Chain-of-LoRA: Enhancing the Instruction Fine-Tuning Performance of Low-Rank Adaptation on Diverse Instruction Set

Xihe Qiu, Teqi Hao, Shaojie Shi, Xiaoyu Tan, and Yu-Jie Xiong

***Abstract*—Recently, large language models (LLMs) with conversational-style interaction, such as ChatGPT and Claude, have gained significant importance in the advancement of artificial general intelligence (AGI). However, the extensive resource requirements during pre-training, instruction fine-tuning (IF), and reinforcement learning through human feedback (RLHF) pose challenges, particularly for individuals and studios with limited resources. Moreover, sensitive data that cannot be deployed on remote training platforms or queried through APIs further exacerbates this issue. To address these limitations, researchers have introduced a parameter-efficient framework called low-rank adaptation (LoRA) for IF on LLMs. However, training individual LoRA networks faces capacity constraints and struggles to adapt to large domains with significant distributional shifts across different tasks. In this letter, we propose a novel framework called chain-of-LoRA to enhance the IF performance of LoRA. Our approach involves training a LoRA network to classify the instruction type and then utilizing task-specific LoRA networks to accomplish the respective tasks. By training multiple task-specific LoRA networks, we exploit a trade-off between performance and disk storage, leveraging the easily expandable and cost-effective nature of disk storage compared to precious graphical resources. Our experimental results demonstrate that our proposed framework achieves comparable performance to typical direct IF on LLMs.**

***Index Terms*—Large language models, low-rank adaptation, instruction fine-tuning.**

I. Introduction

LARGE Language Models (LLMs) [1] are often tailored for various specific applications through a process known as fine-tuning [2], enhancing their capabilities for zero-shot and few-shot generalization across unseen tasks [3], [4]. Their impressive understanding and generative abilities propel the effectiveness of natural language processing (NLP) tasks [5], [6], [7]. Currently, these LLMs serve as comprehensive assistants, offering beneficial and safe advice in response to an extensive range of human inquiries [8]. Nevertheless, the effectiveness of LLMs is significantly affected by the size of the model, seen as a critical factor for emergent capability [9]. Consequently, the alignment process can be resource-intensive, given that the fine-tuning process requires updates to all of the model's parameters [10], [11]. However, LLMs' adaptability to various applications comes at the cost of substantial computational resources [12].

Compared to fine-tuning LLMs with billions of parameters (e.g., GPT-3 with 175 billion trainable parameters [13], Low-Rank Adaptation (LoRA) [14] substantially reduces the number of trainable parameters for downstream tasks, achieving both storage- and compute-efficiency. This is accomplished by freezing the pre-trained model weights and introducing trainable rank-decomposed matrices into each layer of the Transformer architecture. Despite its success, however, LoRA's performance is inferior regarding general IF capability. This is due to LoRA being a low-rank approximation that cannot accommodate all tasks with large distributional shifts, preventing it from achieving comparable general IF capabilities to aligned LLMs.

Is it possible to enhance the IF performance of LoRA to make it comparable to aligned LLMs? In this letter, we propose a novel framework called "chain-of-LoRA". Our approach involves training a task-selection LoRA network to classify instruction types and then utilizing task-specific LoRA networks to perform corresponding tasks. This training process can be conducted in a fully supervised manner. By training multiple task-specific LoRA networks, we aim to exploit the complementary strengths of different networks in subgroup tasks, in that the distributions exhibit statistical similarity. This chain-of-LoRA framework enables improved IF performance by effectively leveraging the capabilities of LoRA for diverse tasks, ultimately enhancing the overall efficiency and adaptability of the model.

Our framework comprises two primary components. Firstly, we train LoRA models that align with the number of labels specific to each task, which we refer to as task-specific LoRA. Secondly, we introduce a dedicated LoRA model for task selection, known as task-selection LoRA. During the training of task-specific LoRA, only data corresponding to the particular task is utilized. The format of this LoRA training is similar to the conventional IF [15], [16], where the input consists of instructions and the output corresponds to the desired output for the given task. During the training process of task-selection LoRA, we employ specific prompts to determine the task type

Manuscript received 15 November 2023; revised 4 February 2024; accepted 11 March 2024. Date of publication 14 March 2024; date of current version 28 March 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62102241, and in part by Shanghai Municipal Natural Science Foundation under Grant 23ZR1425400. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Panagiotis Petrantonakis. *(Corresponding author: Xiaoyu Tan.)*

Xihe Qiu, Teqi Hao, Shaojie Shi, and Yu-Jie Xiong are with the School of Electronic and Electrical Engineering, Shanghai University of Engineering Science, Shanghai 201620, China.

