Enhancing Label-Free Cross-Task Generalization via Stability and Confidence Aware LoRA Composition

Anonymous ACL submission

Abstract

Recent advances in parameter-efficient fine-tuning have established Low-Rank Adaptation (LoRA) as a commonly used technique for adapting large language models (LLMs) to downstream tasks. Building on LoRA's modularity and low resource requirements, composing multiple LoRA modules has emerged as a promising approach to enhance cross-task generalization. However, in label-free scenarios, two major challenges hinder effective unsupervised LoRA composition: (1) the lack of principled criteria for selecting relevant modules in the absence of task-specific information for unseen tasks, and (2) the reliance on training with task-specific examples to optimize module integration coefficients. To tackle these issues, we propose a two-stage method for stability- and confidence-aware LoRA composition, aimed at enhancing label-free cross-task generalization. In the first stage, we assess module robustness via stability analysis—introducing controlled noise during generation and identifying modules whose outputs remain confident and consistent. In the second stage, we generate pseudo-training data from selected modules and perform confidence-guided filtering to ensure high-quality supervision for model adaptation. Empirical results on seven diverse evaluation tasks demonstrate that our approach improves average ROUGE-L scores and outperforms existing label-free merging baselines on 42% of tasks, showcasing its effectiveness in generalizing to unseen settings without labeled data. Our code is available at https://github.com/8k2aax0e/new-LoRA-Composition-method.

1 Introduction

In recent years, LLMs such as GPT-3 (Brown et al., 2020), Qwen (Bai et al., 2023) and DeepSeek R1(DeepSeek-AI et al., 2025), have significantly advanced the automation of natural language understanding and generation tasks. Despite their impressive capabilities, LLMs still face limitations when adapting to unseen tasks without labeled data. While full fine-tuning offers a direct solution to adapt models to new tasks, it comes with prohibitive computational and storage costs, especially for large-scale models.

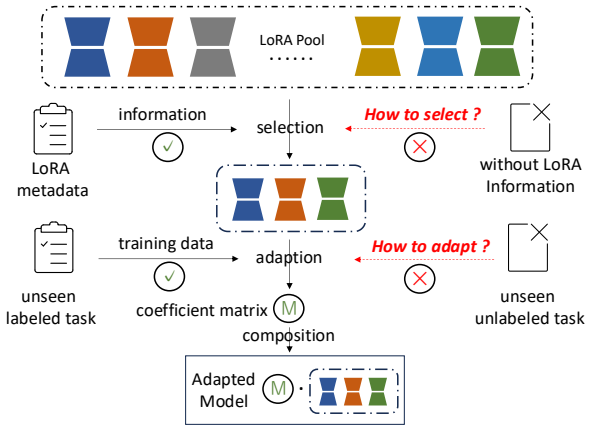


Figure 1: Supervised cross-task generalization enables the construction of task-specific models via module selection and fusion coefficient computation. In contrast, unsupervised cross-task generalization lacks module information and task training data, making it impractical to select appropriate modules or compute their fusion coefficients.

To mitigate this issue, parameter-efficient fine-tuning (PEFT) methods have been proposed, which allow only a small subset of parameters to be updated. Among these, LoRA (Hu et al., 2022) has gained significant popularity for its simplicity and effectiveness. By injecting lightweight trainable matrices into existing model layers, LoRA enables efficient adaptation without modifying the full model. The increasing adoption of LoRA has led to a growing collection of task-specific LoRA modules, many of which are publicly available.

Given the widespread adoption of LoRA in the community, a large number of pre-trained LoRA modules tailored for various tasks are now publicly available. For known tasks, these modules can of-

ten be directly reused without retraining. More recently, a promising direction has emerged: composing multiple LoRA modules to enhance cross-task generalization.

However, in unseen task settings—where no existing module has been explicitly trained—direct reuse becomes insufficient, and compositional adaptation is required. Some prior work, such as LoRAHub (Huang et al., 2024), attempts to address this by selecting relevant modules based on performance over probing examples. Yet this approach assumes access to labeled data or training loss, which is often unavailable. Moreover, many publicly shared LoRA modules lack detailed metadata, and unseen tasks frequently come without any adaptation examples. Consequently, label-free LoRA composition faces two key challenges: (1) the absence of principled, data-free criteria for selecting unlabeled modules relevant to an unseen task; and (2) the reliance on task-specific examples to optimize module integration coefficients, which undermines generalization when such examples are unavailable.

To address these challenges, we propose a **two-stage framework for stability and confidence-aware LoRA composition** to enhance **label-free cross-task generalization**. In the first stage, we perform stability-aware module selection by introducing controlled noise during response generation and analyzing output variations. Modules that consistently produce confident and robust outputs under perturbation are selected for composition. In the second stage, we synthesize pseudo-training data by generating responses with the selected modules. Then, we apply confidence-guided filtering based on perplexity voting among all modules to identify high-quality pseudo-labels, which are used to adapt the composed model.

Unlike prior approaches such as LoRAHub (Huang et al., 2024), which rely on probing examples or access to training loss, our method requires no supervised data or internal module metadata. Compared to unsupervised merging methods like AdaMerging (Yang et al., 2024), which are often limited to classification tasks, our approach is designed for broader applicability across diverse NLP settings. We validate our approach on seven evaluation tasks and show that it not only improves average ROUGE-L scores but also achieves the best performance on 42% of the tasks, demonstrating the effectiveness of our strategy for label-free, cross-task model generalization.

