NoisyTune: A Little Noise Can Help You Finetune Pretrained Language Models Better

Chuhan Wu[†] **Fangzhao Wu**[‡*] **Tao Qi**[†] **Yongfeng Huang**[†]
[†]Department of Electronic Engineering, Tsinghua University, Beijing 100084, China
[‡]Microsoft Research Asia, Beijing 100080, China
{wuchuhan15, wufangzhao, taoqi.qt}@gmail.com
yfhuang@tsinghua.edu.cn

Abstract

Effectively finetuning pretrained language models (PLMs) is critical for their success in downstream tasks. However, PLMs may have risks in overfitting the pretraining tasks and data, which usually have gap with the target downstream tasks. Such gap may be difficult for existing PLM finetuning methods to overcome and lead to suboptimal performance. In this paper, we propose a very simple yet effective method named *NoisyTune* to help better finetune PLMs on downstream tasks by adding some noise to the parameters of PLMs before finetuning. More specifically, we propose a matrix-wise perturbing method which adds different uniform noises to different parameter matrices based on their standard deviations. In this way, the varied characteristics of different types of parameters in PLMs can be considered. Extensive experiments on both GLUE English benchmark and XTREME multilingual benchmark show *NoisyTune* can consistently empower the finetuning of different PLMs on different downstream tasks.

1 Introduction

In recent years, pretrained language models (PLMs) have achieved huge success in NLP (Qiu et al., 2020). Many PLMs such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019) and UniLM (Dong et al., 2019) which are pretrained from large-scale unlabeled corpus in a self-supervised way, have significantly improve various downstream tasks such as reading comprehension (Xu et al., 2019), machine translation (Brown et al., 2020), text classification (Bao et al., 2020), dialog (Wu et al., 2020) and recommendation (Wu et al., 2021) by finetuning on these tasks.

How to effectively finetune PLMs to better empower downstream tasks is an important research problem (Zheng et al., 2021). Many existing NLP methods usually directly finetune PLMs with the labeled data in downstream tasks (Sun et al., 2019). Only a few works explore more effective and robust PLM finetuning methods (Chen et al., 2020; Lee et al., 2020; Aghajanyan et al., 2021; Zhang et al., 2021; Xu et al., 2021). For example, Chen et al. (2020) proposed RecAdam that adds a penalty item to minimize the L_2 distance between the finetuned models and the pretrained models, where the penalty intensity is time-variant during finetuning. Lee et al. (2020) proposed Mixout which randomly replaces part of the parameters in the finetuned model with their original weights in the PLMs. These PLM finetuning methods mainly focus on preventing PLMs from overfitting the limited labeled data in downstream tasks. Besides the overfitting of downstream task data, a rarely studied problem is that the PLMs usually overfit the pretraining tasks and data (Qi et al., 2020), which may have significant gap with the downstream task and data. It is not easy for existing PLM finetuning methods to overcome such gap (Roberts et al., 2020), which may lead to suboptimal performance especially when labeled data in downstream tasks is insufficient.

In order to handle this problem, in this paper we propose a very simple yet effective method named *NoisyTune*, which can help better finetune PLMs for downstream tasks. Different from the

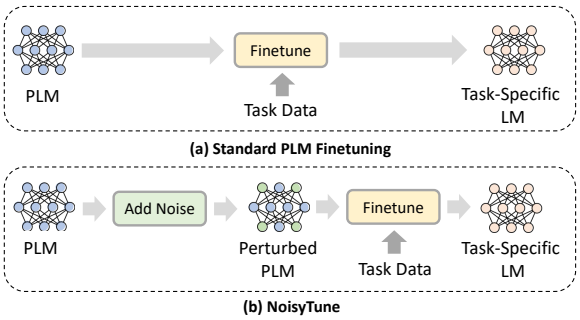

Figure 1: Schematic comparisons between standard PLM finetuning and our *NoisyTune*.

*Corresponding author.

