Domain Expansion: Parameter-Efficient Modules as Building Blocks for Composite Domains

NLP Group #37
Mann Patel, Divyajyoti Panda, Hilay Mehta, Parth Patel, Dhruv Parikh

Abstract

Parameter-Efficient Fine-Tuning (PEFT) is an efficient alternative to full scale fine-tuning, gaining popularity recently. With pre-trained model sizes growing exponentially, PEFT can be effectively utilized to fine-tune compact modules, Parameter-Efficient Modules (PEMs), trained to be domain experts over diverse domains. In this project, we explore composing such individually fine-tuned PEMs for distribution generalization over the composite domain. To compose PEMs, simple composing functions are used that operate purely on the weight space of the individually fine-tuned PEMs, without requiring any additional fine-tuning. The proposed method is applied to the task of representing the 16 Myers-Briggs Type Indicator (MBTI) composite personalities via 4 building block dichotomies, comprising of 8 individual traits which can be merged (composed) to yield a unique personality. We evaluate the individual trait PEMs and the composed personality PEMs via an online MBTI personality quiz questionnaire, validating the efficacy of PEFT to fine-tune PEMs and merging PEMs without further fine-tuning for domain composition.

1 Introduction

Parameter-Efficient Fine-Tuning (PEFT) based methods freeze most of the parameters associated with pre-trained language models (PLMs) and fine-tune only a small sub-set of parameters (adapter parameters) to customize PLMs for downstream tasks. This allows for efficient and quick fine-tuning via PEFT to yield Parameter-Efficient Modules (PEMs), with a reduced model size and memory footprint (Houlsby et al., 2019), trained on domain-specific datasets to be domain experts. Due to the exponentially increasing model sizes associated with PLMs (Zhao et al., 2023), PEFT methods have become increasingly popular (Xu et al., 2023) and the default standard to fine-tune such PLMs. In this project, we choose LoRA (Hu et al., 2021) and IA3 (Liu et al., 2022) state-of-the-art PEFT approaches as our PEM architectures.

Myers-Briggs Type Indicator (MBTI) is a widely used framework to characterize personality traits of individuals (Boyle, 1995). An MBTI personality can be decomposed into 4 key traits, each trait associated with a dichotomy. Specifically, the 4 dichotomies in MBTI are: (i) Extraversion (**E**) - Introversion (**I**) (ii) Sensing (**S**) - Intuition (**N**) (iii) Thinking (**T**) - Feeling (**F**) (iv) Judging (**J**) - Perceiving (**P**). A single MBTI personality, such as **INTJ**, is constructed by composing 4 traits, each trait selected from the two opposite traits in each dichotomy. Fig. 1 shows the composition of traits from dichotomies into the 16 personalities in MBTI.

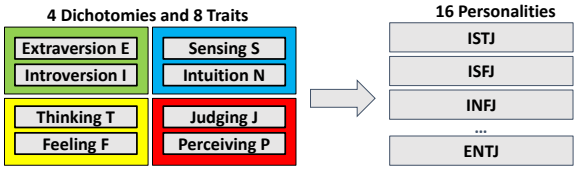

Figure 1: MBTI Dichotomies, Traits and Personalities

In this project, we propose a novel method to represent each of the 16 MBTI personalities using language models. The problem of representing an MBTI personality, via a language model, is formulated as a PEM composition problem. In particular, as the MBTI personalities can be decomposed into their 4 characteristic traits, we first train a PEM for each of these traits, yielding a total of 8 trait PEMs (4 dichotomies $\times$ 2 opposing traits per dichotomy). Next, we compose these individual PEMs via a simple composition function, $f(.)$, operating purely on the weight (parameter) space (of the individual PEMs) and without any additional fine-tuning, to yield a personality PEM for each of the 16 MBTI personalities. We utilize the MBTI personality quiz questionnaire[1] to evaluate both, the individual trait PEMs and the composed personality PEMs.

