

Making Small Language Models Better Multi-task Learners with Mixture-of-Task-Adapters

Yukang Xie^{1,2}, Chengyu Wang², Junbing Yan^{3,2}, Jiyong Zhou^{1,2}, Feiqi Deng¹, Jun Huang²

¹ South China University of Technology, Guangzhou, China

² Alibaba Group, Hangzhou, China ³ East China Normal University, Shanghai, China

ABSTRACT

Recently, Large Language Models (LLMs) have achieved amazing zero-shot learning performance over a variety of Natural Language Processing (NLP) tasks, especially for text generative tasks. Yet, the large size of LLMs often leads to the high computational cost of model training and online deployment. In our work, we present ALTER, a system that effectively builds the multi-task Learners with mixture-of-task-adapters upon small language models (with <1B parameters) to address multiple NLP tasks simultaneously, capturing the commonalities and differences between tasks, in order to support domain-specific applications. Specifically, in ALTER, we propose the Mixture-of-Task-Adapters (MTA) module as an extension to the transformer architecture for the underlying model to capture the intra-task and inter-task knowledge. A two-stage training method is further proposed to optimize the collaboration between adapters at a small computational cost. Experimental results over a mixture of NLP tasks show that our proposed MTA architecture and the two-stage training method achieve good performance. Based on ALTER, we have also produced MTA-equipped language models for various domains.¹

CCS CONCEPTS

• Computing methodologies → Natural language generation.

KEYWORDS

multi-task learning, language model, text generation

ACM Reference Format:

Yukang Xie^{1,2}, Chengyu Wang², Junbing Yan^{3,2}, Jiyong Zhou^{1,2}, Feiqi Deng¹, Jun Huang². 2018. Making Small Language Models Better Multi-task Learners with Mixture-of-Task-Adapters. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 4 pages. <https://doi.org/XXXXXX.XXXXXXX>

1 INTRODUCTION

The rapid emergence of Large Language Models (LLMs) has brought significant changes to the field of Natural Language Processing

¹All datasets are publicly available. The source codes and model checkpoints are released in EasyNLP [16]. URL: <https://github.com/alibaba/EasyNLP/tree/master/examples/mta>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

© 2018 Association for Computing Machinery.

ACM ISBN 978-1-4503-XXXX-X/18/06...\$15.00

<https://doi.org/XXXXXX.XXXXXXX>

(NLP). In particular, LLMs (such as ChatGPT² with 175B parameters) have demonstrated powerful abilities to interact with users and solve various NLP tasks in the zero-shot learning setting, whose performance even approaches or exceeds humans in some tasks [7]. Yet, the impressive performance of LLMs does not cover up their potential drawbacks in two aspects. i) The extremely large parameter size of LLMs leads to unaffordable computational and usage costs, especially when they need to be fine-tuned or deployed privately for specific applications or in resource-constrained environments. ii) The decoder-only architecture makes them difficult to maintain high performance over traditional NLP tasks with a highly limited output space such as text classification [10].

Based on the observation, we revisit the exploitation of *small* language models for multi-task learning across a variety of NLP tasks, which are much easier to train and deploy.³ Yet, improving the multi-task solving capacities of such models has been non-trivial due to the highly limited parameter space. Previously, several works have been conducted to improve the multi-task learning abilities of the transformer architecture. To begin with, Mixture-of-Experts (MoE) [4] divides complex tasks into smaller, more manageable subproblems, each of which is solved by an expert model. Multi-task MoE (MMoE) [8] uses a multi-gating module to generate specific expert weights according to different tasks. Switch Transformers [2] introduces a sparse addition of MoE to solve the problem of large computational efforts. However, the above MoE-based works are mainly proposed to train large language models at scale by adding parallel modules and partitioning the data at the token level. Hence, there is a lack of task-level differentiation for multi-task model adaption. In addition, the collaborative relationships between the internal modules (i.e., experts) for better multi-task learning performance are insufficiently explored.

In our work, we introduce the ALTER system that can effectively extend the multi-task learning capacity of *small* language models to address multiple NLP tasks simultaneously. In ALTER, the Mixture-of-Task-Adapters (MTA) architecture is designed as a lightweight extension to the transformer architecture in order to capture the commonalities and differences between tasks. A two-stage training method is further introduced to slightly adjust the collaboration between adapters for multiple tasks at a small computational cost. We conduct a series of experiments over a mixture of NLP tasks and show that our proposed MTA approach achieves good performance with few additional training costs, compared to standard supervised fine-tuning. We further release several domain-specific MTA-equipped language models and showcase their values to support real-world applications.