Our contributions can be summarized as follows:

- We propose a novel stability-aware LoRA selection method that identifies robust modules without relying on any external supervision or training task datasets.
- We introduce a confidence-aware pseudo-data generation and filtering mechanism that enables effective adaptation of composed LoRA modules in a fully unsupervised manner.
- Our method improves label-free generalization and outperforms existing unsupervised LoRA composition baselines across a range of NLP tasks.

2 Related work

2.1 Model Composition

Model composition in this context includes both full model composition and LoRA module composition. Most model composition methods can be applied to LoRA fusion as well. Depending on the complexity, composition methods can be broadly categorized into three types. The first category consists of simple and intuitive parameter manipulation techniques, such as Task Arithmetic (Ilharco et al., 2023) , TIES-Merging (Yadav et al., 2024), Arrow (Ostapenko et al., 2024), Channel Merging(Zhang et al., 2024) and DARE (Yu et al., 2024). Although these methods are simple and effective in most cases, their primary approach is weight disentangling (Ortiz-Jimenez et al., 2023). As a result, they may not show significant improvements when adapting to tasks with strong structural regularities. The second category involves static composition methods that train parameters based on tasks for adaptation, such as AWD (Xiong et al., 2024) and LoRAHub (Huang et al., 2024). The most complex category includes methods that use gating mechanisms, like LoRAMoE (Dou et al., 2024) and LoRA-Flow (Wang et al., 2024). Whether using static methods or gating-based approaches, these techniques typically require labeled data to function effectively, making them unsuitable for unsupervised settings. Here's the refined version:

There have been attempts to adapt methods for unsupervised tasks, such as AdaMerging (Yang et al., 2024), which uses Shannon entropy in place of cross-entropy. However, this approach is primarily suited for classification tasks. When applied

to natural language generation tasks, it struggles due to the large search space for possible answers, limiting its effectiveness. Our method, through pseudo-data generation and selection, addresses the challenges faced by AdaMerging, offering a more suitable solution for unsupervised adaptation.

2.2 Pseudo-data Generation & Filtering

Pseudo-data generation is widely adopted to address data scarcity, enhancing model generalization through synthetic data (Lee et al., 2013). Common methods include knowledge distillation (Hinton et al., 2015), where a larger model generates data for training a smaller one, and back-translation (Edunov et al., 2018), which enriches parallel corpora by translating monolingual data. Self-instruct (Wang et al., 2022a) extracts instruction-tuning data from a model's own outputs, while LawGPT (Zhou et al., 2024) generates legal texts using task-specific prompts. However, pseudo-data often suffers from quality issues and inherent biases, necessitating effective filtering mechanisms.

Filtering strategies aim to mitigate these biases. Manual filtering, as in InstructGPT (Ouyang et al., 2022), relies on human annotations but is resource-intensive. Single-model self-validation methods, such as SelfCheckGPT (Manakul et al., 2023), detect hallucinations through consistency checks but are limited by the model's inherent biases. Multi-model comparison approaches, like LM vs. LM (Cohen et al., 2023), validate data through cross-checking but demand significant computational resources. Dedicated scoring models, such as RAFT (Dong et al., 2023), rank data using external reward models but lack cross-task adaptability.

In contrast, our method leverages multiple LoRA modules, addressing the limitations of single-model and multi-model approaches. By evaluating confidence across LoRA modules, we reduce computational overhead and mitigate biases without relying on external models, offering a more efficient and generalizable solution.

3 Method

This section outlines our method for adapting a base model to unseen tasks using LoRA and its composition, without relying on labeled data. We begin by identifying the most stable and robust LoRA modules based on their performance and consistency with task-specific data. Next, we describe how pseudo-data is selected for adaptation using a confidence-based approach. Finally, we explain the layer-level fusion of selected LoRA modules to fine-tune the model, enabling effective generalization to the target task.

3.1 Preliminaries

LoRA (Hu et al., 2022) is a PEFT method that adapts LLMs by introducing low-rank parameterizations, while keeping the original weight matrix $W_0 \in \mathbb{R}^{m\times d}$ frozen. It adds two trainable low-rank matrices: $A \in \mathbb{R}^{r\times d}$ and $B \in \mathbb{R}^{m\times r}$, and learns only the update ΔW. The modified weight matrix is:

$$\mathcal{M} \leftarrow W \tag{1}$$

LoRA Composition refers to the concept of combining multiple LoRA modules to adapt the model to unseen tasks. This is typically achieved by using a set of weights $W_{comp} = \{\lambda_1, ..., \lambda_n\}$, where each LoRA weight A_i, B_i is scaled by a factor λ_i and fused into a single weight matrix ΔW.

$$\Delta W = \sum_{i=1}^{N} \lambda_i B_i A_i \tag{2}$$

3.2 Framework

Our objective is to adapt base LLM $\mathcal{M}_{base}$ to a novel, unseen task T using a pool of pre-trained LoRA modules $\mathcal{L}_{pool} = \{L_1, \ldots, L_i\}$, where each LoRA module L is a fine-tuned LoRA module. The task T consists of an unlabeled dataset $T = \{x_1, \ldots, x_n\}$, where each x represents a query from T. Our approach enables generalization to new tasks without task-specific supervision by strategically selecting and combining the most relevant LoRA modules through a two-step process:

First, we randomly select j task examples $T_e = \{x_1, ..., x_j\}$ where $T_e \subseteq T$ to represent the task. Each LoRA model $\mathcal{M}_{L_i}$ composed of the base model $\mathcal{M}_{\text{base}}$ and LoRA modules L_i, processesT_e to generatea set of response pairs $\mathcal{D}_{L_i} = \{(x_1, \hat{y}_1), \ldots, (x_k, \hat{y}_k)\}$, where each $\hat{y}_i$ is the response of x_i from $\mathcal{M}_{L_i}$. We then introduce perturbations to each x_i using text attack method (Morris et al., 2020), replacing random words in the query with their synonyms to obtain a perturbed variant$\tilde{x}_i$, and re-evaluate the response pairs $\tilde{\mathcal{D}}$to assess the stability of each LoRA module with respect to T . Based on this stability measure, we select the top$N$most stable LoRA modules to form a task-specific fusion group $\mathcal{L}_{\text{stable}} = \{L_1, ..., L_N\}$.

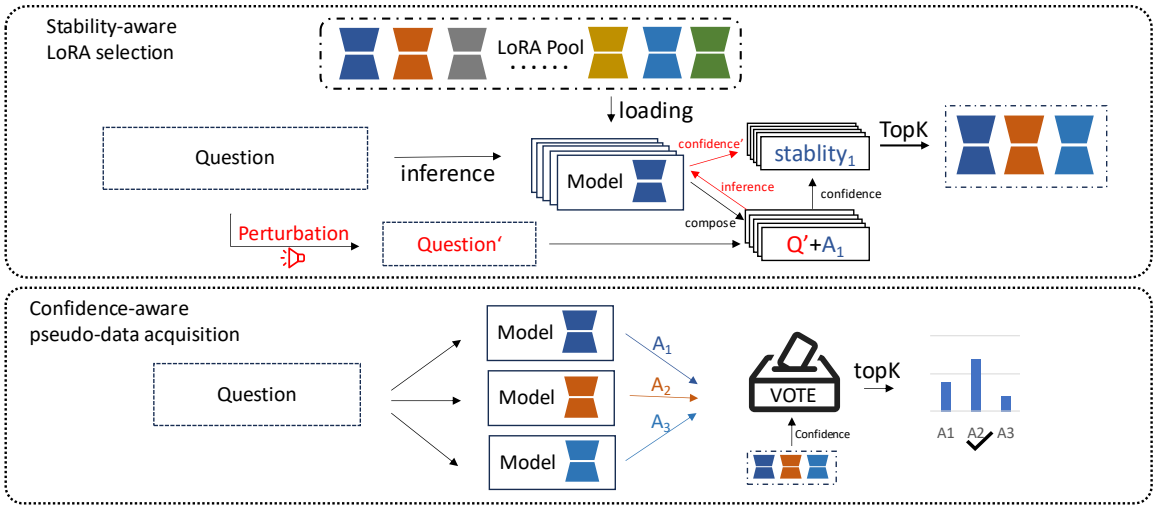


Figure 2: The workflow diagram illustrates two key stages: LoRA selection and pseudo-data acquisition. During the LoRA selection stage, we combine the pseudo-label generated by each LoRA module with a perturbed version of the input question, and then use the same LoRA module to recompute the confidence score. The difference between the original and recomputed confidence scores is defined as the module's stability, which serves as the criterion for selecting LoRA modules. In the pseudo-data selection stage, the generated pseudo-data are passed through all LoRA modules to compute their confidence scores, and the sum of these scores is used to identify high-quality pseudo-data.

Next, using the selected example set T_e, we sample pseudo-data points from a chosen LoRA module L_i in the selected modules pool $\mathcal{L}_{\text{stable}}$. These pseudo-data points are then evaluated by the other LoRA modules in $\mathcal{L}_{\text{stable}}$ for confidence scoring. This process is repeated N times, and all pseudo-data along with their confidence scores are aggregated. From this pool, we select the top K highest-scoring data, denoted as $\mathcal{D}_{\text{confident}} = \{(x_1, \hat{y}_1), \ldots, (x_K, \hat{y}_K)\}$. and we utilize $\mathcal{L}_{\text{stable}}$ and $\mathcal{D}_{\text{confident}}$ to fine-tune and adapt the LoRA modules for the target task. This approach ensures a effective adaptation process, even in the absence of labeled data.

The following provides a more detailed description of the method.

3.3 Noise Injection and Stability-Based LoRA Selection

To select the most robust LoRA modules in $\mathcal{L}_{pool}$ for task T, we evaluate each $x \in T_e$ and apply a greedy search to generate the most confident response $\hat{y}$ for each LoRA adapted Model $\mathcal{M}_{L_i}$ which composed by $L_i \in \mathcal{L}_{pool}$. The confidence of $\mathcal{M}_{L_i}$ for $(x, \hat{y})$ is given by: $P((x, \hat{y})|\hat{y}^{<(t)}, \mathcal{M}_{L_i})$, where $(x, \hat{y}^{<(t)})$ represents the preceding tokens of $(x, \hat{y})$, defined as: $(x, \hat{y}^{<(t)}) = \{x^{(1)}, \ldots, x^{(n)}, \hat{y}^{(1)}, \ldots, \hat{y}^{(t)}\}$.