Xiaoyu Tan is with INF Technology (Shanghai) Company Ltd., Shanghai 201210, China (e-mail: txywilliam1993@outlook.com).

Digital Object Identifier 10.1109/LSP.2024.3377590

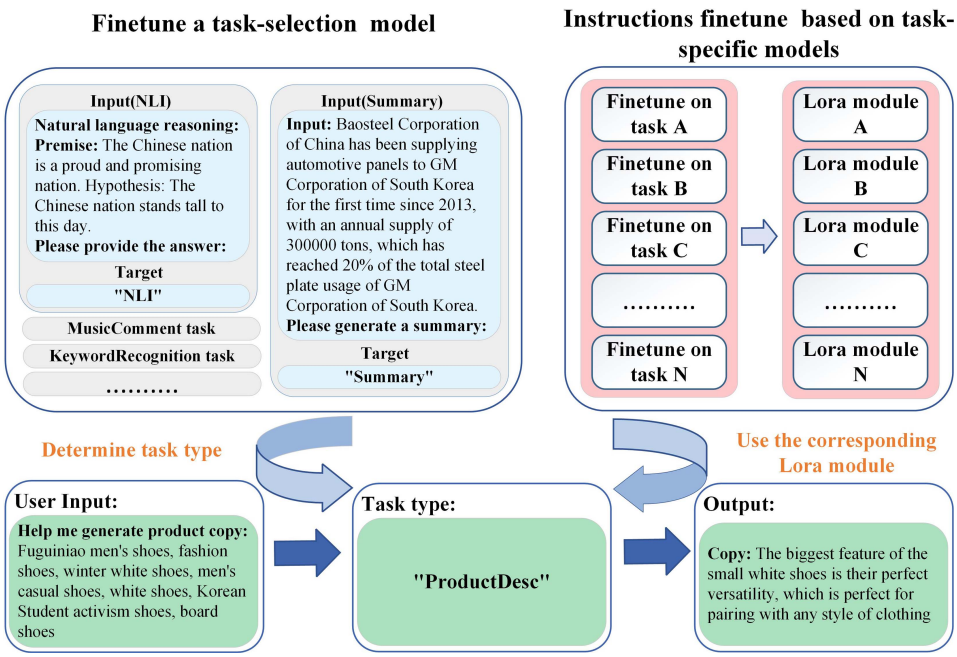

Fig. 1. Overall framework of our proposed method with $task\_prompt$. It consists of (left) task-selection LoRA model and (right) task-specific LoRA model. The task-selection LoRA model (left) analyzes user input instructions and generates corresponding prompts to prompt the framework to select the appropriate (right) task-specific LoRA to complete corresponding downstream tasks.

corresponding to the input instruction. During deployment, our framework processes and analyzes the user-input instruction by inputting it into the task-selection LoRA. This enables us to determine the task type associated with the instruction. Following this, the framework selects the appropriate task-specific LoRA to generate a response tailored to the instruction. The operation process is structured linearly within a single model framework, where transitions between tasks are facilitated by swapping out different LoRA modules. This sequential workflow enhances the model's flexibility across diverse tasks and improves its efficiency and effectiveness in specific tasks. The main contributions can be summarized as follows:

- We propose a novel learning framework (i.e., chain-of-LoRA) to enhance the general IF performance of LoRA.
- Extensive experiments on the generation instruction dataset demonstrate that our chain-of-LoRA outperforms the individual LoRA in terms of model performance. The evaluation demonstrates a significant quality improvement when compared to prior individual LoRA.
- Our model demonstrates high effectiveness and efficiency; it can be performed on consumer-grade graphics cards (i.e., RTX 3090), achieving both storage- and computational-efficiency.

