Evaluating and Explaining Prompt Sensitivity of LLMs Using Interactions

Anonymous authors
Paper under double-blind review

Abstract

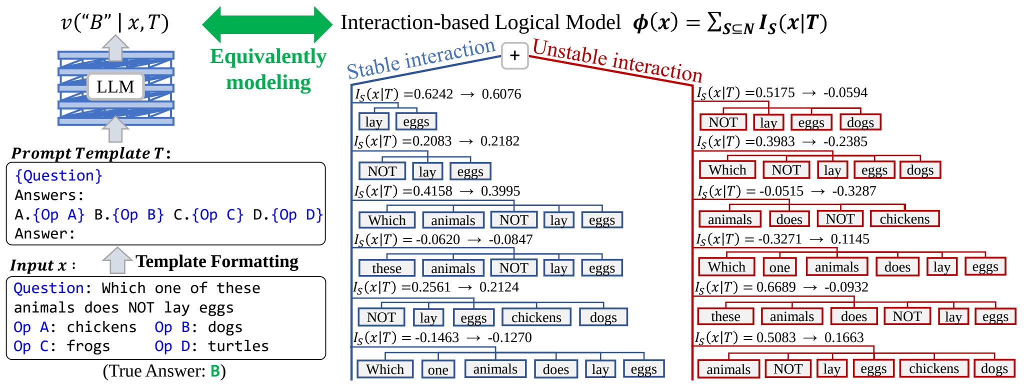
The remarkable capabilities of large language models (LLMs) are often undermined by their instability. That is, LLMs are sensitive to prompts, as even subtle and semantically irrelevant changes in prompts can cause dramatic fluctuations in performance, a phenomenon known as prompt sensitivity. Previous studies typically evaluate prompt sensitivity by comparing the LLM's final outputs when prompts change. However, such coarse-grained metrics fail to explain the internal reasons for prompt sensitivity. In this paper, we introduce the game-theoretic interaction framework as a fine-grained tool to analyze prompt sensitivity of LLMs. Specifically, we disentangle the output score of the LLM into a set of interactions. Each interaction represents a nonlinear relationship associated with a combination of input variables. We discover that subtle changes to prompts can trigger significant instability in interactions, even when the final outputs of the LLM remain the same. To this end, we propose an Interaction-based Prompt Sensitivity (IPS) metric by quantifying changes in interactions when we introduce subtle changes to prompts. We apply the proposed IPS metric to 50 open-source LLMs and uncover four factors that reduce the prompt sensitivity of LLMs, including supervised fine-tuning, increased model scales, dense architectures, and few-shot learning. More crucially, we discover a common mechanism by which these four factors reduce prompt sensitivity: all these four factors tend to reduce the prompt sensitivity of low-order interactions (*i.e.*, interactions involving few input variables).

1 Introduction

LLMs have demonstrated exceptional proficiency in numerous natural language processing tasks (Srivastava et al., 2022; Zhou et al., 2023b; Annepaka & Pakray, 2024; Chang et al., 2024), a success largely driven by the effectiveness of prompting. However, this power is undermined by prompt sensitivity. That is, semantically unimportant changes to the prompt can result in divergent outputs (Sclar et al., 2023). Current research (Sclar et al., 2023; Chatterjee et al., 2024; Lu et al., 2024; Alzahrani et al., 2024; Errica et al., 2025; Razavi et al., 2025) on evaluating prompt sensitivity only focuses on the LLM's final output. These metrics typically measure changes in performance, such as task accuracy or output consistency, across different prompt variations. As coarse-grained measures, these output-based metrics only reveal the consequences of prompt sensitivity (*e.g.*, the LLM's prediction changes from one answer to another one) but fail to explain its underlying reasons.

In this paper, we aim to evaluate the prompt sensitivity of LLMs from a fine-grained perspective and investigate the underlying reasons why certain factors can decrease the prompt sensitivity of LLMs. Recent research (Chen et al., 2024; Zhou et al., 2024; Ren et al., 2025a;b) has utilized interactions to explain the fine-grained inference logic of deep neural networks (DNNs). Inspired by these studies, we introduce the interactions framework to fine-grainedly analyze the prompt sensitivity of LLMs.

