# Enhancing Low-Resource LLMs Classification with PEFT and Synthetic Data

**Parth Patwa**[1], **Simone Filice**[2*], **Zhiyu Chen**[1],
**Giuseppe Castellucci**[1], **Oleg Rokhlenko**[1], **Shervin Malmasi**[1]
[1]Amazon, USA
[2]Technology Innovation Institute, Israel
[1]{parthptw, zhiyuche, giusecas, olegro, malmasi}@amazon.com, [2]simone.filice@tii.ae

**Abstract**

Large Language Models (LLMs) operating in 0-shot or few-shot settings achieve competitive results in Text Classification tasks. In-Context Learning (ICL) typically achieves better accuracy than the 0-shot setting, but it pays in terms of efficiency, due to the longer input prompt. In this paper, we propose a strategy to make LLMs as efficient as 0-shot text classifiers, while getting comparable or better accuracy than ICL. Our solution targets the low resource setting, i.e., when only 4 examples per class are available. Using a single LLM and few-shot real data we perform a sequence of generation, filtering and Parameter-Efficient Fine-Tuning steps to create a robust and efficient classifier. Experimental results show that our approach leads to competitive results on multiple text classification datasets.

**Keywords:** LoRA, PEFT, LLMs, Few-Shot Learning, Text Classification

## 1. Introduction

Recent years have been characterized by a paradigm shift in text classification. Large Language Models (LLMs) offer valid alternatives to the traditional approach of fine-tuning pre-trained models (e.g., BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019)) on annotated datasets. In-Context Learning (ICL) is a first option, where a LLM learns how to solve a task by simply observing a few examples provided in its prompt (i.e., without any fine-tuning stage) (Brown et al., 2020). Another alternative is the 0-shot setting, where the LLM directly solves a task by simply following the provided instructions (i.e., without any example). Instruction-fine tuned models like Flan-T5 (Chung et al., 2022), Instruct-GPT (Ouyang et al., 2022), or ChatGPT excel in this setting. The advantage of these two alternatives with respect to the traditional approach is that they can be used to bootstrap a model when data is scarce or totally absent.

The 0-shot setting is generally more appealing since it does not require any data, however, the few-shot ICL setting typically leads to better results by only leveraging a small number of samples (e.g., less than 10 examples). Obtaining a few annotated data might not be a significant drawback, as a practitioner with moderate domain knowledge can readily create a small number of examples manually. However, a major disadvantage is the higher computational cost, latency, and memory requirements associated with the longer prompt, which needs to contain illustrative examples. A possible solution to leverage the few available examples without incurring the ICL inference costs would be to use them for fine-tuning the LLM, which is possible by using some Parameter-Efficient Fine Tuning (PEFT) techniques (Liu et al., 2022c; Lester et al., 2021; Hu et al., 2021; Liu et al., 2021). Unfortunately, as we will demonstrate in the experimental section, PEFT is not effective with very few examples, due to under-fitting or over-fitting phenomena.

In this paper, we propose a solution to the low-resource PEFT for text classification with LLMs by defining a framework that enables faster, cheaper and more accurate inference than ICL in such a scenario. We hypothesize that LLMs already have some knowledge of how to solve a classification task, but the sub-optimal usage of the available resources (i.e., the few-shot examples) results in low PEFT performance under the low-resource setting. On the contrary, LLMs typically excel in generation tasks, hence we frame an auxiliary data augmentation task that we use to unlock the LLM classification capabilities. Our method consists of three steps. First, we use the LLM to generate synthetic examples for each class of the text classification task we target. Then, we use the same LLM in the ICL setting to classify the examples and clean the data by removing label-inconsistent generated examples. Finally, we fine-tune the LLM with PEFT using the generated and cleaned data. Our experiments show that the resulting classifier reaches accuracy levels comparable to or better than the ICL setting in three different text classification tasks while being a lot more efficient ($\sim$2x to 5x speed boost). In these generate-filter-train stages we always use the same LLM to demonstrate that what leads to a good accuracy is just a better usage of the few available examples and

*Work done while at Amazon.