II. Methods

A. Task-Specific and Task-Selection LoRA

Given an instruction dataset D_{IF}, each data point (x, y, t) contains an instruction x, desiable response y, and a task label t, which is labeled from a task set $t \in T = \{t_1, t_2, \ldots, t_n\}$. Since the D_{IF} is a general dataset that contains diverse types of tasks, the parameter-efficient training methods of training one model based on LLMs cannot generalize across all the tasks. To overcome this issue, as shown in Fig. 1, we train $n + 1$ LoRA models. This includes n task-specific LoRA models denoted as $M = \{M_1, M_2, \ldots, M_n\}$ for each task and one task-selection LoRA model for discriminating the task type of instruction, which is referred to as P. During the training of the task-specific LoRA, only the data corresponding to the specific task is used for training. We can optimize the model sets M by minimizing the cross-entropy between the model generation $\hat{y} = M_t(x)$ and desirable response y for $t \in T = \{t_1, t_2, \ldots, t_n\}$. This optimization process is identical to the typical IF, in which we provide instructions as input and obtain the corresponding output from LLMs [17], [18].

Subsequently, the task-selection LoRA model P is trained to perform task discrimination. We design a prompt $task\_prompt$ to construct a new task selection instruction with x, which is shown in Fig. 1 Then, we can optimize the task-selection model P by minimizing the cross-entropy loss between $\hat{t} = P(task\_prompt, x)$ and task label t. This optimization process is also identical to the typical IF [18], [19], [20] and aforementioned training of M.

For system inference, we first infer the task-selection model $i = P(task\_prompt, x)$ and get the predicted task i. Then, we parse the resulting string to identify the corresponding model and infer the task-specific LoRA model M_i which is trained on the i task, to get the response $y' = M_i(x)$.

B. Training of LoRA

LoRA [14] follows the idea that pre-trained language models with low "intrinsic dimensionality" exhibit efficient learning

even when subjected to random projections into smaller subspaces [21]. It involves freezing the pre-training weight, denoted as W_0, throughout the training process and constraining its update using the following formula:

$$W_0 + \Delta W = W_0 + BA, \tag{1}$$

where both A and B contain trainable parameters, and where $W_0 \in R^{d\times k}$, $B \in R^{d\times r}$, $A \in R^{r\times k}$, $r \ll min(d, k)$

The framework of the training process and reasoning process of the entire model is shown in Fig. 1. It consists of a task selection module P and n instruction fine-tuned task-specific modules M_x, where $x \in n$. We train the two separately and combine them when performing system inference.

C. Evaluation Metrics

We comprehensively assess chain-of-LoRA by conducting a rigorous comparison with state-of-the-art language models. We evaluate perplexity, task-selection model accuracy, and output helpfulness scores rated by GPT-3.5-Turbo and human annotators.

Perplexity [22] assesses LLMs' performance by quantifying sentence prediction quality in a test dataset. Lower perplexity implies superior predictions, derived from the model's word-level predicted probabilities using a specific formula:

$$Perplexity = 2^{-\frac{1}{N}\sum_{i=1}^{N}\log_2 P(w_i|w_1,w_2,\ldots,w_{i-1})} \tag{2}$$

Here, N is the total number of sentences in the test dataset, w_i represents the i-th word in the sentence, and $P(w_i|w_1, w_2, \ldots, w_{i-1})$ is the probability of the model predicting the next word w_i given the previous context.

The effectiveness of the prediction module is evaluated by the **accuracy**, which is described as follows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \tag{3}$$

where TP, TN, FP, and FN represent the true positive rate, true negative rate, false positive rate, and false negative rate, respectively.

GPT-3.5-Turbo evaluation is implemented to automatically rate the model outputs based on their helpfulness. For this evaluation, we directly utilize the prompt used in previous work [23] to rate the outputs from 1 to 6.

For *human evaluation*, we generate a single response for each test prompt. Subsequently, we employ humans to score the LLM-generated content on three levels: excellent, pass, and fail. This labeling method has been used in [23] to evaluate the quality of the LLM outputs. Here we follow the same requirement introduced in [23].