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics
Volume 2: Short Papers, pages 680 - 685

standard finetuning paradigm (Fig. 1 (a)) which directly finetunes PLMs on the downstream task data, the key idea of *NoisyTune* is to add a small amount of noise to perturb PLMs parameters before finetuning (Fig. 1 (b)). It can help prevent PLMs from overfitting the tasks and data in the pretraining stage, and reduce the gap between pretraining and downstream tasks. Since PLMs have different types of parameters which usually own different characteristics, in *NoisyTune* we use a matrix-wise perturbing method that adds uniform noise with different intensities to different parameter matrices according to their standard deviations for better adaptation. We conduct extensive experiments on two widely used NLP benchmarks, namely, GLUE (Wang et al., 2018) for English language understanding and XTREME (Hu et al., 2020) for multilingual language understanding. The results show *NoisyTune* can empower the finetuning of different PLMs on many different downstream NLP tasks to consistently achieve better performance. In addition, the results show *NoisyTune* can be easily combined with many existing PLM finetuning methods and further improve their performance.

2 NoisyTune

The goal of *NoisyTune* is for more effective finetuning of PLMs on downstream tasks. The motivation of *NoisyTune* is that PLMs are well pretrained on some unlabeled corpus with some self-supervision tasks, and they may overfit these pretraining data and tasks (Qi et al., 2020), which usually have gap with the downstream task and data. It may be difficult for PLMs to effectively adapt to downstream tasks especially when labeled data in these tasks are limited, which is usually the case. Motivated by the dueling bandits mechanism (Yue and Joachims, 2009) that adds randomness to the model for exploration, as shown in Fig. 1, we propose to add some noise to the parameters of PLMs before finetuning them on downstream tasks to do some "exploration" in parameter space and reduce the risk of overfitting the pretraining tasks and data.

PLMs usually have different kinds of parameter matrices, such as query, key, value, and feedforward network matrices (Devlin et al., 2019). Different parameter matrices in the PLMs usually have different characteristics and scales. For example, some researchers found that the self-attention parameters and the feed-forward network parameters in Transformers have very different properties, such as rank and density (Wang et al., 2020). Thus, adding unified noise to all parameter matrices in PLMs may not be optimal for keeping their good model utility. To handle this challenge, we propose a matrix-wise perturbing method that adds noise with different intensities to different parameter matrices according to their variances. Denote the parameter matrices (or scalars/vectors) in a PLM as $[\mathbf{W}_1, \mathbf{W}_2, ..., \mathbf{W}_N]$, where N is the number of parameter matrix types. Denote the perturbed version of the parameter matrix $\mathbf{W}_i$ as $\tilde{\mathbf{W}}_i$, which is computed as follows:

$$\tilde{\mathbf{W}}_i = \mathbf{W}_i + U(-\frac{\lambda}{2}, \frac{\lambda}{2}) * std(\mathbf{W}_i), \quad (1)$$

where std stands for standard deviation. The function $U(a, b)$ represents uniform distribution noise ranged from a to b, and λ is a hyperparameter that controls the relative noise intensity.[1] We can see that in *NoisyTune* parameters in PLMs with higher variance will be added with stronger noise. In addition, in some PLMs there are some constant matrices, such as token type embeddings in RoBERTa (Liu et al., 2019). They will not be perturbed because their standard deviation is 0. It can ensure that these constant matrices will not be accidentally activated by additional noise.

NoisyTune is a simple and general plug-and-play technique that can be applied to the finetuning of any PLM on any task, simply by inserting the following PyTorch-style code before finetuning:

```
for name,para in model.named_parameters():
    model.state_dict[name][:] +=
    (torch.rand(para.size())-0.5)
    *noise_lambda*torch.std(para)
```

3 Experiments

3.1 Datasets and Experimental Settings

We conduct extensive experiments on two widely used benchmarks for PLM evaluation. The first one is GLUE (Wang et al., 2018), which is a benchmark for English language understanding that contains different tasks like natural language inference, sentiment analysis and sentence similarity evaluation. The second one is XTREME (Hu et al., 2020), which is a benchmark for multilingual language understanding. It covers 40 languages and contains

[1] Note that $U(a, b)$ is a matrix with the same shape with $\mathbf{W}_i$ rather than a scalar.

four groups of tasks, including sentence classification, structured prediction, sentence retrieval and question answering. More details of these benchmarks can refer to their original papers and official websites. Since the test labels of GLUE are not released, following (Bao et al., 2020) we report results on the dev set of GLUE. The XTREME results are evaluated on the test set. The hyperparameter λ is 0.15 on GLUE and is 0.1 on XTREME. The searching range of hyperparameters in our work are listed in Table 1.