To quantify model confidence, we compute the average conditional probability of (x, y) given its preceding tokens across all time steps t:

$$\mathrm{C}(x, y, \mathcal{M}_{L_i}) = \frac{1}{t} \sum_{k=1}^{t} P((x, y)|y^{<(k)}, \mathcal{M}_{L_i}). \tag{3}$$

Next, we introduce noise into the input by applying the existing perturbation method noise(x) (Morris et al., 2020), replacing a certain proportion of words in x with their synonyms. This process generates N perturbed version $\tilde{x}^n = \text{noise}(x)$.

To assess the stability of each LoRA module, we compute the absolute difference in confidence between the original and perturbed inputs:

$$d(x, \tilde{x}, \hat{y}, \mathcal{M}_{L_i}) = \mathrm{C}(x, \hat{y}, \mathcal{M}_{L_i}) - \mathrm{C}(\tilde{x}, \hat{y}, \mathcal{M}_{L_i}) \tag{4}$$

We then compute the average confidence change across all perturbations:

$$\bar{d} = \frac{1}{N} \sum_{k=0}^{N} d(x, \tilde{x}^k, \hat{y}, \mathcal{M}_{L_i}) \tag{5}$$

The stability score, denoted as $\text{score}^{L_i}_{\text{stable}}$, is defined as the mean absolute deviation from $\bar{d}$

$$\text{score}^{L_i}_{\text{stable}} = \frac{1}{N} \sum_{k=0}^{N} |d(x, \tilde{x}^k, \hat{y}, \mathcal{M}) - \bar{d}| \tag{6}$$

Finally, after multiple noise samplings and averaging the resulting stability scores, we select the N LoRA modules with the smallest stability scores. The selected set of robust modules is denoted as: $\mathcal{L}_{stable} = \{L_1, \ldots, L_N\}$.

3.4 Pseudo-Data Generation and Confidence-Based Selection

To generate high-quality pseudo-data for task adaptation, we employ a sampling-and-filtering approach. For each $x \in T_e$, we perform N diverse random samplings using the model M_{L_i} generated by LoRA module L_i, resulting in a pseudo-data set $\mathcal{D}_{L_i} = \{(x_1, \hat{y}_1^1), (x_1, \hat{y}_1^2), \ldots, (x_n, \hat{y}_n^N)\}$. We then compute the confidence score for each $(x, \hat{y}) \in \mathcal{D}_{L_i}$ $\text{score}_{\text{conf}}^{(x,\hat{y})}$ using other LoRA modules $L_j \in \mathcal{L}_{\text{stable}}$, given by the following formula:

$$\text{score}_{\text{conf}}^{(x,\hat{y})} = \frac{1}{T} \sum_{j=0}^{N} \sum_{t=0}^{T} \text{C}(x, \hat{y}^{(t)}, M_{L_j}) \quad (7)$$

where $L_j \neq L_i$. After sampling across all modules in $\mathcal{L}_{\text{stable}}$, we select the top-$K$ pseudo-data pairs of $(x, \hat{y})$ as a set $\mathcal{D}_{\text{confident}} = \{(x_1, \hat{y}_1), \ldots, (x_1, \hat{y}_K), \ldots, (x_N, \hat{y}_K)\}$ with the highest $\text{score}_{\text{conf}}$ as training data, and use them to adapt $\mathcal{L}_{\text{stable}}$.

3.5 LoRA Module Composition and Adaptation

At this point, we have $\mathbb{L}_{\text{stable}}$ and $\mathcal{D}_{\text{confident}}$. Next, we construct a weight matrix $W_{\text{comp}} \in \mathbb{R}^{\text{layernum} \times \text{LoRAnum}}$ for layer-level model fusion, resulting in the composed model M_{comp}. The adaptation process can be expressed as follows:

$$\hat{W}_{\text{comp}} = \underset{W_{\text{comp}}}{argmax} \left\{ \mathcal{L}(\mathcal{M}_{\text{comp}} | \mathcal{D}_{\text{confident}}) \right\} \quad (8)$$

where the likelihood loss function is defined as:

$$\mathcal{L}(\mathcal{M}_{\text{comp}} | \mathcal{D}_{\text{confident}}) = \sum_{i=1}^{N} P(y_i | \mathcal{M}_{\text{comp}}) \quad (9)$$

$\hat{W}_{\text{comp}}$ is the final adapted model for task T.

4 Experiments

In this section, we first describe the experimental setup, including the training and evaluation datasets, along with the classification of evaluation categories used in the main results. We then detail the experimental process, covering LoRA module training, selection, and the hyperparameters for pseudo-data generation and filtering. Next, we introduce the baseline methods used for comparison to highlight the effectiveness of our approach. Finally, we present the main results and ablation studies.

4.1 Experimental Setup

4.1.1 Datasets and Task Partitions

The Super-NaturalInstructions (SNI) dataset (Wang et al., 2022b) includes 76 distinct task categories and 1,616 fine-grained task instances, each guided by specific instructions. It serves as a strong benchmark for evaluating cross-task generalization. In our experiments, we split the dataset into training and testing sets: 56 categories (about 75%) are used for training, and the remaining 20 categories (25%) are reserved for evaluating performance on unseen tasks.