*LREC-COLING 2024*, pages 6017–6023
20-25 May, 2024. 

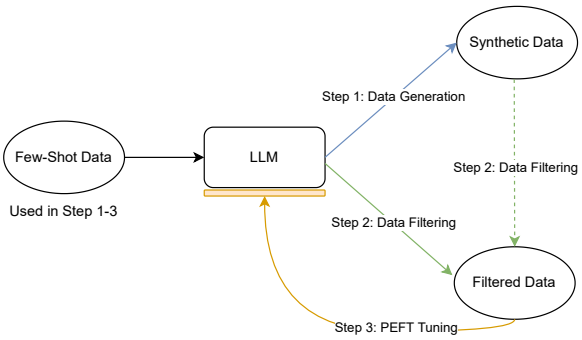


Figure 1: The overview of our method. First, very few real data points are used to generate synthetic data using ICL. Then, the synthetic data is filtered using ICL by LLM again. Finally, the filtered data and the real data are combined to train the LLM using LoRA.

not the employment of any other resource (e.g., another LLM) which might bring additional knowledge to solve the task.

The rest of the paper is organized as it follows: in Section 2 we discuss the related works. In Section 3 we present our method, while in Sections 4 and 5 we discuss the experimental setting and the results, respectively. Finally, Section 6 derives the conclusions.

## 2. Related Work

LLMs demonstrate impressive capabilities to solve Natural Language Understanding tasks. For instance, classification tasks can be approached in a generative way, i.e., by asking the LLM to generate the class name associated with an input example. 0-shot and ICL are two variants of this paradigm. Recent models (e.g., GPT-3, (Brown et al., 2020), Flan-T5 (Chung et al., 2022), ChatGPT (Ouyang et al., 2022), Falcon (Penedo et al., 2023) or Vicuna (Zheng et al., 2023)) reach impressive results in both settings, and researchers started considering using LLMs as annotators (Rosenbaum et al., 2022; Zhu et al., 2023; He et al., 2023). For instance, Rosenbaum et al. (2022) propose a method that uses LLMs to generate and annotate data for Intent Classification and Slot Filling. He et al. (2023) propose a two-step approach where they first use ChatGPT to generate a few-shot Chain-of-Thought prompt, which they then use to annotate unlabeled data. Results are competitive with human annotators, but their classification procedure is relatively slow since the LLM is invoked twice with prompts that need to contain both examples and explanations. Conversely, we propose a solution whose computational complexity at inference time corresponds to the 0-shot setting case. Even if these results are impressive, they still might not reach the state-of-the-art performance achievable when LLMs are fully fine-tuned on large data. Fine-tuning LLMs is extremely expensive, but a viable solution is offered by Parameter Efficient Fine-Tuning (PEFT) techniques (Liu et al., 2022c; Lester et al., 2021; Hu et al., 2021; Liu et al., 2021), where a pre-trained model is fine-tuned by only updating a small number (e.g., 0.01%) of added or selected parameters. These methods report results that match the performance of full fine-tuning when large training datasets are available. On the contrary, there has been relatively little focus on (parameter-efficient) fine-tuning in low-resource settings. Our paper targets this scenario, as we assume we can access only a few annotated examples (e.g., four per class) and no unlabeled data. A work operating in a similar setting is Liu et al. (2022b), where the authors propose a novel PEFT technique that is demonstrated to work well in low resource settings when the PEFT weights are pre-trained and multiple tasks are trained in parallel. We differ from their work as we do not pre-train the PEFT weights and we target a single task at a time, without assuming (possibly related) data from other tasks is available. Relaxing this assumption is especially useful when dealing with very peculiar tasks not sharing similarities with other available datasets. Hence, to the best of our knowledge, we are the first to improve few-shot PEFT without additional resources (external datasets or models).

