Understanding the Prompt Sensitivity

Yang Liu **Chenhui Chu**
Kyoto University
yangliu@nlp.ist.i.kyoto-u.ac.jp, chu@i.kyoto-u.ac.jp

Abstract

Prompt sensitivity, which refers to how strongly the output of a large language model (LLM) depends on the exact wording of its input prompt, raises concerns among users about the LLM's stability and reliability. In this work, we consider LLMs as multivariate functions and perform a first-order Taylor expansion, thereby analyzing the relationship between meaning-preserving prompts, their gradients, and the log probabilities of the model's next token. We derive an upper bound on the difference between log probabilities using the Cauchy-Schwarz inequality. We show that LLMs do not internally cluster similar inputs like smaller neural networks do, but instead disperse them. This dispersing behavior leads to an excessively high upper bound on the difference of log probabilities between two meaning-preserving prompts, making it difficult to effectively reduce to 0. In our analysis, we also show which types of meaning-preserving prompt variants are more likely to introduce prompt sensitivity risks in LLMs. In addition, we demonstrate that the upper bound is strongly correlated with an existing prompt sensitivity metric, `PromptSensiScore`. Moreover, by analyzing the logit variance, we find that prompt templates typically exert a greater influence on logits than the questions themselves. Overall, our results provide a general interpretation for why current LLMs can be highly sensitive to prompts with the same meaning, offering crucial evidence for understanding the prompt sensitivity of LLMs. Code for experiments is available at https://github.com/ku-nlp/Understanding_the_Prompt_Sensitivity.

1 Introduction

Large language models (LLMs) usually show sensitivity to even minor variations in prompts, such as wording, prompt template, or even minor spelling errors, although these variations do not change the meaning of the prompt (Chatterjee et al., 2024). This phenomenon can be described as LLMs' prompt sensitivity, which can amplify the output variance, making the model's output unreliable. To quantify this effect, researchers (Zhuo et al., 2024; Chatterjee et al., 2024) have made considerable efforts to assess the sensitivity of LLMs to minor variations in prompts. Also, Sun et al. (2024) have attempted to improve the generalization ability of LLMs through reinforcement learning from human feedback (RLHF; Christiano et al., 2017) or instruction tuning (Wei et al., 2021). However, even minor changes such as prompt formatting to the wording of the prompts still can lead to the prompt sensitivity of these models (Sclar et al., 2024).

Although prompt sensitivity in LLMs is frequently highlighted, its generation mechanism remains poorly understood. For example, we still do not understand why a set of meaning-preserving prompts can yield completely different outputs by an LLM. This open issue leads to a lack of credibility in previous benchmark-based prompt sensitivity evaluations (Zhuo et al., 2024; Chatterjee et al., 2024) and the arbitrary practice of fine-tuning LLMs by increasing training samples (Liu et al., 2025; Dong et al., 2024). Previous studies (Zhuo et al., 2024; Chatterjee et al., 2024) calculate a metric to represent a model's sensitivity to wording changes in prompts based on its output. However, they make only limited contributions to understanding the prompt sensitivity of LLMs and fail to guide fundamental breakthroughs.

Contrary to previous studies, we aim to understand the prompt sensitivity of LLMs using a mathematical analysis method: Taylor expansion (Taylor, 1715). In this study, we focus on transformer-based LLMs. Specifically, we formalize an LLM as a continuous multivariate function that outputs the log probability of the model's next token. The hidden states are responsible for converting the prompts in discrete space into the continuous representation space. Then, we use the first-order Taylor expan-

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 44362–44388

sion of this function to connect the hidden states of the prompt with the output log probabilities. Furthermore, we monitor the changes in hidden states of two meaning-preserving prompts' across layers to explain the prompt sensitivity of LLMs.

Our analysis starts with an image classification task. We observe that ResNet (He et al., 2016) internally produces clustering behavior to achieve high classification accuracy. Next, we build connections between two prompts using Taylor expansion, derive an upper bound for the log probability difference via the Cauchy-Schwarz (Cauchy, 1821; Schwarz, 1890) inequality, and reveal why LLMs exhibit prompt sensitivity by observing their different behaviors compared to traditional neural networks (RQ1). We then investigate which types of prompt modifications are more likely to lead to prompt sensitivity (RQ2). We also find that the upper bound correlates strongly with an existing prompt sensitivity metric (RQ3). Furthermore, by analyzing the variance of LLMs' logits, we observe that in existing LLMs, prompt templates exert a greater influence on logits than the questions themselves (RQ4).

2 Neural Networks Are Functions

A neural network is a mathematical relationship that maps inputs to outputs (LeCun et al., 2015; Nielsen, 2015; Goodfellow et al., 2016). If the input is a vector $\boldsymbol{x} \in \mathbb{R}^d$ and the output is a scalar $y \in \mathbb{R}$, then a single-layer neural network can be represented as:

$$y = \sigma(\boldsymbol{w}^\top \boldsymbol{x} + b) \tag{1}$$

where $\boldsymbol{w}$ is the weight vector, b is the bias, and $\sigma(\cdot)$ is the activation function, such as sigmoid (Rumelhart et al., 1986), ReLU (Nair and Hinton, 2010; Glorot et al., 2011), etc. If we consider this neural network as a function $y = f(\boldsymbol{x})$. The one-time inference using this neural network can be interpreted as input vector $\boldsymbol{x}$ to the function $f(\boldsymbol{x})$, outputting the scalar y. In this section, we start by explaining why deep neural networks are composite functions. Then, we introduce intra-class mean distances, a simple representation of space distances. Finally, we interpret why deep neural networks can perform classification tasks from an interesting perspective: that a neural network is a function.

Deep neural networks are compositions of functions. A deep neural network defines a function as a composition of simpler functions. In particular, it is composed of layer-by-layer composites of affine transformations and activation functions (Cybenko, 1989; Hornik et al., 1989; Murphy, 2012). Formally, the affine transformation of the layer l is:

$$A_l(\boldsymbol{x}) = \boldsymbol{W}_l \boldsymbol{x} + \boldsymbol{b}_l \tag{2}$$

where $\boldsymbol{x} \in \mathbb{R}^{d_{l-1}}$ is the output vector of layer $l-1$, $\boldsymbol{W}_l \in \mathbb{R}^{d_l \times d_{l-1}}$ is the weight of the layer l, and $\boldsymbol{b}$ is the bias of layer l. Then, the affine transformation $A_l(\boldsymbol{x})$ is composed using the activation function σ_l as follows:

$$g_l = \sigma_l \circ A_l \tag{3}$$

The general mapping of the deep neural network of layer L is as follows:

$$F = g_L \circ g_{L-1} \circ \cdots \circ g_1 \tag{4}$$

where the composite function F is a continuous mapping from the input space to the output space (Goodfellow et al., 2016).

Intra-class compactness. Intra-class compactness (Yan et al., 2020) refers to how close or tightly clustered the samples or data points of the same class are in the feature space. Typically, an ideal classifier requires ensuring high intra-class compactness. To remove the influence of vector dimension on distance metrics, we first perform L^2 normalization on the feature vectors, then use the Euclidean distance between the normalized vectors as the metric:

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \|\boldsymbol{x}_i - \boldsymbol{x}_j\| \tag{5}$$

As all vectors are normalized to the unit hypersphere, this distance reflects only directional differences and is equivalent to cosine similarity, e.g., $\|\boldsymbol{x}_i - \boldsymbol{x}_j\| = \sqrt{2 - 2\cos\theta_{ij}}$ (see Appendix B), where θ_{ij} is the angle between the two vectors. We denote the samples for class c as $\mathcal{J}_c$. We use the intra-class mean distance as the metric. The distance of samples in class c is defined as follows:

$$D_{\text{intra}}^{(c)} = \frac{1}{|\mathcal{J}_c|(|\mathcal{J}_c|-1)} \sum_{\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{J}_c, i \neq j} d(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{6}$$

We use the average of the distances over all classes as the metric for intra-class compactness:

$$D_{\text{intra}} = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} D_{\text{intra}}^{(c)} \tag{7}$$

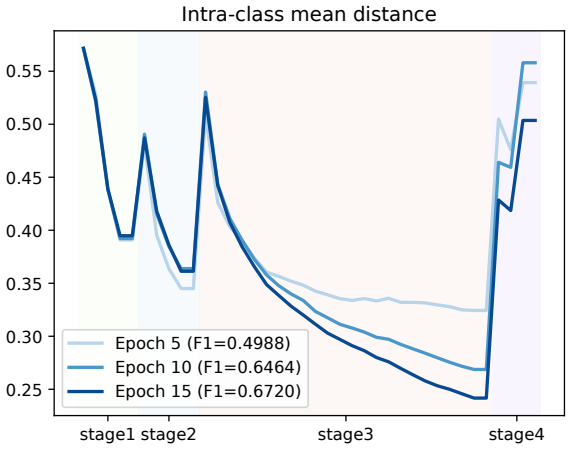


Figure 1: Intra-class mean distances for CIFAR-10 across different training epochs.

where $\mathcal{C}$ is the class set and $|\mathcal{C}|$ denotes the total number of classes. Enhancing intra-class compactness, i.e., decreasing D_{intra}, can improve the neural network's classification performance (Liu et al., 2016; Yan et al., 2020).

Intra-class mean distance of ResNet on CIFAR-10. To illustrate the internal behavior of neural networks while performing classification tasks, we investigate how the intra-class compactness of the feature maps changes across each stage of the neural network. We take the application of ResNet (we pick `ResNet-101`; He et al., 2016) on the CIFAR-10 dataset (Krizhevsky et al., 2009) as an example. ResNet is typically divided into four stages, each consisting of multiple residual blocks stacked together and outputting the feature maps of the input image. Stages 1, 2, 3, and 4 consist of 3, 4, 23, and 3 blocks, respectively. Here, we focus on the block level of ResNet to analyze the features output by each block and the input features output by the stem module of ResNet.

We trained this neural network on the CIFAR-10 dataset for 20 epochs, achieving the highest F1 score of 0.7048 on the testing set at epoch 18.[1] As shown in Figure 1, we compare the intra-class mean distance of features across three epochs: epoch 5, 10, and 15; their F1 scores gradually increased. A low intra-class mean distance indicates high intra-class compactness. At the stage level, we observe that the intra-class mean distance gradually decreases from stage 1 to stage 3, then increases in stage 4. This indicates that stages 1 to 3 are performing clustering, while stage 4 is classifying feature differences within classes. The behavior of classifying feature differences within classes also occurs in stages 2 and 3. In stage 3, which demonstrated the best clustering performance, epochs with lower intra-class mean distances yield higher F1 scores. This indicates that **the neural network achieves stronger classification performance by improving its clustering behavior**. From the functional perspective, clustering brings samples of the same class closer together, while continuous functions produce similar outputs for similar inputs. In this paper, based on the above analysis, we formulate LLMs as functions and investigate their prompt sensitivity using Taylor expansion.

[1]For more hyperparameters, see Appendix C.

3 Interpretation of Prompt Sensitivity

The prompt sensitivity of LLMs usually refers to minor variations in prompts causing LLMs to respond with different results (Zhuo et al., 2024; Chatterjee et al., 2024). In this paper, we narrow it down to describe "how prompt p_0 and its meaning similar prompt p_1 cause the LLMs to respond with different log probabilities of the model's next token y_t." The natural language prompts or their tokenized tokens reside in a discrete space, while their embeddings represented by the embedding layer or hidden states output by a specific transformer block can be regarded as variables in the continuous representation space.

3.1 LLMs Are Multivariable Functions

In § 2, we observe that ResNet exhibits clustering behavior to achieve significantly higher accuracy and F1 score. In this section, we generalize this interpretation to LLMs. Unlike classification neural networks, which project the feature representations into a class space, LLMs project the feature representations into a vocabulary space to predict the next token (Vaswani et al., 2017).

In LLMs' inference stage, when an LLM predicts the next token, it first maps the input tokens into embeddings by the embedding layer and adds positional encodings to form a sequence representation. Then, the sequence representation passes through several transformer blocks sequentially. In the self-attention module of each transformer block, a causal mask is applied to block tokens to the right of the current position, ensuring that each current position only depends on the content to its left. In this way, each position ultimately obtains a hidden state vector that contains only the prefix informa-

tion. When the model processes the entire input sequence, it predicts the next token using the hidden state of the last position. This hidden state is projected to the vocabulary space via the output layer (typically a linear layer and softmax).

Now, suppose we input a prompt containing L tokens into an LLM. The model maps each token in this prompt to a D-dimensional embedding. Following the analysis in § 2, We consider an LLM as a multivariable function, where the output of the embedding layer serves as the function's input. The log probability of the next token is treated as the model's output value. The difference between the log probabilities of two meaning-preserving inputs (prompts) can be interpreted as a measure of prompt sensitivity. A smaller difference indicates lower prompt sensitivity of the model.