III. Experiments and Performance

A. Data Description and Implementation Details

We extensively evaluate the performance of our model using the widely used YeungNLP/firefly-train-1.1 M dataset[1] [24]. This dataset encompasses 23 distinct tasks related to Chinese natural language processing, which is suitable to test the capability of our proposed chain-of-LoRA. This dataset has taken careful measures to create quality instruction templates and to implement effective data preprocessing strategies, ensuring the data we work with is both non-repetitive and simulates human-like interactions. To preserve task independence and representativeness, we exclude the data generated by "BELLE" [25] in this dataset because the "BELLE" is a self-instruct dataset that contains general instructions from different tasks. Hence, the final dataset we used for training and testing comprises 1.1 million samples.

We randomly split 20% for testing, 20% for validation, and 60% for training. For the experiments conducted in Section II-I-B, we report the results evaluated on the test dataset. For human evaluation, we randomly sample 200 prompts per evaluation from the test set for each annotator. We experiment on one A100 (80 G) GPU for efficiency. The hyper-parameter we used for training follows the original Bloomz model [26] and the recommendation in [24]. We implemented and thoroughly evaluated three models of varying complexities on a standalone NVIDIA RTX 3090 GPU, leveraging chain-based computational techniques to significantly improve memory optimization. Our approach guarantees efficient and smooth inference for LLMs on consumer-grade GPUs by promptly freeing up GPU memory as soon as the task-selection model finishes executing.

B. Main Results

1) Perplexity Evaluation: We conducted experiments using three different parameter sizes of the Bloomz models [26], which have been processed to reduce the model size by pruning the token embedding, with parameter sizes of 396 m, 1b4, and 6b4, respectively. To perform a comprehensive comparison, we organized the aforementioned 23 task types into distinct categories and separately applied IF on LoRA modules for each category. Conversely, we merged these 23 diverse task types into a unified mixed-task dataset and conducted holistic IF to train the LoRA module. In particular, we trained a specific model set M, for each task and also developed a consolidated mixed-task model M_{mix} for mixed-task training. Subsequently, we further trained a task-selection model P to make task selections. Our proposed chain-of-LoRA framework consists of the selection model P, combined with 23 different task models M. Table I shows that adopting a mixed-task LoRA approach notably decreases the model's performance in terms of perplexity, except for OpenQA tasks. In contrast, our framework exhibited considerably lower perplexity scores and demonstrated its effectiveness. This improvement emphasizes the strength of our framework, which, by leveraging the inherent generative capabilities of the LLMs, is capable of generating plausible responses even in scenarios where the task selection mechanism might not be perfectly accurate.

2) Task-Selection Accuracy: Additionally, we conduct further experiments to demonstrate the effectiveness of the task-selection module P. Due to the nature of generation, it is often challenging to directly generate the exact target strings we

[1][Online]. Available: https://huggingface.co/datasets/YeungNLP/firefly-train-1.1M/tree/main

TABLE I
ZERO-SHOT PERPLEXITY OF M, MIXED-TASK LORA, AND PROPOSED CHAIN-OF-LORA

Task Type	Bloomz-396m	Bloomz-1b4	Bloomz-6b4
AncientPoem	120.99	89.26	3.81
ClassicalChinese	33.96	26.47	2.90
Composition	29.57	21.65	7.95
Cot	4.71	4.41	2.68
Couplet	32.61	25.58	2.78
Dictionary	27.55	21.02	2.96
JinYongGeneration	42.24	29.66	22.10
KeywordRecognition	23.74	16.84	2.75
LyricGeneration	11.68	10.22	4.18
MRC	33.14	23.60	2.66
MusicComment	40.89	33.09	6.32
NER	21.17	18.35	2.64
NLI	15.37	12.45	2.61
ProductDesc	26.29	20.80	3.26
Program	3.44	3.21	2.72
ProseGeneration	67.42	55.86	27.15
OpenQA	63.15	43.07	3.96
SentimentAnalyze	20.43	24.45	2.70
Story Generation	29.61	22.21	9.06
Summary	32.95	22.74	2.77
TextCorrection	14.59	11.91	2.77
TextMatching	6.32	5.96	2.61
Translation	30.19	22.21	2.78
Mixed-task-LoRA	37.29	24.81	7.04
Chain-of-LoRA	2.78	2.24	1.95

