FT-MDT: Extracting Decision Trees from Medical Texts via a Novel Low-rank Adaptation Method

Yuheng Li[1,*], **Jiechao Gao**[2,†,*], **Wei Han**[3], **Wenwen Ouyang**[4], **Wei Zhu**[5,†], **Hui Yi Leong**[6]

[1]Johns Hopkins University, Baltimore, MD, United States [2]Stanford University, Stanford, CA, United States
[3]Independent Researcher, Austin, TX, United States. Email: palebluedot.milkyway@gmail.com
[4]Carnegie Mellon University, Pittsburgh, PA, United States [5]University of Hong Kong, Hong Kong, HK, China
[6]University of Chicago, Chicago, IL, United States

Abstract

Knowledge of the medical decision process, which can be modeled as medical decision trees (MDTs), is critical to building clinical decision support systems. However, current MDT construction methods rely heavily on time-consuming and laborious manual annotation. To address this challenge, we propose PI-LoRA (Path-Integrated LoRA), a novel low-rank adaptation method for automatically extracting MDTs from clinical guidelines and textbooks. We integrate gradient path information to capture synergistic effects between different modules, enabling more effective and reliable rank allocation. This framework ensures that the most critical modules receive appropriate rank allocations while less important ones are pruned, resulting in a more efficient and accurate model for extracting medical decision trees from clinical texts. Extensive experiments on medical guideline datasets demonstrate that our PI-LoRA method significantly outperforms existing parameter-efficient fine-tuning approaches for the Text2MDT task, achieving better accuracy with substantially reduced model complexity. The proposed method achieves state-of-the-art results while maintaining a lightweight architecture, making it particularly suitable for clinical decision support systems where computational resources may be limited.

1 Introduction

Medical decision processes are critical to building effective clinical decision support systems, and these processes can be effectively modeled as medical decision trees (MDTs). The ability to automatically extract MDTs from clinical guidelines and textbooks would significantly reduce the reliance on time-consuming manual annotation while enabling the development of more robust decision support tools. However, the complex hierarchical nature of medical decision-making presents unique challenges for automated extraction, requiring sophisticated natural language processing techniques that can capture both the structural and semantic aspects of clinical guidelines.

Previous research (Zhu et al., 2024a) on the Text2MDT task has primarily focused on either end-to-end frameworks using large language models or pipeline approaches, without fully investigating parameter-efficient fine-tuning (PEFT) methods (Ding et al., 2022) for this specific application. In addition, while PEFT methods like LoRA (Hu et al., 2021) have shown promise in reducing model complexity, existing LoRA-based approaches for Text2MDT suffer from critical limitations in rank allocation. Methods such as AdaLoRA (Zhang et al., 2023a) rely on sensitivity-based importance scores that are unreliable as they only consider how a single parameter change affects the model under the assumption that no other parameters change. Similarly, approaches like SoRA (Ding et al., 2023) and SaLoRA (Hu et al., 2023) use architectural parameters that cannot reliably reflect the quality or importance of LoRA ranks, leading to suboptimal performance in the context of medical decision tree extraction where precise structural representation is essential.

To address these limitations, we propose PI-LoRA (Path-Integrated LoRA), a novel low-rank adaptation method that overcomes the deficiencies of existing rank allocation techniques for the Text2MDT task. Our approach draws inspiration from Shapley value theory, treating each LoRA module as an independent participant in a cooperative game to measure its contribution to overall model performance. Additionally, we integrate gradient path information to capture synergistic effects between different modules, enabling more effective and reliable rank allocation. This framework ensures that the most critical modules receive ap-

[*]These authors contribute equally.
[†]Corresponding author. For any inquiries, please contact: michaelwzhu91@gmail.com, jiechao@stanford.edu.

propriate rank allocations while less important ones are pruned, resulting in a more efficient and accurate model for extracting medical decision trees from clinical texts.

Extensive experiments on medical guideline datasets demonstrate that our PI-LoRA method significantly outperforms existing PEFT approaches for the Text2MDT task, achieving better accuracy with substantially reduced model complexity. The proposed method achieves state-of-the-art results while maintaining a lightweight architecture, making it particularly suitable for clinical decision support systems where computational resources may be limited. Our contributions include the first comprehensive exploration of PEFT methods for Text2MDT, a novel PI-LoRA framework that overcomes limitations in existing rank allocation methods, and empirical evidence showing that our approach achieves superior performance compared to both pipeline and end-to-end LLM-based methods while requiring significantly fewer parameters.

2 Related works

2.1 Tree data extraction from text

There is a rich history of NLP tasks that aim to extract tree structures from a given text. The most fundamental task in NLP is syntax analysis, which aims to express the syntactic structure of a sentence into a syntactic tree (Zhang, 2020). Parsing often relies on a specific grammar, which is used to refine the output structures of syntax and semantics. Two of the most popular grammars are constituent parsing and dependency parsing. Text2Tree is also seen in many application scenarios. Math word problems (MWPs) (Zhang et al., 2022b; Zhao et al., 2023) extract mathematical expressions from the unstructured texts and try to improve the neural networks' capabilities in math problem solving by asking the model to understand the tree structure. Semantic parsing (Kamath and Das, 2018), which transforms unstructured text into an SQL query, has promising application potential in areas like dialogue systems, search engines, and business intelligence. Text2MDT (Zhu et al., 2024a; al., 2022; Zhu et al., 2023a,b) aims to automatically extract medical decision trees from clinical guidelines and textbooks to support the development of clinical decision support systems without relying on time-consuming manual annotation. It proposes two different approaches for Text2MDT - an end-to-end framework using large language models and a pipeline framework - demonstrating that the LLM-based method outperforms the pipeline approach while a lightweight pipelined method achieves comparable performance with significantly smaller model complexity.