3.2 Taylor Expansion of LLMs

We use the Taylor expansion to build connections between two meaning-preserving prompts. For simplicity, we denote the log probability difference between two meaning-preserving prompts $\mathbf{h}_0$ and $\mathbf{h}_1$ as $\Delta \log \pi(y_t \mid \mathbf{h})$. Here,

$$\Delta \log \pi(y_t \mid \mathbf{h}) = \log \pi(y_t \mid \mathbf{h}_1) - \log \pi(y_t \mid \mathbf{h}_0), \tag{8}$$

where $\pi = \mathrm{softmax}(\mathbf{z})$ is the softmax of the logits $\mathbf{z}$ output by the LLM. Formally, we can express the relationship between the hidden states $\mathbf{h}_0$ and $\mathbf{h}_1$ by Taylor expansion[2] as follows:

$$\underbrace{\Delta \log \pi(y_t \mid \mathbf{h})}_{1\times 1} = \underbrace{\nabla_{\mathbf{h}} \log \pi(y_t \mid \mathbf{h}_0)^\top}_{1\times D} \underbrace{(\Delta \mathbf{h})}_{D\times 1} + \mathcal{O}(\|\Delta \mathbf{h}\|^2), \tag{9}$$

where $\mathcal{O}(\|\Delta \mathbf{h}\|^2)$ is the remainder term of the Taylor expansion. $\nabla_{\mathbf{h}} \log \pi(y_t \mid \mathbf{h}_0)$ indicates the gradient vector and $\Delta \mathbf{h} = \mathbf{h}_1 - \mathbf{h}_0$ indicates the difference between the feature representations of the two prompts. It is calculated through element-wise subtraction, thus capturing not only semantic differences between the two prompts but also variations in their expressive styles. In this paper, unless otherwise specified, we set the correct answer of the question as y_t. Discussions regarding other tokens as y_t are provided in Appendix H.

3.3 Upper Bound

From the properties of the Taylor expansion, we know that when the distance between $\mathbf{h}_0$ and $\mathbf{h}_1$ is sufficiently close, the remainder term $\mathcal{O}(\|\Delta \mathbf{h}\|^2)$ will vanish faster than $\|\Delta \mathbf{h}\|^2$ as $\mathbf{h}_1 \to \mathbf{h}_0$. Moreover, in this paper, $\mathbf{h}_0$ and $\mathbf{h}_1$ are two meaning-preserving prompt words that reside in a close semantic space. Based on this condition, we rewrite Eq. (9) in the following form:

$$\Delta \log \pi(y_t \mid \mathbf{h}) \approx \nabla_{\mathbf{h}} \log \pi(y_t \mid \mathbf{h}_0)^\top \Delta \mathbf{h}. \tag{10}$$

Then, we obtain the following inequality by calculating the L2 norm:

$$|\Delta \log \pi(y_t \mid \mathbf{h})| \leq \|\nabla_{\mathbf{h}} \log \pi(y_t|\mathbf{h}_0)\| \cdot \|\Delta \mathbf{h}\|, \tag{11}$$

where $\|\cdot\|$ is the L2 norm. This inequality tells us that $|\Delta \log \pi(y_t \mid \mathbf{h})|$ has an upper bound $\|\nabla_{\mathbf{h}} \log \pi(y_t|\mathbf{h}_0)\| \cdot \|\Delta \mathbf{h}\|$. If the upper bound is significantly low, $|\Delta \log \pi(y_t \mid \mathbf{h})|$ can be approximated as 0, meaning the two meaning-preserving prompts receive equal log probabilities of the model's next token.

Calculate the gradient. We represent the gradient matrix as follows:

$$\nabla_{\mathbf{h}} \log \pi(y_t \mid \mathbf{h}_0)^\top = G(\mathbf{h}_0), \tag{12}$$

where $G(\mathbf{h}_0) \in \mathbb{R}^D$ represents the gradient vector of $\mathbf{h}_0$. The gradient for the i-th dimension is calculated as follows:

$$g(\mathbf{h}[i]) = \nabla_{\mathbf{h}[i]} \log \pi(y_t \mid \mathbf{h}) \tag{13}$$

The gradient $g(\mathbf{h}[i])$ is usually named the saliency score (Simonyan et al., 2013; Li et al., 2016). Unlike Yin and Neubig (2022), who use the L1 norm to calculate the saliency score for each input token, we take the L2 norm of the gradient vector to obtain the saliency score of the input $\mathbf{h}$ as follows:

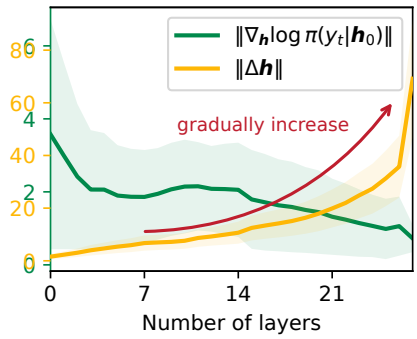
$$S_{GN}(\mathbf{h}) = \|\nabla_{\mathbf{h}} \log \pi(y_t \mid \mathbf{h})\|_2 = \sqrt{\sum_i |g(\mathbf{h}[i])|^2} \tag{14}$$

$S_{GN}(\mathbf{h})$ is the overall contribution of $\mathbf{h}$ to the log probability of the model's next token.

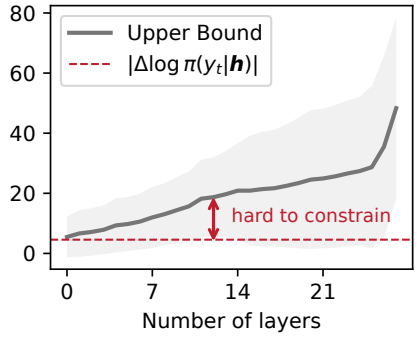
4 Experimental Verifications

In this section, we verify our analytical results in practical settings. We consider four multiple-choice question (MCQ) datasets commonly used to evaluate prompt sensitivity (Zhuo et al., 2024; Chatterjee et al., 2024), ARC Challenge (Clark et al., 2018), CommonSenseQA (Talmor et al.,

[2] Appendix A provides the first-order Taylor expansion for both univariate and multivariate cases.



(a) The trend of $\|\nabla_{\mathbf{h}} \log \pi(y_t|\mathbf{h}_0)\|$ and $\|\Delta \mathbf{h}\|$.



(b) The trend of the upper bound.

Figure 2: Key results of RQ1: (a) indicates the trend of $\|\nabla_{\mathbf{h}} \log \pi(y_t|\mathbf{h}_0)\|$ and $\|\Delta \mathbf{h}\|$ across layers. (b) indicates the trend of the upper bound across layers. The upper bound is calculated by multiplying $\|\nabla_{\mathbf{h}} \log \pi(y_t|\mathbf{h}_0)\|$ by $\|\Delta \mathbf{h}\|$.

2019), MMLU (Hendrycks et al., 2021), and OpenBookQA (Mihaylov et al., 2018). To further examine whether our analysis generalizes beyond MCQ tasks, we additionally include the open-ended generation dataset Alpaca (Rohan et al., 2023). Each MCQ sample is a multiple-choice question with a correct option as the target token, and each Alpaca sample consists of an instruction, where the last token of its reference response is taken as the target token. We randomly select 500 examples to create our test set from each dataset. We consider 12 prompt templates[3] proposed by Zhuo et al. (2024) as the meaning-preserving prompts for LLMs. We combine the 12 prompt templates with 500 samples from each dataset. We perform all our experiments on four model series: `Pythia-410M/1B/1.4B` (Biderman et al., 2023), `GPT2-small/medium/large` (Radford et al., 2019b), `Qwen1.5-0.5B/1.8B/4B` (Bai et al., 2023), and `Llama3.2-1B/3B` (Touvron et al., 2023).

[3]All templates are provided in Appendix E.1.

4.1 Why Do LLMs Exhibit Prompt Sensitivity? (RQ1)

In this section, we explain why LLMs exhibit prompt sensitivity. Specifically, we observe the trend of the upper bound ($\|\nabla_{\mathbf{h}} \log \pi(y_t|\mathbf{h}_0)\| \cdot \|\Delta \mathbf{h}\|$) in Eq. (11) across LLM layers. As shown in Figure 2a, we illustrate the trends of $\|\nabla_{\mathbf{h}} \log \pi(y_t|\mathbf{h}_0)\|$ (the green line) and $\|\Delta \mathbf{h}\|$ (the yellow line) across the layers of `Llama3.2-3B` on the ARC Challenge dataset.[4] We observe that the gradient values are higher in the earlier layers of the model and lower in the later layers. However, unlike the clustering behavior observed in traditional classification tasks, $\|\Delta \mathbf{h}\|$ **gradually increases** from close to 0 to approximately 70 across the model layers. Because the upper bound is calculated by multiplying $\|\nabla_{\mathbf{h}} \log \pi(y_t|\mathbf{h}_0)\|$ by $\|\Delta \mathbf{h}\|$. Additionally, the gradient is not less than 0 (Figure 2b). The increase of $\|\Delta \mathbf{h}\|$ leads to an increasing trend of the upper bound across layers, making it impossible to converge to sufficiently low values and **hard to constrain** $|\Delta \log \pi(y_t \mid \mathbf{h})|$ to 0 via the upper bound. This means that the log probabilities of the next token for the two meaning-preserving prompts are not exactly equal.

In summary, we have the following interpretation for prompt sensitivity of LLMs. First, LLMs do not exhibit the clustering behavior that is found in traditional neural networks. This clustering behavior serves as a crucial role in allowing neural networks to accurately perform classification tasks. Secondly, as LLMs tend to pull meaning-preserving prompts farther apart in the representation space, this leads to giving $|\Delta \log \pi(y_t \mid \mathbf{h})|$ a large upper bound $\|\nabla_{\mathbf{h}} \log \pi(y_t|\mathbf{h}_0)\| \cdot \|\Delta \mathbf{h}\|$. This makes it hard to constrain $|\Delta \log \pi(y_t \mid \mathbf{h})|$ to 0. In other words, because LLMs do not exhibit clustering behavior for meaning-preserving prompts, they can only learn each sample individually during training. This cannot guaranty that the model fits meaning-preserving prompts to the same degree, leading to different outputs. To further verify the causal role of $\|\Delta \mathbf{h}\|$, we conduct an activation steering experiment that directly forces $\|\Delta \mathbf{h}^{(l)}\| = 0$ at a chosen layer; steering consistently reduces the observed prompt sensitivity, empirically confirming this causal role. Details and full results are provided in Appendix G.

[4]Experimental results for other models are provided in Appendix F.1.

4.2 Which Types of Prompt Modifications Are More Likely to Lead to Higher Upper Bounds? (RQ2)

Understanding which types of prompt modifications are more likely to lead to prompt sensitivity is important. This can provide evidence for anticipating and preventing biases caused by prompt sensitivity. To investigate which types of prompt modifications are more likely to lead to higher prompt sensitivity, we create seven modification types to modify the prompt template. Our seven modification types are as follows:[5]

1. Modification first ($\mathbf{h}_{first}$): replace one token in the first half of the seed prompt template with a synonymous token.

2. Modification latter ($\mathbf{h}_{latter}$): replace one token in the latter half of the seed prompt template with a synonymous token.

3. Misalignment fewer ($\mathbf{h}_{fewer}$): modify a few tokens in the seed prompt template to make them slightly token misalignment.

4. Misalignment more ($\mathbf{h}_{more}$): modify the token order in the seed prompt template to make them significant token misalignment.

5. Typographical errors ($\mathbf{h}_{typo}$): apply keyboard-level typos (insertion, omission, transposition, or substitution) to up to k randomly-selected words in the seed prompt.

6. Orthographic errors ($\mathbf{h}_{orth}$): apply k surface-level formatting perturbations (extra spaces, missing spaces, random case flips, or extra punctuation) to the seed prompt.

7. Paraphrases ($\mathbf{h}_{para}$): use an LLM (gpt-5.4; OpenAI, 2025) to rewrite the seed prompt by replacing exactly k words with semantically equivalent alternatives.

Types 5 to 7 are applied to the prompt body rather than the template. A word here means an alphabetic token of at least three letters. For $\mathbf{h}_{typo}$, each inserted or substituted character is drawn from the QWERTY-neighbor set of the original character, with letter case preserved. For $\mathbf{h}_{orth}$, each perturbation is drawn uniformly from four operations, duplicating an existing space, removing the space after a sentence-ending period, flipping the case of a single letter inside a word, or inserting one of ',' / '.' / ';' / ':' after a word, none of which inserts, deletes, or substitutes letters within a word. For $\mathbf{h}_{para}$, the LLM returns the rewritten prompt together with k (original, replacement) pairs; outputs that fail the word-count check are retried, and results are cached so that the same paraphrase is reused across all 11 models.

We randomly select 500 samples from each dataset. Types 1 to 4 are evaluated on the four MCQ datasets only, as they are defined on a fixed MCQ template; types 5 to 7 are additionally evaluated on the open-ended generation dataset Alpaca. For modification types 1 to 4, we create three different variants per type. For types 5 to 7, we vary the severity $k \in \{1, 2, 3\}$ and produce one variant per $(type, k)$ pair, yielding 3 variants per type.[6] We refer to the seed prompt template as $\mathbf{h}_{seed}$ and the seven modified versions as $\mathbf{h}_{first}$, $\mathbf{h}_{latter}$, $\mathbf{h}_{fewer}$, $\mathbf{h}_{more}$, $\mathbf{h}_{typo}$, $\mathbf{h}_{orth}$, and $\mathbf{h}_{para}$, respectively. From Eq. (11), when $\mathbf{h}_0$ is fixed, the upper bound of $|\Delta \log \pi(y_t \mid \mathbf{h})|$ is determined by $\|\Delta\mathbf{h}\|$. This implies that a higher $\|\Delta\mathbf{h}\|$ imposes a looser constraint on $|\Delta \log \pi(y_t \mid \mathbf{h})|$. Therefore, we calculate $\|\Delta\mathbf{h}\|$ between $\mathbf{h}_{seed}$ and the seven types of modifications for comparison.