²<https://chat.openai.com/>

³By *small* language models in our work, we refer to language models with <1B parameters, compared to ChatGPT-like models with over 100B parameters.

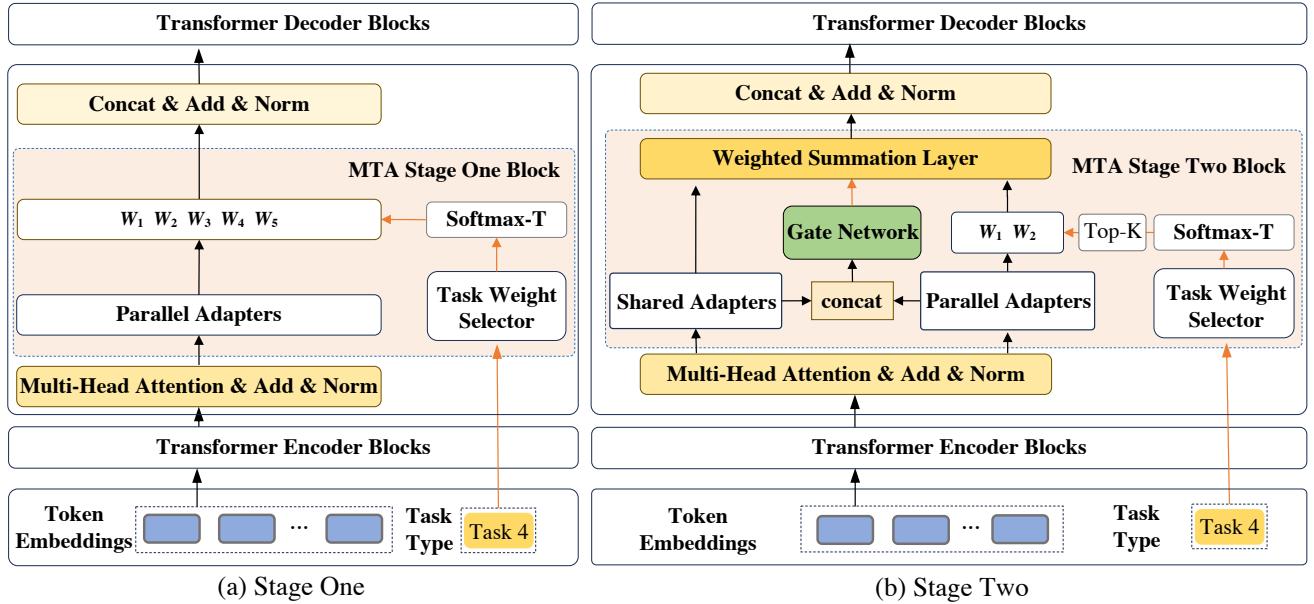


Figure 1: The replacement of the FFN (Feed Forward Network) module in a specific transformer layer with the MTA (Mixture-of-Task-Adapters) module, the structure of the adapter mirrors the FFN structure.

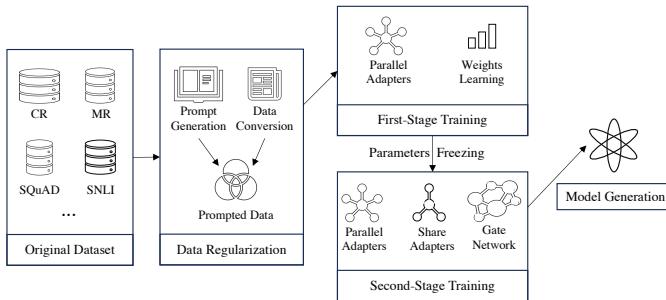


Figure 2: The system pipeline of ALTER.

2 THE ALTER SYSTEM

In this section, we first introduce our system pipeline. After that, we describe the structure of our MTA architecture and how to learn the parameters of MTA on a multi-task dataset. Specifically, the model is trained in two stages, with the parameters of the whole model fine-tuned in the first stage. In the second stage, we add the shared adapter module to improve the model performance by training only the parameters of the MTA module with other parameters frozen.