For each training category, we train a separate LoRA module using 10,000 randomly sampled instances to ensure broad task coverage. During evaluation, up to 2,000 instances are sampled per category from the evaluation set to measure task-specific performance efficiently. We use ROUGE-L as the primary evaluation metric across all tasks.

For clarity, the 20 evaluation categories are grouped into 7 broader tasks. Performance is averaged across categories within each task, and the overall average is also computed at the category level.

4.1.2 Implementation Details

Base Model Configuration. We employ LLaMA-7B (Touvron et al., 2023) as the base model in our experiments. This open-source foundation model is widely recognized for its strong performance across a broad range of benchmarks, making it an ideal starting point for our adaptation approach.

LoRA Training. To implement LoRA, we apply it to all linear projection layers within the attention mechanism, specifically targeting the query (q), key (k), value (v) layers, as well as the feed-forward layers of the attention mechanism. The adaptation rank is set to 16, and a dropout rate of 0.01 is used to prevent overfitting. We optimize the model using the AdamW optimizer with a learning rate and a batch size of 16, balancing computational efficiency and training stability. Due to hardware limitations, we restrict the maximum context length to 1024 tokens.

LoRA	Data	Veri.	Quez.	Gen.	Ana.	Inter.	Logi.	Cls.	Avg.
empty	zero-shot	7.34	23.08	10.61	6.26	7.29	7.87	7.27	10.23
all	TA(Ilharco et al., 2023)	34.94	**23.15**	**21.33**	17.43	32.06	9.88	31.08	24.5
random	random	38.58	17.95	17.58	18.38	31.79	9.63	30.41	23.53
	all	36.48	19.53	16.41	17.33	31.17	10.04	29.53	22.92
	TA(Ilharco et al., 2023)	39.3	15.46	18.82	18.2	32.79	8.51	29.15	23.39
	AM(Yang et al., 2024)	39.0	14.36	18.5	18.6	34.28	8.45	28.49	23.33
	confidence	38.98	17.74	18.49	18.08	31.95	9.5	30.32	23.69
entropy	random	39.0	14.36	18.5	**18.6**	34.28	8.45	28.49	23.33
	all	29.53	18.46	15.92	11.94	29.89	**10.28**	28.0	20.48
	TA(Ilharco et al., 2023)	25.24	15.04	16.61	10.68	28.93	9.81	29.7	19.26
	AM(Yang et al., 2024)	26.56	13.89	17.2	10.5	30.9	8.52	29.89	19.56
	confidence	28.38	15.61	17.07	11.52	30.64	9.76	29.86	20.3
stability	random	38.98	17.74	18.49	18.08	31.95	9.5	30.32	23.69
	all	37.87	22.72	19.54	11.9	34.05	7.48	34.95	24.13
	TA(Ilharco et al., 2023)	37.44	21.07	20.24	11.92	30.13	10.21	30.75	23.23
	AM(Yang et al., 2024)	35.94	19.62	19.8	10.19	26.68	6.77	30.7	21.57
	confidence	**40.92**	22.32	20.26	12.01	**34.58**	8.37	**35.9**	**24.95**

Table 1: Performance Comparison of Different Methods. The first two columns represent the LoRA selection method and the pseudo-data selection method, respectively, and the last column represents the average performance across 20 categories.

Token Replacement and Module Selection. To select the most stable LoRA modules, we evaluate each candidate using five randomly chosen queries. Stability is assessed by replacing 40% of the tokens with synonyms using the TextAttack framework (Morris et al., 2020) and measuring the change in output confidence. The top 10 modules with the smallest confidence drop are selected for composition.

During data synthesis, each selected module generates 10 pseudo-training examples per query, resulting in a final dataset of 1,000 examples from 10 different queries.

Pseudo-Data Acquisition. To encourage diversity, we use sampling instead of beam search, setting the temperature to 0.3 and applying a repetition penalty of 2 (Keskar et al., 2019). This setup helps generate varied and non-repetitive responses.

Model Merging and Adaptation. we set the learning rate to 5×10^{-4}, with the number of epochs set to 5 and a batch size of 10. This configuration ensures that the adaptation process is completed within 5 minutes on a single A100 80GB GPU.

4.2 Baselines

LoRA Module Selection. We compare our stability-based selection with three alternatives: random selection, minimum Shannon entropy selection inspired by AdaMerging (Yang et al., 2024), and a no-selection baseline.

Pseudo-Data Selection and Task Adaptation. We evaluate our method against several baselines, including zero-shot reasoning, Task Arithmetic (TA) (Ilharco et al., 2023), and hierarchical AdaMerging (AM). Additionally, we report results for variants using random selection and no selection to provide further comparison.

For Task Arithmetic, we set the combination coefficient to $\frac{1}{N}$, where N is the number of LoRA modules. In hierarchical AdaMerging, the learning rate for learning combination coefficients is set to 5×10^{-4}. To ensure fair comparison, we fix the number of training epochs to 50, matching the training steps used in our proposed method.