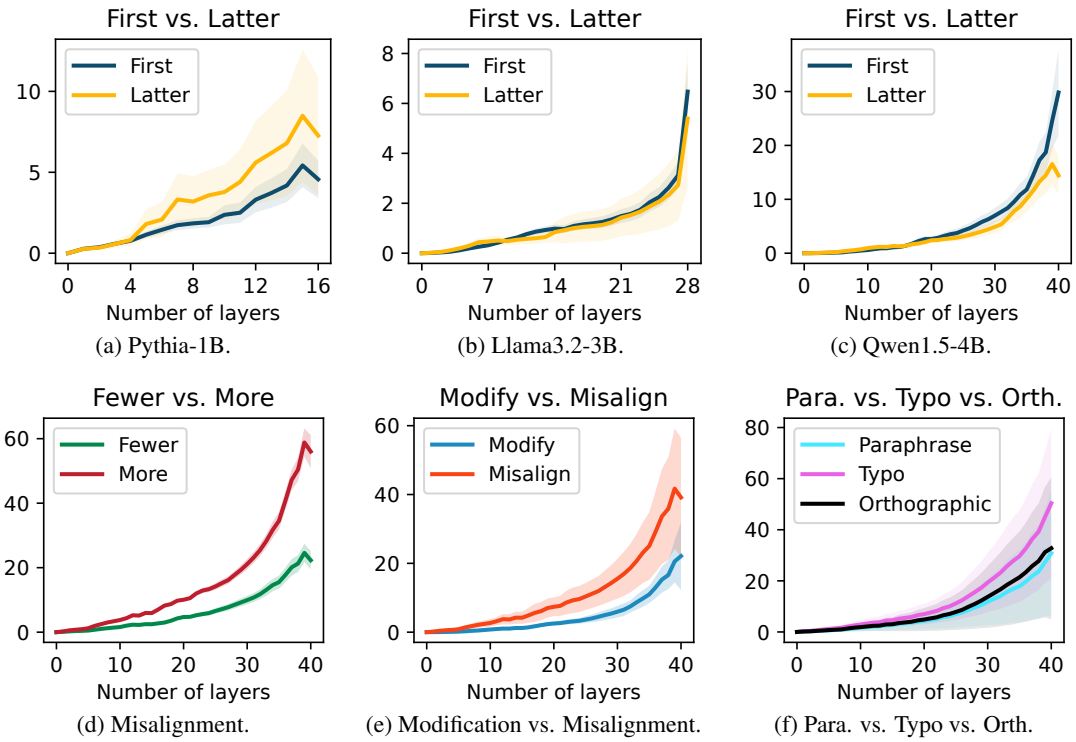
Figure 3 shows the comparison results. Experimental results show that when comparing $\mathbf{h}_{first}$ and $\mathbf{h}_{latter}$, smaller models such as `Pythia-1B` (Figure 3a) exhibit a higher $\|\Delta\mathbf{h}\|$ of $\mathbf{h}_{latter}$ than that of $\mathbf{h}_{first}$. However, as model size increases, such as in `Llama3.2-3B` (Figure 3b), the $\|\Delta\mathbf{h}\|$ of $\mathbf{h}_{latter}$ becomes comparable to that of $\mathbf{h}_{first}$. When model size increases to 4B, such as `Qwen1.5-4B` (Figure 3c), $\mathbf{h}_{first}$'s $\|\Delta\mathbf{h}\|$ surpasses $\mathbf{h}_{latter}$'s $\|\Delta\mathbf{h}\|$. Within the `Qwen1.5` and `Llama3.2` series, smaller models are more sensitive to latter-half modifications, while larger models are more sensitive to first-half modifications. This size-dependent transition is not observed in the older `GPT2` and `Pythia` series (Appendix F.2).

Figure 3d shows the comparison results between tokens with fewer ($\mathbf{h}_{fewer}$) and more ($\mathbf{h}_{more}$) misalignments in the prompt. We observe that different numbers of token misalignments produce significant differences in $\Delta\mathbf{h}$. Specifically, $\mathbf{h}_{more}$ is more likely than $\mathbf{h}_{fewer}$ to lead to prompt sensitivity.

In addition, we compare the trends of $\|\Delta\mathbf{h}\|$ for

[5] For details of the prompt templates, please refer to Appendix E.2.

[6] Here, $type$ refers to one of $\mathbf{h}_{typo}$, $\mathbf{h}_{orth}$, or $\mathbf{h}_{para}$; each type thus consists of three variants, one per $k \in \{1, 2, 3\}$.



(a) Pythia-1B. (b) Llama3.2-3B. (c) Qwen1.5-4B.

(d) Misalignment. (e) Modification vs. Misalignment. (f) Para. vs. Typo vs. Orth.

Figure 3: Key results of RQ2: (a), (b), and (c) indicate the trend of $\|\Delta\mathbf{h}\|$ when modifying the first and latter half of the prompt templates. (d), (e), and (f) are results on `Qwen1.5-4B`; (d) and (e) use the ARC Challenge dataset, (f) uses the Alpaca dataset. (d) indicates the trend of $\|\Delta\mathbf{h}\|$ when the prompt templates have fewer and more tokens misaligned. (e) indicates the trend of $\|\Delta\mathbf{h}\|$ between modification and misalignment. (f) indicates the trend of $\|\Delta\mathbf{h}\|$ among paraphrase, typo, and orthographic modifications. Full results are provided in Appendix F.2

modification and misalignment. Here, modification refers to the average result of $\mathbf{h}_{first}$ and $\mathbf{h}_{latter}$, while misalignment denotes the average result of $\mathbf{h}_{fewer}$ and $\mathbf{h}_{more}$. As shown in Figure 3e, misaligned prompt tokens are more likely to lead to prompt sensitivity than modified prompt tokens.

Figure 3f compares $\mathbf{h}_{typo}$, $\mathbf{h}_{orth}$, and $\mathbf{h}_{para}$ on the Alpaca dataset. We observe that $\mathbf{h}_{typo}$ induces the highest $\|\Delta\mathbf{h}\|$, followed by $\mathbf{h}_{orth}$ and then $\mathbf{h}_{para}$. This indicates that character-level typographical errors are more likely to lead to prompt sensitivity than surface-level orthographic perturbations or word-level paraphrases, as typos disrupt subword tokenization most aggressively while paraphrases preserve the majority of tokens. See Appendix F.2 for more experimental results.

4.3 What Is the Relationship Between the Upper Bound and an Existing Prompt Sensitivity Metric? (RQ3)

Many studies (Zhuo et al., 2024; Chatterjee et al., 2024) propose prompt sensitivity metrics to evaluate the prompt sensitivity of LLMs. These metrics are typically single values that represent the sensitivity of LLMs to different meaning-preserving prompt templates. According to Eq. (11), a higher upper bound makes it more difficult for LLMs to achieve $|\Delta \log \pi(y_t \mid \mathbf{h})|$ close to 0. Therefore, what is the relationship between the upper bound and the prompt sensitivity of LLMs? To answer this question, we compare the prompt sensitivity metric `PromptSensiScore` (PSS; Zhuo et al., 2024) with the upper bound. PSS is a value ranging from 0 to 1, where a lower value indicates lower prompt sensitivity.[7] Theoretically, LLMs with smaller upper bounds should have a higher chance of achieving lower PSS. Figure 4 shows the relationship between the average upper bound across layers of LLMs and PSS. We can observe that within the same model series, the average upper bound is positively correlated with PSS. This indicates that the smaller the average upper bound, the greater the chance that LLMs achieve lower prompt sensitivity. The positive correlation between upper bounds and PSS further indicates that upper bounds influence the prompt sensitivity of LLMs.

[7]The calculation method for PSS is provided in Appendix D.

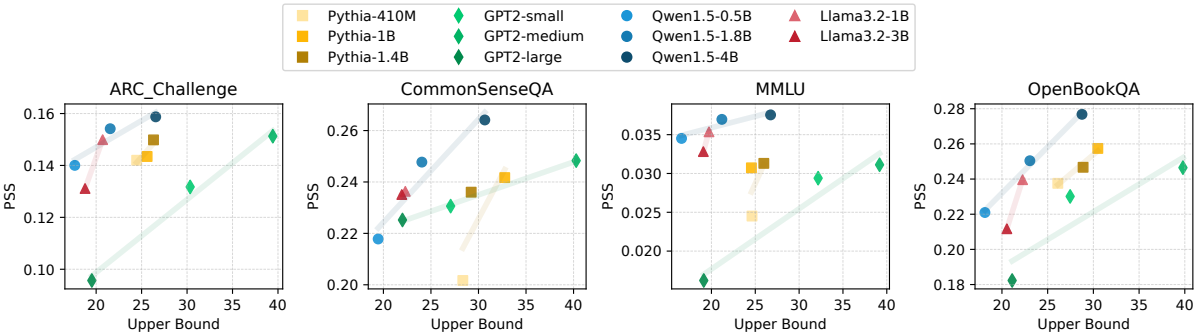


Figure 4: The relationship between the average upper bound across layers of LLMs and PSS. The lines represent linear fits of points within the same model series.

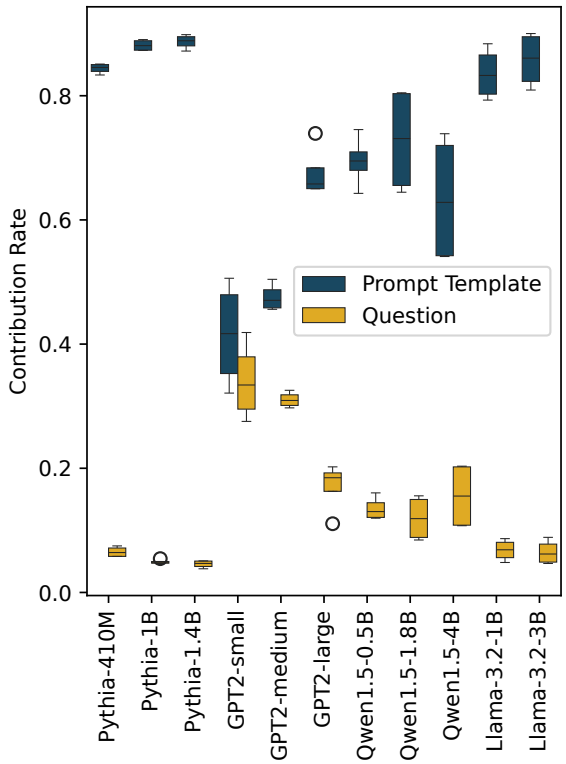


Figure 5: Comparison of contribution rates of prompt templates and questions to logits.

4.4 Which Factor Contributes to the Change of Logits? (RQ4)

Wu and Varshney (2025)'s study indicates that LLMs tend to cluster the same task samples. This inspires us to evaluate whether the outputs of LLMs are more influenced by the prompt template or the question itself. In particular, we construct an ordinary least squares regression model that uses different prompt templates and questions to predict the logit of the model's next token. Subsequently, we perform an analysis of variance (ANOVA) on the regression results to calculate each factor's contribution to the total variance and further determine each factor's proportion of contribution relative to the total sum of squares. Figure 5 compares the contributions of the prompt template and question to the logits. The prompt template is the primary factor explaining the logit variation. Except for `GPT2-small` and `GPT2-medium`, the contribution rate of prompt templates to logits significantly exceeds that of questions. In addition, the `Pythia` series model exhibits the lowest variance in contribution rate across all datasets, while the `Qwen1.5` series model shows the highest variance in contribution rate across all datasets. This indicates that the `Qwen1.5` series model is significantly sensitive to different datasets.

5 Related Work

5.1 Prompt Sensitivity of LLMs

LLMs have strong in-context learning capabilities (Brown et al., 2020), enabling them to perform diverse tasks based on prompts, often without requiring additional fine-tuning (Radford et al., 2019a; Raffel et al., 2020; Gao et al., 2021). However, the stability and reliability of this learning approach remain controversial (Weber et al., 2023). Existing studies indicate that model outputs are highly dependent on multiple factors, such as the choice and order of examples (Liu et al., 2022; SU et al., 2023; Lu et al., 2022; Zhao et al., 2021), the definition of input labels (Min et al., 2022), and the phrasing of prompts (Gu et al., 2023; Sun et al., 2024). Beyond these factors, LLMs exhibit extreme sensitivity to minor changes in prompt structure or phrasing, even when such alterations preserve semantic meaning. This phenomenon has been systematically explored in numerous studies (Voronov et al., 2024; Mizrahi et al., 2024),

indicating that subtle modifications to prompts can significantly impact model outputs. Furthermore, to characterize and compare the prompt sensitivity of different models, numerous studies (Zhuo et al., 2024; Chatterjee et al., 2024) have constructed specialized benchmarks to quantify and evaluate models' robustness to prompt perturbations. Contrary to previous work, this study attempts to represent LLMs as functions, leveraging Taylor expansion to explain the mechanism behind prompt sensitivity from the function perspective. It provides both theoretical foundations and empirical evidence to explain why LLMs exhibit prompt sensitivity.