2.2 Parameter efficient fine-tuning

Parameter-efficient fine-tuning (PEFT) optimizes only a small subset of new parameters while freezing the backbone model during LLM adaptation (Ding et al., 2022; Zhang et al., 2023b). Contemporary approaches fall into three categories: (a) *Addition-based methods* that incorporate supplementary modules (e.g., Adapters (Houlsby et al., 2019), Prefix/Prompt tuning (Li and Liang, 2021; Lester et al., 2021)); (b) *Specification-based methods* that selectively adjust or prune internal parameters (Ben-Zaken et al., 2021); and (c) *Reparameterization-based methods* that project adaptive parameters into low-dimensional subspaces (Aghajanyan et al., 2021), building on the intrinsic dimensionality concept.

LoRA (Hu et al., 2021) exemplifies reparameterization by optimizing low-rank decompositions of weight updates, showing strong performance across models (Dettmers et al., 2023). However, its fixed-rank design lacks guidance for module-specific rank allocation. Subsequent methods address this by dynamically adapting LoRA parameters: (a) Dy-LoRA (Valipour et al., 2023) trains multi-rank modules simultaneously via random rank sampling. (b) AdaLoRA (Zhang et al., 2023a) allocates weights using SVD ($\Delta W = P \Lambda Q$) and importance-based pruning. (c) SoRA (Ding et al., 2023) reduces redundancy via l_0 regularization and proximal gradients. (d) SaLoRA (Hu et al., 2023) enables module-differentiated ranks using Lagrange multipliers.

3 Method

3.1 Task formulation

The Text2MDT task involves the systematic reconstruction of medical decision trees from clinical documentation. Given a textual input $X = [x_1, x_2, ..., x_{n_{text}}]$ comprising n_{text} lexical units, the objective is to synthesize the pre-order traversal sequence $T = [N_1, N_2, ..., N_{n_{node}}]$ representing n_{node} structural elements within the medical decision tree. This encoding uniquely captures the hierarchical decision logic embedded in clinical guidelines.

Each decision node integrates three functional components as formalized below:

$$\begin{aligned} \text{Node} &= \{\text{Role, Triplets, Logical\_Rel}\}, \\ \text{Role} &= \Diamond \text{ or } \Box, \\ \text{Triplets} &= (t_1, t_2, ..., t_{n_{tri}}), \\ \text{Logical\_Rel} &= \text{and, or, null}, \end{aligned} \tag{1}$$

with critical specifications: (a) Role distinguishes node functionality—$\Diamond$ indicates conditional assessment nodes, whereas $\Box$ designates therapeutic decision nodes. (b) Triplets aggregates n_{tri} subject-relation-object constructs $(t_1, t_2, ..., t_{n_{tri}})$, each $t = (sub, rel, obj)$ encoding clinical entities or procedural instructions. (c) Logical_Rel establishes inter-triplet connectivity, defaulting to null when $n_{tri} \leq 1$.

3.2 Prompt template

According to (Zhu et al., 2024a), the Text2MDT task contains three sub-tasks: (a) triplet extraction; (b) node grouping; (c) tree assembling. The prompt templates and introductions to these sub-tasks are presented in Appendix A.

We also consider utilizing the LLMs for the end-to-end framework. Since this task is complex, it is natural that the idea of chain-of-thought (COT) (Wei et al., 2022) could benefit our task. (Zhu et al., 2024a) constructs a series of different COT-style prompts and responses, and they find that a set of prompt-response template referred to as COT-Gen-3 performs the best. The prompt and response template are presented in Appendix A.

3.3 PI-LoRA: Path-Integrated LoRA

Note that the previous works on Text2MDT only considers the vanilla LoRA method for fine-tuning LLMs, neglecting the other PEFT methods or more advanced variants of LoRA. Thus, in this work, we first analyze the limitations of the current LoRA methods, and then propose a novel LoRA method to enhance the fine-tuning performance.

3.3.1 Analysis of Problems in Existing Methods

We now reflect on the previous representative works on LoRA rank allocation. AdaLoRA (Zhang et al., 2023a) first consider re-arrage the rank distributions of LoRA modules on the LLM backbone. It achieve this objective by first initialize all the LoRA modules with a large number of ranks, and prune the less important ranks gradually along with the training procedure. AdaLoRA utilize a sensitivity based importance score (Michel et al., 2019),

$$\text{ipt}(w) = \|w \nabla_w \mathcal{L}\| \tag{2}$$

which measures how much the training loss will change if the LoRA parameters change. However, (Zhang et al., 2022a) pointed out that this importance measure is unreliable, since it only considers how one parameter change affects the model under the hypothesis that no other parameter changes occur, and have not consider its importance under different model statuses.

AutoLoRA (Zhang et al., 2024) builds upon the methodology of differentable neural architecture search and bi-level optimization (Liu et al., 2019). It considers each LoRA rank as a neural network operation and assigns a learnable architectural parameter. Its objective is to select the best LoRA architecture, which relies on the learned architectural parameters' values as the importance scores. SoRA (Ding et al., 2023) and SaLoRA (Hu et al., 2023) are similar to AutoLoRA except that the architectural parameters are learned with a normal optimization procedure on the training set (Bi et al., 2020). The LoRA ranks with higher architectural weights are kept while others are pruned. However, as pointed out by (Chen and Hsieh, 2020), the architectural parameters can not reliably reflect the quality or importance of the LoRA ranks.