2 Related Work

Several prior works discuss language model composition for distribution generalization to composite domain tasks. Specifically, (Wortsman et al., 2022; Matena and Raffel, 2022; Jin et al., 2023; Ilharco et al., 2023), perform a full fine-tuning of PLMs across tasks and/or domains (each PLM initialized from the same pre-trained checkpoint) and merge/compose the fine-tuned models by performing simple arithmetic operations on model parame-

[1] https://www.16personalities.com/free-personality-test

ters, improving the composed model performance. On the other hand, (Pfeiffer et al., 2021; Wang et al., 2022), compose PEMs by fusing individual PEM outputs via a learnable module or via mixture-of-experts, both requiring post-composition fine-tuning. We utilize the method proposed by (Zhang et al., 2023) to compose PEMs *without* requiring additional fine-tuning, applying it to the problem of representing MBTI personalities through PEMs.

Prior works that combine language models and MBTI personalities typically explore language models to classify the MBTI personality associated with an excerpt of text. (dos Santos and Paraboni, 2022; Keh and Cheng, 2019; Mehta et al., 2020; Vásquez and Ochoa-Luna, 2021) are works that fine-tune PLMs in order to classify text data with an MBTI personality. (Fernau et al., 2022) aligns a chatbot to the MBTI personality of its user in order to improve chatbot usability.

However, no prior work uses PEMs as building blocks to compose personalities via PEM traits, fine-tuning language models (PEMs) to behave as individuals with trait/personality characteristics. Further, the proposed method is general, simple and efficient: PEMs are inexpensive to train, result in smaller model sizes and require no additional fine-tuning data post-composition for distribution generalization.

3 Problem Description

To formally define our problem, we consider N domains, each domain representing an individual task and/or application. Given these N domains, $D_1, D_2, ..., D_N$ we fine-tune a PEM for each such domain, yielding N PEMs, $M_{\boldsymbol{\theta}_1}, M_{\boldsymbol{\theta}_2}, ..., M_{\boldsymbol{\theta}_N}$, where $\boldsymbol{\theta}_i$ indicates the (adapter) parameters of the i^{th} PEM model, $M_{\boldsymbol{\theta}_i}$. We wish to compose such N fine-tuned PEMs via a composing function $f(.)$ applied to the (adapter) parameters of the individual PEMs, to yield a composite PEM $\mathcal{M}_{\boldsymbol{\theta}_C}$ with composite (adapter) parameters $\boldsymbol{\theta}_C$ that generalizes over the composite task $D_C = \bigcup_{i=1}^{N} D_i$.

$$\boldsymbol{\theta}_C = f(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, ..., \boldsymbol{\theta}_N) \quad (1)$$

$$D_C = \bigcup_{i=1}^{N} D_i \quad (2)$$

Note that $\boldsymbol{\theta}_i$ are adapter parameters representing a small fraction of the PLM model parameters utilized for (parameter-efficient) fine-tuning (PEFT). Additionally, the fine-tuning (PEFT) performed for each D_i is initiated from an identical PLM model checkpoint, across all the domains over which composition for distribution generalization is to be performed.

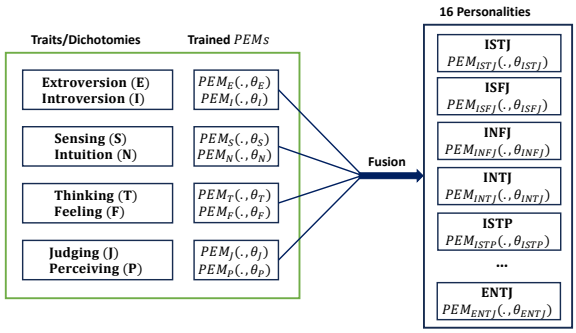

Figure 2: Method Overview

4 Proposed Method

In our project, the individual domains D_i (section 3) correspond to the individual traits **E, I, S, N, T, F, J, P** defined in section 1 (2 opposing traits for each of the 4 dichotomies). The individual trait PEMs are first trained for each trait (domain) to yield 8 base PEMs. Next, we merge/compose 4 trait PEMs (selecting exactly one trait from each dichotomy) to yield a personality PEM (for a total of $2^4 = 16$ personality PEMs). The overview of our proposed method can be seen in Fig. 2.