Hyperparameters	Range
Learning rate	{7e-6, 1e-5, 2e-5, 3e-5}
Epoch	{3, 5, 7, 10, 15, 20}
Batch size	{8, 16, 32}
Noisy intensity	{0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3}

Table 1: Searching ranges of different hyperparameters in our experiments.

Following (Zheng et al., 2021), in sentence retrieval tasks we first train the models on the XNLI dataset, and then use the average of token representations produced by the hidden layer that yields the best performance. In order not to harm the alignment of token embeddings across different languages, we do not add noise to the token embeddings in multilingual PLMs. We repeat experiments 5 times with different random seeds and report the average scores.

3.2 Performance Evaluation

On the GLUE benchmark, we compare the performance of directly finetuning the base version of BERT (Devlin et al., 2019), XLNET (Yang et al., 2019), RoBERTa (Liu et al., 2019) and ELECTRA (Clark et al., 2020) with that of finetuning them after applying *NoisyTune*. On the XTREME benchmark, we compare the performance of directly finetuning both base and large versions of XLM-R (Conneau et al., 2020) with that of their variants obtained by applying *NoisyTune*. The results on these two benchmarks are shown in Tables 2 and 3, respectively. On the XTREME datasets, we report two types of results. The first one is zero-shot crosslingual transfer from English to other languages, and the second one is learning models on both English and translated data.

According to these results, *NoisyTune* can consistently improve the performance of different PLMs on different tasks in both English and multilingual settings. In addition, the performance improvement brought by *NoisyTune* is usually larger on relatively small datasets (e.g., RTE, CoLA and WNLI). These results indicate that when labeled data in downstream tasks is insufficient, it is quite difficult to effectively finetune PLMs starting from the original parameters which usually overfit the pretraining tasks and data. The experimental results validate that *NoisyTune* can properly perturb PLMs with a little noise to explore different parameter spaces and reduce the overfitting problem, making PLMs easier to be adapted to downstream tasks.

3.3 Which Noise to Use and How?

In this section we study which kind of noise is more suitable for *NoisyTune*. In addition, we explore whether our proposed matrix-wise perturbing method is better than using a unified global noise for all model parameters in PLMs. We compare five methods, including (1) *NoisyTune* without any noise; (2) *NoisyTune* with a global Gaussian noise; (3) *NoisyTune* with a global uniform noise; (4) *NoisyTune* with matrix-wise Gaussian noise; (5) *NoisyTune* with matrix-wise uniform noise. The results on GLUE are shown in Fig. 2, and the results on XTREME show similar patterns. We find that adding global noise with the same distribution to all the PLM parameters will harm the model performance. This is because different parameter matrices in PLMs have very different distributions and characteristics (Wang et al., 2020). Simply adding a unified global noise to all the parameter matrices is not optimal. The results show that matrix-wise noise is a much better choice, since the different characteristics of different parameter matrices can be taken into consideration. In addition, we find an interesting phenomenon that adding uniform noise is better than Gaussian noise. This may be because Gaussian noise has wider ranges and some extreme values may affect the model performance. Thus, we use matrix-wise uniform noise in *NoisyTune*.

3.4 Combination with Existing PLM Finetuning Methods

From Fig. 1, it is very clear that *NoisyTune* is independent of the specific PLM finetuning method, since it is applied at the stage before finetuning PLM on the task-specific data. Thus, it is very easy to combine *NoisyTune* with any kind of existing PLM finetuning method. In this section, we explore whether *NoisyTune* has the potential to empower the existing PLM finetuning techniques to achieve better performance. Here we select two

