RGL: A Simple yet Effective Relation Graph Augmented Prompt-based Tuning Approach for Few-Shot Learning

Anonymous ACL submission

Abstract

Pre-trained language models (PLMs) which carry generic knowledge can be a good starting point for adapting to downstream applications. However, it is difficult to generalize PLMs to new tasks with only a limited number of labeled samples given. In this work, we show that Relation Graph augmented Learning RGL method can obtain better performance in few-shot natural language understanding tasks. During learning, RGL constructs a relation graph based on the label consistency between samples in the same batch, and learns to solve the resultant node classification and link prediction problems of the relation graphs. In this way, RGL fully exploits the limited supervised information, which can boost the tuning effectiveness. Extensive experiments on benchmark tasks show that RGL consistently improve the performance of prompt-based tuning strategies.

1 Introduction

Pre-trained language models (PLMs), such as BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019), has become the standard workhorse for nowadays natural language processing (NLP) tasks. A direct way of leveraging these PLMs is to fine-tune them by taking gradient descent w.r.t. the objective of downstream tasks. However, tuning the large PLM by a few labeled samples has a high risk of overfitting (Dodge et al., 2020; Zhang et al., 2021; Gunel et al., 2020). Besides, as PLMs are trained by an objective different from the downstream tasks, the ability of PLM may not be fully exploited. Recently, prompt-based tuning methods emerge and show promising results on adapting PLMs to new tasks with a few labeled samples (Liu et al., 2021). In particular, prompts are used to reformulate the downstream tasks into the same form of pre-training tasks such that the gap between pre-training and fine-tuning is reduced (Brown et al., 2020; Schick and Schütze, 2021a). Concretely, prompt-based tuning strategies rewrite the input sequence into cloze-style question with masks (Schick and Schütze, 2021a). The input sequence is rewritten as prompts, while the corresponding label is replaced by answer tokens. Some methods use hard prompts and answers which use text strings with semantic meaning conveyed (Schick and Schütze, 2021b; Tam et al., 2021; Gao et al., 2021), while others take learnable parameters as soft prompts and answers (Liu et al., 2021; Li and Liang, 2021; Lester et al., 2021). In addition, multiple prompts can be used to boost the performance of prompt-based tuning (Brown et al., 2020; Schick and Schütze, 2021b). While the above strategies improve few-shot performance, they pay less attention to representation learning through fully utilizing supervisions from few-shot training datasets.

In this work, we propose a simple yet effective relation graph augmented approach which can enhance the performance of prompt-based tuning strategies PLM in few-shot natural language understanding tasks. Specifically, our proposal aims at fully exploiting the limited supervised information via **R**elation **G**raph augmented **L**earning, we thus call the proposed method RGL. RGL first constructs batch-wise relation graph, where every node refers to a labeled sample and the edge between nodes refers to the similarity between the two samples. RGL establishes edge in the relation graph w.r.t whether the two samples are from the same class and regularizes the similarity of representations learned by PLMs between every two samples to fit the edge of relation graph. RGL can easily scale up as the relation graph is constructed w.r.t. only a mini-batch of sampled data points per iteration. Empirical results on benchmark datasets show that RGL can consistently improve the performance of prompt-based tuning. We make our codes[1] publicly available.

[1] https://github.com/[XXX]/RGL

2 Background

In this paper, we target at generalizing a pre-trained language model (PLM) to text classification tasks with a few labeled examples. Following the problem definition of (Schick and Schütze, 2021b; Gao et al., 2021), each task $\mathcal{T}$ with label space $\mathcal{Y}$ consists of three datasets: (i) training dataset $\mathcal{D}_{\text{train}} = \{(\mathbf{x}_i, y_i)\}$ containing a few labeled examples where $\mathbf{x}_i$ is the sequence and y_i is the corresponding label, (ii) development(validation) dataset $\mathcal{D}_{\text{dev}}$ containing the same number of samples as $\mathcal{D}_{\text{train}}$ and is used for model selection, and (iii) testing dataset $\mathcal{D}_{\text{test}}$ containing samples to be predicted.