## 3. Methodology

We explore a low-resource setting where we have few training examples per class and no unlabeled data. ICL methods could achieve reasonable performance with few-shot samples but inference cost is high due to long prompts. PEFT methods like LoRA are known to be more efficient than ICL at inference. However, we find that LoRA performs worse than ICL in data-scarce settings (see Tab. 1). In this paper, we aim to explore the potential of combining the strengths of PEFT and ICL methods for achieving efficient and effective text classification. Hence we propose to augment the training data with synthetic data to better align the generation and classification capability of LLMs and to ensure that PEFT is performed on a decent amount of data. Our method has 3 steps: generate data, filter data, and train. An overview of our method is shown in Figure 1.

**Generation step:** Chavan et al. (2023) show that in a few-shot setting, the performance of PEFT significantly improves as the number of training samples per class increases. We also observe similar results in our initial experiments (presented in section 5. Further, since ICL performs well, we hypothesize that the model has the inherent knowledge to

| Model | Method | #real | #syn | SST2 | | TREC | | AG News | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | acc | inf. time | acc | inf. time | acc | inf. time |
| Vicuna7b | 0-shot | 0 | 0 | 0.55 | 0.27 | 0.16 | 0.28 | 0.36 | 0.5 |
| Vicuna7b | ICL | 4 | 0 | **0.95** | 0.6 | 0.60 | 0.9 | 0.75 | 2.5 |
| Vicuna7b | LoRA | 4 | 0 | 0.51 | 0.27 | 0.49 | 0.28 | 0.35 | 0.5 |
| Vicuna7b | LoRA | 25 | 0 | 0.89 | 0.27 | **0.84** | 0.28 | **0.86** | 0.5 |
| Vicuna7b | ours | 4 | 21 | 0.90 | 0.27 | 0.79 | 0.28 | 0.82 | 0.5 |
| Vicuna13b | 0-shot | 0 | 0 | 0.85 | 0.37 | 0.36 | 0.38 | 0.31 | 1.83 |
| Vicuna13b | ICL | 4 | 0 | 0.93 | 1.2 | 0.75 | 1.7 | 0.80 | 4.36 |
| Vicuna13b | LoRA | 4 | 0 | 0.84 | 0.37 | 0.62 | 0.38 | 0.64 | 1.83 |
| Vicuna13b | LoRA | 25 | 0 | 0.93 | 0.37 | **0.93** | 0.38 | 0.84 | 1.83 |
| Vicuna13b | ours | 4 | 21 | **0.93** | 0.37 | 0.81 | 0.38 | **0.86** | 1.83 |
| Vicuna7b | LoRA | Full | 0 | 0.97 | 0.27 | 0.98 | 0.28 | 0.95 | 0.5 |

Table 1: Accuracy results and inference times (in seconds) on three classification tasks. In bold the best performing method for the model in a data-scarce setting. "Full" data refers to the use of the entire available training data. #real and #syn refer to the number of real and synthetic examples per class used for training.

**Few examples of movie reviews having positive sentiment are given. Generate more positive reviews**
**Text:** [Positive review 1]
**Label:** Positive
**...**
**Text:** [Positive review 4]
**Label:** Positive
**Text:** [the model generates this]

Figure 2: An example of a prompt used for generating positive reviews for SST2 data. Four examples of the positive class are provided in the prompt.

**Classify the sentiment of the given movie review into Positive or Negative**
**Text:** [review 1]
**Label:** [Label 1]
**...**
**Text:** [review 8]
**Label:** [Label 8]
**Text:** [generated review]
**Label:** [Predicted Label]

Figure 3: An example of a prompt used for classifying the sentiment of a movie review. Four examples per class are given in the prompt in a random order.

solve the classification task and that the low PEFT results are due to sub-optimal usage of the available resources (the few shot examples). To fill this gap, we first use the LLM $\mathcal{L}$ in the ICL setting to generate synthetic data which we can use to augment the few shot examples at our disposal. We generate examples for each class in the targeted classification task. An example of the prompt we used in this step is shown in Figure 2.

