Model Merging using Geometric Median of Task Vectors

Aakash Gupta[1] and **Siddharth Gupta**[1]

[1]Vision and Language Group, Indian Institute of Technology Roorkee, Roorkee, Uttarakhand, India ,
aakash_g@ece.iitr.ac.in, siddharth_g@cs.iitr.ac.in

Abstract

Training high-performing large language models (LLMs) from scratch is an expensive and complex task. Model merging techniques offer a more computationally efficient alternative, where pretrained LLMs are fine-tuned on specific tasks and then combined to produce a versatile model capable of handling a broad range of tasks, including reasoning, coding, mathematics, conversation, and tool usage. Unlike traditional fine-tuning or ensemble methods, our approach to merging is less computationally intensive. We represent each fine-tuned model with a "Task Vector" relative to a pretrained "Base LLM" , derived from the LoRA (Low Rank Adaptation) weights of the fine-tuned models. By computing the geometric median of these task vectors in high-dimensional space using Weiszfeld's iterative algorithm and adding it to the "Base LLM" weights, we create a unified model that generalizes effectively across tasks. This efficient method achieves state-of-the-art performance on benchmark tests while reducing computational demands.
Code available at https://github.com/iMmOrTaL2121/geometric_median_llm_merging.git.

1 Introduction

Model merging, [9] or model fusion, combines the parameters of multiple models with unique strengths into a single, unified model. Unlike ensemble methods, which require high memory and processing power, model merging consolidates knowledge into one streamlined model, reducing computational costs, memory usage, and latency. This efficient technique enhances generalization across tasks and is ideal for resource-constrained or low-latency environments, as it does not require access to the original training data or extensive computation and training.

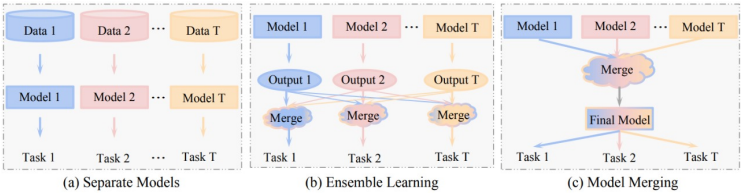

Figure 1: An illustration of the ensemble learning paradigm versus the model merging paradigm. (a) T separate models for T tasks, (b) Assemble T separate models for T tasks, (c) A merged model for T tasks.

Combining multiple fine-tuned LLMs with a pretrained "Base LLM" can yield a model capable of excelling across a wide range of tasks, including reasoning, coding, mathematics, conversation, and tool use. In our approach, we introduce a method that represents each fine-tuned LLM as a

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

"task vector" [4] relative to the Base LLM, utilizing LoRA [2] (Low-Rank Adaptation) weights. By computing the geometric median of these task vectors through Weiszfeld's iterative algorithm [1] and adding it to the Base LLM, we effectively merge the capabilities of different models, achieving strong performance on benchmark tests.

2 Background

2.1 Task Vectors

For our understanding, [4] a "task" is instantiated by a dataset and a loss function used for fine-tuning. Let $\theta_{\text{pre}} \in \mathbb{R}^d$ be the weights of a pre-trained base model, and $\theta_t^{\text{ft}} \in \mathbb{R}^d$ the corresponding weights after fine-tuning on task t. The task vector $\tau_t \in \mathbb{R}^d$ is given by the element-wise difference between θ_t^{ft} and θ_{pre}, i.e.,

$$\tau_t = \theta_t^{\text{ft}} - \theta_{\text{pre}}.$$

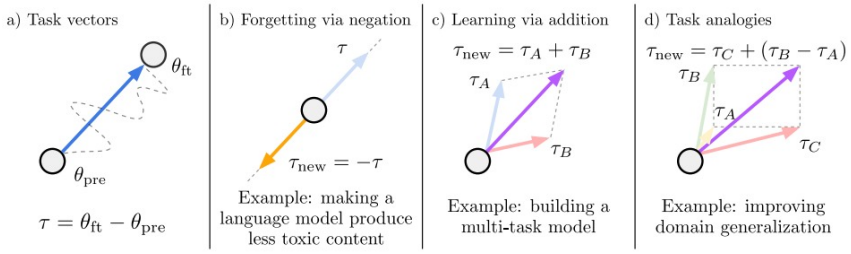

Figure 2:

For all operations, the model weights obtained by applying τ_{new} are given by

$$\theta_{\text{new}} = \theta + \lambda \tau_{\text{new}},$$

where the scaling term λ is determined using held-out validation sets. In our approach, we set $\lambda = 1$.