5.2 LLMs as Functions

In recent years, some studies have attempted to characterize LLMs from the perspective of function mapping (Brown et al., 2020; Wei et al., 2022). This perspective abstracts an LLM as a function mapping x to a distribution over y. In other words, given a prompt x, the model defines a distribution $P(y \mid x)$ for an output y. This functional representation facilitates a unified understanding of model behavior across different tasks and provides a theoretical framework for analyzing LLM generalization and robustness. Notably, it has also been shown that transformers themselves serve as universal approximators of sequence-to-sequence functions (Yun et al., 2020), further reinforcing the perspective that LLMs are functions. Building on this idea of function mapping, some studies consider prompt engineering as a design problem for function call interfaces, investigating how different prompt formats alter the properties of the function mapping (Liu et al., 2023). In our study, we consider LLMs as composite functions that can be split into the feature processing part and the function part between any transformer blocks. This split allows us to perform a Taylor expansion on any part of the models for analysis.

6 Conclusion

Prompt sensitivity, which describes how LLMs produce different outputs in response to meaning-preserving prompts, raises user concerns about the stability and reliability of LLMs. To investigate the underlying mechanisms of prompt sensitivity and to better understand LLMs, we started by considering LLMs as multivariate continuous functions. We pointed out that improving classification accuracy requires internal clustering behavior within neural networks. Then, we applied the first-order Taylor expansion to LLMs. By observing changes in hidden states across all layers, we found that transformer-based LLMs lacked this clustering behavior, leading to a high upper bound on the difference in log probabilities between two prompts. We also identified which types of modifications are more likely to lead to prompt sensitivity. Moreover, the upper bound of the difference in log probabilities correlated positively with an existing prompt sensitivity metric. Counterintuitively, our analysis revealed that prompt templates contributed more significantly to logits than the questions themselves. In the future, we will attempt to introduce higher-order Taylor terms (such as second-order terms implemented via the Hessian matrix) to achieve more precise and faithful bounds. We also plan to extend this work to analyze the entire log probability space and multi-step generation process.

Limitations

One limitation of this work is we only considered the log probabilities of a single dimension for the model's next token, implicitly requiring the entire logit distribution of the next token to remain consistent across meaning-preserving prompts. This requirement poses a significant challenge for LLMs. Moreover, we employed only a first-order Taylor expansion. Given that LLMs are naturally highly complex functions, this linear approximation may introduce some errors. In the future, exploring higher-order Taylor expansions could yield more precise approximations.

Acknowledgments

This work was supported by JST BOOST, Grant Number JPMJBS2407. We thank the constructive comments from the anonymous reviewers, which helped improve this work. We also appreciate the careful attention of the meta reviewer.

References

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, and 1 others. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, and 1 others. 2023. Pythia: A suite for analyzing large language

models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Augustin Louis Baron Cauchy. 1821. *Cours d'analyse de l'École Royale Polytechnique*. Imprimerie royale.

Anwoy Chatterjee, H S V N S Kowndinya Renduchintala, Sumit Bhatia, and Tanmoy Chakraborty. 2024. POSIX: A prompt sensitivity index for large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 14550–14565, Miami, Florida, USA. Association for Computational Linguistics.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

George Cybenko. 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314.

Guanting Dong, Hongyi Yuan, Keming Lu, Chengpeng Li, Mingfeng Xue, Dayiheng Liu, Wei Wang, Zheng Yuan, Chang Zhou, and Jingren Zhou. 2024. How abilities in large language models are affected by supervised fine-tuning data composition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 177–198, Bangkok, Thailand. Association for Computational Linguistics.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. 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 3816–3830, Online. Association for Computational Linguistics.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323. JMLR Workshop and Conference Proceedings.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. *Deep learning*, volume 1. MIT Press.

Jiasheng Gu, Hongyu Zhao, Hanzi Xu, Liangyu Nie, Hongyuan Mei, and Wenpeng Yin. 2023. Robustness of learning from task instructions. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13935–13948, Toronto, Canada. Association for Computational Linguistics.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Thomas Little Heath. 1981. *A history of Greek mathematics*, volume 1. Courier Corporation.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. 1989. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.

Alex Inglis. 1940. James gregory. tercentenary memorial volume. edited by hw turnbull. pp. viii, 524; 5 plates. 25s. 1939.(published for the royal society of edinburgh by g. bell & sons). *The Mathematical Gazette*, 24(259):125–129.

Alex Krizhevsky and 1 others. 2009. Learning multiple layers of features from tiny images.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature*, 521(7553):436–444.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in nlp. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 681–691.

David C. Lindberg. 1992. *The Beginnings of Western Science: The European Scientific Tradition in Philosophical, Religious, and Institutional Context, 600 B.C. to A.D. 1450*. University of Chicago Press, Chicago.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, Dublin, Ireland and Online. Association for Computational Linguistics.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM computing surveys*, 55(9):1–35.

Weiyang Liu, Yandong Wen, Zhiding Yu, and Meng Yang. 2016. Large-margin softmax loss for convolutional neural networks. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 507–516. JMLR.org.

Yang Liu, Masahiro Kaneko, and Chenhui Chu. 2026. On the alignment of large language models with global human opinion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 37673–37681.

Ziche Liu, Rui Ke, Yajiao Liu, Feng Jiang, and Haizhou Li. 2025. Take the essence and discard the dross: A rethinking on data selection for fine-tuning large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6595–6611, Albuquerque, New Mexico. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098, Dublin, Ireland. Association for Computational Linguistics.

Jean-Claude Martzloff. 2007. *A history of Chinese mathematics*. Springer.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Moran Mizrahi, Guy Kaplan, Dan Malkin, Rotem Dror, Dafna Shahaf, and Gabriel Stanovsky. 2024. State of what art? a call for multi-prompt LLM evaluation. *Transactions of the Association for Computational Linguistics*, 12:933–949.

Kevin P Murphy. 2012. *Machine learning: a probabilistic perspective*. MIT press.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

Michael A Nielsen. 2015. *Neural networks and deep learning*, volume 25. Determination press San Francisco, CA, USA.

OpenAI. 2025. Introducing gpt-5. https://openai.com/index/introducing-gpt-5/. Accessed: 2026-04-17.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019a. Language models are unsupervised multitask learners.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019b. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Taori Rohan, Gulrajani Ishaan, Zhang Tianyi, Dubois Yann, Li Xuechen, Guestrin Carlos, Liang Percy, and B. Hashimoto Tatsunori. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

Hermann Amandus Schwarz. 1890. Ueber ein die flächen kleinsten flächeninhalts betreffendes problem der variationsrechnung: Festschrift zum siebzigsten geburtstage des herrn karl weierstrass. In *Gesammelte Mathematische Abhandlungen: Erster Band*, pages 223–269. Springer.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2024. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. In *The Twelfth International Conference on Learning Representations*.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.

Hongjin SU, Jungo Kasai, Chen Henry Wu, Weijia Shi, Tianlu Wang, Jiayi Xin, Rui Zhang, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. 2023. Selective annotation makes language models better few-shot learners. In *The Eleventh International Conference on Learning Representations*.

Jiuding Sun, Chantal Shaib, and Byron C Wallace. 2024. Evaluating the zero-shot robustness of instruction-tuned language models. In *The Twelfth International Conference on Learning Representations*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Brook Taylor. 1715. *Methodus incrementorum directa*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Anton Voronov, Lena Wolf, and Max Ryabinin. 2024. Mind your format: Towards consistent evaluation of in-context learning improvements. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 6287–6310, Bangkok, Thailand. Association for Computational Linguistics.

Lucas Weber, Elia Bruni, and Dieuwke Hupkes. 2023. The icl consistency test. *arXiv preprint arXiv:2312.04945*.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Xinbo Wu and Lav R. Varshney. 2025. Transformer-based causal language models perform clustering. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 5347–5372, Albuquerque, New Mexico. Association for Computational Linguistics.

Luo Yan, Wong Yongkang, Mohan Kankanhalli, and Qi Zhao. 2020. G-softmax: Improving intraclass compactness and interclass separability of features. *IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS*, 31(2):685.

Kayo Yin and Graham Neubig. 2022. Interpreting language models with contrastive explanations. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 184–198, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank Reddi, and Sanjiv Kumar. 2020. Are transformers universal approximators of sequence-to-sequence functions? In *International Conference on Learning Representations*.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12697–12706. PMLR.

Jingming Zhuo, Songyang Zhang, Xinyu Fang, Haodong Duan, Dahua Lin, and Kai Chen. 2024. ProSA: Assessing and understanding the prompt sensitivity of LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1950–1976, Miami, Florida, USA. Association for Computational Linguistics.

A Taylor Expansion

A.1 Background

The roots of Taylor expansion can be traced back to early thoughts on infinity, such as the paradoxes of divisibility proposed by the ancient Greek philosopher Zeno (Lindberg, 1992), as well as the "method of exhaustion" developed by Archimedes (Heath, 1981) and later by Liu Hui (Martzloff, 2007), which laid the foundation for approximating infinite processes through finite steps. In the 14th century, Indian mathematician Madhava of Sangamagrama and his successors in the Kerala school developed series expansions for functions such as sine, cosine, and arctangent, marking the earliest concrete examples of power series methods analogous to later Taylor expansions (Lindberg, 1992). In the 17th century, Newton and Gregory independently developed general methods for expanding functions (Inglis, 1940). Later, Brook Taylor first systematically proposed an expansion method applicable to general functions in 1715, forming the basis of today's Taylor expansions (Taylor, 1715). In our study, we consider LLMs as functions and employ first-order Taylor expansions to connect prompts, their gradients, and the logit of the model's next token, thereby analyzing the constraint relationships among them.

A.2 The First-order Taylor Expansion

In mathematics, the Taylor series or Taylor expansion of a function is an infinite sum of terms that are expressed in terms of the function's derivatives at a single point. The partial sum formed by the first $n+1$ terms of a Taylor series is a polynomial of degree n that is called the nth Taylor polynomial of the function. Taylor polynomials are approximations of a function, which become generally more accurate as n increases. The first-order Taylor expansion in one variable of $f(x)$ about $x = a$ is as follows:

$$\begin{aligned} f(x) =& f(a) + f'(a)(x-a) \\ &+ \mathcal{O}((x-a)^2) \quad (x \to a). \end{aligned} \tag{15}$$

where $\mathcal{O}(x-a)$ indicates the infinitesimal term of higher order than $(x-a)$, and $x \to a$ indicates that this equality holds as x approaches a. In other words, this expansion is a local approximation describing the behavior of $f(x)$ near $x = a$.

For more complex multivariate scenarios, we suppose $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable at the point $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$. Then the first-order Taylor expansion of f at $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ is:

$$\begin{aligned} f(\boldsymbol{x}) = f(\boldsymbol{a}) + \nabla f(\boldsymbol{a}) \cdot (\boldsymbol{x} - \boldsymbol{a}) \\ + \mathcal{O}(\|\boldsymbol{x} - \boldsymbol{a}\|^2) \quad (\boldsymbol{x} \to \boldsymbol{a}). \end{aligned} \tag{16}$$

where $\nabla f(\boldsymbol{a}) = \left(\frac{\partial f}{\partial x_1}(\boldsymbol{a}), \frac{\partial f}{\partial x_2}(\boldsymbol{a}), \ldots, \frac{\partial f}{\partial x_n}(\boldsymbol{a})\right)$ is the gradient. In the expression $\mathcal{O}(\|\boldsymbol{x} - \boldsymbol{a}\|)$, the norm $\|\cdot\|$ can be any norm (such as the Euclidean norm (2-norm) or vector norm) on $\mathbb{R}^n$, because all norms in finite-dimensional spaces are equivalent. The $\mathcal{O}(\|\boldsymbol{x} - \boldsymbol{a}\|^2)$ means the remainder term that vanishes faster than $\|\boldsymbol{x} - \boldsymbol{a}\|^2$ as $\boldsymbol{x} \to \boldsymbol{a}$. The operator $\cdot$ denotes the dot product.

B Proof

Statement. Suppose $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are normalized unit vectors, i.e., $\|\boldsymbol{x}_i\|^2 = \|\boldsymbol{x}_j\|^2 = 1$, and θ_{ij} is the angle between them, then the following holds:

$$\|\boldsymbol{x}_i - \boldsymbol{x}_j\| = \sqrt{2 - 2\cos\theta_{ij}} \tag{17}$$

Proof. As $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are unit vectors,

$$\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 = \langle \boldsymbol{x}_i - \boldsymbol{x}_j,\ \boldsymbol{x}_i - \boldsymbol{x}_j \rangle \tag{18}$$

$$= \|\boldsymbol{x}_i\|^2 + \|\boldsymbol{x}_j\|^2 - 2\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \tag{19}$$

$$= 1 + 1 - 2\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \tag{20}$$

$$= 2 - 2\|\boldsymbol{x}_i\|\,\|\boldsymbol{x}_j\|\cos\theta_{ij} \tag{21}$$

$$= 2 - 2\cos\theta_{ij}. \tag{22}$$

Taking square roots yields: $\|\boldsymbol{x}_i - \boldsymbol{x}_j\| = \sqrt{2 - 2\cos\theta_{ij}}$.

C Hyperparameters for Training ResNet.

To ensure stable optimization and efficient convergence of the `ResNet-101` network on the CIFAR-10 dataset, a carefully designed hyperparameter configuration scheme was employed during training.