4.3 Main Results

Our main results, presented in Table 1, demonstrate that the two-stage approach—combining stability-based LoRA selection with confidence-based data selection—achieves state-of-the-art performance, reaching an average accuracy of 24.95% across all tasks. This marks a 144% relative improvement over the zero-shot baseline (10.23% $\rightarrow$ 24.95%). Our method consistently outperforms existing fusion techniques, delivering absolute gains of 6.7% over Task Arithmetic (23.39%) and 15.7% over AdaMerging (21.57%). Notably, it sets new state-of-the-art results in Verification(40.92%, +1.6%)

and Classification (35.9%, +5.8% over full-data baselines). Moreover, it surpasses Task Arithmetic in 5 out of 7 tasks, with substantial improvements of 3.8% in Verification and 6.8% in Classification. While our method consistently outperforms zero-shot learning across all tasks, Logic remains particularly challenging, with all approaches scoring below 10.28%.

4.4 Ablation Studies

4.4.1 The Efficiency of LoRA selection

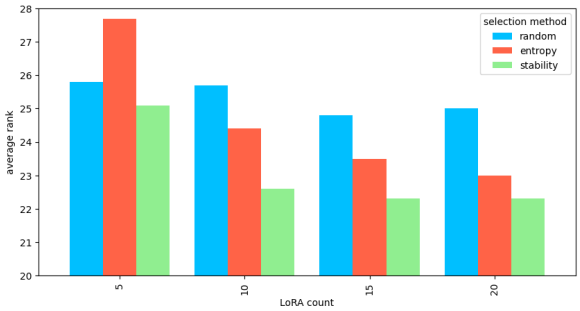


Figure 3: Average rank for selection methods

To better demonstrate the effectiveness of LoRA selection, we evaluate 54 trained LoRA modules across 20 categories per test. For each task, we measure the performance of each LoRA module (detailed in the Appendix Figure **??**) and rank them in descending order based on their performance. The average rankings are presented in Figure 3. The results clearly show that, compared to random selection and entropy-based selection, our stability-based method consistently selects LoRA modules with higher average ranks, highlighting its effectiveness.

5 Analyze and Discussion

In this section, we analyze how different LoRA module selection strategies impact final model performance and investigate the underlying reasons for the observed variations. We also provide a detailed account of our pseudo-data generation process and discuss why the performance of generated pseudo-data often differs significantly from that of the final adapted models.

5.1 The Efficiency of Pseudo-data selection

Table 1 shows that among the three LoRA selection methods, our confidence-based pseudo-data selection outperforms Task Arithmetic in 12 out of 14 tasks, except for random LoRA selection. Additionally, it achieves a higher overall average in two of the three methods. Within the stability-based approach, our method surpasses other data selection methods in 4 out of 7 tasks, further validating the effectiveness of our pseudo-data selection method.

5.2 Stability-Driven LoRA Selection and Task-Specific Capabilities

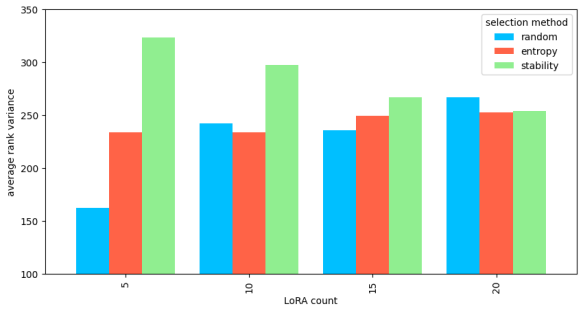


Figure 4: Average rank variance for selection methods

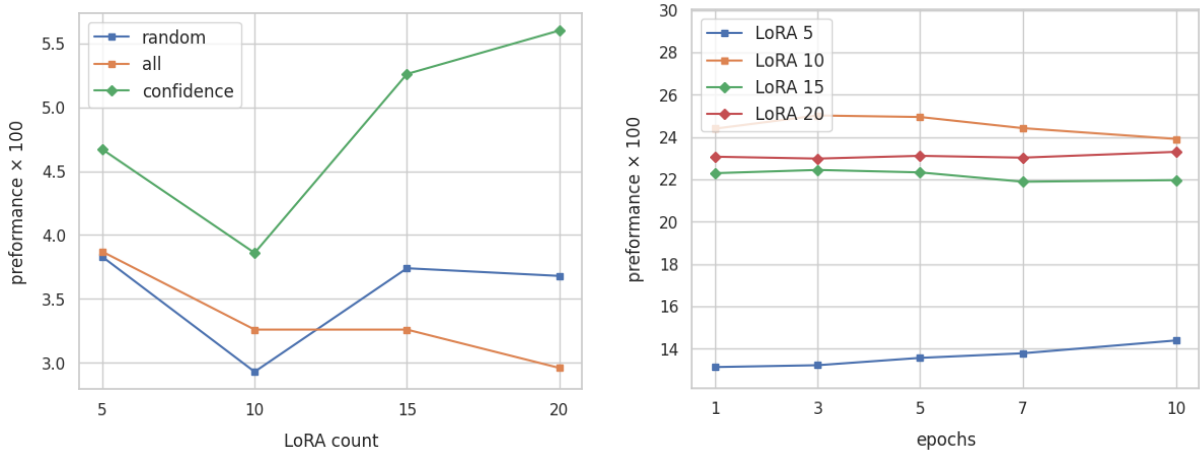
To explore the link between model stability and task performance, we analyzed the variance in average rankings across different LoRA fusion settings (5, 10, 15, and 20 modules), as shown in Figure 4. Together with Figure 3, the results show that our stability-based selection method tends to yield higher variance. This indicates that the selected modules are often more extreme in quality, as stability alone does not always reflect actual performance. While some low-confidence modules may be selected, stronger models typically show greater stability and confidence, making them more reliable and robust for task adaptation.