2.2 LoRA (Low-Rank Adaptation)

LoRA [2]is a method used to efficiently fine-tune large language models (LLMs) by injecting trainable low-rank matrices into the model's architecture. In our approach, we have specifically used PEFT [7](Parameter Efficient Fine-Tuned) models for model merging . This reduces the number of parameters that need to be updated during fine-tuning, making the process of finetuning faster and more memory-efficient, while maintaining performance on downstream tasks.

LoRA modifies a weight matrix W in the neural network as follows:

$$W' = W + \Delta W$$

where W is the frozen pretrained weight matrix of the base model, and ΔW is the low-rank adaptation learned during fine-tuning.

ΔW is expressed as:

$$\Delta W = BA$$

Here, $A \in \mathbb{R}^{r \times d}$ and $B \in \mathbb{R}^{d \times r}$, where r is much smaller than the dimension d of the original weight matrix W. This decomposition reduces the number of trainable parameters from d^2 to $2dr$, making the process fast and highly memory efficient.

- LoRA A: This is a low-rank matrix of size $r \times d$ that maps the lower-dimensional representation back to the original dimension d.
- LoRA B: This is a low-rank matrix of size $d \times r$ that is learned during fine-tuning. It projects the input (or intermediate representations) to a lower-dimensional space of rank r.

3 Method

The composition $\Delta W = BA$ allows for an efficient update of the original weight matrix W without needing to directly update W itself . The product of these two low-rank matrices approximates the weight update that would normally be learned by directly updating W.

In our approach, the fine-tuned LLM's we used had 23 encoder blocks and 23 decoder blocks, and each encoder or decoder block has LoRA A weights with dimension $A \in \mathbb{R}^{16\times 2048}$ and LoRA B weights with dimension $B \in \mathbb{R}^{2048\times 16}$.

3.1 Computing weights of merged LLM

A transformer block has many encoder and decoder blocks [8]. Also from section 2.2 , we can say that the matrix $\Delta W = BA$ is the corresponding task vector of a given fine-tuned LLM for a given encoder/decoder block (parameter matrix of the $W \in \mathbb{R}^{2048\times 2048}$) . Similarly we find task vectors for all fine-tuned LLMs for the same block. We flatten all these task vector matrices to a high dimensional vector $\mathbb{R}^{1\times 4194304}$ and then find the geometric median of all these flattened vectors treating each of these vectors as a point in multidimensional space to get a "net task vector" for that block which is finally added to the corresponding block in the pretrained base LLM after reshaping to the original size $\mathbb{R}^{2048\times 2048}$ of block's parameter .We are finding the geometric median , so that our model is able to optimally perform in all the tasks.

3.2 Geometric Median

The geometric median [1] is a point in multidimensional space that minimizes the sum of distances to a set of given points. It generalizes the concept of the median from 1D to higher dimensions.

Mathematically,given a set of points $X = \{x_1, x_2, \ldots, x_n\}$ in Euclidean space $\mathbb{R}^d$, the geometric median M is the point that minimizes the sum of Euclidean distances to the given points:

$$M = \arg\min_{y\in\mathbb{R}^d} \sum_{i=1}^{n} \|y - x_i\| \tag{1}$$

where:

- M is the geometric median (the point to be found).
- $\|y - x_i\|$ denotes the Euclidean distance between point y and x_i.

3.3 Weiszfeld Iterative Algorithm

The Weiszfeld algorithm [1] is an iterative method used to compute the geometric median of a set of points in multi-dimensional space. The geometric median minimizes the sum of Euclidean distances from a point to a set of given points. This is different from the arithmetic mean, which minimizes the sum of squared distances. Mathematically, the objective function being minimized is:

$$f(x) = \sum_{i=1}^{n} w_i \|x - p_i\|$$

Where:

- x is the point to be found.
- p_i are the given points.
- w_i are optional weights (if present, they give different importance to the distances from the point x to each p_i).In our approach, we have taken w_i=1 for all i .
- $\|x - p_i\|$ is the Euclidean distance between the point x and the point p_i.