To enhance the effectiveness of the LoRA scoring mechanism, we identify the key challenge as overcoming the limitations of sensitivity scores (the foundation of the AdaLoRA method). Our primary inspiration stems from Shapley value theory (Lundberg and Lee, 2017). This theory models module evaluation as a cooperative game framework, treating each module as an independent participant. The Shapley value $\Phi(m)$ for a neural network module m is defined by the following game-theoretic equation:

$$\Phi_m = \frac{1}{|\mathcal{S}_k|} \sum_{A \subseteq \mathcal{S}_k \setminus \{m\}} \Big[\mathcal{V}(A \cup \{m\}) - \mathcal{V}(A) \Big] \tag{3}$$

where $\mathcal{S}_k$ represents the space of all possible module combinations, and $\mathcal{V}(\cdot)$ denotes the coalition utility function. This framework captures synergistic effects through multi-dimensional interaction evaluation. While computational complexity remains challenging, its theoretical foundation provides crucial insights for our research.

Our second inspiration comes from gradient path integration techniques. The integrated gradients method (Sundararajan et al., 2017) measures the importance by integrating along paths:

$$\text{ipt}(\omega) = \omega * \int_{\alpha=0}^{1} \frac{\partial F(\alpha\omega)}{\partial \omega} d\alpha \tag{4}$$

This method constructs an interpolation path between baseline input $\boldsymbol{x}'$ (e.g., zero vector) and target input $\boldsymbol{x}$, quantifying feature contributions through gradient integration. Essentially, it computes gradients under different parameter scaling coefficients α, though significant computational overhead arises from multiple forward passes.

3.3.2 PI-LoRA

Building on the above analysis, we propose PI-LoRA to improve importance evaluation of LoRA modules in LLMs. Given any parameter $w \in \text{LoRA}_{m,l}$, loss function $\mathcal{L}$, and zero baseline, we construct the importance scoring function:

$$s(w) = \left| w \int_{0}^{1} \nabla_w \mathcal{L}(\alpha w) d\alpha \right| \tag{5}$$

$$\approx \frac{|w|}{K} \left\| \sum_{k=1}^{K} \nabla \mathcal{L}(\tfrac{k}{K} w) \right\| \tag{6}$$

Equation (6) employs the trapezoidal rule with $K > 0$ equidistant points to approximate high-dimensional integration, addressing the strong non-convexity of $\mathcal{L}$ in LLM parameter spaces.

Equation (6) requires K gradient computations, yielding $O(K)$ complexity. To reduce computational load, we design a stochastic sampling strategy. Assume the training process contains P training steps. During the p-th mini-batch of fine-tuning, we uniformly sample α_p from $\{\frac{1}{K}, ..., \frac{K-1}{K}, \frac{K}{K}\}$, with single-point approximation:

$$\tilde{s}^{(p)}(w) = \|(\alpha_p * w) * \nabla \mathcal{L}(\alpha_p * w)\| \tag{7}$$

After P mini-batches, parameter importance is obtained via temporal aggregation:

$$\tilde{s}(w) = \frac{1}{P} \sum_{p=1}^{P} \tilde{s}^{(p)}(w). \tag{8}$$

This approach reduces complexity to $O(1)$ per parameter per batch, significantly enhancing feasibility for large-scale models. Then the sensitivity score for the whole LoRA module is calculated as the average score:

$$\tilde{s}(\text{LoRA}_{m,l}) = \frac{\sum_{w \in \text{LoRA}_{m,l}} \tilde{s}(w)}{|\text{LoRA}_{m,l}|}. \tag{9}$$

3.3.3 Algorithm Overview

We now present the overall procedure for our PI-LoRA method, which is presented in Algorithm 1. We can see that PI-LoRA is a LoRA pruning method equipped with our LoRA scoring method. We calculate the LoRA importance scores from Eq 9, and then prune the LoRA parameters that receive the lowest scores.

Algorithm 1 PI-LoRA

```
Input: Train data $\mathcal{D}$; the number of training steps $P$; randomly initialized LoRA modules $\text{LoRA}_{m,l}$ ($l < L$, $m \in S_{\text{LoRA}}$); targeted number of LoRA modules $N_{\text{LoRA}}$.
1: for $p = 1$ to $P$ do
2:     Sample a mini-batch data from $\mathcal{D}$;
3:     Sample $\alpha_p$ from $\{\frac{1}{K}, ..., \frac{K-1}{K}, \frac{K}{K}\}$;
4:     Conduct a forward pass on Ba and compute the gradients of the LoRA modules;
5:     Calculate the sensitivity scores $\tilde{s}^{(p)}(\text{LoRA}_{m,l})$ via Eq 8;
6:     Accumulate the average score of $\text{LoRA}_{m,l}$ for each LoRA module via Eq 9.
7: end for
8: Prune the LoRA modules that receives the lowest scores, so that remaining LoRA parameters meet the targeted number of LoRA modules $N_{\text{LoRA}}$.
```

4 Experiments

4.1 Datasets and evaluation metrics

In this work, we mainly use the Text2MDT task for evaluation. Readers are referred to (Zhu et al., 2024a) for detailed introductions and dataset statistics. In order to evaluate how different models perform on the Text2MDT task, we now define the following evaluation metrics:

- For triplet extraction, we follow (Zhu et al., 2023) to adopt the triplet-level precision (Prec), recall (Rec) and F1 scores as evaluation metrics.