The system pipeline is shown in Figure 2. Given a collection of datasets of any arbitrary domain (possibly including text classification, language inference, generation, etc.), we regularize all datasets by reformatting the data and adding corresponding prompts, thus obtaining a multitasking dataset that is uniformly formatted. The data are trained in one stage with parallel adapters and weight learning to minimize the interference of different tasks, and then parameter freezing, introduction of shared adapters and gate[8] networks to further improve the collaboration between different tasks. We will subsequently elaborate on the details.

2.1 First-Stage Training for Obtaining Task-to-Adapter Correspondence

In the first stage of training, our main goal is to learn the correspondence between the adapters and tasks. We abandon the token-level input format in MoE [2] and instead favor the sentence-level input, the entire sentence is taken as a whole and is not broken down into individual tokens to be entered into a module. In our work, we propose the MTA (Mixture-of-Task-Adapters) architecture, which can be seen as parallel adapter blocks. As shown in Figure 1, in the first stage, the MTA module mainly consists of Parallel Adapters and Task Weights Selector.

To establish a correspondence between tasks and adapters, with each adapter being dedicated to a specific type of task, we introduce a bias to the initialization of the weights in the Task Weight Selector. This ensures that the model is capable of enabling task-to-adapter correspondence. Using Softmax-T amplifies the effect of the bias and enhances the model's adaptation capability. For example, the weights of the type of i -th task W_i can be initialized as follows:

$$W_i = \left[\frac{1}{N}, \frac{1}{N}, \frac{1+\lambda}{N}, \dots, \frac{1}{N} \right] \quad (1)$$

where N denotes the number of parallel adapters, $\frac{1+\lambda}{N}$ refers to the third gating value, guiding the third adapter to pay more attention to the i -th type of task. Note that during model training, these weights are also learnable, making our mechanism self-adaptive. The MTA module during the first phase of training has the following formula:

$$A(x) = \text{Concat}(A_1(x), \dots, A_N(x)) \quad (2)$$

$$MTA_{out1} = softmax\left(\frac{W}{T}\right) \cdot A(x) \quad (3)$$

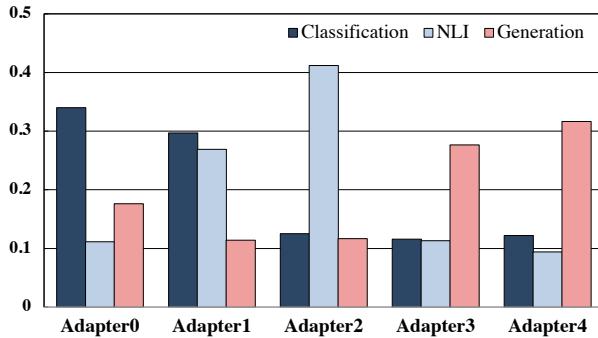


Figure 3: Visualization of adaptive weights obtained after first-stage training reveals that weights are strongly correlated with the type of task.

where $A_i(x)$ is the output of the i -th adapter network w.r.t. the input x . W denotes the manually initialized matrix of gates for all the types of tasks. T is the sharpening coefficient. By introducing the gate values, we guide each adapter to make major contributions to one type of tasks only. This approach enables us to retain the specific characteristics of individual tasks. Take our experimental multi-task datasets as an example. The self-adaptive weights learned during the first stage are shown in Figure 3. In addition, we insert a special token “[START]” to each input sample, aiming to capture the semantics of the whole input sequence.

2.2 Second-Stage Training for Adjustment of Collaboration Relationships

After completing the first stage of training, we inherit all the weights from the first stage. To further adjust the collaborative relationship between adapters, we introduce additional shared adapters and a gate network, while freezing all training parameters except for the MTA module. The gate network is used to generate adaptive weights, aiming to obtain collaborative relationships among various adapters for different tasks by capturing the “[START]” information in the hidden layer. The formula for the gate network is as follows:

$$A^*(x) = \text{concat}(\sum_{i=1}^K A_i(x) \cdot W_i, S(x)) \quad (4)$$

$$W^* = G(\text{concat}(S(x)[\text{START}], A^*(x)[\text{START}])) \quad (5)$$

$$\text{MTA}_{out2} = A^*(x) \cdot W^* \quad (6)$$

where $A^*(x)$ denotes the combined representations of shared and top- K adapter modules. $S(x)$ is the output of the shared adapters. G denotes a gate network with linear layers and activation functions.