In our approach, we iteratively approximate the geometric median until either 300 iterations are reached, or the loss function satisfies the condition .

$$|f(x)| < \epsilon \quad (\epsilon = 10^{-8}, \text{ in our case})$$

4 Results

We evaluated our approach using multiple fine-tuned LLMs across various NLP tasks, including text classification, question answering, text generation, text-to-text generation, sentence similarity, comprehension and understanding, search and retrieval, natural language generation, paraphrasing, extraction, and process understanding. The table 1 compares the performance of merging various number of models using two methods: GeoMed, which computes the geometric median of task vectors, and WeightAvg, a baseline method that computes the average of all task vectors. The results indicate that as the number of models merged increases, the evaluation metrics show a substantial improvement, highlighting the ability of the merged large language model (LLM) to generalize across multiple tasks. Our method demonstrates effective merging, achieving robust performance across diverse tasks and indicating enhanced adaptability and generalization.

Merge Method	Rouge1	Rouge2	RougeL	RougeLsum	BLEU
GeoMed (20)	**0.365169018**	**0.168960446**	**0.311908921**	**0.321912100**	**0.085560895**
WeightAvg (20)	0.363765969	0.167650662	0.310080274	0.309277511	0.083514798
GeoMed (15)	**0.363746168**	**0.168716755**	**0.310760179**	**0.311333478**	**0.083514798**
WeightAvg (15)	0.3647378596	0.167141560	0.310224524	0.310449948	0.083514798
GeoMed (6)	**0.362287888**	**0.166269069**	**0.319156270**	**0.319113298**	**0.083994781**
WeightAvg (6)	0.361478874	0.169796169	0.310843997	0.311794364	0.082005513
GeoMed (2)	**0.355095178**	**0.164083656**	**0.317939794**	**0.299276737**	**0.089607033**
WeightAvg (2)	0.353798629	0.164048906	0.316438994	0.298291665	0.079670733

Table 1: Performance metrics (ROUGE and BLEU) for different model configurations. As shown in Appendix A.7, several finetuned models were used for various tasks such as Text Classification, Question Answering, and Sentence Similarity etc.

The metrics used are ROUGE and BLEU [3]. We obtained these results by applying our method to the XSum dataset, which is used for text summarization. This is one of many NLP tasks, and our method generalizes well across this task.

5 Conclusion

We propose an efficient model merging technique using geometric median computation over task vectors, offering a resource-friendly alternative to traditional fine-tuning and ensemble methods. Representing each fine-tuned model as a "Task Vector" relative to a pretrained "Base LLM" and leveraging LoRA weights, we capture specialized knowledge effectively. By computing the geometric median with Weiszfeld's iterative algorithm, we create a unified model that excels across tasks with reduced computational demands. This method achieves state-of-the-art results on benchmark tests, combining high performance with scalability.

References

[1] Amir Beck and Shoham Sabach. Weiszfeld's method: Old and new results. *Journal of Optimization Theory and Applications*, 164:1–40, 2015.

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

[3] Taojun Hu and Xiao-Hua Zhou. Unveiling llm evaluation focused on metrics: Challenges and solutions. *arXiv preprint arXiv:2404.09135*, 2024.

[4] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

[5] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[6] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[7] George Pu, Anirudh Jain, Jihan Yin, and Russell Kaplan. Empirical analysis of the strengths and weaknesses of peft techniques for llms. *arXiv preprint arXiv:2304.14999*, 2023.

[8] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[9] Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. Model merging in llms, mllms, and beyond: Methods, theories, applications and opportunities. *arXiv preprint arXiv:2408.07666*, 2024.

A Appendix / supplemental material

A.1 Geometric Median

In geometry, the geometric median [1] of a discrete point set in a Euclidean space is the point minimizing the sum of distances to the sample points. This generalizes the median, which has the property of minimizing the sum of distances or absolute differences for one-dimensional data. It provides a measure of central tendency in higher dimensions.The geometric median minimizes the sum of the L_2 distances of the samples.

The special case of the problem for three points in the plane (that is, $m = 3$ and $n = 2$ in the definition below) is sometimes also known as *Fermat's problem* .Its solution is known as the *Fermat point* of the triangle formed by the three sample points. The geometric median may in turn be generalized to the problem of minimizing the sum of *weighted* distances, known as the *Weber problem* .