Specifically, given a sentence $\boldsymbol{x}$ with n input variables (*e.g.*, words or tokens) indexed by $N = \{1, 2, \ldots, n\}$, an interaction represents an intricate nonlinear relationship associated with a specific combination of input variables. Consider the sentence $\boldsymbol{x} =$ *"He is a green hand."* In this context, the idiom "green hand" carries the meaning of "beginner". The joint presence of the input variables in the set $S = \{\textit{green}, \textit{hand}\} \subseteq N$ triggers a special interaction effect. This interaction effect, denoted as I_S, pushes the network's inference towards the semantic meaning of *"beginner."* Li & Zhang (2023) have mathematically demonstrated that the scalar output $v(\boldsymbol{x})$ of a DNN is always equivalent to the



$v(\text{“}B\text{”} \mid x, T)$
LLM
Equivalently modeling
Interaction-based Logical Model $\boldsymbol{\phi}(\boldsymbol{x}) = \sum_{S\subseteq N} \boldsymbol{I}_S(\boldsymbol{x}|\boldsymbol{T})$
Stable interaction
+
Unstable interaction
Prompt Template $\boldsymbol{T}$:
Input $\boldsymbol{x}$:
Template Formatting
$I_S(x|T)$ =0.6242 → 0.6076
lay eggs
$I_S(x|T)$ =0.2083 → 0.2182
NOT lay eggs
$I_S(x|T)$ =0.4158 → 0.3995
Which animals NOT lay eggs
$I_S(x|T)$ = -0.0620 → -0.0847
these animals NOT lay eggs
$I_S(x|T)$ =0.2561 → 0.2124
NOT lay eggs chickens dogs
$I_S(x|T)$ = -0.1463 → -0.1270
Which one animals does lay eggs
$I_S(x|T)$ =0.5175 → -0.0594
NOT lay eggs dogs
$I_S(x|T)$ =0.3983 → -0.2385
Which NOT lay eggs dogs
$I_S(x|T)$ = -0.0515 → -0.3287
animals does NOT chickens
$I_S(x|T)$ = -0.3271 → 0.1145
Which one animals does lay eggs
$I_S(x|T)$ = 0.6689 → -0.0932
these animals does NOT lay eggs
$I_S(x|T)$ = 0.5083 → 0.1663
animals NOT lay eggs chickens dogs

```
{Question}
Answers:
A.{Op A} B.{Op B} C.{Op C} D.{Op D}
Answer:
```

```
Question: Which one of these
animals does NOT lay eggs
Op A: chickens  Op B: dogs
Op C: frogs     Op D: turtles
```

(True Answer: B)

Figure 1: Using interactions for fine-grained analysis of prompt sensitivity. Given an input $\boldsymbol{x}$ and a prompt template T, the LLM's output score $v(\text{“}B\text{”}|\boldsymbol{x}, T)$ is equivalent to the output of an interaction-based logical model $\phi(\boldsymbol{x})$, i.e., $v(\text{“}B\text{”}|\boldsymbol{x}, T) = \phi(\boldsymbol{x}) = \sum_{S\subseteq N} I_S$. In this way, we can uncover the underlying reasons contributing to the prompt sensitivity of LLMs by analyzing detailed interaction patterns. Specifically, we divide interactions into those that are stable to prompt changes (*i.e.* stable interaction) and those that are unstable to prompt changes (*i.e.* unstable interaction).

output of an interaction-based logical model $\phi(\boldsymbol{x}) = \sum_{S\subseteq N} I_S$. That is, $v(\boldsymbol{x}) = \phi(\boldsymbol{x}) = \sum_{S\subseteq N} I_S$. Thus, the inference logic of a DNN can be explained by a set of interactions between input variables.