**Filtering step:** We first apply a basic filtering step to discard duplicates and malformed generations (i.e., too short or too long texts). On manual inspection of the generated data, we found some label-inconsistent generations (i.e., data that are not valid examples of the class they should represent). We hypothesize this is due to hallucination. To identify and remove these cases, we classify the generated data using ICL with $\mathcal{L}$. The prompt used for this stage is similar to the one shown in Figure 3. If the predicted label does not match the intended label from the generation step, we discard the generated example. We repeat these generation-filtering steps until we produce $N$ new data samples for each class in the targeted classification task.

**Training step:** Finally, we use the filtered data along with the few (4 per class) real examples for the PEFT of the LLM $\mathcal{L}$ with LoRA. Note that $\mathcal{L}$ is used for all 3 steps, as we want to validate our hypothesis that $\mathcal{L}$ does not need additional knowledge to work in the PEFT setting, but only a more stable training process which can be guaranteed by the self-generated synthetic examples.

**Inference** Conversely to the ICL setting, the LLM does not use any example at inference time. Note that the three steps (generate, filter, train) are used only at training time, i.e., there is no impact on the inference latency.

## 4. Experiments

We use the Vicuna LLM (Zheng et al., 2023), which is based on LLaMA (Touvron et al., 2023). We selected Vicuna as it is the best-performing model


(a) Real data

(b) Synthetic data

Figure 4: Word clouds of the real and synthetic data belonging to the positive class in SST2.

among those whose weights were publicly available at the time of our experiments, according to the Chatbot Arena Leaderboard[1]. We experiment with 2 model sizes - Vicuna-7b and Vicuna-13b which have 7 billion and 13 billion parameters, respectively. We show results on the official test sets of 3 datasets - SST2 (sentiment analysis, 2 classes) (Pang and Lee, 2005), AG News (news topic classification, 4 classes) (Zhang et al., 2016), and TREC (question classification, 6 classes) (Li and Roth, 2002). To generate synthetic data, we use random sample decoding with temperature = 1.0, top_k = 50 and num_beams = 1 or 2.

For a fair comparison across models and methods, throughout our experiments, we do not tune the LoRA hyper-parameters (we set rank=8, alpha=32, and dropout=0.1). Similarly, we make minimal changes to the prompt for all ICL experiments and do not perform any prompt engineering.

The code is implemented using hugging face and PEFT library (Mangrulkar et al., 2022) with torch backend. All models are trained on 4 v100 GPUs of 16GB each. All LoRA models are trained for 100 epochs. The language modeling loss (next token prediction) is optimized using the Adam optimizer. Batch size is set to 2 for Vicuna-13b and 8 for Vicuna-7b. We run each experiment multiple times with different seeds and different few-shot examples. We observe negligible deviation across runs and report the average accuracy. We also report the results of several baselines, including 0-shot, ICL, and vanilla LoRA trained with different numbers of real examples. We also report LoRA trained with the full training set. This could be considered a potential upper-bound reachable in high resource settings and gives results comparable to the respective SoTA methods on the 3 datasets (Raffel et al., 2023; Cer et al., 2018; Yang et al., 2019).

[1] https://lmsys.org/blog/2023-05-25-leaderboard/

## 5. Results and Analysis

Table 1 reports the accuracy results and inference times on three tasks. As expected, the few-shot ICL is more accurate than the 0-shot method but significantly slower. Across model sizes and datasets, vanilla LoRA performs a lot worse than ICL in a few shot setting (4 real data points per class). Furthermore, it is even worse than or comparable to the 0-shot method in some cases. Conversely, our generate-filter-train approach is comparable to ICL on the SST2 data and considerably outperforms the ICL baseline on the other datasets. For example, when using Vicuna-7b model on the TREC data, our method gives 0.84 accuracy whereas ICL gives only 0.6 accuracy. Our method is also comparable to or slightly worse than training LoRA with 25 real samples per class. Note that the inference time of LoRA is lower and independent of training data size, whereas the inference time of few-shot ICL is higher and increases with an increase in training data (Liu et al., 2022a). As expected, the overall performance is generally better with Vicuna-13b than with Vicuna-7b.