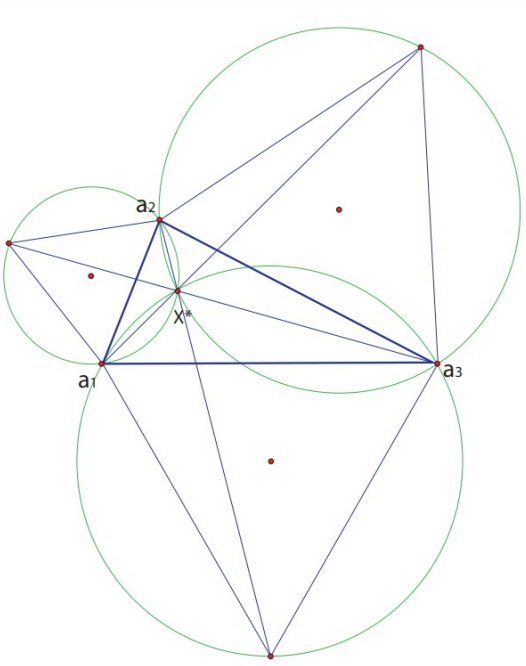

Figure 3: Given a (proper) triangle, formed by three points a_1, a_2 and a_3, construct three equilateral triangles such that each contains one of the edges from the triangle $a_1a_2a_3$. Then, circumscribe each equilateral triangle. The unique point of intersection of these three circles is the point that yields the minimum distance to the points a_1, a_2, and a_3; it is called "the Torricelli Point" and denoted by x^*.

A.2 The Fermat-Weber Problem

The Fermat-Weber problem, described verbally at the beginning of the paper, can be formulated mathematically as the problem of seeking $x \in \mathbb{R}^d$ that solves

$$\min_x \left\{ f(x) = \sum_{i=1}^m \omega_i \|x - a_i\| \right\}, \tag{FW}$$

where $\omega_i > 0$, $i = 1, 2, \ldots, m$ are given weights and the vectors $a_1, a_2, \ldots, a_m \in \mathbb{R}^d$ are given anchors.

To understand the result that Weiszfeld aimed to prove, let us first write down the expression of the gradient of the objective function of problem (FW):

$$\nabla f(x) = \sum_{i=1}^m \omega_i \frac{x - a_i}{\|x - a_i\|}, \quad x \notin A, \tag{2}$$

where $A = \{a_1, a_2, \ldots, a_m\}$ denotes the set of anchors. Note that the gradient is only defined on points different from the anchors.

Theorem 1 (Weiszfeld's original result)**.** *[1] Suppose that the anchors are not collinear.*

(a) Problem (FW) has a unique optimal solution.

(b) Let x^* be the optimal solution of problem (FW). If $x^* \notin A$, then

$$\nabla f(x^*) = \sum_{i=1}^m \omega_i \frac{x^* - a_i}{\|x^* - a_i\|} = 0. \tag{3}$$

If $x^* = a_i$ for some $i \in \{1, 2, \ldots, m\}$, then the following inequality holds:

$$\sum_{\substack{j=1 \\ j \neq i}}^m \omega_j \frac{x^* - a_j}{\|x^* - a_j\|} \leq \omega_i. \tag{4}$$

As noted in the introduction, the theorem was not new and was already established by Sturm in 1884. The anchors $a_1, a_2, \ldots, a_m$ are said to be *collinear* if they reside on the same line, i.e., there exist $y, d \in \mathbb{R}^d$ and $t_1, t_2, \ldots, t_m \in \mathbb{R}$ such that $a_i = y + t_i d$ for all $i = 1, 2, \ldots, m$.