Coarse-grained analysis vs. fine-grained analysis. Traditional analysis of prompt sensitivity can only coarsely reflect whether the LLM's prediction alters when the prompt changes. Beyond traditional analysis, we introduce an interaction-based logical model $\phi(\boldsymbol{x})$ as a fine-grained analytical tool to analyze the stable and unstable interactions encoded by the LLM for each specific sample. As shown in Figure 1, when we keep the same input $\boldsymbol{x}$ but introduce semantically equivalent alterations to the prompt template T (*e.g.*, change “Answers” to “ANSWERS”), traditional coarse-grained analysis can only reflect that the LLM's prediction changes from the correct answer “B” to the incorrect answer “A” but fails to explain why or how this change occurs. In contrast, our interaction-based analysis precisely identifies unstable interactions that may contribute to the prompt sensitivity of the LLM. An interaction is considered stable if its effect changes minimally (e.g., $I_{\{lay,\ eggs\}}$ shifts from 0.6242 to 0.6076). Conversely, an interaction is deemed unstable if its effect fluctuates dramatically (e.g., $I_{\{NOT,\ lay,\ eggs,\ dogs\}}$ shifts from 0.5175 to -0.0594). Strikingly, we find that unstable interactions exist even when the LLM's final output remains the same. This indicates that our fine-grained analysis reveals potential instability that is entirely invisible to traditional, output-level metrics.

Building on these findings, we leverage the interaction framework to propose a fine-grained metric to evaluate the prompt sensitivity of LLMs, which is termed **I**nteraction-based **P**rompt **S**ensitivity (**IPS**). This metric quantifies changes in interaction patterns when LLMs process different prompts. We apply the proposed IPS metric to evaluate the prompt sensitivity of 50 open-source LLMs. However, drawing conclusions about which LLM families are more or less sensitive from this ranking is challenging, as an LLM's prompt sensitivity stems from multiple, intertwined factors.

To this end, we conduct a series of comparative experiments to disentangle these factors, discovering four factors that reduce the prompt sensitivity of LLMs: **(1)** Supervised fine-tuning reduces the prompt sensitivity. Instruct/chat models (with supervised fine-tuning) exhibit lower prompt sensitivity than base models. **(2)** LLMs with larger parameter numbers exhibit lower prompt sensitivity. **(3)** Dense models are generally less sensitive than mixture-of-experts (MoE) models. MoE architectures scale up model capacity with minimal computational overhead, but they may result in weaker stability. **(4)** Few-shot learning considerably reduces prompt sensitivity compared to 0-shot learning.

More crucially, we have explored and uncovered a common underlying mechanism that explains how the four aforementioned factors reduce prompt sensitivity: they primarily reduce the instability of low-order interactions (i.e., interactions between a small number of input variables) in LLMs. This finding is counterintuitive, as our experiments demonstrate that high-order interactions (i.e., interactions involving many input variables) tend to exhibit the highest sensitivity, while low-order interactions are inherently less sensitive. Unexpectedly, these factors further stabilize the already stable low-order interactions, yet remain ineffective in addressing the more pronounced sensitivity of high-order interactions. This finding offers a novel perspective on the prompt sensitivity of LLMs.

2 Related Work

Prompt sensitivity of LLMs. Previous studies (Sun et al., 2023; Zhu et al., 2024a; Sclar et al., 2023; Alzahrani et al., 2024) demonstrated that LLMs are highly sensitive to minor perturbations or semantically unimportant alterations to the prompt, which can lead to significant performance variation. Such prompt sensitivity presents a considerable risk to the reliability and credibility of LLMs in practical applications. Existing metrics (Sclar et al., 2023; Chatterjee et al., 2024; Lu et al., 2024; Zhuo et al., 2024; Errica et al., 2025; Razavi et al., 2025) for evaluating the prompt sensitivity of LLMs typically measure shifts in final outputs, such as task accuracy or output consistency. However, these metrics are coarse-grained and fail to probe the LLM's internal logic. In this paper, we propose a fine-grained metric that evaluates prompt sensitivity based on interactions. This framework enables us to uncover the underlying mechanisms of prompt sensitivity in LLMs.