Model	MNLI Acc	QNLI Acc	QQP Acc	RTE Acc	SST Acc	MRPC Acc	CoLA MCC	STS PCC	WNLI Acc	Avg.
BERT	84.4	91.5	90.9	67.7	93.0	87.1	58.1	89.4	54.4	79.6
BERT+NoisyTune	84.7	91.8	91.2	68.8	93.4	88.0	59.0	90.1	56.1	80.3
XLNET	86.6	91.6	91.2	72.9	94.4	88.1	59.6	89.6	57.5	81.3
XLNET+NoisyTune	86.9	91.9	91.4	73.8	94.7	88.6	60.1	90.0	58.6	81.8
RoBERTa	87.5	92.7	91.7	77.1	94.5	90.1	62.9	90.8	59.2	82.9
RoBERTa+NoisyTune	87.8	93.1	91.9	78.8	94.9	90.6	63.6	91.1	60.3	83.6
ELECTRA	88.4	92.9	91.7	75.2	94.9	88.2	64.2	90.1	62.0	83.1
ELECTRA+NoisyTune	88.7	93.2	92.1	76.4	95.2	88.7	64.9	90.5	63.4	83.7

Table 2: Results of different methods on the GLUE dev set.

Model	Sentence Pair		Structured Prediction		Sentence Retrieval		Question Answering			
	XNLI	PAWS-X	POS	NER	BUCC	Tatoeba	XQuAD	MLQA	TyDiQA	
Metrics	Acc	Acc	F1	F1	Acc	Acc	F1/EM	F1/EM	F1/EM	Avg.
Fine-tune multilingual model on English training set (Cross-lingual Transfer)										
XLM-R_{base}	74.8	84.8	75.5	61.6	77.6	73.8	71.9/56.6	65.2/47.0	55.5/38.4	70.0
XLM-R_{base}+NoisyTune	75.2	85.1	76.0	62.1	78.2	74.5	72.3/57.1	65.5/47.4	56.0/39.2	70.5
XLM-R_{large}	79.0	86.3	72.7	62.3	79.2	76.0	76.2/60.4	71.4/53.0	65.0/45.0	72.4
XLM-R_{large}+NoisyTune	79.3	86.5	73.5	63.2	79.9	76.8	76.7/61.0	71.9/53.6	65.4/45.6	73.0
Fine-tune multilingual model on all training sets (Translate-Train-All)										
XLM-R_{base}	78.5	88.2	76.2	62.6	79.6	79.4	75.0/61.5	67.8/50.1	63.8/47.6	73.3
XLM-R_{base}+NoisyTune	78.9	88.6	76.8	63.1	80.0	79.8	75.4/61.8	68.0/50.4	64.1/48.1	73.7
XLM-R_{large}	82.3	90.3	77.3	67.3	82.5	82.7	80.0/65.6	72.9/54.4	66.3/47.6	76.4
XLM-R_{large}+NoisyTune	82.5	90.5	77.8	67.9	82.9	83.0	80.4/66.1	73.3/54.9	66.8/48.2	76.8

Table 3: Results of different methods on the XTREMRE test set.

well-known PLM finetuning for experiments, i.e., RecAdam (Chen et al., 2020) and Mixout (Lee et al., 2020). The experimental results are summarized in Fig. 3. We find that combining *NoisyTune* with existing PLM finetuning techniques can further improve their performance. This is because *NoisyTune* aims to address the overfitting of pretraining signals while these methods aim to prevent overfitting in downstream tasks. Thus, *NoisyTune* and these PLM finetuning methods are complementary, and they can be empowered by *NoisyTune* to achieve better performance.

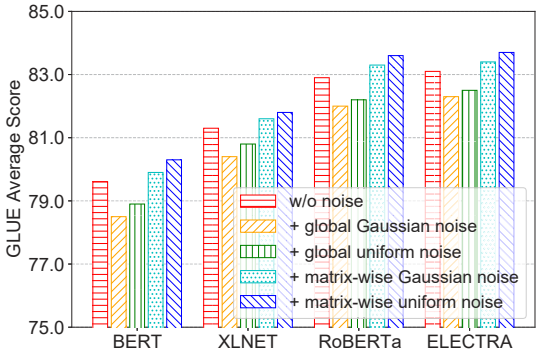

Figure 2: Different noise types and perturbing methods.

3.5 Empirical Analysis of NoisyTune

Next, we empirically analyze why *NoisyTune* can help PLM finetuning. We compare the accuracy of BERT with and without *NoisyTune* finetuned with different percentage of samples on the MRPC dataset.[2] The results are shown in Fig. 4. We find *NoisyTune* can consistently improve PLMs under different amounts of data, especially when less training data is used. This is because the perturbed PLMs may have lower risks of overfitting the pretraining tasks and have better generalization abilities, which is especially beneficial for finetuning PLMs on downstream task with limited data.