In prompt-based tuning, each input sample $(\mathbf{x}_i, y_i)$ is reformulated as a pattern-verbalizer pair (PVP) (Schick and Schütze, 2021a) in terms of $(p(\mathbf{x}_i), v(y_i))$. The pattern mapping function $p(\cdot)$ maps $\mathbf{x}_i$ to cloze questions with masks. For example, a single sentence "$\mathbf{x}_i$ = [CLS]s[SEP]" can be mapped as "$p(\mathbf{x}_i)$ = [CLS]s It was [MASK].[SEP]", where [CLS] and [SEP] are special start and end tokens. As for a sentence pair "$\mathbf{x}_i$ = [CLS]s_1[SEP]s_2[SEP]", it can be mapped as "$p(\mathbf{x}_i)$ = [CLS]s_1[MASK], s_2[SEP]". The verbalizer $v(\cdot)$ maps y_i to tokens expressing the semantic meaning of y_i. For examples, "positive/negative" can be mapped as "good/bad". With PVPs, the token embedding $\mathbf{h}_i^{[\text{MASK}]}$ is taken as the representation of $\mathbf{x}_i$. The class prediction $\hat{\mathbf{y}}_i$ contains conditional probability distribution of each possible class label given $\mathbf{x}_i$, whose entry corresponds to y_i is estimated as

$$q(y_i|\mathbf{x}_i) = \frac{\exp(p([\text{MASK}] = v(y_i)|p(\mathbf{x}_i)))}{\sum_{y_j \in \mathcal{Y}} \exp(p([\text{MASK}] = v(y_j)|p(\mathbf{x}_i)))} = \frac{\exp(\mathbf{w}_{v(y_i)}^\top \cdot \mathbf{h}_i^{[\text{MASK}]})}{\sum_{y_j \in \mathcal{Y}} \exp(\mathbf{w}_{v(y_j)}^\top \cdot \mathbf{h}_i^{[\text{MASK}]})}, \quad (1)$$

where $\mathbf{w}_v$ is the logit vector of token v existing in the vocabulary. Let $\mathbf{y}_i$ be a one-hot vector with all 0s but a single one denoting the index of the ground truth class label $y_i \in \{1, \ldots, C\}$. The model is optimized with respect to loss $\mathcal{L}_{\text{CE}}$ defined as

$$\mathcal{L}_{\text{CE}} = \sum_{i=1}^{N} -\log(\hat{\mathbf{y}}_i)^\top \mathbf{y}_i, \quad (2)$$

where $(\cdot)^\top$ denotes the transpose operation.

3 RGL: Our Proposed Method

In this section, we present the proposed RGL (Figure 1). We manage to exploit more supervised signals out of the training samples by constructing and learning on batch-wise relation graphs, which can boost the effectiveness of prompt-based tuning.

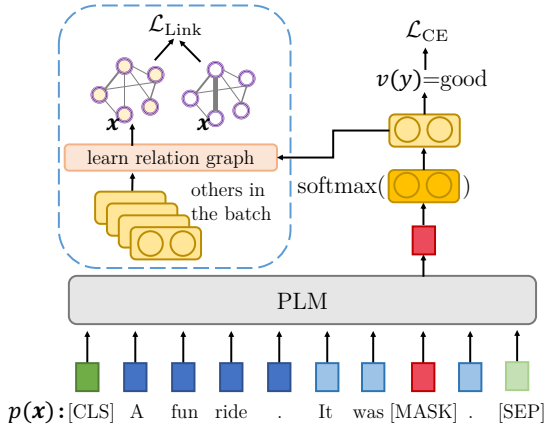


Figure 1: A high-level illustration of prompt-based tuning with the proposed RGL (marked by the square with blue dotted lines).

3.1 Defining the Relation Graphs

Consider a mini-batch $\mathcal{B} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ containing N randomly sampled sequence-label pairs, whose indexes are kept in $\mathcal{I} = \{1, \ldots, N\}$. We try to exploit more supervised information by modeling its relation graph. Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ denotes the relation graph among the N training samples in $\mathcal{B}$. In particular, $\mathcal{V}$ is a set of nodes where each node $v_i \in \mathcal{V}$ corresponds to one training sample $\mathbf{x}_i$, and $\mathcal{E} = \{e_{ij}\}$ is a set of edges between the N training samples. As we mainly consider text classification tasks, the edge e_{ij} between a node v_i and another node v_j is established if these nodes come from the same class. Formally, e_{ij} is set as

$$e_{ij} = \begin{cases} 1 & \text{if } y_j = y_i \\ 0 & \text{otherwise} \end{cases}. \quad (3)$$

Note that (3) is just an example of defining e_{ij} in classification tasks, which already obtains good performance. One can define e_{ij} in other ways as extension, such as modeling both the intra- and inter-class relations (Kim et al., 2019), using auxiliary information to it, and using real-valued e_{ij} for regression tasks.