Using game-theoretic interactions to explain DNNs. Traditional methods for explanations (Tenney et al., 2020; Zhang et al., 2020) often lack mathematical guarantees of faithfulness, meaning their outputs may not accurately represent the internal logic employed by the DNN. To this end, Ren et al. (2023a) proposed to use interactions between input variables to explain DNNs and provided a series of theoretical guarantees for the method's validity. Furthermore, it has been empirically discovered (Li & Zhang, 2023) and theoretically proven (Ren et al., 2024) that a DNN typically encodes only a sparse set of interactions. At the application level, the interaction framework has proven effective in a wide range of complex tasks, including adversarial transferability (Deng et al., 2024), model generalization (Chen et al., 2024; Zhou et al., 2024), model training process (Ren et al., 2025a;b) and overfitting (Zhou et al., 2023a; Ren et al., 2023b). In this paper, we use the interaction framework to analyze the prompt sensitivity of LLMs.

3 Interaction-based Analysis of Prompt Sensitivity

3.1 Preliminaries: Interactions

This section introduces the definition of interactions, as well as the mathematical guarantees of interaction-based explanation. Given a well-trained DNN v and an input sentence $\boldsymbol{x}$ with n input variables (*e.g.*, words) indexed by $N = \{1, 2, \ldots, n\}$, let $v(\boldsymbol{x}) \in \mathbb{R}$ denote the scalar output of the DNN. Here we set $v(\boldsymbol{x}) = \log \frac{p(y=y^*|\boldsymbol{x})}{1-p(y=y^*|\boldsymbol{x})} \in \mathbb{R}$, where $p(y = y^*|\boldsymbol{x})$ represents the probability of generating the ground truth token y^* given the input $\boldsymbol{x}$. We define a surrogate logical model $\phi(\boldsymbol{x})$ to match the scalar output $v(\boldsymbol{x})$ of the DNN. Recent studies (Li & Zhang, 2023; Ren et al., 2023a) have proved Theorem 1, which shows that the output score $v(\boldsymbol{x})$ on any randomly masked[1] input $\boldsymbol{x}_T$ can be accurately calculated by the following surrogate logical model $\phi(\cdot)$.

$$\phi(\boldsymbol{x}_T) \triangleq \phi(\boldsymbol{x}_\emptyset) + \sum\nolimits_{S \subseteq N} \mathbb{1}(S \mid \boldsymbol{x}_T) \cdot I_S, \tag{1}$$

where the AND trigger function $\mathbb{1}(S \mid \boldsymbol{x}_T) \in \{0, 1\}$ represents an **AND relationship** between input variables in S, which can also be termed **AND interaction pattern**. The scalar weight I_S quantifies the effect of an AND relationship, which can also be termed **interaction effect**. An AND relationship is activated only by the joint presence of all input variables in the set S, *i.e.*, all input variables in S are not masked. For instance, given the input sentence $\boldsymbol{x} =$ *"He is a green hand,"* the co-occurrence of the input variables in the set $S = \{green, hand\}$ contributes a numerical effect I_S that pushes the surrogate logical model's inference towards the semantic meaning of *"beginner."* If an AND interaction S is triggered, *i.e.*, $\mathbb{1}(S \mid \boldsymbol{x}_T) = 1$, the corresponding interaction effect I_S is added to the output of the logical model. Otherwise, if any word in S is masked and the AND interaction is not triggered, *i.e.*, $\mathbb{1}(S \mid \boldsymbol{x}_T) = 0$, the interaction effect I_S is not added to the output of the logical model. $\boldsymbol{x}_\emptyset$ represents that all input variables in N are masked.

