1}_{\pm 0.2}$	$90.5_{\pm 0.5}$	$\mathbf{89.4}_{\pm 0.4}$	$83.0_{\pm 0.5}$	$85.3_{\pm 0.7}$	$\mathbf{82.7}_{\pm 0.3}$	**89.1**
	90.0	95.8	91.3	91.2	95.2	91.3	89.5	83.0	86.3	83.4	89.7

Compacter (Mahabadi et al., 2021), and slightly worse than Adapter (Houlsby et al., 2019), across the ten tasks. However, we observe that the gap is mostly from sentence-pair tasks while IDPG-PHM is very efficient in single-sentence tasks. For example, IDPG performs 1.2pts better than Adapter over the five single-sentence tasks. We hypothesize that since the current prompt tuning method is embedding-based while Adapter/Compacter are layer-based, the sentence-pair tasks require steering model attention within each layer. We discuss additional steps to improve the performance in Section 6.

4.3 Efficiency

Table 2 lists the number of trainable parameters for different methods excluding the classification head. The general goal for efficient transfer learning is to train models with fewer parameters while achieving better performance. The ideal model in Figure 1 should be the one on the left-top corner. Traditional prompt-tuning method only requires training a token embedding table with a few thousand parameters. However, its performance is worse than a lightweight adapter model (e.g., Compacter with 104K parameters). Our proposed method, especially the IDPG-PHM, falls in the gap between prompt-tuning and adapter model, since it only requires training 105K parameters and performs better than the Compacter.

Method	Parameters
Transformer Fine-tune (Liu et al., 2019)	355M
Adapter (Houlsby et al., 2019)	6.2M
Compacter (Mahabadi et al., 2021)	104k
Prompt-tuning (Lester et al., 2021)	5K
+ Re-param (Li and Liang, 2021)	14M
IDPG-DNN	1.5M
IDPG-PHM	105K

Table 2: Number of trainable parameters of different methods. Note that we did not include the parameters from classification heads.

4.4 Performance in low-resource scenario

We further evaluate our proposed method in the low-resource scenario. Following the existing evaluation protocols in the few-shot setting (He et al., 2021), we sample a subset of the training data for each task with size $K \in \{100, 500, 1000\}$ as our training data and another subset with size 1000 as a development set.

In the extreme low-resource case when K=100, IDPG performs 0.9pt better than the traditional prompt tuning method. This improvement is higher than in high-resource scenario, illustrating that our method has better generalization in few-shot settings. When K becomes larger, IDPG-PHM still maintains good results with 0.1pt (K=500) and 0.5pt (K=1000) better accuracy than traditional

Table 3: Low-resource results are evaluated on full test sets. We report both average results (top) and the voting result (bottom) across 5 runs. **Bold** and underline mark the best result in average and voting results, respectively. We report the average of accuracy and F1 for both MRPC and QQP, and average of Pearson and Spearman correlation coefficients for STS-B. For all other tasks, we report accuracy.