To further study the impact of *NoisyTune* on PLM finetuning, we show the relative changes of the L_1-norms of different kinds of parameters in the BERT model during finetuning on the MRPC dataset in Fig. 5.[3] Since the noise we added to PLMs in *NoisyTune* is zero-mean uniform noise, the absolute parameter L_1-norm will not change too much. However, we can see that the relative change of L_1-norms becomes smaller when *NoisyTune* is applied, which indicates that the PLMs can find the (sub)optimal parameters for downstream

[2]We observe similar patterns on other datasets.

[3]The patterns on other datasets are similar.

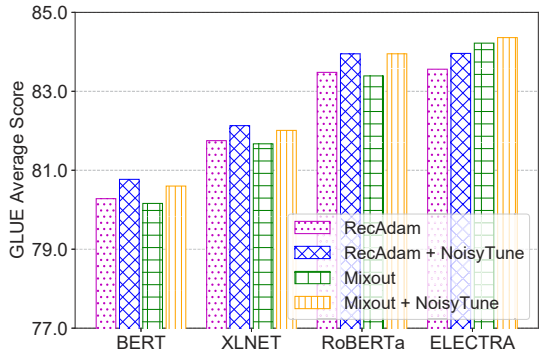

Figure 3: *NoisyTune* can empower many existing PLM finetuning methods to achieve better performance.

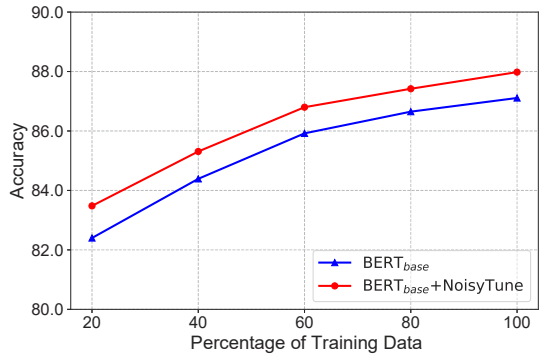

Figure 4: Influence of *NoisyTune* on finetuning.

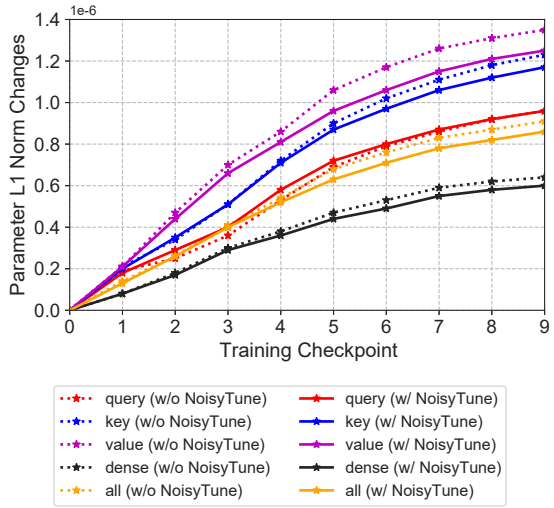

Figure 5: Relative changes of the L_1-norm of different types of parameters in PLM during finetuning.

tasks more easily. This result validates directly finetuning PLMs may need more updates to adapt to downstream tasks, which is due to the overfitting of pretraining tasks, and *NoisyTune* can provide a simple way to alleviate this problem and help finetune PLMs on downstream tasks more effectively.

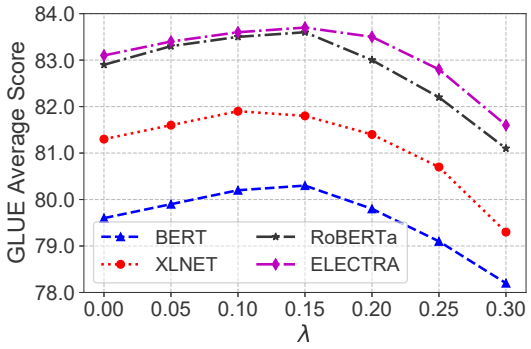

Figure 6: Influence of noise intensity λ.