3.2 Learning with Relation Graphs

On the relation graph $\mathcal{G}$ of mini-batch $\mathcal{B}$, we expand the origin classification task into two problems: (i) a node classification problem to predict the correct class of each node, and (ii) a link prediction problem to connect nodes of the same classes and disconnect nodes from different classes.

The node classification problem corresponds exactly to the original classification task. Therefore, we obtain class prediction $\hat{\mathbf{y}}_i$ of v_i (corresponding to $\mathbf{x}_i$) by (1) and calculate $\mathcal{L}_{\text{CE}}$ loss by (2).

As for the link prediction problem, we establish $\hat{e}_{ij}$ between v_i and v_j based on the relevance between $\hat{\mathbf{y}}_i$ and $\hat{\mathbf{y}}_j$:

$$\hat{e}_{ij} = g(\hat{\mathbf{y}}_i, \hat{\mathbf{y}}_j), \tag{4}$$

where $\hat{\mathbf{y}}_i, \hat{\mathbf{y}}_j$ are obtained by (1), and $g(\cdot,\cdot)$ is simply set as cosine similarity in this paper. There exist other choices to obtain $\hat{e}_{ij}$ such as calculate $g(\mathbf{h}_i^{[\text{CLS}]}, \mathbf{h}_j^{[\text{CLS}]})$ or $g(\mathbf{h}_i^{[\text{MASK}]}, \mathbf{h}_j^{[\text{MASK}]})$ instead. We use $\hat{\mathbf{y}}_i, \hat{\mathbf{y}}_j$ as they carry more semantic information relevant to each class, which are more predictive and obtain better empirical performance. One may also consider using parameterized $g(\cdot,\cdot)$ instead of using cosine similarity. However, considering the limited number of labeled samples, we avoid bringing in extra parameters to reduce the risk of overfitting. To measure the losses of link prediction, We design $\mathcal{L}_{\text{Link}}$ loss as

$$-\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{A}(i)} e_{ij}\log(\hat{e}_{ij}) + (1-e_{ij})\log(1-\hat{e}_{ij}), \tag{5}$$

where $\mathcal{A}(i) = \{j \in \mathcal{I} \text{ and } i \neq j\}$.

For each mini-batch $\mathcal{B}$, we optimize the model to minimize the combination of node classification loss $\mathcal{L}_{\text{CE}}$ and link prediction loss $\mathcal{L}_{\text{Link}}$ as a whole:

$$\mathcal{L}_{\text{CE}} + \alpha\mathcal{L}_{\text{Link}}, \tag{6}$$

where α is a hyperparameter to control the contribution of this $\mathcal{L}_{\text{Link}}$.

3.3 Comparisons with SCL

The most relevant work to RGL is SCL(Gunel et al., 2020) which applies supervised contrastive learning (SCL) on a batch level while fine-tuning PLM (rather than prompt-based tuning PLM). SCL optimizes for the following objective:

$$\mathcal{L}_{\text{CE}} + \beta\mathcal{L}_{\text{SCL}}, \tag{7}$$

where $\mathcal{L}_{\text{SCL}}$ takes the following form:

$$-\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{P}(i)} \log \frac{\exp(f(\mathbf{x}_i)\cdot f(\mathbf{x}_j)/\tau)}{\sum_{k\in\mathcal{A}(i)}\exp(f(\mathbf{x}_i)\cdot f(\mathbf{x}_k)/\tau)}, \tag{8}$$

where $\mathcal{P}(i) = \{j \in \mathcal{A}(i) : y_j = y_i\}$, τ is a hyperparameter. $f(\mathbf{x}_i)$ is the representation of $\mathbf{x}_i$, which is chosen as $\mathbf{h}_i^{[\text{CLS}]}$ in (Gunel et al., 2020) and is changed to $\mathbf{h}_i^{[\text{MASK}]}$ following routine in prompt-based tuning strategies (Liu et al., 2021).