Theorem 1 (Universal matching property, proven in Appendix B)**.** *Given an input $\boldsymbol{x}$ with n input variables, we randomly mask any combinations of input variables to generate 2^n masked inputs $\{\boldsymbol{x}_T \mid T \subseteq N\}$. For every masked input $\boldsymbol{x}_T$, when the scalar weight I_S in the logical model $\phi(\cdot)$ are set to $I_S = \sum_{S' \subseteq S} (-1)^{|S|-|S'|} \cdot v(\boldsymbol{x}_{S'}), \phi(\boldsymbol{x}_\emptyset) = v(\boldsymbol{x}_\emptyset)$, the output of the logical model $\phi(\cdot)$ can always match the DNN's output score $v(\cdot)$.*

$$\forall\, T \subseteq N, \;\; v(\boldsymbol{x}_T) = \phi(\boldsymbol{x}_T) \tag{2}$$

[1] It is common to use a specific token or embedding to mask input variables of a DNN, *e.g.*, replacing the target token with a specific [MASK] token. See Appendix A for an introduction to masking strategies.

In addition to Theorem 1, **the sparsity property also provides a theoretical guarantee for the faithfulness of interaction-based explanation**. The sparsity property shows that a DNN only encodes a sparse set of interactions with salient effects. That is, only a small subset of all 2^n AND interactions in Theorem 1, termed salient interactions, have a significant impact on the logical model's output. In contrast, the vast majority of interactions have negligible effects and are considered noise patterns, *i.e.*, $I_S \approx 0$. The sparsity property of AND interactions has been proven by Ren et al. (2024). We follow Li & Zhang (2023) to extract AND-OR interactions[2] by learning to decompose the LLM's output. The extraction process is framed as an optimization problem where a LASSO-like loss is minimized to pursue higher sparsity of interactions (please see Appendix D for details of extracting AND-OR interactions). Such a technique has proven effective by both theoretical proofs (Li & Zhang, 2023) and extensive empirical validation (Ren et al., 2023a; Zhou et al., 2024).

3.2 Verifying the Faithfulness of Considering Interactions as Inference Patterns used by LLMs

Before utilizing the interaction framework to analyze the prompt sensitivity of LLMs, we need to **theoretically prove and experimentally validate** the faithfulness of using interactions to explain LLMs. **In theory**, Theorem 1 guarantees that the surrogate logical model's output $\phi(\cdot)$ can always match the LLM's output $v(\cdot)$ for all 2^n masked samples. Since the logical model's output $\phi(\cdot)$ is composed entirely of the sum of all interaction effects as defined in Eq. (1), **we can consider interactions as the detailed inference patterns that constitute the LLM's internal logic**. Therefore, we can evaluate the prompt sensitivity of LLMs by measuring the instability of interaction patterns.

In practice, we conduct experiments to verify whether LLMs encode sparse interactions. Consider the multiple choice question (MCQ) in Figure 1 as an example. Let Q denote the set of all words in the question, *e.g.*, Q = {*Which, one, of, these, animals, does, NOT, lay, eggs*}, M denote the set of all words in the options, *e.g.*, M = {*chickens, dogs, frogs, turtles*}, and T denote the set of all words in the prompt template, *e.g.*, T = {*Answers:, A., B., C., D., Answer:*}. Specifically, we take words[3] in $Q \cup M$ as input variables and compute the interaction effect I_S of all interactions $S \subseteq Q \cup M$, as shown in Theorem 1. Meanwhile, we treat words in prompt template T as background context. We follow Li & Zhang (2023) to extract AND and OR interactions. Thus, given an input $\boldsymbol{x} = Q \cup M$ with n words, we can obtain a set of 2^{n+1} interactions, including 2^n AND interactions and 2^n OR interactions[2]. For each interaction effect I_S, we apply min-max normalization to it. Specifically, $\widetilde{I}_S \triangleq sgn(I_S) \cdot \frac{|I_S| - Min}{Max - Min}$, where *Min* and *Max* are the minimum and maximum absolute values of all 2^{n+1} interaction effects; $sgn(I_S) = \frac{I_S}{|I_S|}$ represents the sign of I_S, thus preserving the positive or negative effect of the interaction. Figure 2 (a) shows the distribution of all absolute values of normalized interaction effects $|\widetilde{I}_S|$. Results[4] verify that only a small set of interactions have salient effects, while most of the interactions have negligible effects and can be considered as noise patterns. Figure 2 (b) compares the LLM's true output $v(\boldsymbol{x}_T)$ for all 2^n masked inputs against the logical model using only the most salient interactions. Even when using just the top 3% or top 5% of all interactions, the matching error is minimal. This empirically demonstrates that the LLM's output can be faithfully approximated by a small, sparse set of salient interactions[4].