- For node grouping, we define a Levenshtein ratio (Navarro, 2001) style score, NG_LR, for this subtask.

- For the tree assembling subtask and also the whole Text2MDT task, we define three metrics: (a) the accuracy of decision tree extraction (Tree_Acc); (b) the F1 score of decision paths (DP_F1); (c) Lenvenshtein ratio of the decision tree (Tree_LR).

4.2 Baselines

On the Text2MDT task, We compare our method with the current SOTA PEFT baseline methods.

LoRA and its variants we consider the following LoRA variants as baselines: (a) the original LoRA (Hu et al., 2021) which are considered by (Zhu et al., 2024a); (b) AdaLoRA (Zhang et al., 2023a), which adaptively adjust the LoRA parameters among different Transformer modules. (c) AutoLoRA (Zhang et al., 2024), which utilize the bi-level optimization method (Liu et al., 2019) to learn the LoRA ranks' importance scores. (d) MOELoRA (Liu et al., 2023), which considers each LoRA module as a mixture of single-rank LoRA experts. (e) DoRA (Liu et al., 2024).

Other PEFT methods We also consider the most recent PEFT methods: (a) Parallel-Adapter proposed by He et al. (2021); (b) Learned-Adapter (Zhang et al., 2023b). (c) P-tuning v2 (Liu et al., 2021). (d) IAPT (Zhu et al., 2024b). (e) BitFit (Ben-Zaken et al., 2021). (f) $(IA)^3$ (Liu et al., 2022), which multiplies learnable vectors to the hidden states in different modules of the Transformer layer. (g) SSP (Hu et al., 2022).

The baselines are implemented using their open-sourced codes. We only adjust the hyper-parameters related to tunable parameter numbers to fairly compare the baseline methods and our method.

4.3 Experimental settings

LLM backbones The main experiments use the most recent open-sourced LLM, Qwen 2.5 7B models (Yang et al., 2025) as the pretrained backbone model. In the ablation studies, we will also use the Baichuan 2 7B models (Yang et al., 2023) and GLM-4-9B-Chat[1]. When fine-tuning a LLM, we only consider the supervised fine-tuning (SFT) setting (Ouyang et al., 2022). After receiving a prompt or instruction, all the predictions are generated using the language modeling head (LM head). No additional prediction heads are installed for making categorical or numerical predictions. For decoding during inference, we use beam search with beam size 3.

Implementation details for PI-LoRA For our PI-LoRA method, each LoRA module is initialized with rank $r = 16$. For our PI-LoRA method, K is set to 25. And at the end of the PI-LoRA training procedure, half of the LoRA modules will be pruned. And the remaining LoRA placements will be used as the LoRA setting for re-initialization and retraining. The Adam optimizer (Loshchilov and Hutter, 2017) is employed throughout all experiments. The loss objective is MSE. The learning rate is set to 1e-4, and the number of learning rate warm-up steps is 100. The batch size is set to 32, with the help of gradient accumulation technique. We run validation on the valid set after each epoch. If the validation loss does not drop for 5 epochs, then the training will stop. The gradient checkpoints with the lowest validation loss will be used to make predictions on the test set.

During the final fine-tuning stage, all the LoRA modules are randomly initialized according to the allocation setting delivered by the previous stage. And training hyper-parameters are set to be the same with the previous stage. In every 200 steps, the model is evaluated on the dev set. Patience is set to 10, that is, if the model does not achieve a lower development set loss for 10 evaluation runs, the training stops. The best checkpoint on the dev set is used to run predictions on the test set.

4.4 Main results

Results on the sub-tasks In this setup, We compare our proposed method with baseline PEFT methods by employing these methods in fine-tuning on the Text2MDT task. The experimental results are presented in Table 1. We present the encoder-based methods from (Zhu et al., 2024a) as comparison. Table 1 reveals that our PI-LoRA method outperforms the baseline methods across all seven tasks, with comparable tunable parameters. In particular, PI-LoRA outperforms the previous SOTA LoRA-based baselines like AdaLoRA, AutoLoRA, DoRA, and MOELoRA with comparable or less tunable parameters. These results demonstrate that our method excels at downstream task adaptation of large language models. In addition, we can see that our PI-LoRA method outperforms all the encoder-based methods.

Results on the end2end framework In this set of experiments, we adopt the end2end framework and fine-tune the LLM to complete the whole MDT generation process upon the corresponding prompt. The results are detailed in Table 2. Consistent with earlier findings (Table 1), PI-LoRA achieves supe-

[1]https://huggingface.co/zai-org/glm-4-9b-chat-hf

Subtask	Triplet extract			Node Grouping	Tree assembling		
Metric	Prec	Rec	F1	NG_LR	Tree_Acc	DP_F1	Tree_LR
LLM APIs							
GPT-4	0.783	0.815	0.798	0.916	0.672	0.786	0.893
Encoder-based methods							
UNIRE	0.913	0.881	0.896	-	-	-	-
TPLinker	0.909	0.878	0.893	-	-	-	-
CasRel	0.882	0.891	0.886	-	-	-	-
Sep-Biaffine	0.893	0.897	0.895	-	-	-	-
NG-Biaffine	-	-	-	0.962	-	-	-
NG-TableFilling	-	-	-	0.961	-	-	-
TreeAssemble-Biaffine	-	-	-	-	0.735	0.841	0.937
TreeAssemble-TableFilling	-	-	-	-	0.741	0.838	0.933
LLM fine-tuning methods							
IA3	0.886	0.902	0.893	0.957	0.746	0.835	0.933
Bitfit	0.879	0.911	0.894	0.962	0.741	0.828	0.924
LoRA	0.901	0.908	0.904	0.973	0.764	0.858	0.952
AdaLoRA	0.903	0.910	0.906	0.978	0.761	0.852	0.948
AutoLoRA	0.914	0.901	0.907	0.976	0.765	0.860	0.954
MOELoRA	0.910	0.907	0.908	0.975	0.764	0.859	0.953
DoRA	0.906	0.913	0.909	0.978	0.767	0.862	0.959
PI-LoRA (ours)	0.911	0.916	0.913	0.981	0.772	0.884	0.967