Our RGL is different from SCL in three aspects: (i) RGL constructs relation graphs and aims to approximate the edge labels $\hat{e}_{ij}$ defined in (3), while SCL does not use any precise measures (e.g., edge labels) constraining similarities/distances between intra/inter-class samples; (ii) RGL rules samples from the same class to be connected and otherwise disconnected, while SCL only enforces samples from the same class to be close without explicitly pushing those from different classes to be farther apart; and (iii) RGL estimates edge labels $\hat{e}_{ij}$ using the prediction $\hat{\mathbf{y}}_i$ and $\hat{\mathbf{y}}_i$ to regularize the target task-dependent representations, while SCL uses representation of $\mathbf{x}_i$ (outputs of PLM) which might be irrelevant to the target task.

4 Experiments

Experimental settings. We use RoBERTa-large[2] (Liu et al., 2019) as the PLM. We take PET[3] (Schick and Schütze, 2021b) as the basic prompt-based tuning method. Upon PET, we compare the benefits of applying the proposed RGL versus SCL (Gunel et al., 2020). We implement RGL in PyTorch. All the hyperparameters are selected using the provided development set via grid search following Gao et al. (2021). We first select Adam optimizer learning rate from $\{1e-5, 2e-5, 5e-5\}$ and batch size from $\{2, 4, 8\}$ for PET. Then, we select hyperparameter α from $[0 : 0.2 : 1]$ for RGL and hyperparameters β and τ for SCL. We train all methods for a maximum number of 1000 steps and evaluate the performance on development set every 100 steps.

Dataset. Experiments are performed on a variant of GLUE benchmarks (Wang et al., 2018) for few-shot setting, which is provided by Gao et al. (2021). Gao et al. (2021) provide 5 different training sets and developing sets where each of them consist of 16 labeled samples per class. The averaged performance over these 5 splits are reported. We also evaluate the proposed RGL on the SuperGLUE (Wang et al., 2019) variant proposed by Schick and Schütze (2021b), whose results are put in Appendix due to space limit.

Results. Table 1 shows the results. As shown, both RGL and SCL can bring in additional perfor-

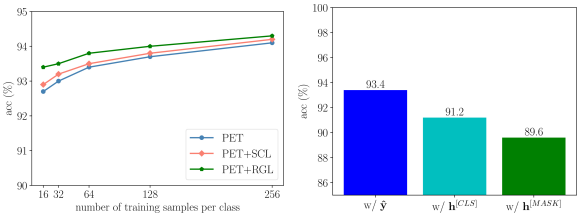
[2] https://huggingface.co/roberta-large.
[3] https://github.com/timoschick/pet.

	SST-2 (acc)	SST-5 (acc)	MR (acc)	CR (acc)	MPQA (acc)	Subj (acc)	TREC (acc)	CoLA (Matt.)
PET	$92.7_{(0.9)}$	$47.4_{(2.5)}$	$87.0_{(1.2)}$	$\mathbf{90.3}_{(1.0)}$	$84.7_{(2.2)}$	$\mathbf{91.2}_{(1.1)}$	$84.8_{(5.1)}$	$9.3_{(7.3)}$
PET+SCL	$92.9_{(1.9)}$	$48.0_{(1.9)}$	$87.1_{(1.8)}$	$\mathbf{90.3}_{(1.5)}$	$84.9_{(2.4)}$	$\mathbf{91.2}_{(1.7)}$	$85.5_{(2.6)}$	$20.9_{(16.5)}$
relative ↑	+0.2	+0.6	+0.1	+0.0	+0.2	+0.0	+0.7	+11.6
PET+RGL	$\mathbf{93.4}_{(0.5)}$	$\mathbf{49.3}_{(1.2)}$	$\mathbf{87.3}_{(0.8)}$	$\mathbf{90.3}_{(0.9)}$	$\mathbf{85.6}_{(1.5)}$	$\mathbf{91.4}_{(1.5)}$	$\mathbf{86.8}_{(2.9)}$	$\mathbf{22.7}_{(14.1)}$
relative ↑	+0.7	+1.9	+0.3	+0.0	+0.9	+0.2	+2.0	+13.4
	MNLI (acc)	**MNLI-mm** (acc)	**SNLI** (acc)	**QNLI** (acc)	**RTE** (acc)	**MRPC** (F1)	**QQP** (F1)	**STS-B** (Pear.)
PET	$68.3_{(2.3)}$	$70.5_{(1.9)}$	$\mathbf{77.2}_{(3.7)}$	$64.5_{(4.2)}$	$69.1_{(3.6)}$	$74.5_{(5.3)}$	$65.5_{(5.3)}$	$71.0_{(7.0)}$
PET+SCL	$69.5_{(3.2)}$	$71.3_{(3.1)}$	$\mathbf{77.2}_{(2.9)}$	$69.2_{(2.7)}$	$69.3_{(4.1)}$	$75.8_{(4.0)}$	$66.7_{(3.7)}$	$71.6_{(6.5)}$
relative ↑	+1.2	+0.8	+0.0	+4.7	+0.2	+1.3	+1.2	+0.6
PET+RGL	$\mathbf{70.8}_{(2.3)}$	$\mathbf{72.7}_{(1.9)}$	$\mathbf{77.5}_{(1.7)}$	$\mathbf{70.3}_{(1.7)}$	$\mathbf{69.7}_{(2.6)}$	$\mathbf{77.0}_{(6.7)}$	$\mathbf{68.8}_{(1.8)}$	$\mathbf{72.5}_{(6.2)}$
relative ↑	+2.5	+2.2	+0.3	+5.8	+0.6	+2.5	+3.3	+1.5