As shown in Figure 6, our network architecture is a ResNet connected to a projection layer and a fully connected layer. This section provides more details. We preprocess images using the following pipeline before feeding them into ResNet:

```
transform = transforms.Compose([
    transforms.Resize(112),
    transforms.CenterCrop(112),
    transforms.ToTensor()
])
```

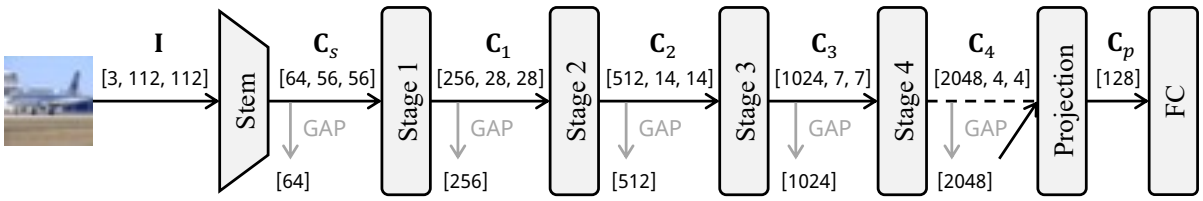


Figure 6: The feature maps' shape of the neural network. The batch size dimension is omitted and "GAP" indicates global mean pooling.

We project the 2048-dimensional features from ResNet's stage 4 output onto a 128-dimensional embedding space, then classify them using a fully connected (classification) layer. The specific network architecture is as follows:

```
proj = nn.Sequential(
    nn.Linear(2048, 512),
    nn.BatchNorm1d(512),
    nn.PReLU(),
    nn.Dropout(p=0.2),
    nn.Linear(512, 128)
)

clf = nn.Linear(128, num_classes)
```

Our experiment employs the cross-entropy loss function with the AdamW optimizer (Loshchilov and Hutter, 2017), using the macro F1 score as the primary evaluation metric. The training process utilizes a batch size of 128 and runs for 20 epochs.

Input and Output Shapes. As shown in Figure 6, we mark the shape of each feature map. Here, "GAP" denotes the global average pooling operation. After performing L^2 normalization on the vector obtained from global average pooling, the distance is calculated using Eq. (5). It is especially worth noting that the output $\mathbf{C}_4$ of "Stage 4" undergoes global average pooling before being fed into the "Projection" layer as input.

D Prompt Sensitivity Metric: PSS

In this section, we introduce PSS (Zhuo et al., 2024). For each set of all prompt variants under the same question, we have:

$$S = \frac{\sum_{p_i, p_j \in P}(|Y(P_i) - Y(P_j)|)}{C(|P|, 2)}, \qquad (23)$$

where $Y(p)$ represents the performance metric under prompt p. In our study, $Y(p)$ refers to correctness. $|Y(P_i) - Y(P_j)|$ represents the absolute value difference in performance metrics between prompt p_i and prompt p_j. $C(|P|, 2)$ represents the count of prompt pairs in the same question. PSS is given by the following:

$$PSS = \frac{1}{N}\sum_{i=1}^{N} S_i, \qquad (24)$$

where N is the total number of questions and S_i is the score for the i-th question.

E Prompt Templates

E.1 Meaning-Preserving Prompt Templates

In this section, we provide the 12 prompt templates provided by Zhuo et al. (2024) mentioned in § 4. For multiple-choice questions with 4 options, the templates are shown in Table 1. We choose 12 prompts for experimentation to ensure data diversity and avoid inaccurate results caused by individual edge cases.

E.2 Modification and Misalignment Prompt Templates

To evaluate which types of prompts may lead to higher prompt sensitivity, we create four prompt templates for quantitative analysis. These four prompt templates are shown in Table 2. These prompt templates are modified from a seed prompt template, which is: "`You are a very helpful AI assistant. Please answer the following questions:\nQuestion: {question}\nA. {A} B. {B} C. {C} D. {D}\nPlease choose the best option and respond only with the option of the correct answer (A, B, C, or D).\nAnswer:`"

Our experimental implementation process is as follows: We first randomly select 500 samples from each of the four datasets. We then combine these samples with both the seed prompt template and our modified 12 prompt templates, creating 6,500 prompts for each dataset. These prompts feed into the LLMs for testing.

F More Experimental Results

F.1 More Results of RQ1

Figure 7 shows the trends of $\|\nabla_{\mathbf{h}} \log \pi(y_t|\mathbf{h}_0)\|$ and $\|\Delta\mathbf{h}\|$ across layers for all models on the four datasets. We can observe that the trends in $\|\nabla_{\mathbf{h}} \log \pi(y_t|\mathbf{h}_0)\|$ and $\|\Delta\mathbf{h}\|$ across models within the same series are similar across all datasets, indicating that our findings are broadly applicable. Although $\|\Delta\mathbf{h}\|$ may suddenly decrease in certain layers of some models (for example, `Pythia-1B/1.4B` and `GPT2-small/medium/large`), $\|\nabla_{\mathbf{h}} \log \pi(y_t|\mathbf{h}_0)\|$ simultaneously increases abruptly to prevent the upper bound from dropping too low. Figure 8 shows the trends of upper bounds. Experimental results indicate that upper bounds exhibit increasing trends across all models and datasets, aligning with the conclusion of our RQ1: although gradients decrease across layers, the increase in $\|\Delta\mathbf{h}\|$ prevents the upper bound from becoming sufficiently low for $|\Delta \log \pi(y_t \mid \mathbf{h})|$ to approach 0.

F.2 More Results of RQ2

In this section, we provide the results of all models across the four datasets. Figure 9 compares "First" and "Latter." We observe that the ordering between $\|\Delta\mathbf{h}\|$ for "First" and "Latter" is not universal, but depends on the model series. Within the `Pythia` series, $\|\Delta\mathbf{h}\|$ for "Latter" is consistently higher than $\|\Delta\mathbf{h}\|$ for "First" across all examined sizes (`Pythia-410M`, `Pythia-1B`, and `Pythia-1.4B`). Within the `GPT2` series, the ordering varies inconsistently with model size: $\|\Delta\mathbf{h}\|$ for "Latter" is lower than for "First" on `GPT2-small` and `GPT2-medium`, but higher on `GPT2-large`, so no monotonic size trend is observed. Within the `Qwen1.5` series, we observe a clear size-dependent transition: $\|\Delta\mathbf{h}\|$ for "Latter" is higher than for "First" on `Qwen1.5-0.5B` and `Qwen1.5-1.8B`, becomes comparable on intermediate sizes, and is exceeded by $\|\Delta\mathbf{h}\|$ for "First" on `Qwen1.5-4B`. Within the `Llama3.2` series, $\|\Delta\mathbf{h}\|$ for "Latter" is higher than for "First" on `Llama3.2-1B`, and the two become comparable on `Llama3.2-3B`, which loosely follows the same size-dependent transition. These results refine the earlier conclusion: the size-dependent transition from latter-sensitive to first-sensitive holds within more recent model series (`Qwen1.5` and `Llama3.2`), but does not extend to older series (`Pythia` and `GPT2`). We attribute this cross-series variation to differences in pretraining data and objectives across model eras. Figure 10 compares "Fewer" and "More," in all cases, $\|\Delta\mathbf{h}\|$ for "More" is higher than $\|\Delta\mathbf{h}\|$ for "Fewer." Figure 11 compares "Modify" and "Misalign," in all cases, $\|\Delta\mathbf{h}\|$ for "Modify" is higher than $\|\Delta\mathbf{h}\|$ for "Misalign." Figure 12 compares "Paraphrase," "Typo," and "Orthographic" across all 11 models and 5 datasets. In most cases, $\|\Delta\mathbf{h}\|$ decreases in the order "Typo" > "Orthographic" > "Paraphrase," consistent with the observation on Alpaca in Figure 3f. All the experimental results align with the conclusions of RQ2.

G Mitigating Prompt Sensitivity via Activation Steering

The Taylor-expansion analysis in § 3 and experimental verification in § 4 identify $\|\Delta\mathbf{h}\|$ as the primary driver of the upper bound on prompt sensitivity. This suggests that reducing $\|\Delta\mathbf{h}\|$ at a target layer should directly reduce the model's output divergence between meaning-preserving prompts. We verify this hypothesis with activation steering, an intervention that forces $\|\Delta\mathbf{h}^{(l)}\| = 0$ at a chosen layer l.

G.1 Method

Given a seed prompt p_A and a meaning-preserving variant p_B, let $\mathbf{h}_A^{(l)}, \mathbf{h}_B^{(l)}$ denote their hidden states at layer l. We construct a steered forward pass for p_A by overwriting $\mathbf{h}_A^{(l)} \leftarrow \mathbf{h}_B^{(l)}$ and continuing the forward computation with the original model weights for layers $l+1, \ldots, L$. This sets $\|\Delta\mathbf{h}^{(l)}\| = 0$ at the intervention layer. We then measure the resulting prompt sensitivity as $|\Delta \log \pi(y_t \mid \mathbf{h})|$ between the steered forward of p_A and the natural forward of p_B.

G.2 Experiments

We apply steering at three depths $l \in \{L/4, L/2, 3L/4\}$ for all 11 models on the four MCQ datasets. Figure 13 reports $|\Delta \log \pi(y_t \mid \mathbf{h})|$ averaged over meaning-preserving prompt pairs, comparing the non-steered baseline (red) with the steered forward pass (green).

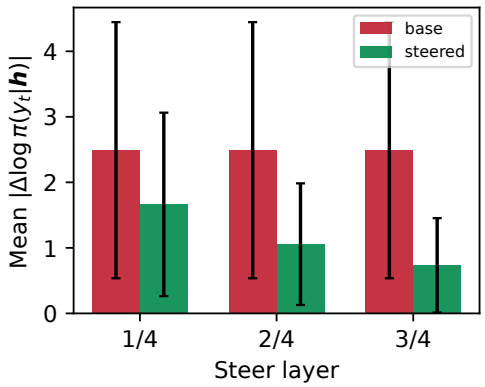


Figure 13: Activation steering on `Qwen1.5-4B` (ARC Challenge). Bars show the mean $|\Delta \log \pi(y_t \mid \mathbf{h})|$ before (red) and after (green) steering at three layer depths $l \in \{L/4, L/2, 3L/4\}$.

Steering consistently reduces prompt sensitivity, and the reduction grows as the intervention layer goes deeper. For example, on `Qwen1.5-4B` with the ARC Challenge, the baseline $|\Delta \log \pi(y_t \mid \mathbf{h})|$ is 2.49; steering at $L/4$, $L/2$, and $3L/4$ reduces it to 1.66, 1.06, and 0.73 respectively. This confirms that $\|\Delta \mathbf{h}\|$ plays a causal role in the observed prompt sensitivity predicted by our Taylor-expansion analysis: forcing $\|\Delta \mathbf{h}\|$ to zero at any layer proportionally lowers the downstream log-probability divergence, with the effect being strongest when the intervention occurs closer to the output. Full results across the 11 models and 4 datasets are provided in Figure 15. Across all combinations, steering reduces $|\Delta \log \pi(y_t \mid \mathbf{h})|$ at every intervention depth, with monotonically larger reductions at deeper layers. This pattern further confirms that $\|\Delta \mathbf{h}\|$ causally drives prompt sensitivity.

H Other Tokens as y_t in Eq. (9)

In the definition of Eq. (9), y_t can be any token in the vocabulary of the LLMs. This means that the logits of two meaning-preserving prompts should be equal at every position. In this appendix, we analyze whether different values of y_t affect the experimental results. Specifically, in addition to the correct answer, we randomly set y_t to the following two types of values:

1. y_i: A random sample of incorrect answers from the question's options.

2. y_n: A random sample of numbers between 0 and 9.

We mainly demonstrate the effects of the next token y_t on the following two terms:

1. The upper bound $\|\nabla_{\mathbf{h}} \log \pi(y_t|\mathbf{h}_0)\| \cdot \|\Delta \mathbf{h}\|$ of different y_t.

2. The absolute difference of the log probabilities $|\Delta \log \pi(y_t \mid \mathbf{h})|$.

Figure 14 shows the comparison of the upper bounds for y_c, y_i, and y_n. We find that when y_t is either the correct or incorrect option token, the trends of their upper bounds are similar. The upper bound when y_t is a number is higher than the upper bound when y_t is an option (correct or incorrect).

As shown in Figure 16, we compare the absolute differences of the log probabilities for y_c, y_i, and y_n. We can observe that when the next token is an option (correct or incorrect), the value of the absolute difference of the log probabilities is relatively close. However, when the next token is not an option, the value of the absolute difference of the log probabilities becomes significantly lower. This is because the shape of the output probability distribution is significantly sharp, assigning higher probabilities to option tokens and lower probabilities to non-option tokens (Liu et al., 2026). Consequently, the difference between the two lower values has the chance to be relatively lower. In summary, our analysis holds true for any next token y_t.

Table 1: The meaning-preserving prompt templates for ARC Challenge, MMLU, and OpenBookQA datasets. For the CommonSenseQA dataset, the number of options changes from four to five, so option 'E' should be added accordingly. Gray text indicates template slots that need to be replaced.

prompt 1	{question}\nA. {A}\nB. {B}\nC. {C}\nD. {D}\nAnswer:
prompt 2	Question:\n{question}\nA. {A}\nB. {B}\nC. {C}\nD. {D}\nAnswer:
prompt 3	Question:\n{question} A. {A} B. {B} C. {C} D. {D}\nAnswer:
prompt 4	Could you provide a response to the following question: {question} A. {A} B. {B} C. {C} D. {D}
prompt 5	Please answer the following question:\n{question}\nA. {A}\nB. {B}\nC. {C}\nD. {D}
prompt 6	Please address the following question:\n{question}\nA. {A}\nB. {B}\nC. {C}\nD. {D}\nAnswer:
prompt 7	You are a very helpful AI assistant. Please answer the following questions: {question} A. {A} B. {B} C. {C} D. {D}
prompt 8	As an exceptionally resourceful AI assistant, I'm at your service. Address the questions below:\n{question}\nA. {A}\nB. {B}\nC. {C}\nD. {D}
prompt 9	As a helpful Artificial Intelligence Assistant, please answer the following questions\n{question} A. {A}\nB. {B}\nC. {C}\nD. {D}
prompt 10	Could you provide a response to the following question: {question} A. {A} B. {B} C. {C} D. {D}\nAnswer the question by replying A, B, C or D.
prompt 11	Please answer the following question:\n{question}\nA. {A}\nB. {B}\nC. {C}\nD. {D}\nAnswer the question by replying A, B, C or D.
prompt 12	Please address the following question:\n{question}\nA. {A}\nB. {B}\nC. {C}\nD. {D}\nAnswer this question by replying A, B, C or D.