Table 1: Results for each subtask in Text2MDT. The average results in five different runs are reported. The best results are in bold.

Method	Tree_Acc	DP_F1	Tree_ER
LoRA	0.510	0.646	0.911
AdaLoRA	0.510	0.651	0.907
AutoLoRA	0.510	0.648	0.914
MOELoRA	0.520	0.657	0.917
DoRA	0.520	0.660	0.923
PI-LoRA (ours)	0.550	0.679	0.936

Table 2: Overall results of the end2end methods using different PEFT methods. The average results in five different runs are reported. The best results are in bold.

Value of k	Tree_Acc	DP_F1	Tree_ER
$K = 1$	0.510	0.652	0.906
$K = 5$	0.520	0.664	0.923
$K = 10$	0.530	0.672	0.928
$K = 15$	0.540	0.678	0.934
$K = 25$	0.550	0.679	0.936
$K = 50$	0.550	0.677	0.935
$K = 100$	0.550	0.679	0.935

Table 3: Results of different values for the K parameter.

rior performance compared to the baseline methods across all benchmarks. These results underscore PI-LoRA's efficacy in improving instruction-tuning performance for LLMs, highlighting its potential as a robust alternative to existing parameter-efficient adaptation strategies.

4.5 Ablation studies and further analysis

Sensitivity analysis on the K parameter In our main experiments (Table 1 and 2), we set the K parameter to 25. Now we change its value to {1, 5, 10, 15, 50, 100}, and investigate how it affects the fine-tuning performance. The experimental results are presented in Table 3. The results show that $K > 10$ results in reasonable performances, and our PI-LoRA method is robust to this parameter. However, low K results in sub-optimal performance. This is natural: low K values means that our method can not properly approximate the path integral's process, making our method to reduce to the AdaLoRA method.

On the stability of our scoring method On a given LLM backbone, we need to investigate whether our LoRA scoring and pruning is stable since it involves random sampling. We run the whole LoRA scoring procedure under 3 different random seeds, and compare the LoRA importance scores obtained in each run. We calculate the similarities between these three sets of scores pairwise. The similarity score is measured using Spearman rank correlation. Note that these three results are not included in the evaluation of the previous ex-

	Seed 1	Seed 2	Seed 3
Seed 1	1.00	0.94	0.93
Seed 2	-	1.00	0.92
Seed 3	-	-	1.00

Table 4: The pairwise similarity scores for the LoRA ranks' importance estimations obtained under three random seeds.

Method	Tree_Acc	DP_F1	Tree_ER
Results for Baichuan 2 7B			
LoRA	0.490	0.632	0.898
AdaLoRA	0.490	0.634	0.903
Ours	0.51	0.645	0.909
Results for GLM-4-9B-Chat			
LoRA	0.540	0.678	0.936
AdaLoRA	0.540	0.676	0.935
Ours	0.560	0.688	0.942

Table 5: Results for different PEFT methods, when the backbone LLMs are Baichuan 2 7B and GLM-4-9B-Chat.

periments. We present the pairwise correlations in Table 4. From the results, we can see that the importance scores of the LoRA ranks obtained under different random seeds have very high correlations, indicating that the LoRA scores obtained by our method is stable with respect to random seeds.

Ablation studies of the LLM backbone Our main experiments are conducted on the Qwen 2.5 7B model. To demonstrate the wide applicability of our method, we now conduct experiments on the Baichuan 2 7B and GLM-4-9B-Chat. The results are reported in Table 5. We can see that on these two backbones, our method can also outperform the baseline methods.

5 Conclusion

In this paper, we presented FT-MDT, a novel framework for extracting medical decision trees from clinical texts through a novel low-rank adaptation method. The proposed approach addresses Text2MDT, the task of of building clinical decision support systems by automating the extraction of medical decision trees from clinical guidelines and textbooks, eliminating the need for time-consuming manual annotation. The key innovation of our work lies in the development of PI-LoRA (Path-Integrated LoRA), which overcomes the limitations of existing rank allocation methods in LoRA by leveraging Shapley value theory and gradient path integration. Unlike previous approaches that rely on unreliable sensitivity scores, PI-LoRA provides a more effective and theoretically grounded mechanism for module-specific rank allocation, significantly improving the efficiency and performance of the model. Our experimental results demonstrate that the PI-LoRA method achieves superior performance compared to previous approaches. This work not only advances the state-of-the-art in medical decision tree extraction but also provides a valuable framework for LoRA-based LLM fine-tuning.

Future work will explore extending this framework to handle more complex medical decision-making scenarios and investigate its applicability across diverse medical domains with varying levels of structural complexity. The principles underlying PI-LoRA may also inspire more effective parameter-efficient fine-tuning methods for other natural language processing tasks involving hierarchical structure extraction.