5.3 Results from the Synergistic Effects

We argue that using stability-based module selection or confidence-based pseudo-data filtering alone is not sufficient for optimal results. Stability-based selection often includes both strong and weak modules, which can degrade adaptation if used without filtering. However, this diversity enables the generation of more varied pseudo-data—some very good, some poor. By applying multi-module confidence filtering, we can effectively discard the low-quality examples and retain the high-quality ones. This combination leads to better overall performance and outperforms baseline methods.

5.4 Pseudo-Data Selection Effectiveness and Performance Gap After Adaptation

As shown in Figure 5a, our confidence-based method consistently outperforms both random selection and no selection across all four LoRA



(a) An evaluation of generated pseudo-data performance across various data selection methods and differing LoRA count.

(b) The impact of training epochs on the performance of different LoRA count.

Figure 5: The line charts illustrate the performance evaluation results using stability-based LoRA selection with 5, 10, 15, and 20 LoRA modules.

counts. However, comparing the confidence-based results in Figure 5a with those in Figure 5b reveals a significant performance gap: while the pseudo-data alone yields a score of only 3.86, the final adapted model (e.g., LoRA count = 10) achieves an average of 24.54. We attribute this to the importance of restoring the effectiveness of individual LoRA modules during adaptation. Proper pseudo-data adaptation helps fully leverage each merged module. Importantly, our confidence-based method selects data that better aligns with the target task. However, as shown by the trends for LoRA counts of 10, 15, and 20, excessive training can lead to overfitting, causing performance to decline over time.

5.5 Variance-Induced Performance Decline and Adaptive Mitigation in LoRA Configurations

In Figure 5b, we observe that the performance for LoRA count = 5 is noticeably lower than for other counts. We propose several possible explanations. As shown in Figure 4, the performance variance for LoRA count = 5 is high, suggesting that the stability-based selection method may have included both strong and weak LoRA modules. With only five modules, each one has a significant impact, so a poorly performing module can greatly reduce the overall performance of the merged model. This also explains the unique trend in Figure 5b: as training progresses, the influence of the weaker module diminishes, resulting in a steady performance improvement across epochs.

5.6 Impact of LoRA Count and Epochs on Task Performance

As shown in Figure 5b, the best performance is achieved when the LoRA count is 10. However, in Figure 5a, the pseudo-data performance for LoRA count = 10 is relatively poor. This suggests that pseudo-data selection alone is not the sole factor determining the performance of the merged model. Effective adaptation also requires selecting sufficiently strong LoRA modules to enhance performance, choosing an appropriate number of LoRA modules to mitigate the impact of poorly performing ones, and setting a suitable number of training epochs to properly adapt different LoRA modules.

6 Conclusion

In this paper, we introduced an unsupervised LoRA composition method to improve cross-task generalization without labeled data. Our approach selects robust LoRA modules based on stability, generates pseudo-data, and applies confidence-based filtering using other modules in the selected pool. A new model is then composed and adapted specifically for the target task. We evaluated our method on the SNI dataset, grouping 20 categories into 7 broader tasks. Through extensive analysis, including ablations and case-specific insights, we demonstrated that our approach outperforms baseline methods on 3 out of 7 tasks, highlighting its effectiveness for generalizing to unseen tasks in a label-free setting.

Limitation

Performance limitations. Our proposed module integration method is designed for fully label-free task settings, where the integration coefficients are trained solely based on pseudo-labels generated by the model itself. As a result, the learned coefficients primarily serve to align and combine modules, offering limited performance gains. In contrast, integration methods that utilize labeled data can typically achieve a significantly higher performance ceiling.

Requirement for module uniformity. In the current implementation, all LoRA modules to be integrated must share the same rank. Modules with differing ranks cannot be fused, which imposes a constraint on module compatibility and limits flexibility in practical deployment.

Instability induced by stability-based selection. This stability-based selection criterion may lead to the inclusion of both consistently helpful and consistently detrimental modules, potentially undermining task performance. Therefore, it cannot be used as a standalone module selection strategy and must be combined with our confidence-aware pseudo-data selection method.

References

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen Technical Report. *arXiv preprint*. ArXiv:2309.16609 [cs].

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. *arXiv preprint*. ArXiv:2005.14165 [cs].

Roi Cohen, May Hamri, Mor Geva, and Amir Globerson. 2023. Lm vs lm: Detecting factual errors via cross examination. *arXiv preprint arXiv:2305.13281*.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. 2023. Raft: Reward ranked finetuning for generative foundation model alignment. *arXiv preprint arXiv:2304.06767*.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Wei Shen, Limao Xiong, Yuhao Zhou, Xiao Wang, Zhi-

heng Xi, Xiaoran Fan, Shiliang Pu, Jiang Zhu, Rui Zheng, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024. LoRAMoE: Alleviating World Knowledge Forgetting in Large Language Models via MoE-Style Plugin. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1932–1945, Bangkok, Thailand. Association for Computational Linguistics.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. *arXiv preprint arXiv:1808.09381*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. 2024. Lorahub: Efficient cross-task generalization via dynamic loRA composition. In *First Conference on Language Modeling*.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2023. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations*.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL: A conditional transformer language model for controllable generation. *CoRR*, abs/1909.05858.