Table 2: Our prompt templates for ARC Challenge, MMLU, and OpenBookQA datasets. For the CommonSenseQA dataset, the number of options changes from four to five, so option 'E' should be added accordingly. Gray text indicates template slots that need to be replaced. Green indicates the modified token in the first half of the prompt. Red indicates the modified token in the latter half of the prompt. Orange indicates the token causing the misalignment in the prompt. Blue indicates that the prompt is completely misaligned.

Seed prompt	You are a very helpful AI assistant. Please answer the following questions:\nQuestion: {question}\nA. {A} B. {B} C. {C} D. {D}\nPlease choose the best option and respond only with the option of the correct answer (A, B, C, or D).\nAnswer:
Modification first	You are a very useful AI assistant. Please answer the following questions:\nQuestion: {question}\nA. {A} B. {B} C. {C} D. {D}\nPlease choose the best option and respond only with the option of the correct answer (A, B, C, or D).\nAnswer: You are a very smart AI assistant. Please answer the following questions:\nQuestion: {question}\nA. {A} B. {B} C. {C} D. {D}\nPlease choose the best option and respond only with the option of the correct answer (A, B, C, or D).\nAnswer: You are a very friendly AI assistant. Please answer the following questions:\nQuestion: {question}\nA. {A} B. {B} C. {C} D. {D}\nPlease choose the best option and respond only with the option of the correct answer (A, B, C, or D).\nAnswer:
Modification latter	You are a very helpful AI assistant. Please answer the following questions:\nQuestion: {question}\nA. {A} B. {B} C. {C} D. {D}\nPlease choose the best option and respond only with the option of the suitable answer (A, B, C, or D).\nAnswer: You are a very helpful AI assistant. Please answer the following questions:\nQuestion: {question}\nA. {A} B. {B} C. {C} D. {D}\nPlease choose the best option and respond only with the letter of the correct answer (A, B, C, or D).\nAnswer: You are a very helpful AI assistant. Please answer the following questions:\nQuestion: {question}\nA. {A} B. {B} C. {C} D. {D}\nPlease choose the best option and respond only with the choice of the correct answer (A, B, C, or D).\nAnswer:
Misalignment fewer	You are a very helpful AI assistant. Please answer the following questions:\nQuestion: {question}\nA. {A} B. {B} C. {C} D. {D}\nPlease choose the best option and respond only with the option of the answer (A, B, C, or D) below.\nAnswer: You are a very helpful AI assistant. Please answer the following questions:\nQuestion: {question}\nA. {A} B. {B} C. {C} D. {D}\nPlease choose the best option and respond only with the option of the answer (A, B, C, or D) carefully.\nAnswer: You are a very helpful AI assistant. Please answer the following questions:\nQuestion: {question}\nA. {A} B. {B} C. {C} D. {D}\nPlease choose the best option and respond only with the option of the answer (A, B, C, or D) now.\nAnswer:
Misalignment more	Please choose the best option and respond only with the option of the answer (A, B, C, or D) below.\nYou are a very helpful AI assistant. Please answer the following questions:\nQuestion: {question}\nA. {A} B. {B} C. {C} D. {D}\nAnswer: Please choose the best option and respond only with the option of the answer (A, B, C, or D) carefully.\nYou are a very helpful AI assistant. Please answer the following questions:\nQuestion: {question}\nA. {A} B. {B} C. {C} D. {D}\nAnswer: Please choose the best option and respond only with the option of the answer (A, B, C, or D) now.\nYou are a very helpful AI assistant. Please answer the following questions:\nQuestion: {question}\nA. {A} B. {B} C. {C} D. {D}\nAnswer:

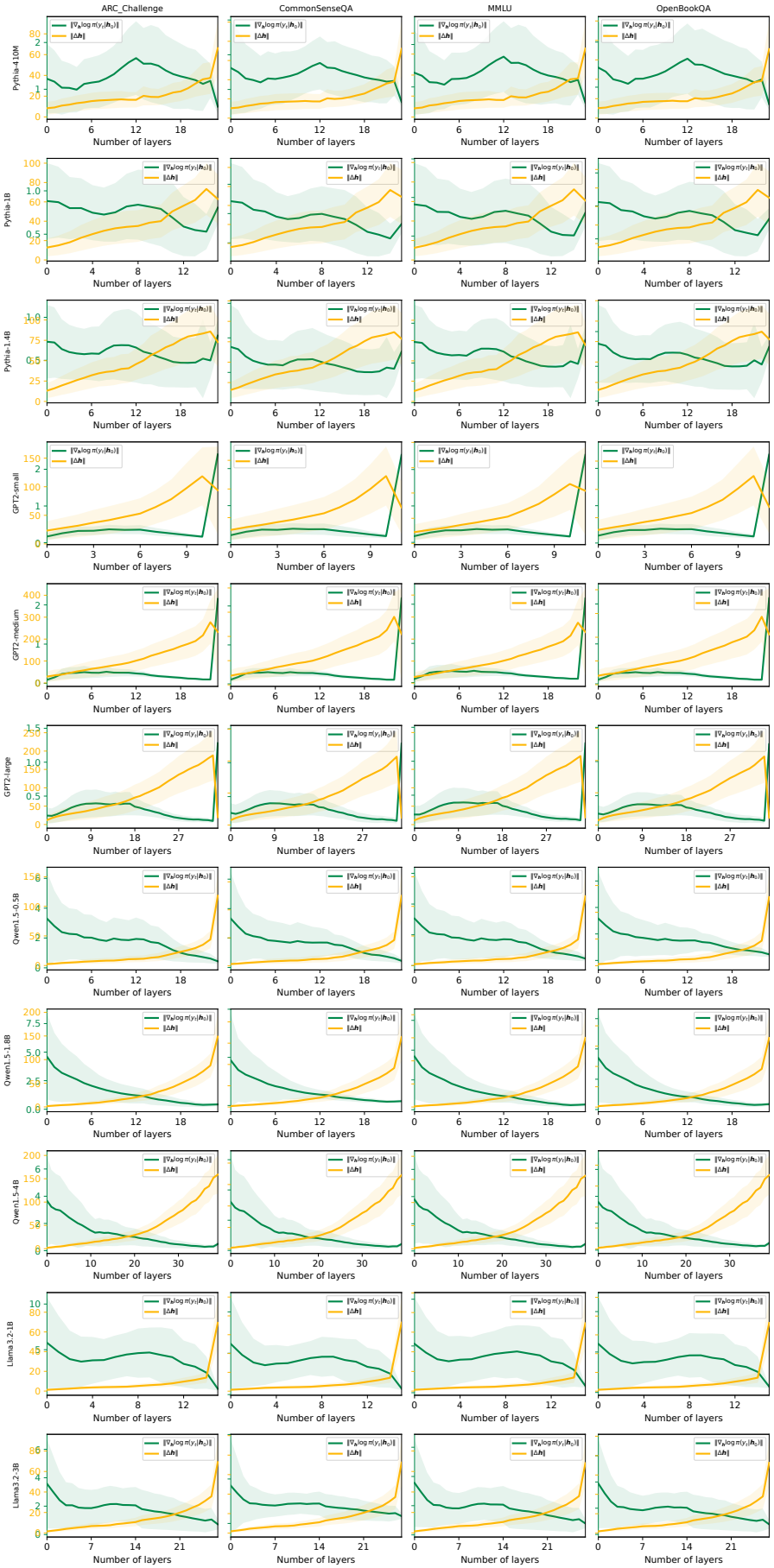


Figure 7: The trends of $\|\nabla_{\mathbf{h}} \log \pi(y_t|\mathbf{h}_0)\|$ and $\|\Delta\mathbf{h}\|$ across layers for all models on the four datasets.

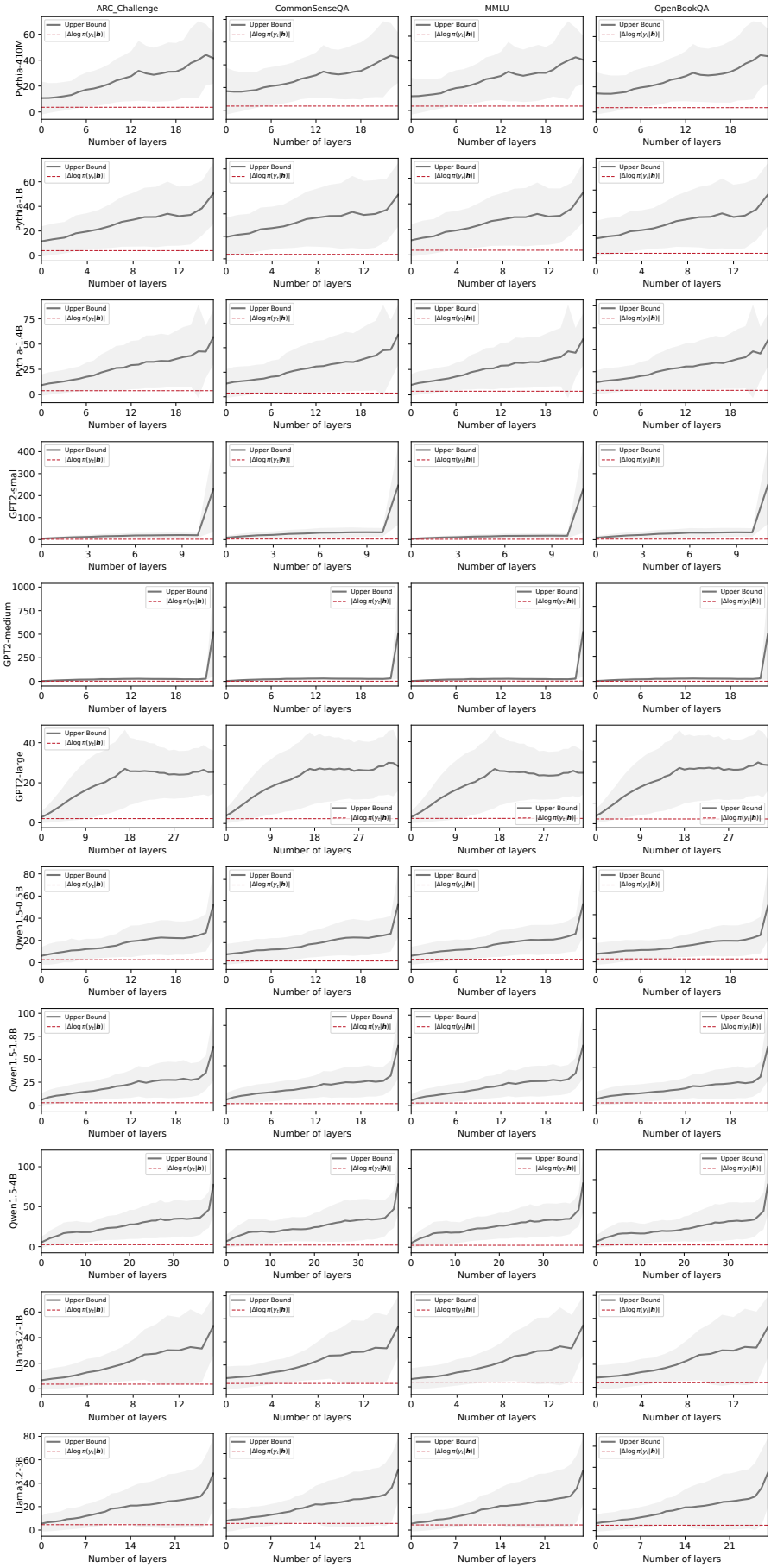


Figure 8: The trends of upper bounds across layers for all models on the four datasets.