4.1 Trait PEMs via PEFT

In order to train the individual trait PEMs, we utilize 2 state-of-the-art PEFT methods/architectures - (i) LoRA (ii) IA3. PEFT based approaches, including LoRA and IA3, work by freezing parameters associated with the base PLM excluding certain fine-tuning parameters (adapter parameters or adapters) (Fig. 3). During the backward pass, the frozen parameters stay fixed (unmodified) while the adapter parameters are allowed to learn (update) from the domain data. PEFT, thus, (i) reduces effective model size (ii) makes training inexpensive/quick (iii) reduces the number of domain data samples required for fine-tuning (iv) can be used to train several diverse domain expert PEMs, efficiently.

LoRA LoRA (Hu et al., 2021) (Low-Rank Adaptation) is a prevalent PEFT method. For layers in a transformer that transform an input $\boldsymbol{x} \in \mathbb{R}^k$ to $\boldsymbol{h} \in \mathbb{R}^d$ via weight matrices, LoRA modifies this transformation as below,

$$\boldsymbol{h} \leftarrow \boldsymbol{h} + \boldsymbol{BAx} \quad (3)$$

$$\boldsymbol{\theta}_{LoRA} = \{\boldsymbol{B}, \boldsymbol{A}\} \quad (4)$$

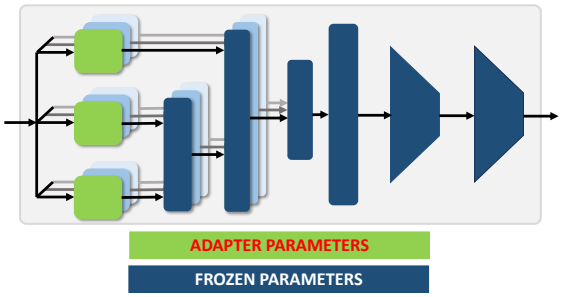

Figure 3: PEFT Training

In eq. 3, $\boldsymbol{B} \in \mathbb{R}^{d \times r}$ and $\boldsymbol{A} \in \mathbb{R}^{r \times k}$ are the LoRA adapter parameters $\boldsymbol{\theta}_{LoRA}$ with $r \ll \min(d, k)$. While the modification in eq. 3 can be done for any $\boldsymbol{x} \rightarrow \boldsymbol{h}$ transformation within a transformer, it is conventionally applied for the query and value vector generation in each transformer layer. In our project, we follow this convention.

IA3 IA3 (Liu et al., 2022) is a PEFT method which was proposed for few-shot learning. In order to perform PEFT, IA3 introduces learnable scale vectors $\boldsymbol{l}_k$, $\boldsymbol{l}_v$ and $\boldsymbol{l}_{ff}$. A learnable scale vector $\boldsymbol{l} \in \mathbb{R}^d$ is applied to a vector $\boldsymbol{h} \in \mathbb{R}^d$ as below,

$$\boldsymbol{h} \leftarrow \boldsymbol{h} \odot \boldsymbol{l} \tag{5}$$

In eq. 5, $\boldsymbol{h}$ is a vector generated within the transformer while $\boldsymbol{l}$ is the adapter parameter for IA3. Particularly, $\boldsymbol{l}_k$ and $\boldsymbol{l}_v$ are applied to key and value vectors, respectively, in the multi-headed self-attention mechanism and $\boldsymbol{l}_{ff}$ is applied to the intermediate activations of the feed-forward network, in a transformer (in each layer). Thus, $\boldsymbol{\theta}_{IA3} = \{\boldsymbol{l}_k, \boldsymbol{l}_v, \boldsymbol{l}_{ff}\}$.