A.3 Weiszfeld's Method

Theorem 2 (Optimality Condition)**.** *[1] The optimality condition $\nabla f(x^*) = 0$ is the necessary and sufficient condition for unconstrained convex minimization problems at points where the objective function is differentiable. When anchors are not collinear, the optimal solution is unique and satisfies the fixed-point relation.*

A.4 Fixed-Point Iteration

The optimal solution can be expressed as:

$$x^* = \frac{\sum_{i=1}^m \omega_i a_i / \|x^* - a_i\|}{\sum_{i=1}^m \omega_i / \|x^* - a_i\|}, \tag{5}$$

or in operator form:

$$x^* = T(x^*), \tag{6}$$

where the operator $T : \mathbb{R}^d \setminus A \to \mathbb{R}^d$ is defined by:

$$T(x) := \frac{\sum_{i=1}^m \omega_i a_i / \|x - a_i\|}{\sum_{i=1}^m \omega_i / \|x - a_i\|}. \tag{7}$$

We have thus shown that, for any $y \in \mathbb{R}^d \setminus A$,

$$y = T(y) \quad \text{if and only if} \quad \nabla f(y) = 0. \tag{8}$$

Weiszfeld's Method
Initialization. $x_0 \in \mathbb{R}^d \setminus A$.
General Step (k = 0, 1, ...):

$$x_{k+1} = T(x_k), \tag{9}$$

where $T(x_k)$ represents the transformation defined in the method.

A.5 Algorithm

Algorithm 1 Weiszfeld Algorithm for Minimizing $f(x) = \sum_{i=1}^{n} w_i \|x - p_i\|$

Input: Points $p_i \in \mathbb{R}^d$, $i = 1, \ldots, n$, with optional weights $w_i > 0$. Initial guess $x^{(0)} \in \mathbb{R}^d$.

$k \leftarrow 0$
repeat
  if $x^{(k)} \neq p_i$ for all $i \in \{1, \ldots, n\}$ **then**
    Compute the weighted average:

$$x^{(k+1)} \leftarrow \frac{\sum_{i=1}^{n} \frac{w_i}{\|x^{(k)} - p_i\|} p_i}{\sum_{i=1}^{n} \frac{w_i}{\|x^{(k)} - p_i\|}}$$

  else
    Apply a small perturbation to avoid division by zero.
  end if
  $k \leftarrow k + 1$
until convergence (i.e., $\|x^{(k+1)} - x^{(k)}\| < \epsilon$ for a small ϵ)

Output: Geometric median $x^{(k+1)}$.

A.6 Evaluation Metrics

We mostly use ROUGE, BLEU, Brevity Penalty as our Evaluation Metrics for summarization,question answering and text-generation tasks, etc.[3]

A.6.1 ROUGE (Recall-Oriented Understudy for Gisting Evaluation)

ROUGE [5] is mostly recall-focused, meaning it evaluates how much of the reference (ground truth) content is captured by the generated content. There are different variations:

ROUGE-N: Measures the overlap of n-grams between the generated text and reference text.

$$\text{ROUGE-N} = \frac{\sum_{\text{ngram} \in \text{reference}} \text{Count}_{\text{match}}(\text{ngram})}{\sum_{\text{ngram} \in \text{reference}} \text{Count}(\text{ngram})} \tag{10}$$

where $\text{Count}_{\text{match}}(\text{ngram})$ is the number of n-grams that match between the generated text and reference, and $\text{Count}(\text{ngram})$ is the total count of n-grams in the reference.

ROUGE-L: Based on the longest common subsequence (LCS), ROUGE-L captures sentence fluency and grammatical structure.

$$\text{ROUGE-L} = \frac{\text{LCS}(X, Y)}{\text{Length}(Y)} \tag{11}$$

where $\text{LCS}(X, Y)$ is the length of the longest common subsequence between the generated text X and reference Y, and $\text{Length}(Y)$ is the length of the reference text.

A.6.2 BLEU (Bilingual Evaluation Understudy)

BLEU [6] is primarily a precision-focused metric, measuring how much of the generated text matches the reference text on an n-gram basis. It's especially popular in machine translation.

Formula: BLEU score combines n-gram precision for multiple n-grams (e.g., unigrams, bigrams, etc.) and takes the geometric mean of these scores, followed by a brevity penalty (discussed below).

For N-gram precision (up to 4-grams, typically):

$$P_n = \frac{\sum_{\text{ngram} \in \text{candidate}} \text{Count}_{\text{clip}}(\text{ngram})}{\sum_{\text{ngram} \in \text{candidate}} \text{Count}(\text{ngram})} \tag{12}$$

where $\text{Count}_{\text{clip}}(\text{ngram})$ is the clipped count of each n-gram in the candidate that appears in the reference (capped at the reference's count).