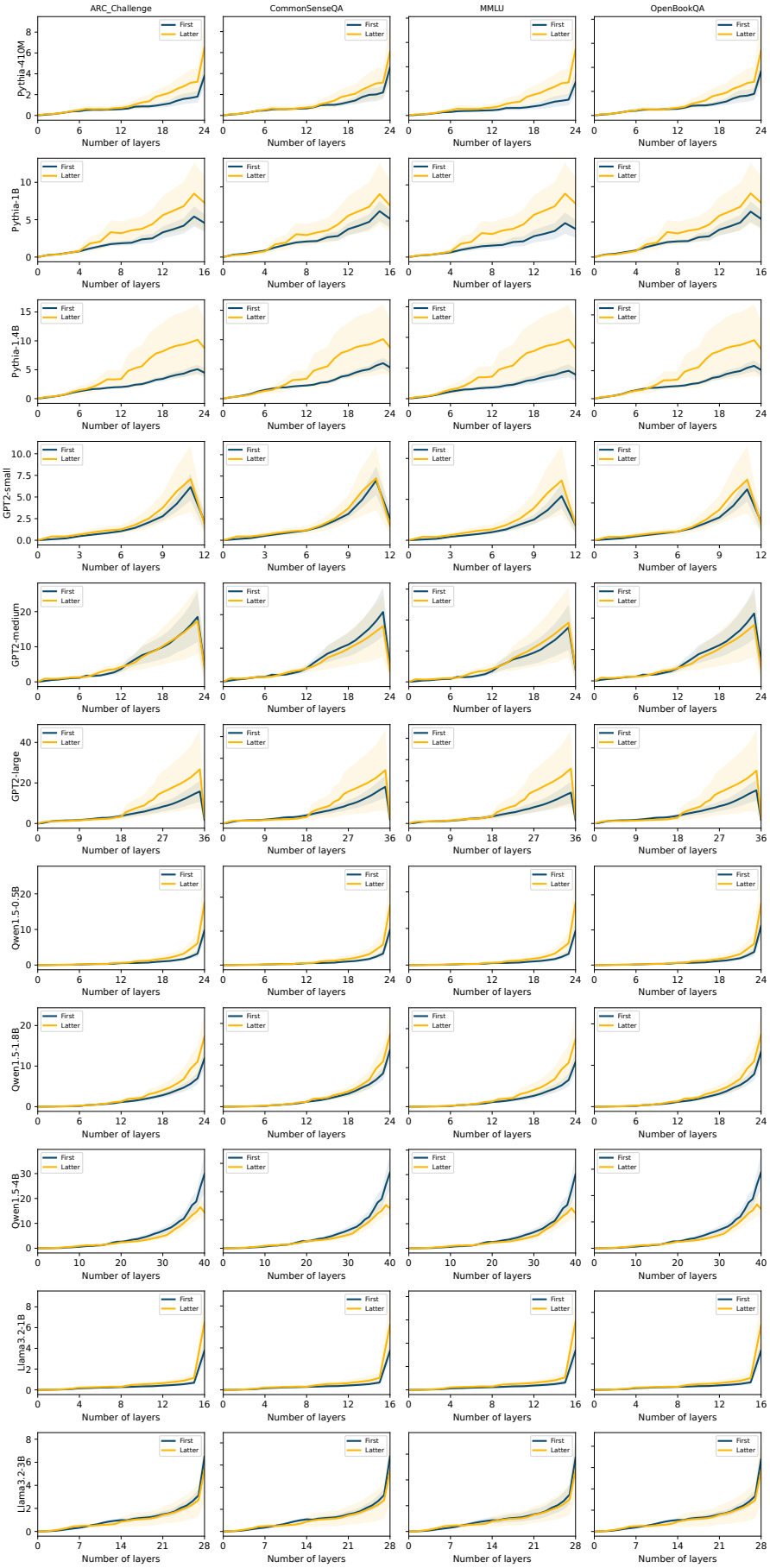


Figure 9: The comparison results between “First” and “Latter” across all models and datasets.

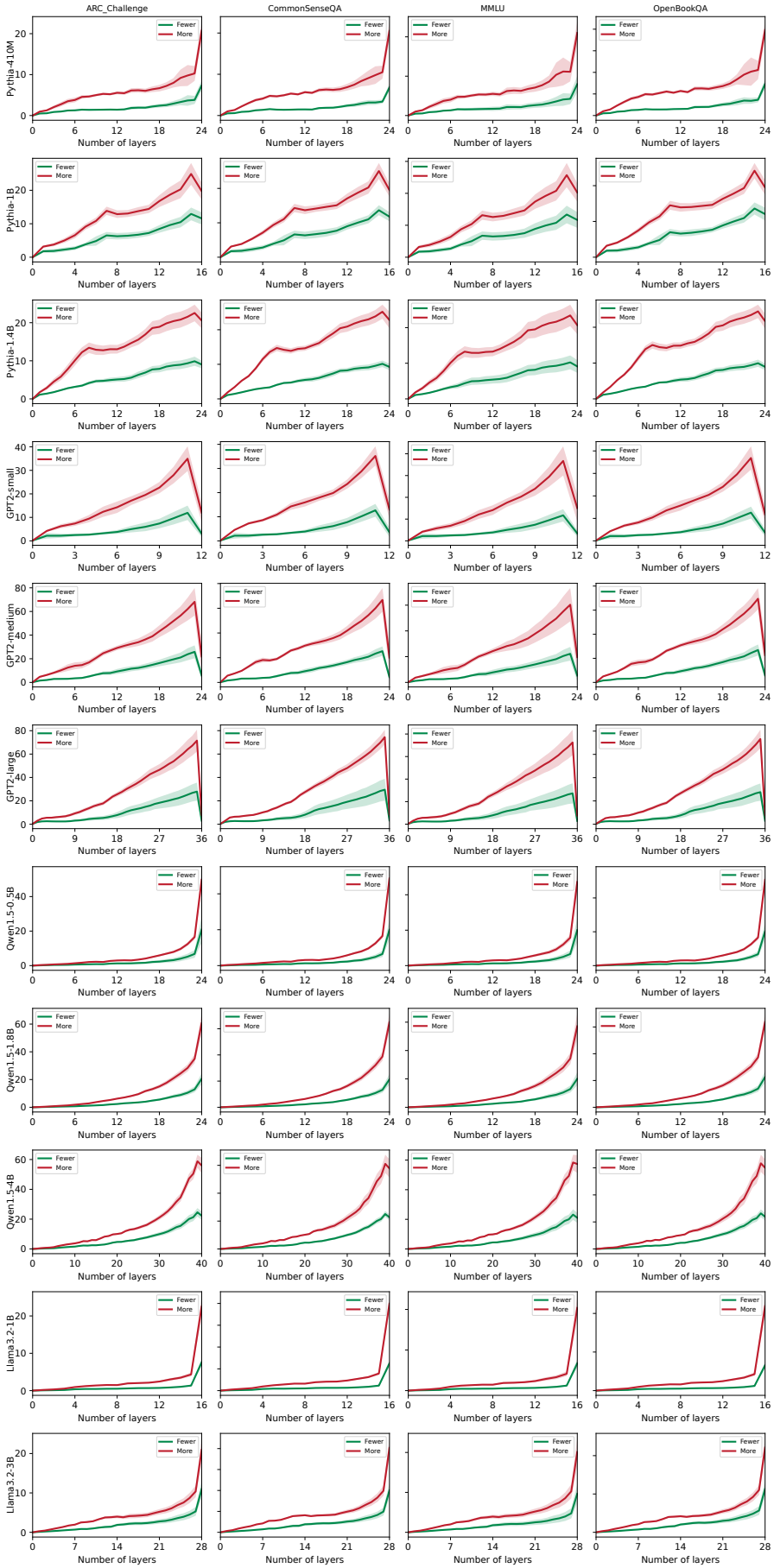


Figure 10: The comparison results between "Fewer" and "More" across all models and datasets.

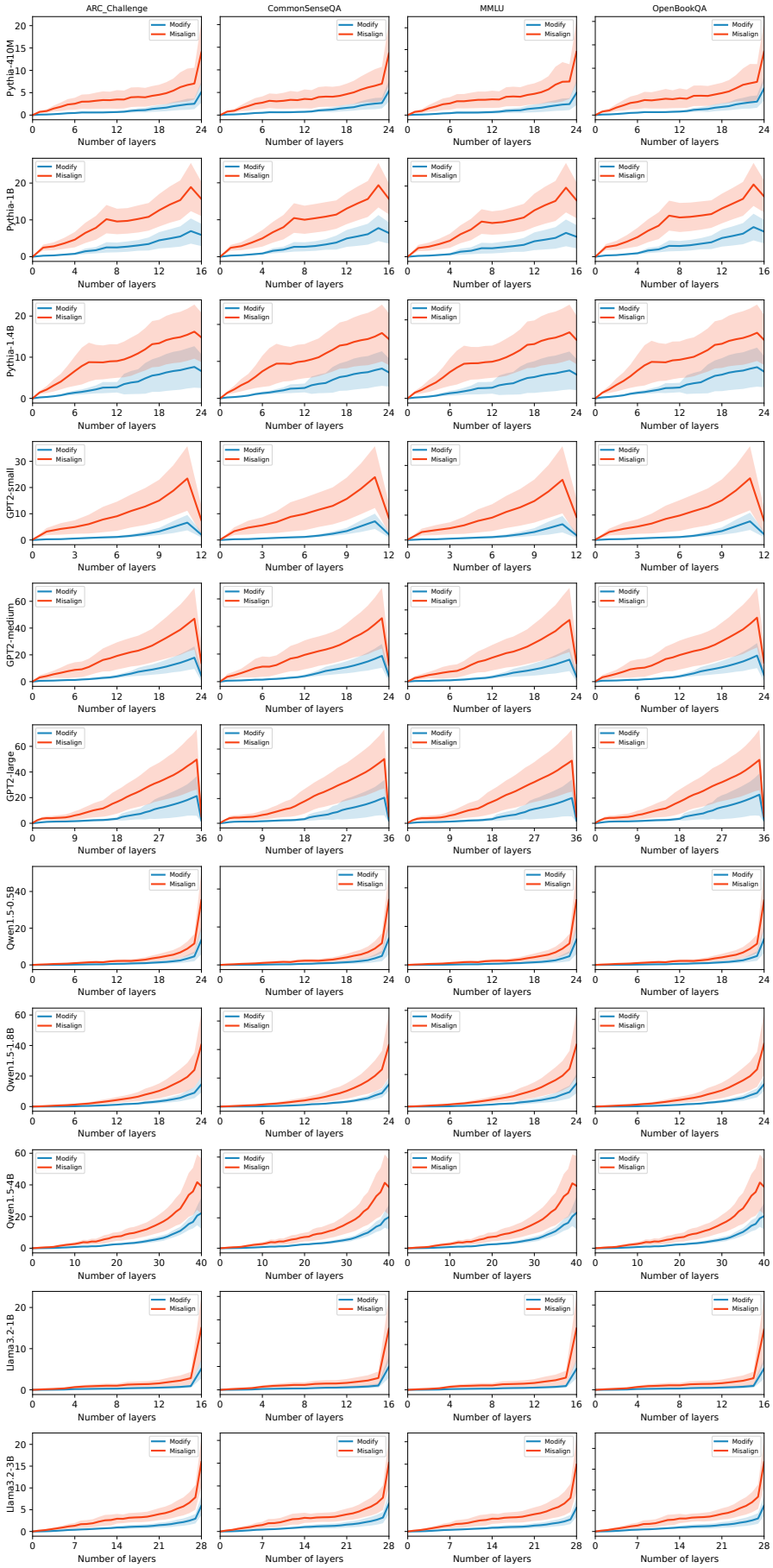


Figure 11: The comparison results between “Modify” and “Misalign” across all models and datasets.

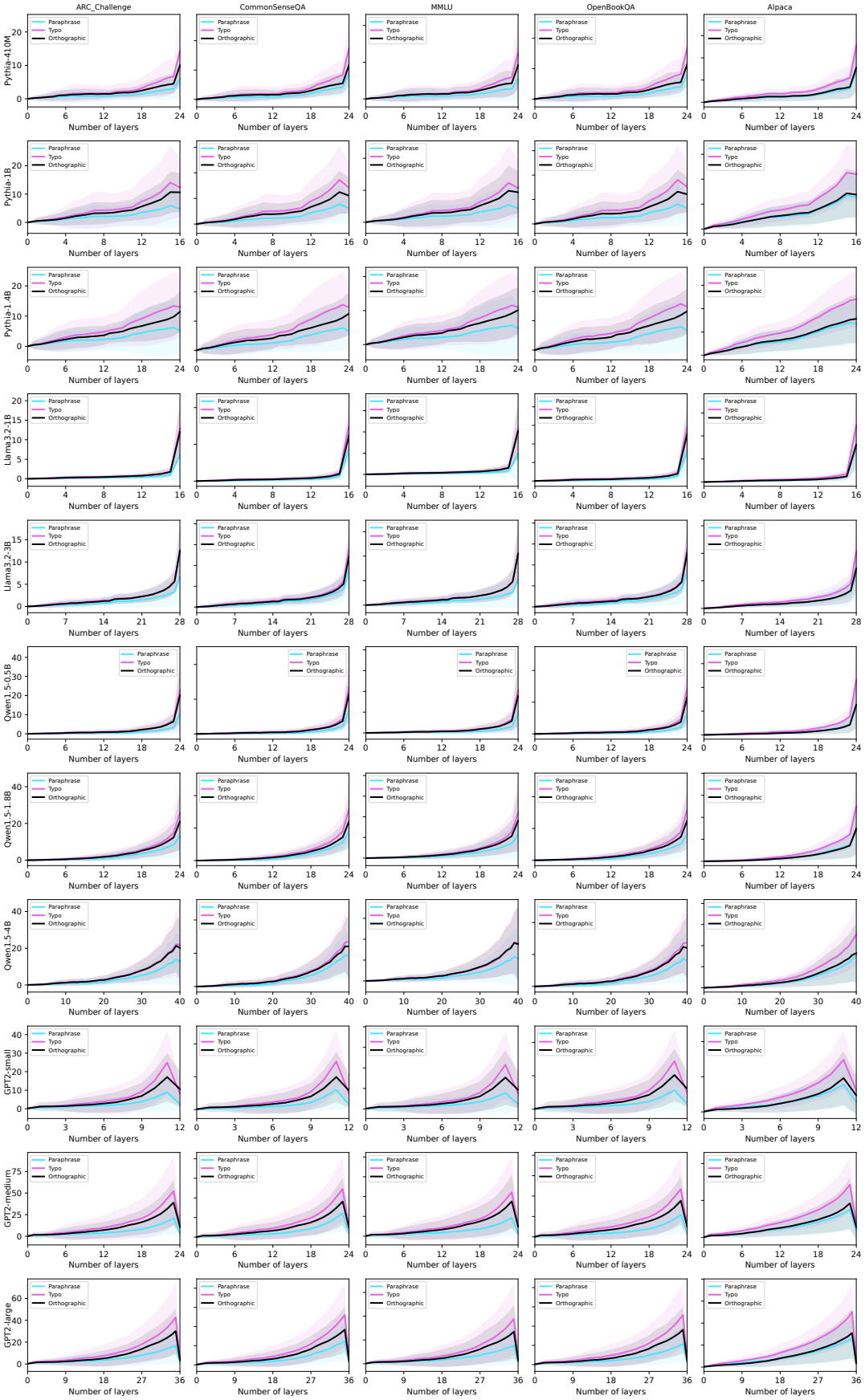


Figure 12: The comparison results between "Paraphrase," "Typo," and "Orthographic" across all models and datasets.