Method	MPQA	Subj	CR	MR	SST-2	QNLI	RTE	MRPC	STS-B	QQP	Avg
K = 100											
prompt tuning	$75.9_{\pm 1.6}$	$86.8_{\pm 0.8}$	$72.9_{\pm 1.4}$	$74.1_{\pm 1.4}$	$82.9_{\pm 2.0}$	$\mathbf{82.7}_{\pm 0.2}$	$86.5_{\pm 0.6}$	$\mathbf{80.0}_{\pm 1.3}$	$70.2_{\pm 3.1}$	$\mathbf{76.5}_{\pm 0.4}$	78.9
+ MNLI	76.9	87.4	72.7	74.7	84.0	83.1	87.7	81.1	72.8	76.7	79.7
IDPG-PHM	$\mathbf{79.0}_{\pm 3.7}$	$\mathbf{87.6}_{\pm 1.1}$	$75.0_{\pm 1.6}$	$76.2_{\pm 1.3}$	$87.6_{\pm 1.3}$	$80.4_{\pm 1.2}$	$86.3_{\pm 0.5}$	$79.3_{\pm 0.4}$	$\mathbf{70.9}_{\pm 2.5}$	$76.1_{\pm 0.6}$	**79.8**
	78.2	89.4	74.8	77.5	86.1	81.3	85.6	80.5	73.5	76.5	80.3
IDPG-DNN	$78.0_{\pm 2.1}$	$84.2_{\pm 1.6}$	$\mathbf{76.3}_{\pm 4.5}$	$\mathbf{77.4}_{\pm 0.5}$	$\mathbf{89.6}_{\pm 1.2}$	$81.1_{\pm 0.8}$	$\mathbf{87.4}_{\pm 0.8}$	$78.8_{\pm 1.3}$	$70.6_{\pm 2.8}$	$74.1_{\pm 0.9}$	**79.8**
	74.6	85.2	76.5	78.4	92.5	82.3	88.4	79.6	72.8	74.5	80.5
K = 500											
prompt tuning	$82.4_{\pm 1.3}$	$91.2_{\pm 0.1}$	$86.8_{\pm 0.4}$	$84.6_{\pm 0.8}$	$88.6_{\pm 1.0}$	$\mathbf{86.3}_{\pm 0.4}$	$86.5_{\pm 0.4}$	$80.0_{\pm 0.4}$	$77.4_{\pm 1.9}$	$77.8_{\pm 0.3}$	84.2
+ MNLI	82.8	91.8	86.8	85.5	89.2	86.7	85.9	78.6	79.2	77.9	84.4
IDPG-PHM	$81.6_{\pm 2.7}$	$\mathbf{91.4}_{\pm 0.7}$	$85.8_{\pm 2.0}$	$85.8_{\pm 0.5}$	$88.5_{\pm 1.3}$	$85.0_{\pm 0.4}$	$86.3_{1.3}$	$81.9_{\pm 0.8}$	$78.3_{\pm 1.5}$	$\mathbf{78.1}_{\pm 0.3}$	84.3
	82.5	92.5	85.6	86.9	89.4	86.3	86.6	82.7	79.7	78.6	85.1
IDPG-DNN	$\mathbf{84.8}_{\pm 0.7}$	$90.8_{\pm 0.6}$	$\mathbf{89.7}_{\pm 1.0}$	$\mathbf{86.1}_{\pm 2.8}$	$\mathbf{90.4}_{\pm 1.6}$	$84.8_{\pm 0.3}$	$\mathbf{87.7}_{\pm 0.7}$	$\mathbf{82.0}_{\pm 1.4}$	$\mathbf{79.1}_{\pm 2.3}$	$77.1_{\pm 0.4}$	**85.3**
	85.4	92.2	90.8	88.1	92.1	85.3	87.4	82.8	81.6	78.0	86.4
K = 1000											
prompt tuning	$83.9_{\pm 2.0}$	$92.6_{\pm 0.4}$	$87.2_{\pm 1.4}$	$86.7_{\pm 0.3}$	$89.9_{\pm 1.0}$	$\mathbf{86.9}_{\pm 0.1}$	$86.4_{\pm 0.7}$	$82.5_{\pm 0.3}$	$82.9_{\pm 1.3}$	$78.6_{\pm 0.3}$	85.8
+ MNLI	84.6	92.8	87.6	87.1	90.5	87.4	85.6	82.2	84.1	78.8	86.1
IDPG-PHM	$83.4_{\pm 1.7}$	$\mathbf{93.4}_{\pm 0.9}$	$89.2_{\pm 0.8}$	$88.0_{\pm 0.9}$	$90.2_{\pm 1.0}$	$85.5_{\pm 0.6}$	$86.9_{\pm 0.6}$	$\mathbf{83.1}_{\pm 0.4}$	$83.9_{\pm 0.8}$	$\mathbf{78.9}_{\pm 0.4}$	86.3
	85.2	94.2	90.0	88.6	91.2	86.2	86.3	84.5	85.2	79.3	87.1
IDPG-DNN	$\mathbf{85.9}_{\pm 0.8}$	$93.3_{\pm 1.2}$	$\mathbf{89.9}_{\pm 0.8}$	$\mathbf{89.6}_{\pm 1.1}$	$\mathbf{92.2}_{\pm 0.8}$	$85.2_{\pm 1.3}$	$\mathbf{87.7}_{\pm 0.8}$	$82.5_{\pm 0.9}$	$\mathbf{84.7}_{\pm 0.9}$	$78.0_{\pm 0.8}$	**86.9**
	86.6	94.2	90.3	90.2	92.8	86.8	87.7	82.9	85.2	78.6	87.5

prompt tuning. We also observe that when K is small, our method sometimes has a higher variance than the standard prompt tuning method. We suspect that this may be due to bad initialization that leads our model to non-optimal parameters.

4.5 Intrinsic Study

We conduct several ablation studies including exploration of different generator architectures and impact of selecting different prompt positions.

4.5.1 Generator Architecture Exploration

We explore three different architectures for the proposed PHM-based generator: (i) Residual: a residual structure (He et al., 2016) is applied to add the sentence representation to each generated tokens; (ii) LayerNorm: layer normalization (Ba et al., 2016) is also added to normalize the generated token embedding; (iii) residual + layerNorm: a mixed model that uses both the residual component and LayerNorm. Note that, to balance the token embedding and sentence embedding, we apply LayerNorm to each embedding first, then after the add-up, use LayerNorm again to control the generated tokens. We observe that adding LayerNorm slightly improves the voting results, while residual performs slightly worse. One surprising result is that the mixed model of Residual and LayerNorm has significantly poorer performance compared to other methods.