TABLE II
TASK ACCURACY OF THE TASK-SELECTION MODEL P

Evaluate	Bloomz-396m	Bloomz-1b4	Bloomz-6b4
Acc.	82.24%	86.46%	95.06%
Top-k Acc.	84.42%	89.46%	97.32%

TABLE III
DIFFERENT TRAINING STRATEGIES FOR TASK SELECTION MODULES P

Evaluation	Bloomz-LoRA	Bloomz-SFT	BERT-SFT
Acc.	82.24%	86.96%	83.52%
Top-k Acc.	84.42%	89.34%	87.89%

desire. However, by examining the generated content, we found that the correct target always appears in the generated output. Therefore, we implement the Knuth-Morris-Pratt algorithm to match the desired task targets. To further mitigate the potential error encountered by the task selection model due to similar instructions, we have employed a Top-k operation to enhance the precision of the task selection process. We evaluated the performance of the task-selection model using three different parameter sizes. Experimental results in Table II highlight the favorable performance of our framework in the classification task of predicting instruction labels, achieving both high top-k accuracy when $k = 1, 3$. We also observed that the efficacy of the framework can be enhanced by increasing the scales of the base models.

We examined various strategies for training task-selection models, using Bloomz-396 M and BERT-base as case studies. The findings are detailed in Table III. When we applied IF to the entire model, we observed a slight improvement in predictive accuracy. However, the computational resources required for such fine-tuning were significantly high, making this method cost-inefficient compared to the use of LoRA. While employing a smaller network like BERT as a task selection strategy is feasible, we prefer to maintain framework consistency and operational simplicity by consistently using LoRA modules within the same model architecture for task selection. In future research, we aim to investigate using other smaller networks to improve our framework.

TABLE IV
EFFECTIVENESS OF CHAIN-OF-LORA

Evaluation	Bloomz-396m	Bloomz-1b4	Bloomz-6b4
		Direct IF	
GPT	2.53	3.07	3.41
Fail	0.61	0.42	0.11
Pass	0.38	0.50	0.69
Excellent	0.10	0.08	0.20
		Mixed-task-LoRA	
GPT	2.03	2.45	2.97
Fail	0.72	0.51	0.32
Pass	0.28	0.48	0.64
Excellent	0.00	0.01	0.40
		Chain-of-LoRA	
GPT	2.47	2.98	3.34
Fail	0.63	0.43	0.12
Pass	0.36	0.52	0.72
Excellent	0.01	0.05	0.16

3) GPT-3.5-Turbo and Human Evaluation: To further validate the effectiveness of our proposed framework, we conducted response quality evaluation by GPT-3.5-Turbo and human annotators, which follows the evaluation processes introduced in Section II-C. GPT-3.5-Turbo will provide a score ranging from 1 to 6 to measure the helpfulness of responses with respect to the given instructions. For human evaluation, we invite five human annotators to label all the responses and report the average results. Similar to [23], we report the pass rate, excellent rate, and fail rate for each evaluation. As depicted in Table IV, our proposed approach considerably enhances the Mixed-task LoRA method's effectiveness in GPT-3.5-Turbo evaluations, delivering results on par with typical direct IF methods. In terms of human evaluation, it's noteworthy that our methodology significantly reduces failure rates to a level similar to direct IF, while notably optimizing pass rates. These results also confirm observations from related work [23], highlighting a strong correlation between human evaluations and automatic evaluations conducted with other independent LLMs.

IV. CONCLUSION

Our proposed chain-of-LoRA framework addresses the challenges of resource-intensive IF on LLMs with LoRA. By training a task-selection LoRA to classify instruction types and leveraging task-specific LoRA networks for respective tasks, we achieve comparable performance to typical direct IF on LLMs. This approach achieves a trade-off between performance and disk storage, rendering it more viable for individuals and studios with limited resources. Both automated evaluation and human assessment demonstrate that our chain-of-LoRA framework presents a promising solution to enhance the adaptability and efficiency of IF in the context of conversational-style LLMs.