3.3 Using interactions as a fine-grained tool to analyze prompt sensitivity

The verification of the faithfulness of considering salient interactions as detailed inference patterns encoded by LLMs enables us to employ them as a fine-grained analytical tool. Specifically, for an LLM, we visualize the changes in salient interactions encoded by the LLM for the same input under a pair of prompt templates. As Figure 3 shows, when the LLM's outputs are different, nearly half of the salient interactions reverse their sign (from positive to negative, or vice versa), another 30%-40% change significantly in magnitude, and only a small fraction (10%-20%) remain stable. Strikingly, even when the LLM's output remains the same, the majority (60%-80%) of the salient interactions

[2]The OR interaction is proved to be a specific AND interaction. Please see Appendix C for proof.

[3]We take words instead of tokens as input variables because different LLMs may divide the same word into different tokens. For example, Llama-2-7B tokenizes the word "Elements" into two tokens "Element" and "s", while Qwen3-8B treats it as one token.

[4]Results on more LLMs in Appendix F exhibit the same conclusion.

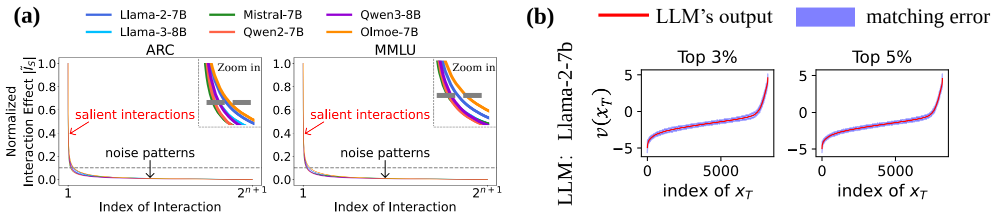


Figure 2: (a) Verifying the sparsity of interactions. We show absolute values of normalized interactions in a descending order. LLMs all encode a small number of salient interactions, while most of the interaction effects are negligible. (b) Verifying the quality of universal matching for any 2^n masked inputs. The red line plots outputs of the LLM in an ascending order.

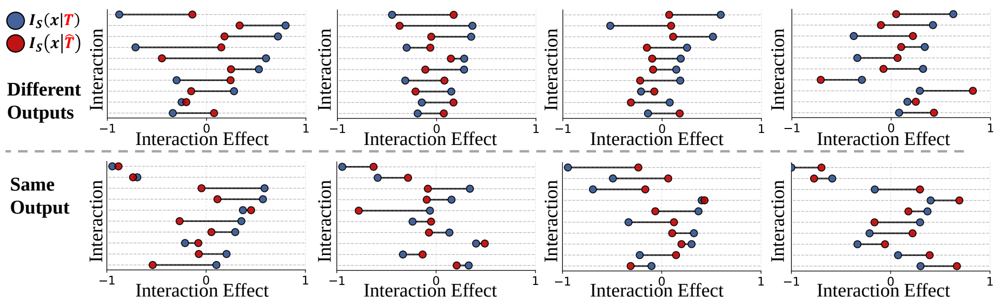


Figure 3: Case studies of interaction-level analysis revealing latent instability.The plots compare the interaction effects encoded by the LLM for the same input $\boldsymbol{x}$ under two semantically equivalent prompt templates, T (blue) and $\hat{T}$ (red). The bottom row shows examples where the LLM's output remains the same; the top row shows examples where it changes. Results show that interaction effects are highly unstable, changing in either sign or magnitude, even when the output of the LLM remains the same. This highlights a critical risk of prompt sensitivity that is invisible to output level.

are still unstable. This offers preliminary evidence that semantically irrelevant alterations to the prompt template can lead to significant changes in the salient interaction patterns.