Architecture	Avg	Voting
PHM	86.1	86.9
+residual	85.9	86.7
+LayerNorm	86.1	87.1
+residual+LayerNorm	77.8	81.2

Table 4: Ablation study on generator architecture.

4.5.2 Prompt Position

As we discussed in Section 3.2.2, the prompt position has a direct impact on the prediction results. We conduct a comprehensive study of the prompt position for our proposed method in both supplementary training and downstream fine-tuning phases.

Looking at the prompt position in downstream tasks first, Figure 6(a) shows that for both standard prompt tuning and our proposed method, the best position is 0 for single-sentence tasks and 1 for sentence-pair tasks. This result is intuitive for single-sentence tasks since prompt in position 0 can be regarded as the premise and original input sentence as the hypothesis. For sentence-pair tasks, we hypothesize that inserting prompt into position 1 can better align the two input sentences. Fig-

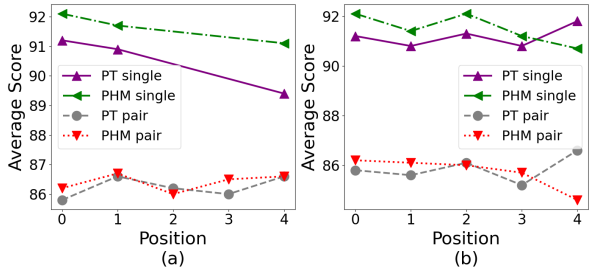


Figure 6: Impact of prompt position on (a) downstream tasks; (b) supplementary training phase.

ure 6(b) illustrates the effect of prompt position on the supplementary training phase. It is interesting that IDPG achieves best results in position 0 while the standard prompt-tuning achieves the best results in position 4 for both single-sentence and sentence-pair tasks.

5 Related Work

Supplementary Training: Existing works (Phang et al., 2018; Liu et al., 2019) have observed that starting from the fine-tuned MNLI model results in a better performance than directly from the vanilla pre-trained models for RTE, STS, and MRPC tasks. A series of work (SentenceBERT (Reimers and Gurevych, 2019), BERT-flow (Li et al., 2020), SimCSE (Gao et al., 2021)) explored intermediate training to improve STS tasks. All of them applied pre-fine tuning on NLI datasets. More recently, EFL (Wang et al., 2021) proposed a task transformation paradigm, improving single sentence tasks with less labels using rich sentence-pair datasets.

Adapter Tuning: Adapter tuning has emerged as a novel parameter-efficient transfer learning paradigm (Houlsby et al., 2019; Pfeiffer et al., 2020b), in which adapter layers – small bottleneck layers – are inserted and trained between frozen pre-trained transformer layers. On the GLUE benchmark, adapters attain within 0.4% of the performance of full fine-tuning by only training 3.6% parameters per task. Compactor (Mahabadi et al., 2021) substitutes the down-projector and up-projector matrices by a sum of Kronecker products, reducing the parameters by a large margin while maintaining the overall performance.

Prompting: Hand-crafted prompts were shown to be helpful to adapt generation in GPT-3 (Brown et al., 2020). Existing works including LM-BFF (Gao et al., 2020; Wang et al., 2021) explored the prompt searching in a few-shot setting.

Recently, several researchers have proposed continuous prompts training to overcome the challenges in discrete prompt searching. Prefix tuning (Li and Liang, 2021) prepends a sequence of trainable embeddings at each transformer layer and optimizes them for natural language generation tasks. Two contemporaneous works – prompt tuning (Lester et al., 2021) and P-tuning (Liu et al., 2021b), interleave the training parameters in the input embedding layer instead of each transformer layer. All these methods focus on task-specific prompt optimization. Our proposed method, IDPG, is the first prompt generator that is not only task-specific but also instance-specific.

6 Conclusion and Discussion

We have introduced IDPG, an instance-dependent prompt generation model that generalizes better than the existing prompt tuning methods. Our method first factors in a instance-dependent prompt, which is robust to data variance. Parameterized Hypercomplex Multiplication (PHM) is applied to shrink the training parameters in our prompt generator, which helps us build an extreme lightweight generation model. Despite adding fewer parameters than prompt tuning, IDPG shows consistent improvement. It also outperforms the lightweight adapter tuning methods such as Compacter while using similar amount of trainable parameters. This work provided a new research angle for prompt-tuning of a pre-trained language model. We also obtained a comprehensive understanding of how to smooth the distribution gap in intermediate training through our extensive experiments. These results could be further improved in future by:

Multi-layer Prompt Generation: Existing work (Li and Liang, 2021; Liu et al., 2021a) showed that we can insert prompts or prefixes into each layer of pre-trained LM, which can further improve the performance while slightly increasing number of trainable parameters. One promising direction is to combine our PHM-based generation model with the multi-layer prompt tuning.