Relying only on the first stage parallel adapters leads to direct interactions between tasks and makes it difficult to achieve a better state of collaboration. In the second stage, the top- k selection can eliminate direct interference between different tasks. To compensate for the collaborative effect of synchronous deletion without reintroducing direct interference, a shared adapter structure is added as a transition module for information interaction to moderate the direct interference between tasks.

Method	Classification	NLI	QA	Score
ChatGPT	35.88	43.90	72.15	50.70
BART	88.23	78.00	72.92	79.72
GPT-2	91.58	79.63	71.88	81.03
Switch Trans.	91.10	82.85	62.70	78.88
T5-Base	92.47	84.78	80.00	0.8575
Ours-Base	93.26	86.34	81.28	86.96
T5-Large	93.81	88.54	77.82	86.72
Ours-Large	94.01	88.71	82.70	88.47

Table 1: Overall evaluation results. ChatGPT is evaluated under the zero-shot learning setting and is for reference only. NLI and QA are short for natural language inference and question answering, respectively.

Model	Score (T5-base)	Score (T5-large)
Full implement.	86.96	88.47
Vanilla fine-tuning	85.75	86.72
w/o. second-stage	86.58	87.81
w/o. parameter freeze	86.60	87.40

Table 2: Ablation results of two-stage training with different strategies on T5-base and T5-large models.

3 EXPERIMENTAL EVALUATION

In this section, we evaluate the effectiveness of our proposed approach over public datasets.

3.1 Datasets and Experimental Settings

We employ three common types of NLP tasks (text classification, natural language inference task, and question answering) to construct multi-task datasets. The datasets include:

- Text classification: CR [3], MR [9], SST-2 [14] and TREC [15];
- Natural language inference: SNLI [1];
- Question answering (text generation): SQuAD [13].

Following the work of T5 [12], we add appropriate prompts for all the tasks to convert them into a uniform input format. For classification and inference tasks, we match the model outputs with the ground-truth and report the prediction accuracy. For generation, we constructed the data in such a way as to generate questions based on the answers, we report the Rouge-L score [6], measuring the similarity between generated and reference texts. In the implementation, we load the pre-trained weights from T5-base, with 24 transformer layers. Baseline models include the base versions of BART [5], GPT-2 [11] and Switch Transformer [2]. The performance of ChatGPT is also reported for reference. For our method, we also conduct comparative experiments on T5-large.

3.2 General Experimental Results

In the first set of experiments, we mix multiple tasks into a test set, and then obtain four scores via a unified scoring criterion. Several points can be observed from Table 1. Firstly, our model achieves higher overall results than other models, demonstrating the effectiveness of our proposed method. Secondly, when compared to the token-level processing method of Switch Transformer, our sentence-level approach to task processing exhibits a significant

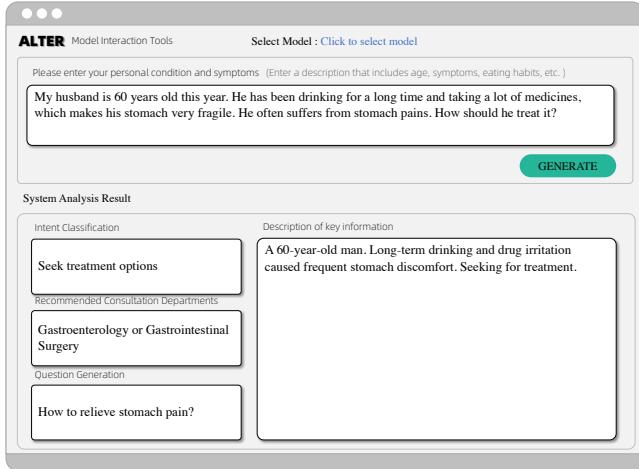


Figure 4: The inference process and results of the model trained on the PromptCBLUE dataset presented on WebUI.

performance advantage on this multi-task dataset. Finally, we compare the results of our proposed model with ChatGPT-turbo. The outputs generated by ChatGPT-turbo are more difficult to control due to its intrinsic properties. Therefore, we relax the scoring criteria to include answer words as long as they are correct for both classification tasks and natural language inference tasks. Despite this relaxation, our model still outperforms the ChatGPT-turbo baseline, demonstrating its effectiveness even against larger models after fine-tuning on common multi-task datasets.