4.2 Personality PEMs via PEM Composition

Using LoRA and IA3, we train 8 trait PEMs (each, for both LoRA and IA3). To obtain a personality PEM, we compose (merge) the trait PEMs, without further fine-tuning post-composition. Given trait PEMs $M_{\boldsymbol{\theta}_{T_i}}$ where T_i is the i^{th} dichotomy trait, $i \in \{1, 2, 3, 4\}$ for the 4 dichotomies, we compose PEMs as below,

$$\boldsymbol{\Theta}_P = \sum_{i=1}^{4} \boldsymbol{\theta}_{T_i} \tag{6}$$

In eq. 6, $P = T_1T_2T_3T_4$, is an MBTI personality type and $\boldsymbol{\Theta}_P$ are the adapter parameters obtained for the composite PEM $\mathcal{M}_{\boldsymbol{\Theta}_P}$ representing that personality. Note that the resulting composite PEM has the same architecture as the individual combining PEMs. The adapter parameters of the composite PEM is given by a simple sum (composing function $f \rightarrow \sum$) over the adapter parameters of the individual PEMs, leading to distribution generalization over the combining (trait) domains.

Fundamental works such as word2vec (Mikolov et al., 2013) motivate such simple PEM composition rule for distribution generalization. Further, (Matena and Raffel, 2022; Wortsman et al., 2022; Jin et al., 2023) hypothesize that models which are fine-tuned from the same initial PLM checkpoint often lie within similar error bins. Thus, the parameters of such models can simply be added up to yield a superior model with improved generalization.

5 Experimental Results

We worked on two tasks, zero-shot classification for classifying the prompt into one of the seven categories on the 7-point Likert scale (strongly agree, agree, slightly agree, neutral, slightly disagree, disagree and strongly disagree), and text generation which acted as a chatbot for answering questions.

For zero-shot classification, we utilize the BERT base model as the PLM backbone. In order to fine-tune PEM adapters for each individual trait, we insert a classification head (adapter parameter) on top of a BERT base model. For text classification, we use the RoBERTa model as the base model.

A PEM is fine-tuned for a trait such that it responds to input questions asked to it as if an individual with that trait would respond. Similarly, post-composition, the composite PEM responds to input questions in alignment with the resultant personality obtained from the composing (combining) trait (PEMs).

Synthetic Dataset Generation In order to train the individual trait PEMs, we synthetically generate 8 trait datasets (one per trait) by prompting ChatGPT-4. A generated data sample, from ChatGPT-4, for a given trait dataset, is in the format {'question': <*question*>, 'answer': <*answer*>}. A <*question*> corresponds to a question (statement)[2] relevant to the given trait (dataset). For zero-shot classification, <*answer*> is the response in the 7 classes that aligns with the trait. An example sample for the **Introversion** trait dataset is shown below,

[2]Note that the question can also just be a simple statement to be classified

question: I find it energizing to be around other people for long periods of time.
answer: Strongly Disagree

For text generation, <*answer*> is the response to the question provided that aligns with the trait. An example sample for the **Introversion** trait dataset is shown below,

question: Describe a perfect weekend for you.
answer: A perfect weekend for me would involve spending time alone or with one or two close friends, perhaps exploring a new book or diving into a personal project.

We generate 154 samples per trait dataset. ChatGPT-4 is prompted [3] as below for zero-shot classification and text generation respectively, as an example, to generate trait dataset for the trait **Judging**,

prompt: Draft me 154 question with answers being in the 7 classes which can be used to distinguish between **Judger** and **Perceiver** MBTI traits. Save the questions in a JSON format, responding as a **Judger**. Generate examples such that there are at least 5 samples for each of the 7 classes.

prompt: Draft me 154 question with answers which can be used to check whether the person is a **Judger** or a **Perceiver**. Save the questions in a JSON format, responding as a **Judger**.

PEM Composition Instead of using eq. 6 directly to compose trait PEMs, we use the below,

$$\mathbf{\Theta}_P = \sum_{i=1}^{4} \lambda_i \boldsymbol{\theta}_{T_i} \tag{7}$$

In eq. 7, the hyper-parameter set $\{\lambda_i\}$ is selected such that each $\lambda_i \in (0, 1)$ and $\sum_i \lambda_i = 1$. An optimal set of $\{\lambda_i^*\}$ is obtained by sweeping over a set of possible values (following the constraints, with a granularity of 0.1) and selecting the $\{\lambda_i\}$ maximizing the (personality) evaluation score.