Table 1: Test performance obtained on GLUE variant (Gao et al., 2021). The evaluation metrics follow Wang et al. (2018). The best results (according to the pairwise t-test with 95% confidence) are highlighted in bold.

mance gain. In particular, RGL can improve the performance of PET by 2.38 on average, while SCL only improves PET by 1.46 on average. Moreover, RGL obtains more stable results as the variances are smaller than the others.

Model analysis. Figure 2(a) plots the effect of varying the number of labeled training samples. As shown, all methods obtain better performance given more training samples, while PET+RGL consistently outperforms the others. We further consider different ways of obtaining $\hat{e}_{ij}$ in (4): (i) **w/ $\mathbf{h}^{[\mathbf{CLS}]}$** which sets $\hat{e}_{ij} = \cos(\mathbf{h}_i^{[\text{CLS}]}, \mathbf{h}_i^{[\text{CLS}]})$ where $\cos(\cdot,\cdot)$ denotes the cosine similarity function; (ii)**w/ $\mathbf{h}^{[\mathbf{MASK}]}$** which sets $\hat{e}_{ij} = \cos(\mathbf{h}_i^{[\text{MASK}]}, \mathbf{h}_i^{[\text{MASK}]})$; and (iii) **w/ $\hat{\mathbf{y}}$** which is the one adopted in RGL and sets $\hat{e}_{ij} = \cos(\hat{\mathbf{y}}_i, \hat{\mathbf{y}}_j)$. Results in Figure 2(b) show that RGL outperforms the others. This validates that class prediction carries more relevant information to discriminate samples.



(a) Effect of labeled samples. (b) Effect of estimating $\hat{e}_{ij}$.

Figure 2: Model analysis on SST-2 task.

Visualization. Figures 3 plots the t-SNE visualization of the learned sample embeddings. It apparently shows that, when combining RGL with both fine-tuning and PET, the distances of deep representations between any two inter-class samples are much longer than the intra-class distances. Furthermore, PET+RGL can separate two classes of samples with clear margin while concentrating samples of every class closely to the center of the group, resulting in better discrimination capability.

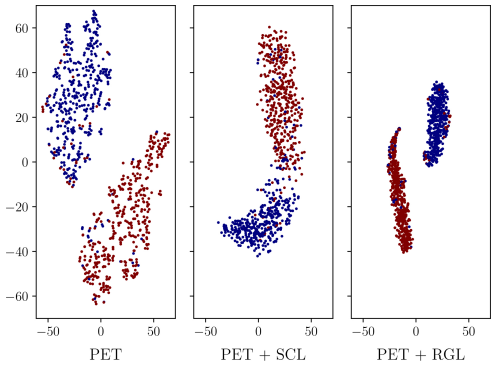


Figure 3: t-SNE visualization on SST-2 task.

5 Conclusion

We present RGL to fully exploit the limited supervised information of few-shot natural language understanding tasks. During learning, RGL constructs batch-wise relation graphs based on label consistency between samples, and explicitly tunes the pre-trained language models to solve the resultant node classification and link prediction problems. In this paper, we provide one way of relation graph learning. This can be further extended to broader applications where other ways of relation graph learning worth trying. In addition, one can explore how to avoid the interference of noisy samples.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, Chris 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 *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901.

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, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online. Association for Computational Linguistics.