REFERENCES

[1] J. Huang et al., "Large language models can self-improve," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2023, pp. 1051–1068.

[2] J. Phang, Y. Mao, P. He, and W. Chen, "HyperTuning: Toward adapting large language models without back-propagation," in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 27854–27875.

[3] S. Zhang et al., "OPT: Open pre-trained transformer language models," 2022, *arxiv: 2205.01068*.

[4] A. Chowdhery et al., "Palm: Scaling language modeling with pathways," *J. Mach. Learn. Res.*, vol. 24, pp. 1–113, 2023.

[5] C. Zhou et al., "A comprehensive survey on pretrained foundation models: A history from BERT to ChatGPT," 2023, *arXiv:2302.09419*.

[6] W. X. Zhao et al., "A survey of large language models," 2023, *arXiv:2303.18223*.

[7] X. Tan et al., "Self-criticism: Aligning large language models with their understanding of helpfulness, honesty, and harmlessness," in *Proc. Conf. Empirical Methods Natural Lang. Process.: Ind. Track*, 2023, pp. 650–662.

[8] S. Bubeck et al., "Sparks of artificial general intelligence: Early experiments with GPT-4," 2023, *arXiv:2303.12712*.

[9] J. Yang et al., "Harnessing the power of LLMs in practice: A survey on ChatGPT and beyond," *ACM Trans. Knowl. Discov. Data*, 2023, doi: 10.1145/3649506.

[10] H. Liu et al., "Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 1950–1965.

[11] M. Jia et al., "Visual prompt tuning," in *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 709–727.

[12] M. S. Ansari, B. F. Cockburn, and J. Han, "An improved logarithmic multiplier for energy-efficient neural computing," *IEEE Trans. Comput.*, vol. 70, no. 4, pp. 614–625, Apr. 2021.

[13] T. Brown et al., "Language models are few-shot learners," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 1877–1901.

[14] E. J. Hu et al., "LoRA: Low-rank adaptation of large language models," in *Proc. Int. Conf. Learn. Representations*, 2021.

[15] Y. Wang et al., "Super-naturalinstructions: Generalization via declarative instructions on 1600+ NLP tasks," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2022, pp. 5085–5109.

[16] C. Xu, D. Guo, N. Duan, and J. McAuley, "Baize: An open-source chat model with parameter-efficient tuning on self-chat data," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2023, pp. 6268–6278.

[17] S. Mishra, D. Khashabi, C. Baral, and H. Hajishirzi, "Natural instructions: Benchmarking generalization to new tasks from natural language instructions," 2021, *arXiv:2104.08773*.

[18] H. W. Chung et al., "Scaling instruction-finetuned language models," 2022, *arXiv:2210.11416*.

[19] V. Sanh et al., "Multitask prompted training enables zero-shot task generalization," in *Proc. Int. Conf. Learn. Representations*, 2021.

[20] J. Wei et al., "Chain-of-thought prompting elicits reasoning in large language models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 24824–24837.

[21] A. Aghajanyan, L. Zettlemoyer, and S. Gupta, "Intrinsic dimensionality explains the effectiveness of language model fine-tuning," in *Proc. 59th Annu. Meeting Assoc. Comput. Linguistics 11th Int. Joint Conf. Natural Lang. Process.*, 2021, pp. 7319–7328.

[22] N. Lee, Y. Bang, A. Madotto, M. Khabsa, and P. Fung, "Towards few-shot fact-checking via perplexity," in *Proc. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2021, pp. 1971–1981.

[23] C. Zhou et al., "LIMA: Less is more for alignment," in *Proc. Adv. Neural Inf. Process. Syst.*, 2024.

[24] B. Peng, L. Song, Y. Tian, L. Jin, H. Mi, and D. Yu, "Stabilizing RLHF through advantage model and selective rehearsal," 2023, *arXiv:2309.10202*.

[25] G. Yan et al., "Belle: Be everyone's large language model engine," 2023. [Online]. Available:https://github.com/LianjiaTech/BELL

[26] N. Muennighoff et al., "Crosslingual generalization through multitask finetuning," in *Proc. Annu. Meeting Assoc. Comput. Linguistics*, 2023, pp. 15991–16111.