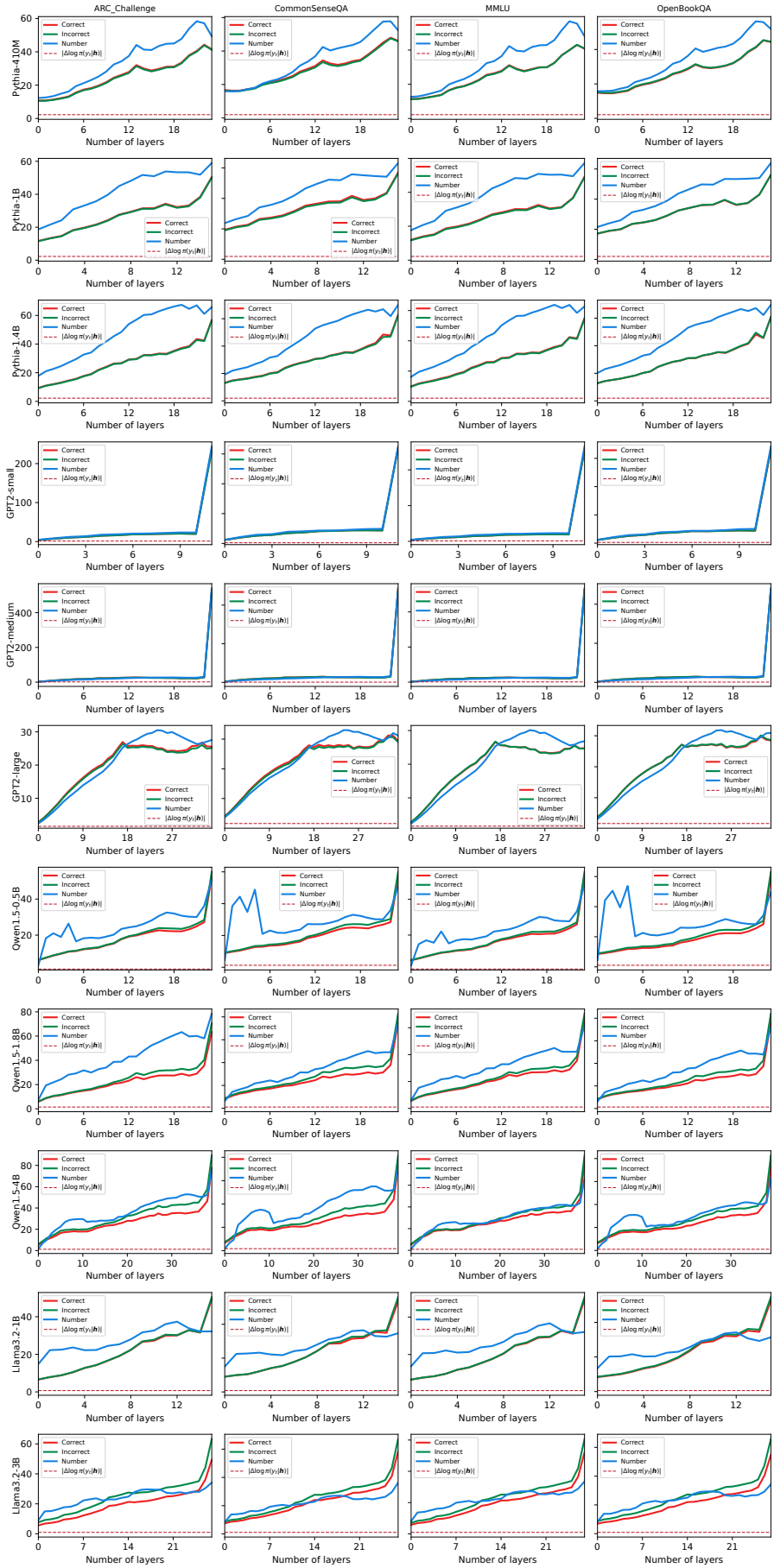


Figure 14: The comparison of the upper bound $\|\nabla_{\mathbf{h}} \log \pi(y_t|\mathbf{h}_0)\| \cdot \|\Delta\mathbf{h}\|$ of different y_t.

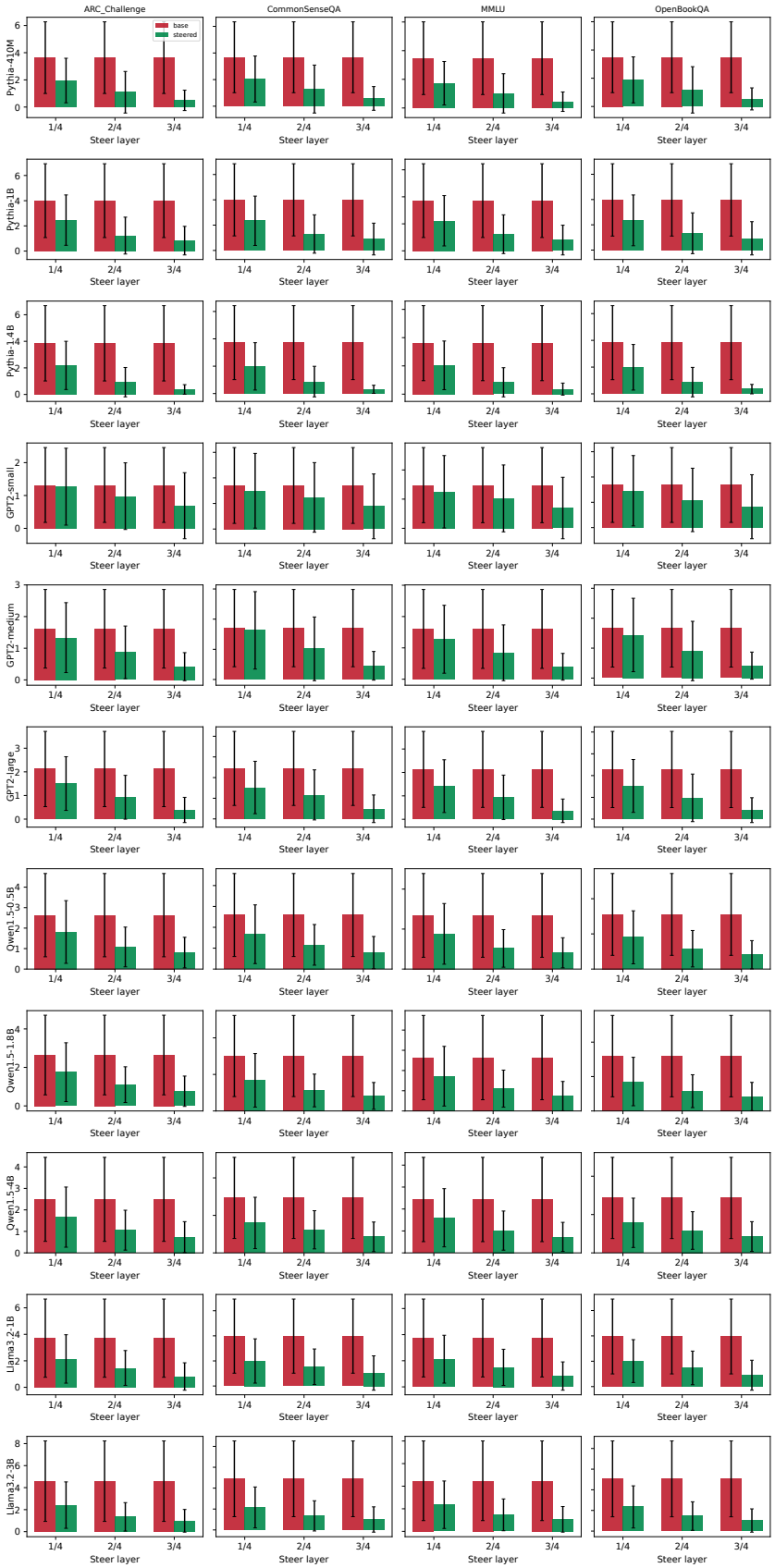


Figure 15: Activation steering across all 11 models on the four MCQ datasets. Each cell shows $|\Delta \log \pi(y_t \mid \mathbf{h})|$ before (red) and after (green) steering at $l \in \{L/4, L/2, 3L/4\}$.

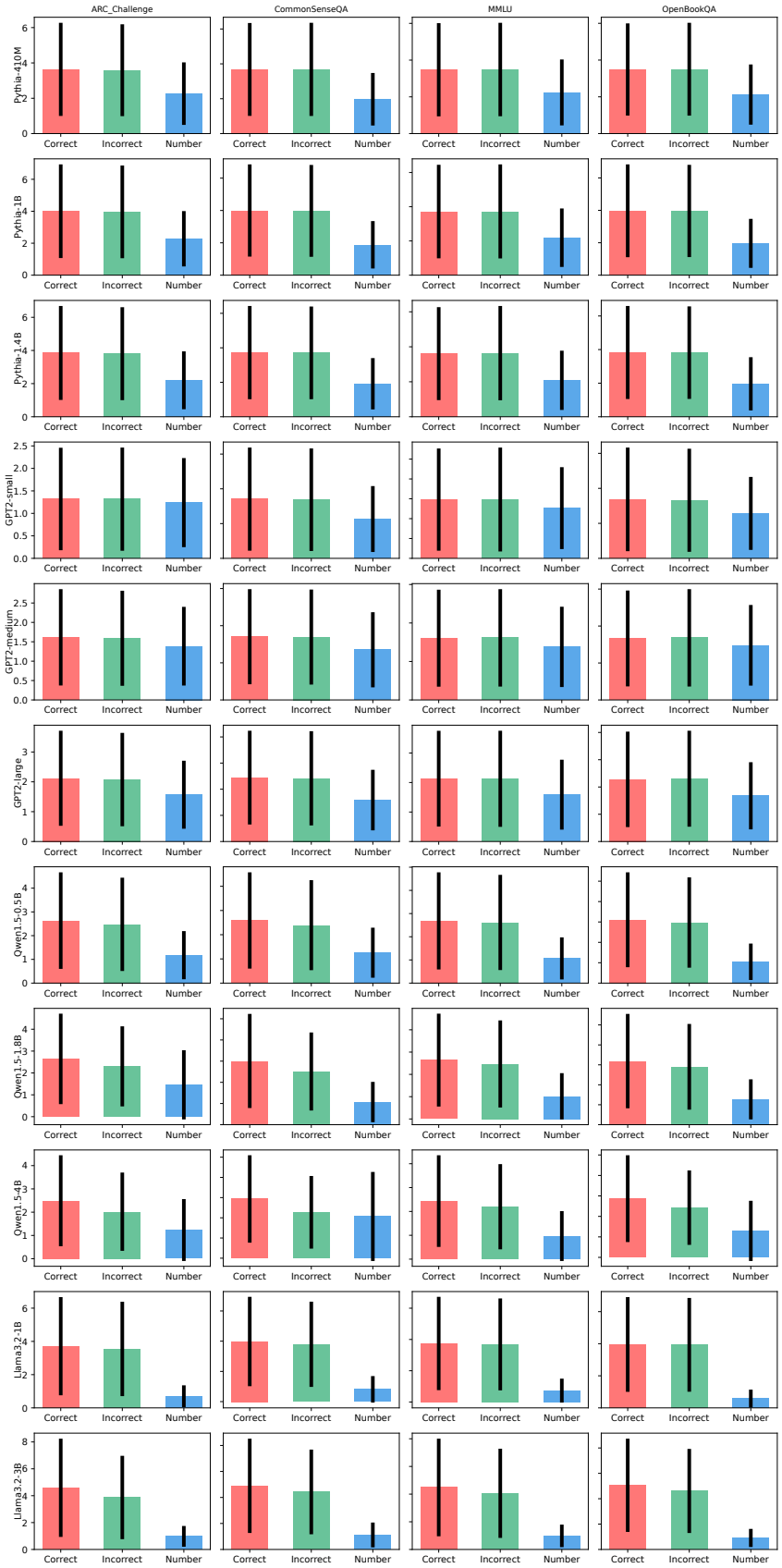


Figure 16: The comparison of the absolute difference of the log probabilities $|\Delta \log \pi(y_t \mid \mathbf{h})|$ of different y_t. "Correct", "Incorrect", and "Number" indicates the next token is y_c, y_i, and y_n, respectively.