**Ablation studies:** We train LoRA on the unfiltered synthetic data (by skipping the filtering step) and find that it achieved an accuracy of 0.68 on the TREC dataset. Using the filtered data the accuracy is 0.79, which shows the usefulness of the filtering step. We manually checked 125 unfiltered synthetic TREC samples and we found that 33 were incorrect (cases of hallucination). For example *"Who is called the Father of Geometry"* was incorrectly generated for the `location` class. However, only 5 samples were incorrect out of 125 examples after filtering. For the SST2 dataset, no sample was identified as incorrectly labeled, hence the filtering step did not affect the performance.

**Effect of Data Size:** We experiment with various data sizes. The results of Vicuna-7b on SST2 and TREC are shown in Figure 5 and 6, respectively. Increasing the size of real data is always beneficial; on the opposite, adding more synthetic data does

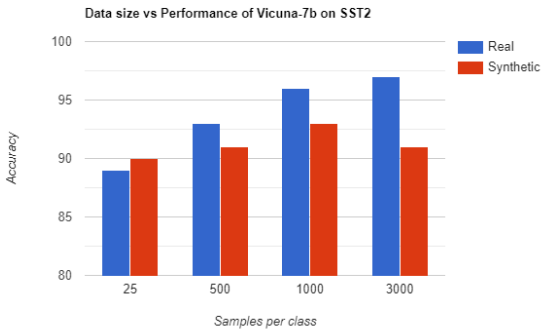


Figure 5: Data size vs performance of Vicuna-7b on SST2.

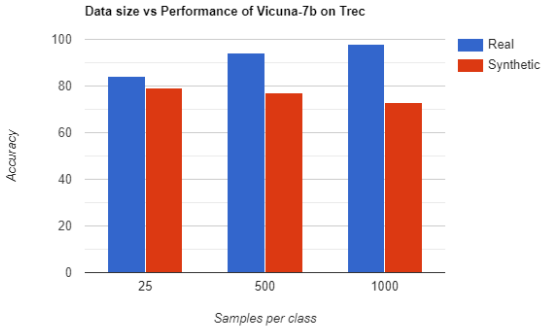


Figure 6: Data size vs performance of Vicuna-7b on TREC. Please note that in the real dataset, some classes have a limited number of instances (possibly lower than the values reported on x-axis). In such case, all instances of these classes are used.

always not provide a clear benefit. The main reason is the lack of diversity in the synthetic data.

**Data diversity:** Figure 7 shows the number of unique tri-grams vs. data size for real and synthetic SST2 data. We can see that for smaller data sizes, the diversity of the real data is comparable to that of synthetic data. However, as the data size increases, the real data diversity increases faster. This shows the difficulty of generating a large amount of diverse synthetic data with only 4 seed examples.

**Qualitative Data Analysis:** Figure 4 shows the word clouds of the real and synthetic examples belonging to the *positive* class of SST2 data. SST2 is a dataset of sentiment analysis of movie reviews. Hence, we can see that words like *film*, *movie* appear in both word clouds. Words that show positive sentiment like *entertaining*, *beautiful*, *funny* also appear in both the word clouds. Further, we see other positive words like *stunning*, *delightful* only in the synthetic data word cloud whereas subtle positive words like *compelling*, *solid* are seen only in the real data word cloud. From this, we can conclude

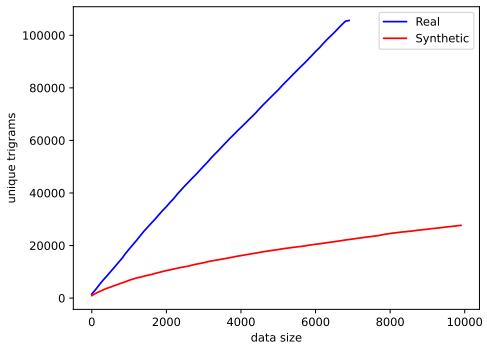


Figure 7: Data size vs unique trigrams in SST2.

that the synthetic data has a slightly different distribution and can capture the meaning of the positive class.