[3]Not the exact prompt. We add a random seed and a temperature parameter to the prompt to ensure that samples are not repeated

Trait	Baseline (%)	LoRA (%)	IA3 (%)
Extroversion	45	70	57
Introversion	55	72	72
Intuitive	42	66	68
Sensor	58	80	80
Thinker	54	64	61
Feeler	46	57	56
Judger	49	64	64
Perceiver	51	65	61

Table 1: MBTI Trait Results

Personality	LoRA (%)					IA3 (%)				
	E/I	S/N	T/F	J/P	✓/✗	E/I	S/N	T/F	J/P	✓/✗
ENFJ	54	51	52	60	✓	54	52	68	71	✓
ENFP	56	63	64	67	✓	63	58	56	56	✓
ENTJ	53	51	61	64	✓	59	54	53	58	✓
ENTP	63	58	53	67	✓	66	66	58	31	✗
ESFJ	79	51	52	51	✓	56	57	56	67	✓
ESFP	56	54	73	67	✓	54	52	53	51	✓
ESTJ	64	69	53	67	✓	63	64	57	67	✓
ESTP	53	52	61	83	✓	53	57	54	51	✓
INFJ	40	59	66	67	✗	53	51	56	64	✓
INFP	44	70	54	63	✗	72	41	52	69	✗
INTJ	57	45	71	64	✗	58	51	52	65	✓
INTP	64	55	59	65	✓	65	51	64	60	✓
ISFJ	53	66	56	71	✓	53	63	51	58	✓
ISFP	53	60	60	64	✓	69	66	43	67	✗
ISTJ	66	78	53	67	✓	57	64	76	57	✓
ISTP	69	58	55	76	✓	64	63	58	71	✓

Table 2: MBTI Personality Results. The scores at a given row correspond to the traits of that personality (ENTJ row scores are for E, N, T and J traits respectively). Note that ✓/✗ indicates whether the composed PEM correctly aligns with the MBTI personality it was composed for or not.

Evaluation We evaluate the fine-tuned individual trait PEMs and the composed personality PEMs using the personality quiz questionnaire at https://www.16personalities.com/free-personality-test. The questionnaire comprises of 60 questions that a user responds to (with a response in one of the 7 classes) to identify their personality type. The questionnaire outputs the users personality type based on the responses, along with a percentage score for each trait (per dichotomy).

The questionnaire is used as input to the PEMs and the output responses are recorded and fed to the online questionnaire via an automated Selenium pipeline to automate evaluation.

Results The results for zero-shot classification are summarized in Table 1 and 2 for individual trait PEMs and personality PEMs, respectively. In Table 1, we see that compared against a baseline BERT not fine-tuned via PEFT, the PEMs obtained for LoRA and IA3, both, outperform the baseline. Further, the trait PEMs correctly align with the trait

they were trained for (seen by a score > 50).

For the personality PEMs, as seen in Table 2, the composed PEMs, correctly align with the personalities that they were composed for in most cases. Instances of mis-alignment are few and appear due to at most one misaligned trait.

The output generated by the baseline and the PEM-equipped text generation model as an extrovert is listed as follows:

> *Prompt*: "Describe about your personal relationships."
>
> *Baseline model response*: "Write a list of 5 specific and specific ways you encourage and support your family members, friends, colleagues, or business associates ..."
>
> *PEM-equipped model response*: "I thrive on connecting with others and building meaningful bonds ..."

Demo We further create an API using Streamlit [4] that allows a user to ask questions to our PEMs and interact with them. The API allows both question-answer style interaction as well as chatbot style conversational interaction.

6 Conclusion and Future Work

From our results, we conclude that PEFT and PEMs are effective domain learners and they can be merged for distribution generalization without additional fine-tuning to represent specialized composites for composite domains.