Beliz Gunel, Jingfei Du, Alexis Conneau, and Veselin Stoyanov. 2020. Supervised contrastive learning for pre-trained language model fine-tuning. In *International Conference on Learning Representations*.

Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. 2021. WARP: Word-level Adversarial ReProgramming. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4921–4933, Online. Association for Computational Linguistics.

Jongmin Kim, Taesup Kim, Sungwoong Kim, and Chang D Yoo. 2019. Edge-labeling graph neural network for few-shot learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11–20.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *International Conference on Learning Representations*.

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

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

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

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

Timo Schick and Hinrich Schütze. 2021a. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269, Online. 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. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2339–2352, Online. Association for Computational Linguistics.

Derek Tam, Rakesh R Menon, Mohit Bansal, Shashank Srivastava, and Colin Raffel. 2021. Improving and simplifying pattern exploiting training. *arXiv preprint arXiv:2103.11955*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. *arXiv preprint arXiv:1905.00537*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q Weinberger, and Yoav Artzi. 2021. Revisiting few-sample BERT fine-tuning. In *International Conference on Learning Representations*.

Appendix: More Experimental Results

All the experiments are conducted on a 32GB NVIDIA Tesla V100 GPU.

In addition to PET (Schick and Schütze, 2021b), we also combine the proposed RGL with other prompt-based tuning strategies. As for baselines to incorporate RGL, we take P-tuning[4] (Liu et al., 2021) as the representative for joint prompt and PLMtuning, and WARP[5] (Hambardzumyan et al., 2021) as the representative for prompt tuning with a fixed PLM. WARP is only evaluated on CB and RTE as in the original paper.

As both P-tuning and WARP use SuperGLUE (Wang et al., 2019) variant proposed by Schick and Schütze (2021b) to evaluate the few-shot performance and use ALBERT[6] (Lan et al., 2020) as the PLM, we adopt them for fairness. Schick and Schütze (2021b) provide one training set which consists 32 samples per task and a testing set. Schick and Schütze (2021b) also use unlabeled samples, which are not used in this paper. Following Liu et al. (2021), a development set consisting of 32 samples per task are randomly drawn for model selection. As only one split is provided, we initialize the parameter with five random seeds and report the averaged results over five runs.

	BoolQ (acc)	**MultiRC** (EM)	**MultiRC** (F1a)	**WiC** (acc)	**WSC** (acc)
P-tuning	$75.2_{(5.2)}$	$32.1_{(1.0)}$	$74.9_{(1.9)}$	$55.3_{(1.5)}$	$80.8_{(2.5)}$
+RGL	$\mathbf{77.4}_{(0.8)}$	$\mathbf{33.5}_{(0.2)}$	$\mathbf{75.6}_{(1.2)}$	$\mathbf{57.3}_{(2.9)}$	$\mathbf{81.7}_{(1.0)}$
relative ↑	+2.2	+1.4	+0.7	+2.0	+0.9
	CB (acc)	**CB** (F1)	**RTE** (acc)	**COPA** (acc)	
P-tuning	$87.5_{(3.0)}$	$82.1_{(6.0)}$	$74.7_{(1.0)}$	$82.3_{(2.5)}$	
+RGL	$\mathbf{88.1}_{(2.1)}$	$\mathbf{84.2}_{(2.3)}$	$\mathbf{75.5}_{(1.3)}$	$\mathbf{83.7}_{(5.1)}$	
relative ↑	+0.6	+2.1	+0.7	+1.4	
WARP	$82.2_{(3.0)}$	$77.5_{(7.2)}$	$72.8_{(0.5)}$		
+RGL	$\mathbf{84.3}_{(2.1)}$	$\mathbf{80.5}_{(4.7)}$	$\mathbf{73.2}_{(1.0)}$		
relative ↑	+2.1	+3.0	+0.4		

Table 2: Test performance obtained on SuperGLUE variant (Schick and Schütze, 2021b). The evaluation metrics follow Wang et al. (2019). The best results (according to the pairwise t-test with 95% confidence) are highlighted in bold.

Table 2 shows the results obtained on SuperGLUE variants. The results show that RGL can consistently boost the performance when it is combined with P-tuning and WARP.

[4] https://github.com/THUDM/P-tuning.
[5] https://github.com/YerevaNN/warp.
[6] https://huggingface.co/albert-xxlarge-v2.