## 6. Conclusion and Future work

In this paper, we introduced a framework to make LLMs more efficient and effective text classifiers in very low-resource settings. The procedure we proposed consists of three steps. In the first step, the LLM is used to augment a very small training set with synthetic data; then, we adopt the LLM to classify the generated data and remove label-inconsistent examples; finally, we use the resulting data to fine-tune the LLM using LoRA. By running experiments on three different classification datasets we demonstrated how training LoRA using the self-generated synthetic data allowed our model to be comparable to or surpass several baselines operating in low resource settings, including 0-shot, ICL, and vanilla LoRA. In future work, we plan to improve the quality of the generated examples by promoting data diversity. Some strategies to improve data diversity include increasing attribute diversity (Yu et al., 2023), logit suppression (Chung et al., 2023) etc.

## 7. Limitations

Our method might not work on tasks that are particularly challenging and hard to catch with only a few examples. In this case, ICL is expected to fail, and similarly, our first two steps are expected to produce low-quality examples making the entire procedure ineffective. Another limitation is that our approach is fully based on LLMs and cannot be applied to low-resource languages where there is no existing LLM working well.

## 8. Ethics

Generating data using LLMs for text classification exposes the resulting classifier to the biases acquired during the LLM pre-training. In our framework, this phenomenon is potentially even amplified, as using the same LLM to generate and filter the data might reinforce such biases. Unfortunately, there is no one-size-fits-all solution for this problem. The biases are dependent on the application domain and on the data distribution to be generated. However, we encourage the readers to be very cautious about using this framework and to take the appropriate actions - for example, compiling a list of the potential biases specific for the target application domain and checking for those in the generated data - to mitigate the potential biases that may get reinforced when using a methodology similar to the one here presented.

## 9. References


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. *Advances in neural information processing systems*, 33:1877–1901.

Daniel Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder.

Arnav Chavan, Zhuang Liu, Deepak Gupta, Eric Xing, and Zhiqiang Shen. 2023. One-for-all: Generalized lora for parameter-efficient fine-tuning.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models.

John Chung, Ece Kamar, and Saleema Amershi. 2023. Increasing diversity while maintaining accuracy: Text data generation with large language models and human interventions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 575–593, Toronto, Canada. Association for Computational Linguistics.

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.

Xingwei He, Zhenghao Lin, Yeyun Gong, A-Long Jin, Hang Zhang, Chen Lin, Jian Jiao, Siu Ming Yiu, Nan Duan, and Weizhu Chen. 2023. Annollm: Making large language models to be better crowdsourced annotators.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *CoRR*, abs/2106.09685.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING*.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. 2022a. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. 2022b. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In *Advances in Neural Information Processing Systems*, volume 35, pages 1950–1965. Curran Associates, Inc.

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

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022c. P-tuning: Prompt tuning can be comparable to

fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68, Dublin, Ireland. Association for Computational Linguistics.

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*.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, and Sayak Paul. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL*. Association for Computational Linguistics.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. 2023. The refinedweb dataset for falcon llm: Outperforming curated corpora with web data, and web data only.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2023. Exploring the limits of transfer learning with a unified text-to-text transformer.

Andy Rosenbaum, Saleh Soltan, Wael Hamza, Yannick Versley, and Markus Boese. 2022. LINGUIST: Language model instruction tuning to generate annotated utterances for intent classification and slot tagging. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 218–241, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Yue Yu, Yuchen Zhuang, Jieyu Zhang, Yu Meng, Alexander J Ratner, Ranjay Krishna, Jiaming Shen, and Chao Zhang. 2023. Large language model as attributed training data generator: A tale of diversity and bias. In *Advances in Neural Information Processing Systems*, volume 36, pages 55734–55784. Curran Associates, Inc.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2016. Character-level convolutional networks for text classification.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena.

Y Zhu, P Zhang, EU Haq, P Hui, and G Tyson. 2023. Can chatgpt reproduce human-generated labels. *A Study of Social Computing Tasks*.