Dong-Hyun Lee et al. 2013. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 896. Atlanta.

Potsawee Manakul, Adian Liusie, and Mark JF Gales. 2023. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. *arXiv preprint arXiv:2303.08896*.

John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 119–126.

Guillermo Ortiz-Jimenez, Alessandro Favero, and Pascal Frossard. 2023. Task Arithmetic in the Tangent Space: Improved Editing of Pre-Trained Models. *arXiv preprint*. ArXiv:2305.12827 [cs].

Oleksiy Ostapenko, Zhan Su, Edoardo Maria Ponti, Laurent Charlin, Nicolas Le Roux, Matheus Pereira, Lucas Caccia, and Alessandro Sordoni. 2024. Towards modular llms by building and reusing a library of loras. *Preprint*, arXiv:2405.11157.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

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. *Preprint*, arXiv:2302.13971.

Hanqing Wang, Bowen Ping, Shuo Wang, Xu Han, Yun Chen, Zhiyuan Liu, and Maosong Sun. 2024. LoRA-flow: Dynamic LoRA fusion for large language models in generative tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12871–12882, Bangkok, Thailand. Association for Computational Linguistics.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022a. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Kuntal Kumar Pal, Maitreya Patel, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Savan Doshi, Shailaja Keyur Sampat, Siddhartha Mishra, Sujan Reddy A, Sumanta Patro, Tanay Dixit, and Xudong Shen. 2022b. Super-NaturalInstructions: Generalization via declarative instructions on 1600+ NLP tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5085–5109, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Feng Xiong, Runxi Cheng, Wang Chen, Zhanqiu Zhang, Yiwen Guo, Chun Yuan, and Ruifeng Xu. 2024. Multi-Task Model Merging via Adaptive Weight Disentanglement. *arXiv preprint*. ArXiv:2411.18729 [cs].

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. 2024. Ties-merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*, 36.

Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. 2024. Adamerging: Adaptive model merging for multi-task learning. In *The Twelfth International Conference on Learning Representations*.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. 2024. Language models are super mario: absorbing abilities from homologous models as a free lunch. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Mingyang Zhang, Jing Liu, Ganggui Ding, Xinyi Yu, Linlin Ou, and Bohan Zhuang. 2024. Channel merging: Preserving specialization for merged experts. *Preprint*, arXiv:2412.15283.

Zhi Zhou, Jiang-Xin Shi, Peng-Xiao Song, Xiao-Wen Yang, Yi-Xuan Jin, Lan-Zhe Guo, and Yu-Feng Li. 2024. Lawgpt: A chinese legal knowledge-enhanced large language model. *arXiv preprint arXiv:2406.04614*.

A Evaluate Category List and Its Classification

To facilitate evaluation, we divided the 20 test categories into 7 tasks based on specific principles. For details, see Table 2.

B knowledge Base LoRA performance

We evaluated the performance of the LoRA modules trained on 54 categories against the test categories. As shown in Figure 6, some tasks exhibit relatively lower performance, while a number of isolated high-performance spots are also observed. This indicates that there are notable differences in performance among the LoRA modules trained on different categories.

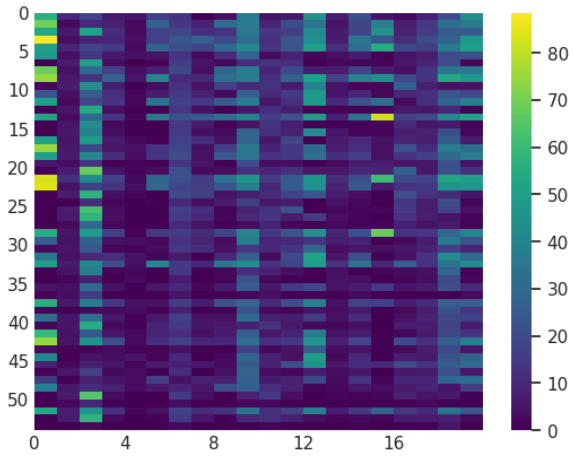


Figure 6: Heatmap of the trained LoRA module's rouge score of evaluation tasks. The x-axis represents evaluation tasks, while the y-axis corresponds to trained LoRA modules.

Task	Sign	Category
Verification	Veri.	fact verification, stereotype detection, stance detection Verify factual accuracy or detect bias.
Questioning	Quez.	question rewriting, question understanding , question answering Process, rewrite, or answer questions.
Generation	Gen.	title generation, dialogue generation, text matching, fill in the blank Generate or optimize text for specific goals.
Analysis	Ana.	pos tagging, word semantics, word analogy Analyze linguistic structure and semantics.
Interaction	Inter.	speaker identification, negotiation strategy detection, intent identification Manage dialogue and negotiation strategies.
Logic	Logi.	sentence ordering, text to code Handle structured reasoning tasks.
Classification	Cls.	text categorization, text quality evaluation Categorize or evaluate text quality.

Table 2: The table describes the classification of categories and their core tasks.