For future work, additional compositions on PEMs can be explored that allow PEMs to unlearn, transfer domain expertise, multi-task and detoxify toxic language models. Trait and personality PEMs can be human evaluated and tested over a population to validate the findings of this work.

[4] API Demo Video: https://tinyurl.com/csci544-group-37

7 Division of Labor

Mann Patel Ideated the project and led the group discussions, automated the evaluation process via a Selenium pipeline and formalized the algorithm, wrote adapter training pipeline, and ran experiments for composition associated with MBTI, worked on project presentation.

Divyajyoti Panda Wrote the evaluation script, performed PEM composition for IA3 through a script to select optimal $\{\lambda_i\}$ for each (personality) PEM, generated the results in Table 1 and 2 for IA3, generated synthetic trait datasets.

Hilay Mehta Wrote the Streamlit API pipeline to allow for users to interact with the trait and personality PEMs, generated synthetic trait datasets using few shot prompting, and setting up the pipeline for merging & training the PEMs.

Parth Patel Performed PEM composition for LoRA through a script to select optimal $\{\lambda_i\}$ for each (personality) PEM, generated the results in Table 1 and 2 for LoRA, assisted with debugging the Streamlit API pipeline, generated synthetic trait datasets, worked on project presentation.

Dhruv Parikh Wrote the training script for training LoRA and IA3 PEMs, helped with debugging text classification pipeline for training the individual PEMs, prepared the project presentation slides and wrote the project final report.

References

Gregory J. Boyle. 1995. Myers-briggs type indicator (mbti): Some psychometric limitations. *Australian Psychologist*, 30(1):71–74.

Vitor dos Santos and Ivandre Paraboni. 2022. Myers-briggs personality classification from social media text using pre-trained language models. *JUCS - Journal of Universal Computer Science*, 28(4):378–395.

Daniel Fernau, Stefan Hillmann, Nils Feldhus, and Tim Polzehl. 2022. Towards Automated Dialog Personalization using MBTI Personality Indicators. In *Proc. Interspeech 2022*, pages 1968–1972.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. *Preprint*, arXiv:1902.00751.

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

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2023. Editing models with task arithmetic. *Preprint*, arXiv:2212.04089.

Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. 2023. Dataless knowledge fusion by merging weights of language models. *Preprint*, arXiv:2212.09849.

Sedrick Scott Keh and I-Tsun Cheng. 2019. Myers-briggs personality classification and personality-specific language generation using pre-trained language models. *Preprint*, arXiv:1907.06333.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Preprint*, arXiv:2205.05638.

Michael Matena and Colin Raffel. 2022. Merging models with fisher-weighted averaging. *Preprint*, arXiv:2111.09832.

Yash Mehta, Samin Fatehi, Amirmohammad Kazameini, Clemens Stachl, Erik Cambria, and Sauleh Eetemadi. 2020. Bottom-up and top-down: Predicting personality with psycholinguistic and language model features. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 1184–1189.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *Preprint*, arXiv:1301.3781.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. Adapterfusion: Non-destructive task composition for transfer learning. *Preprint*, arXiv:2005.00247.

Ricardo Lazo Vásquez and José Ochoa-Luna. 2021. Transformer-based approaches for personality detection using the mbti model. In *2021 XLVII Latin American Computing Conference (CLEI)*, pages 1–7.

Yaqing Wang, Sahaj Agarwal, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan Awadallah, and Jianfeng Gao. 2022. Adamix: Mixture-of-adaptations for parameter-efficient model tuning. *Preprint*, arXiv:2205.12410.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. *Preprint*, arXiv:2203.05482.

Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. 2023. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. *Preprint*, arXiv:2312.12148.

Jinghan Zhang, Shiqi Chen, Junteng Liu, and Junxian He. 2023. Composing parameter-efficient modules with arithmetic operations. *Preprint*, arXiv:2306.14870.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A survey of large language models. *Preprint*, arXiv:2303.18223.