Lightweight Input Representation: The proposed IDPG method relies on pre-trained LM to extract sentence representation, i.e., `[CLS]` token embedding. Obtaining contextualized transformer sentence embedding is often expensive if it is not pre-computed. One open question is to explore reliability on lightweight sentence representations such as GLOVE embedding (Pennington et al., 2014) or token embedding of pre-trained language models.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.

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

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

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Ruidan He, Linlin Liu, Hai Ye, Qingyu Tan, Bosheng Ding, Liying Cheng, Jia-Wei Low, Lidong Bing, and Luo Si. 2021. On the effectiveness of adapter-based tuning for pretrained language model adaptation. *arXiv preprint arXiv:2106.03164*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177.

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

Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. 2020. On the sentence embeddings from pre-trained language models. *arXiv preprint arXiv:2011.05864*.

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

Xiao Liu, Kaixuan Ji, Yicheng Fu, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021a. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021b. Gpt understands, too. *arXiv preprint arXiv: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*.

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021. Compacter: Efficient low-rank hypercomplex adapter layers. *arXiv preprint arXiv:2106.04647*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. *arXiv preprint cs/0409058*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. *arXiv preprint cs/0506075*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2020a. Adapterfusion: Non-destructive task composition for transfer learning. *arXiv preprint arXiv:2005.00247*.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020b. Mad-x: An adapter-based framework for multi-task cross-lingual transfer. *arXiv preprint arXiv:2005.00052*.

Jason Phang, Thibault Févry, and Samuel R Bowman. 2018. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv preprint arXiv:1811.01088*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Timo Schick and Hinrich Schütze. 2020. Exploiting cloze questions for few shot text classification and natural language inference. *arXiv preprint arXiv:2001.07676*.

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

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Sinong Wang, Han Fang, Madian Khabsa, Hanzi Mao, and Hao Ma. 2021. Entailment as few-shot learner. *arXiv preprint arXiv:2104.14690*.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2):165–210.

Aston Zhang, Yi Tay, Shuai Zhang, Alvin Chan, Anh Tuan Luu, Siu Cheung Hui, and Jie Fu. 2021. Beyond fully-connected layers with quaternions: Parameterization of hypercomplex multiplications with $1/n$ parameters. *arXiv preprint arXiv:2102.08597*.

A Appendix

A.1 Experimental Settings

We use RoBERTa-Large (Liu et al., 2019) model implemented by Fairseq (Ott et al., 2019) as our basic model. The detailed model hyperparameters are listed below:

Hyperparam	**Supplmentary**	**Finetune**	**few-shot**
#Layers	24	24	24
Hidden size	1024	1024	1024
FFN inner hidden size	4096	4096	4096
Attention heads	16	16	16
Attention head size	64	64	64
dropout	0.1	0.1	0.1
Learning Rate	linearly decayed	fixed	fixed
Peak Learning Rate	$1e^{-5}$	$5e^{-4}$	$5e^{-4}$
Batch Size	32	{16, 32}	16
Weight Decay	0.1	0.1	0.1
Training Epoch	10	10	50
Adam ε	1e-6	1e-6	1e-6
Adam β_1	0.9	0.9	0.9
Adam β_2	0.98	0.98	0.98

Table 5: Hyperparameters for supplmentary training, fine-tuning, few-shot fine-tuning.

Note that both transformer fine-tuning method including RoBERTa (Liu et al., 2019) and EFL (Wang et al., 2021) used a polynomial learning rate scheduler with 6% of total steps to warm up.

A.2 Datasets

We provide a detailed information in Table 6 for 10 NLU datasets we used.

Corpus	**|Train|**	**|Valiadation|**	**Task**	**Evaluation Metrics**
Single Sentence Tasks				
CR	1,775	2,000	sentiment	accuracy
MR	8,662	2,000	sentiment	accuracy
SUBJ	8,000	2,000	sentiment	accuracy
MPQA	8,606	2,000	opinion polarity	accuracy
SST-2	67,349	1,821	sentiment analysis	accuracy
Sentence Pair Tasks				
QNLI	104,743	5,463	NLI	accuracy
RTE	2,491	278	NLI	accuracy
MRPC	3,668	409	paraphrase	accuracy/F1
QQP	363,846	40,430	paraphrase	accuracy/F1
STS-B	5,749	1,500	sentence similarity	Pearson/Spearman corr.

Table 6: The datasets evaluated in this work.