Limitations

Despite the promising results achieved by our PI-LoRA approach for medical decision tree extraction, several limitations warrant consideration. First, while our method significantly reduces the need for manual annotation, it remains sensitive to the quality and structure of the input medical texts. Clinical guidelines with non-standard formatting, ambiguous phrasing, or complex multi-step decision processes often lead to incomplete or inaccurate tree structures, particularly when the text contains multiple concurrent treatment paths that are not clearly delineated.

Second, the prompt engineering approach, while effective for standard cases, faces challenges with rare medical conditions or emerging treatment protocols that fall outside the training distribution of the LLM. The current triplet relationship categories (e.g., "clinical manifestations," "therapeutic drugs") may not adequately capture specialized medical knowledge in rapidly evolving fields, potentially requiring domain-specific customization for optimal performance.

Third, the PI-LoRA method, while computationally efficient compared to full fine-tuning, still requires careful tuning of the path integration parameters for different medical domains. The current implementation assumes a relatively uniform structure across medical texts, which may not hold for highly specialized clinical guidelines that deviate from the standard binary decision tree format.

Fourth, our evaluation primarily focuses on structural accuracy of the extracted decision trees, but does not fully address the clinical validity of the resulting decision logic. A tree that is structurally correct may still contain medically questionable recommendations due to limitations in the LLM's knowledge base or the prompt design.

Finally, the current implementation is limited to English-language medical texts, which restricts its immediate applicability in non-English speaking healthcare settings without significant additional adaptation. Future work should address these limitations to broaden the method's practical utility across diverse medical contexts and languages.

Ethics Statement

This research involves the extraction of medical decision trees from clinical texts using large language models. We have carefully considered the ethical implications of our work in the healthcare domain and have implemented the following measures:

Data Privacy: All medical texts used in this study were publicly available clinical guidelines or synthetic data generated for research purposes. No patient-identifiable information was included in any of our datasets or models. All data processing and model training complied with institutional privacy policies.

Bias Mitigation: We acknowledge that medical data may contain biases that could affect clinical decision-making. To address this, we implemented a multi-step verification process involving medical professionals to review the extracted decision trees for potential biases in treatment recommendations. We recognize that our current model may not capture all nuances of medical decision-making and recommend future work with more diverse medical data sources.

Safety and Clinical Use: The extracted decision trees are intended to support, not replace, clinical decision-making. We explicitly state that all clinical decisions must be made by qualified healthcare professionals, and our system should not be used as a standalone diagnostic or treatment tool. We emphasize the importance of human oversight in any clinical implementation.

Transparency: We provide a comprehensive description of our methodology, model architecture, and limitations to enable peer review, replication, and critical evaluation of our work. The prompt templates, evaluation metrics, and implementation details are fully documented for transparency.

Potential Misuse: We recognize that this technology could be misused to automate clinical decisions without appropriate oversight. Therefore, we recommend that our system should only be deployed in clinical settings with appropriate regulatory approvals and under the supervision of qualified medical professionals.

Social Impact: While we believe this research has the potential to improve clinical decision support systems and enhance healthcare efficiency, we acknowledge the need for ongoing discussion about the societal implications of increased automation in healthcare, including potential impacts on healthcare workforce dynamics. We commit to ongoing ethical review of our work as the technology develops and is implemented in real-world clinical settings.

References

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. 2021. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7319–7328, Online. Association for Computational Linguistics.

Wenfeng Li，Wei Zhu，Xiaoling Wang，et al. 2022. Text2dt: decision rule extraction technology for clinical medical text. *Journal of Medical Indformatics*, 43(12)：16-22:16–22.

Elad Ben-Zaken, Shauli Ravfogel, and Yoav Goldberg. 2021. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *ArXiv*, abs/2106.10199.

Kaifeng Bi, Lingxi Xie, Xin Chen, Longhui Wei, and Qi Tian. 2020. Gold-nas: Gradual, one-level, differentiable. *ArXiv*, abs/2007.03331.

Xiangning Chen and Cho-Jui Hsieh. 2020. Stabilizing differentiable architecture search via perturbation-based regularization. In *International conference on machine learning*, pages 1554–1565. PMLR.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient Fine-tuning of Quantized LLMs. *arXiv e-prints*, page arXiv:2305.14314.

Ning Ding, Xingtai Lv, Qiaosen Wang, Yulin Chen, Bowen Zhou, Zhiyuan Liu, and Maosong Sun. 2023.

Sparse low-rank adaptation of pre-trained language models. In *Conference on Empirical Methods in Natural Language Processing*.

Ning Ding, Yujia Qin, Guang Yang, Fu Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, Jing Yi, Weilin Zhao, Xiaozhi Wang, Zhiyuan Liu, Haitao Zheng, Jianfei Chen, Yang Liu, Jie Tang, Juan Li, and Maosong Sun. 2022. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models. *ArXiv*, abs/2203.06904.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2021. Towards a unified view of parameter-efficient transfer learning. *ArXiv*, abs/2110.04366.

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. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

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. *arXiv preprint arXiv:2106.09685*.

Shengding Hu, Zhen Zhang, Ning Ding, Yadao Wang, Yasheng Wang, Zhiyuan Liu, and Maosong Sun. 2022. Sparse structure search for parameter-efficient tuning. *ArXiv*, abs/2206.07382.

Yahao Hu, Yifei Xie, Tianfeng Wang, Man Chen, and Zhisong Pan. 2023. Structure-aware low-rank adaptation for parameter-efficient fine-tuning. *Mathematics*.