The BLEU score is then calculated as:

$$\text{BLEU} = \text{Brevity Penalty} \times \exp\left(\sum_{n=1}^{N} w_n \log P_n\right) \tag{13}$$

where w_n is the weight for each n-gram, often set equally (e.g., $w_n = \frac{1}{4}$ for 4-gram BLEU).

A.6.3 Brevity Penalty (BP)

BLEU's Brevity Penalty [6] penalizes short translations that could artificially inflate precision scores. It ensures that a good BLEU score reflects both the precision of n-grams and the length of the generated output.

$$\text{Brevity Penalty (BP)} = \begin{cases} 1 & \text{if } c > r \\ \exp\left(1 - \frac{r}{c}\right) & \text{if } c \leq r \end{cases} \tag{14}$$

where c is the length of the generated text (candidate) and r is the length of the reference text.

- If $c > r$, there's no penalty (BP = 1).
- If $c \leq r$, a penalty proportional to the ratio $\frac{r}{c}$ is applied, reducing the BLEU score for overly short candidates.

A.6.4 Brevity Penalty in Language Generation Tasks

In language generation tasks, like summarization or translation, the brevity penalty controls how much shorter generated outputs are penalized relative to a reference. Here's what each direction typically achieves:

- **More brevity penalty:** A higher penalty (often achieved by setting a lower brevity penalty score) means the model will be discouraged from generating shorter responses and thus produce longer, more complete outputs. This can be useful when concise outputs risk leaving out important information.
- **Less brevity penalty:** A lower penalty (or higher score) allows shorter responses, which can be ideal for concise outputs where detail isn't as critical, like summaries or simpler responses.

In summarization tasks, lower brevity penalties often encourage conciseness. For tasks needing fuller detail (like certain types of translations or responses), higher brevity penalties are better to avoid cutting off important information. In tools like BLEU or ROUGE, one can often adjust brevity to fine-tune according to the goal.

A.7 Finetuned Models Used

No.	Finetuned Model
1	lorahub/flan_t5_xl-dbpedia_14_given_list_what_category_does_the_paragraph_belong_to
2	lorahub/flan_t5_xl-wiki_qa_Topic_Prediction_Question_Only
3	lorahub/flan_t5_xl-anli_r2
4	lorahub/flan_t5_xl- web_questions_question_answer
5	lorahub/flan_t5_xl-duorc_SelfRC_question_answering
6	lorahub/flan_t5_xl-adversarial_qa_dbert_question_context_answer
7	lorahub/flan_t5_xl-wiki_qa_Is_This_True_
8	lorahub/flan_t5_xl-gem_e2e_nlg
9	lorahub/flan_t5_xl-wiki_hop_original_explain_relation
10	lorahub/flan_t5_xl-duorc_SelfRC_title_generation
11	lorahub/flan_t5_xl-glue_mrpc
12	lorahub/flan_t5_xl-glue_cola
13	lorahub/flan_t5_xl-wiki_bio_comprehension
14	lorahub/flan_t5_xl-wiki_bio_key_content
15	lorahub/flan_t5_xl-wiki_bio_guess_person
16	lorahub/flan_t5_xl-wiki_bio_who
17	lorahub/flan_t5_xl-wiki_qa_found_on_google
18	lorahub/flan_t5_xl-gem_web_nlg_en
19	lorahub/flan_t5_xl-duorc_ParaphraseRC_extract_answer
20	lorahub/flan_t5_xl-duorc_SelfRC_extract_answer
21	lorahub/flan_t5_xl-wiqa_what_might_be_the_last_step_of_the_process

Table 2: List of Finetuned Models Used to obtain results of table 1

Number of models	Base Model	Finetuned Models
20	google/flan-t5-xl	(1)(2)(3)(4)(5)(6)(8)(9)(10)(11)(12)(13)(14)(15)(16)(17)(18)(19)(20)(21)
15	google/flan-t5-xl	(1)(2)(3)(4)(5)(6)(8)(9)(10)(13)(14)(18)(19)(20)(21)
6	google/flan-t5-xl	(8)(9)(10)(18)(19)(20)
2	google/flan-t5-xl	(7)(8)

Table 3: As shown in Table 2, several finetuned models were used for various tasks such as Text Classification, Question Answering, and Sentence Similarity etc.