3.6 Hyperparameter Analysis

We study the influence of the most important hyperparameter in *NoisyTune*, i.e., λ, which controls the relative noise intensity. The average GLUE scores w.r.t. different λ values are shown in Fig. 6. We find that when λ is too small or too large, the performance is not optimal. This is because when λ is too small, it is difficult for PLMs to do parameter space exploration and overcome the overfitting problem. While when λ is too large, the useful pretrained knowledge in PLMs may be overwhelmed by random noise. Values between 0.1 and 0.15 are more suitable for *NoisyTune* on the GLUE datasets.

4 Conclusion

In this paper, we propose a very simple but effective method named *NoisyTune*, which can help better finetune PLMs on downstream tasks by adding a little noise to them before finetuning. In *NoisyTune*, we propose a matrix-wise perturbing method that adds noise with different intensities to different kinds of parameter matrices in PLMs according to their variances. *NoisyTune* is a very general method, and is PLM model agnostic, downstream task agnostic, and finetuning method agnostic. Extensive experiments on both monolingual GLUE benchmark and multilingual XTREME benchmark demonstrate *NoisyTune* can consistently empower the finetuning of different PLMs on various downstream tasks to achieve better performance.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant numbers U1936216, U1936208, and 61862002, and the research initiation project of Zhejiang Lab (No. 2020LC0PI01).

References

Armen Aghajanyan, Akshat Shrivastava, Anchit Gupta, Naman Goyal, Luke Zettlemoyer, and Sonal Gupta. 2021. Better fine-tuning by reducing representational collapse. In *ICLR*.

Hangbo Bao, Li Dong, Furu Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Jianfeng Gao, Songhao Piao, Ming Zhou, et al. 2020. Unilmv2: Pseudo-masked language models for unified language model pre-training. In *ICML*, pages 642–652. PMLR.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *NeurIPS*, 33:1877–1901.

Sanyuan Chen, Yutai Hou, Yiming Cui, Wanxiang Che, Ting Liu, and Xiangzhan Yu. 2020. Recall and learn: Fine-tuning deep pretrained language models with less forgetting. In *EMNLP*, pages 7870–7881.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre-training text encoders as discriminators rather than generators. In *ICLR*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *ACL*, pages 8440–8451.

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.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *NIPS*, pages 13063–13075.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. Xtreme: A massively multilingual multi-task benchmark for evaluating cross-lingual generalisation. In *ICML*, pages 4411–4421. PMLR.

Cheolhyoung Lee, Kyunghyun Cho, and Wanmo Kang. 2020. Mixout: Effective regularization to finetune large-scale pretrained language models. In *ICLR*.

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*.

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. Prophetnet: Predicting future n-gram for sequence-to-sequencepre-training. In *EMNLP Findings*, pages 2401–2410.

Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, pages 1–26.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *EMNLP*, pages 5418–5426.

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

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 *BlackboxNLP*, pages 353–355.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*.

Chien-Sheng Wu, Steven CH Hoi, Richard Socher, and Caiming Xiong. 2020. Tod-bert: Pre-trained natural language understanding for task-oriented dialogue. In *EMNLP*, pages 917–929.

Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. 2021. Empowering news recommendation with pre-trained language models. In *SIGIR*, pages 1652–1656. ACM.

Hu Xu, Bing Liu, Lei Shu, and S Yu Philip. 2019. Bert post-training for review reading comprehension and aspect-based sentiment analysis. In *NAACL-HLT*, pages 2324–2335.

Runxin Xu, Fuli Luo, Zhiyuan Zhang, Chuanqi Tan, Baobao Chang, Songfang Huang, and Fei Huang. 2021. Raise a child in large language model: Towards effective and generalizable fine-tuning. In *EMNLP*, pages 9514–9528.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*, pages 5753–5763.

Yisong Yue and Thorsten Joachims. 2009. Interactively optimizing information retrieval systems as a dueling bandits problem. In *ICML*, pages 1201–1208.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q Weinberger, and Yoav Artzi. 2021. Revisiting few-sample bert fine-tuning. In *ICLR*.

Bo Zheng, Li Dong, Shaohan Huang, Wenhui Wang, Zewen Chi, Saksham Singhal, Wanxiang Che, Ting Liu, Xia Song, and Furu Wei. 2021. Consistency regularization for cross-lingual fine-tuning. In *ACL-IJCNLP*, pages 3403–3417.