Aishwarya Kamath and Rajarshi Das. 2018. A survey on semantic parsing. *ArXiv*, abs/1812.00978.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2019. Darts: Differentiable architecture search. *ArXiv*, abs/1806.09055.

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. *ArXiv*, abs/2205.05638.

Qidong Liu, Xian Wu, Xiangyu Zhao, Yuanshao Zhu, Derong Xu, Feng Tian, and Yefeng Zheng. 2023. Moelora: An moe-based parameter efficient fine-tuning method for multi-task medical applications. *arXiv preprint arXiv:2310.18339*.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024. Dora: Weight-decomposed low-rank adaptation. *arXiv preprint arXiv:2402.09353*.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *ArXiv*, abs/2110.07602.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? *Advances in neural information processing systems*, 32.

Gonzalo Navarro. 2001. A guided tour to approximate string matching. *ACM Comput. Surv.*, 33:31–88.

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.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR.

Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. 2023. Dylora: Parameter-efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3274–3287.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Huai hsin Chi, F. Xia, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *ArXiv*, abs/2201.11903.

Aiyuan Yang, Bin Xiao, Bingning Wang, Borong Zhang, Ce Bian, Chao Yin, Chenxu Lv, Da Pan, Dian Wang, Dong Yan, et al. 2023. Baichuan 2: Open large-scale language models. *arXiv preprint arXiv:2309.10305*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Meishan Zhang. 2020. A survey of syntactic-semantic parsing based on constituent and dependency structures. *Science China Technological Sciences*, 63:1898 – 1920.

Qingru Zhang, Minshuo Chen, Alexander W. Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023a. Adaptive budget allocation for parameter-efficient fine-tuning. *ArXiv*, abs/2303.10512.

Qingru Zhang, Simiao Zuo, Chen Liang, Alexander Bukharin, Pengcheng He, Weizhu Chen, and Tuo Zhao. 2022a. Platon: Pruning large transformer models with upper confidence bound of weight importance. In *International conference on machine learning*, pages 26809–26823. PMLR.

Ruiyi Zhang, Rushi Qiang, Sai Ashish Somayajula, and Pengtao Xie. 2024. Autolora: Automatically tuning matrix ranks in low-rank adaptation based on meta learning. *arXiv preprint arXiv:2403.09113*.

Wenqi Zhang, Yongliang Shen, Yanna Ma, Xiaoxia Cheng, Zeqi Tan, Qingpeng Nong, and Weiming Lu. 2022b. Multi-view reasoning: Consistent contrastive learning for math word problem. In *Conference on Empirical Methods in Natural Language Processing*.

Yuming Zhang, Peng Wang, Ming Tan, and Wei-Guo Zhu. 2023b. Learned adapters are better than manually designed adapters. In *Annual Meeting of the Association for Computational Linguistics*.

Xu Zhao, Yuxi Xie, Kenji Kawaguchi, Junxian He, and Qizhe Xie. 2023. Automatic model selection with large language models for reasoning. In *Conference on Empirical Methods in Natural Language Processing*.

Wei Zhu, Wenfeng Li, Xing Tian, Pengfei Wang, Xiaoling Wang, Jin Chen, Yuanbin Wu, Yuan Ni, and Guotong Xie. 2024a. Text2mdt: extracting medical decision trees from medical texts. *arXiv preprint arXiv:2401.02034*.

Wei Zhu, Wenfeng Li, Xiaoling Wang, Wendi Ji, Yuanbin Wu, Jin Chen, Liang Chen, and Buzhou Tang. 2023a. Extracting decision trees from medical texts: An overview of the text2dt track in chip2022. In *Health Information Processing. Evaluation Track Papers*, pages 89–102, Singapore. Springer Nature Singapore.

Wei Zhu, Wenfeng Li, Xiaoling Wang, Wendi Ji, Yuanbin Wu, Jin Chen, Liang Chen, and Buzhou Tang. 2023b. Extracting decision trees from medical texts: An overview of the text2dt track in chip2022. In *Health Information Processing. Evaluation Track Papers*, pages 89–102, Singapore. Springer Nature Singapore.

Wei Zhu, Aaron Xuxiang Tian, Congrui Yin, Yuan Ni, Xiaoling Wang, and Guotong Xie. 2024b. Iapt: Instruction-aware prompt tuning for large language models. *arXiv preprint arXiv:2405.18203*.

Wei Zhu, Xiaoling Wang, Huanran Zheng, Mosha Chen, and Buzhou Tang. 2023. PromptCBLUE: A Chinese Prompt Tuning Benchmark for the Medical Domain. *arXiv e-prints*, page arXiv:2310.14151.

A Prompt templates and response formats for the pipeline framework

A.1 The triplet extraction subtask

In the triplet extraction task asks a language model to predict a series of triplets from the given text. A triplet includes the head entity mention, tail entity mention, and the relation between them. For the triplet extraction sub-task, the template for the input prompt is:

```
Please extract triplets based on the
    following medical guideline text:

[Text]

Instruction: Extract the triplet used to
     describe diagnosis and treatment
    knowledge or clinical information as
     the content of the condition/
    decision node. The triplet
    relationship defines a total of 6
    categories: "clinical manifestations
    ", "therapeutic drugs", "usage and
    dosage", "Treatment plan", "
    Prohibited drugs", "Basic situation"
```

the response should be formated as follows:

```
The triples in the given guideline text
    are as follows:

[triplets]
```

The special token [Text] denotes the input text, and [triplets] denotes a list of triplets.