3.3 Ablation Study and Model Analysis

To further demonstrate the effectiveness of our proposed two-stage training method and the shared adapter, we conduct a series of ablation experiments. Due to space limitations, we show only the most representative set of ablation experiments. As shown in Table 2, models with multiple adapter structures show significant improvements compared to the original model, regardless of whether we use T5-base or T5-large.

Our experiments have revealed that partial parameter fine-tuning performs better and requires less computational resources than full fine-tuning during the second stage of training. Furthermore, a comparison with the training results of the two-stage model architecture using global parameter fine-tuning shows that partial fine-tuning can optimize the collaboration between adapters and significantly improve model performance.

4 DEMONSTRATION SCENARIOS

In this demo, we will show the complete process of how our models are trained and deployed. For domain-specific applications, we have also trained several MTA-equipped language models for various domains. Take medicine for an example. Our model is trained over the multi-task PromptCBLUE dataset⁴, which is a large-scale instruction-tuning dataset in the medical domain in Chinese. The results also prove that our generated model, despite in small parameter size, can generate domain-specific knowledge accurately.

⁴<https://github.com/michael-wzhu/PromptCBLUE>

For better human-machine interactions, we have set up the WebUI interface, as shown in Figure 4.

5 CONCLUSION

We propose ALTER, a system that effectively builds the multi-tAsk Learners with mixTure-of-task-adaptERTs to address multiple NLP tasks simultaneously. Specifically, the Mixture-of-Task-Adapters (MTA) are introduced to help models learn task differences and commonalities to improve model performance on multi-task datasets. We demonstrate that our proposed method can achieve performance comparable to that of large models. In the proposed MTA module, we introduce a two-stage training method that incorporates prior knowledge to obtain basic collaborative model parameters for specific adapters corresponding to specific tasks in the first stage, and further improves model performance by coordinating collaboration among adapters in the second stage. We further demonstrate how to apply our technique for domain-specific applications.

REFERENCES

- [1] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP 2015, Lisbon, Portugal, September 17–21, 2015*. 632–642.
- [2] William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. *J. Mach. Learn. Res.* (2022), 120:1–120:39.
- [3] Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD, Seattle, Washington, USA, August 22–25, 2004*. ACM, 168–177.
- [4] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. 1991. Adaptive Mixtures of Local Experts. *Neural Comput.* 3, 1 (1991), 79–87.
- [5] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv:1910.13461* (2019).
- [6] Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*. ACL, 74–81.
- [7] Yiheng Liu, Tianle Han, Siyuan Ma, Jiayue Zhang, Yuanyuan Yang, Jiaming Tian, Hao He, Antong Li, Mengshen He, Zhengliang Liu, Zihao Wu, Dajiang Zhu, Xiang Li, Ning Qiang, Dinggang Shen, Tianming Liu, and Bao Ge. 2023. Summary of ChatGPT/GPT-4 Research and Perspective Towards the Future of Large Language Models. *CoRR* abs/2304.01852 (2023).
- [8] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H. Chi. 2018. Modeling Task Relationships in Multi-task Learning with Multi-gate Mixture-of-Experts. In *SIGKDD*. 1930–1939.
- [9] Bo Pang and Lillian Lee. 2005. Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales. In *ACL 2005*. 115–124.
- [10] Chengwei Qin, Aston Zhang, Zhuosheng Zhang, Jiaao Chen, Michihiro Yasunaga, and Diyi Yang. 2023. Is ChatGPT a General-Purpose Natural Language Processing Task Solver? *CoRR* abs/2302.06476 (2023).
- [11] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.
- [12] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *CoRR* abs/1910.10683 (2019).
- [13] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250* (2016).
- [14] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *EMNLP 2013, A meeting of SIGDAT*. ACL, 1631–1642.
- [15] Ellen M. Voorhees and Dawn M. Tice. 2000. Building a question answering test collection. In *SIGIR 2000*. ACM, 200–207.
- [16] Chengyu Wang, Minghui Qiu, Taolin Zhang, Tingting Liu, Lei Li, Jianing Wang, Ming Wang, Jun Huang, and Wei Lin. 2022. EasyNLP: A Comprehensive and Easy-to-use Toolkit for Natural Language Processing. In *EMNLP*. ACL, 22–29.