A.2 The node grouping subtask

In the node grouping task, we asks a language model to predict which triplets form a node, and which logical relation the node has. For the node grouping sub-task, the template for the input prompt is:

```
Please combine these triples into
    several nodes based on the following
     medical guideline text and the
    triplet information extracted from
    it, and indicate the logical
    relationship of the triplets within
    this node:

Medical guideline text:

{Text}

The triples in the given guideline text
    are as follows:

{triplets}

Note: If several triples form a node, it
     means that there is an and or or
    logical relationship between these
    triples. If a triple does not have
    an and or or relationship with other
     triples, it means that the triple
    needs to become a node independently
    .
```

the response should be formated as follows:

```
Based on the given guideline text and
    its triplet information, the nodes
    of the decision tree are composed as
     follows:

The following triples constitute a node
    of the decision tree: [triplets].
    The logical relationship of this
    node is: [logical_rel]

The following triples constitute a node
    of the decision tree: [triplets].
    The logical relationship of this
    node is: [logical_rel]
```

The special token [Text] denotes the input text, and [triplets] denotes the list of extracted triplets, and [node] denotes the contents of the node.

A.3 The tree assembling subtask

In the tree assembling task, given the results of the node grouping step, we ask the language model to generate the whole decision tree. For the tree assembling sub-task, the template for the input prompt is:

```
Please form a decision tree based on the
     following medical guideline text
    and the node information extracted
    from it:

Medical guideline text:

[Text]

The nodes in a given guideline text are
    composed as follows:

[nodes]

Note: (1) The diagnosis and treatment
    decision tree is a binary tree
    composed of conditional nodes and
    decision nodes. It aims to express
    guideline text through concise
    structured information. It requires
    not only to dig out the core
    entities and relationships in the
    text, but also to carry out this
    information. They are connected in
    series to form a complete decision-
    making process; (2) In the diagnosis
     and treatment decision-making
    binary tree, non-leaf nodes are
    condition nodes and leaf nodes are
    decision nodes. For the condition
    node, when the condition judgment
    result is "yes", it will go to the
    left child node for the next
    judgment or decision. When the
    condition judgment result is "no",
    it will go to the right child node
    for the next judgment or decision.
    (3) The output of each node is a
    dict, containing three fields: (3a)
    "role", which is the node role type;
     (3b) "triples", which is a list of
    triples; (3c) "logical_rel", which
    represents the node logical
    relationship. (4) The entire
    diagnosis and treatment decision
    tree is arranged into a list using
    the breadth-first strategy.
```

the response should be formated as follows:

```
The diagnosis and treatment decision
    tree extracted based on the given
    guideline text is as follows:

Node [node_idx]: role=[role];
    logical_rel=[logical_rel]; triplets
    =[triplets]

Node [node_idx]: role=[role];
    logical_rel=[logical_rel]; triplets
    =[triplets]

Node [node_idx]: role=[role];
    logical_rel=[logical_rel]; triplets
    =[triplets]
```

The special token [Text] denotes the input text, and [nodes] denotes the list of nodes from the previous subtask. In the response, [node_idx] denotes the index of a node, [triplets] denotes the list of extracted triplets in a node, [logical_rel] denotes the logical relation of the node, and [role] denotes the role label of the node.

A.4 COT-Gen-3

Following (Zhu et al., 2024a), we consider the following template for the COT-Gen-3 method:

```
Please generate a diagnosis and
    treatment decision tree for the
    following medical guideline text:

[Text]

Task description:
(1) Based on the given medical guideline
     text, create a binary tree,
    including conditional nodes and
    decision nodes, to succinctly
    display the guideline content while
    capturing core entities and
    relationships;
(2) Conditional nodes are used for
    judgment, based on the results point
     to the left or right child node to
    make the next decision.
(3) The output of each node is a dict,
    containing three fields:
    - (3a) "role" indicates the node
    type, which can be a condition node
    ("C") or a decision node ("D");
    - (3b) "triples" is a list of
    triples describing diagnosis and
    treatment knowledge or clinical
    information, including "clinical
    manifestations", "therapeutic drugs
    ", "usage and dosage", "treatment
    plan", "Prohibited drugs", "Basic
    situation" six types of
    relationships;
    - (3c) "logical_rel" represents the
    logical relationship between
    multiple triples (the values are and
    , or, null, when there is only one
    triple. The logical relationship is
    null when it is a tuple).
(4) The finally generated diagnosis and
    treatment decision tree is arranged
    into a list according to the breadth
    -first strategy.)

Instructions for the generation steps:
    Please complete the generation of
    the decision tree step by step. (a)
    First extract triples from the above
     text; (b) and then generate a
    complete decision tree.
```

where [Text] is a piece of medical text. According to (Zhu et al., 2024a), the response should be formated as follows:

```
The triples in the given guideline text
    are as follows:

[triplets]

The diagnosis and treatment decision
    tree extracted based on the given
    guideline text is as follows:

Node [node_idx]: role=[role];
    logical_rel=[logical_rel]; triples=[
    triplets]
Node [node_idx]: role=[role];
    logical_rel=[logical_rel]; triples=[
    triplets]
Node [node_idx]: role=[role];
    logical_rel=[logical_rel]; triples=[
    triplets]
```

where [node_idx] represents the node id, [role] means the role of the node, [logical_rel] stands for the logical relation in the node, and [triplets] expresses the contents in the node. From the above prompt template and response, we can see that COT-Gen-3 asks the LM to extract triplets and then generate the whole MDT.