To systematically analyze this instability beyond individual cases, we classify interactions into three distinct types: 1) **Opposite sign**: The interaction's effect reverses its sign (e.g., from positive to negative). Since we only consider salient interactions, this sign change represents the most severe form of instability, as it can completely reverse the LLM's internal logic. 2) **Same sign & different effect**: The interaction maintains its positive or negative influence, but its magnitude changes substantially (i.e., its effect is more than doubled or less than halved). 3) **Same sign & similar effect**: The interaction's sign and magnitude both remain stable, which represents robust, stable interactions.

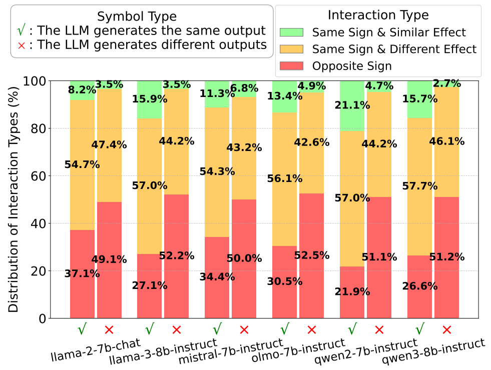


Figure 4: The distribution of salient interactions types of various LLMs.

Figure 4 plots the distribution of three interaction types over all samples, conditioned on whether the LLM's final output changes or remains the same across a pair of prompts. When the LLM generates different outputs, the interaction patterns encoded by the LLM are highly unstable. *Opposite sign* interactions account for approximately 50%, meaning nearly half of the LLMs' salient interactions reverse the sign of their effect. The truly stable *Same sign & similar effect* interactions account for a mere 2.7%-6.8%, while the remaining 42.6%-47.4% of interactions, though maintaining their sign, change significantly in magnitude. More alarmingly, even when the final output of the LLM remains the same, the instability of interactions still exists. While the situation improves—with *Opposite sign* interactions dropping to 21.9%-37.1% and stable *Same sign & similar effect* interactions rising to 8.2%-21.1%—the vast majority of interactions still fall into the two unstable categories.

The above results demonstrate that output-level analysis fails to capture the unreliable internal interaction patterns of LLMs. In contrast, our interaction-based analysis provides a fine-grained per-

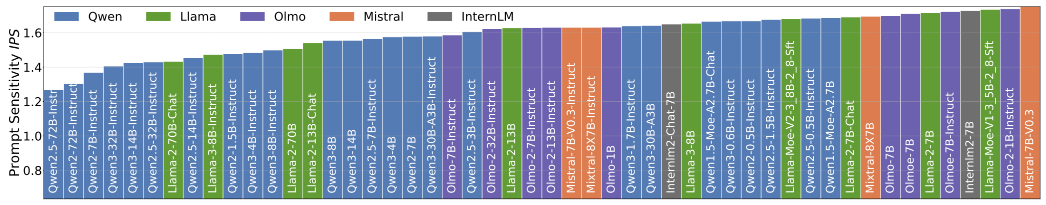


Figure 5: Prompt sensitivity of all the LLMs in an ascending order.

spective to discover the latent instability of LLMs, offering a new analytical tool for quantifying the ratio of stable and unstable interactions on a per-sample basis for any LLM.

4 Evaluating and analyzing the prompt sensitivity of LLMs

4.1 Defining and evaluating interaction-based prompt sensitivity

We propose an interaction-based metric to measure the prompt sensitivity of LLMs. Given an MCQ dataset $\mathcal{D}$, for any input $\boldsymbol{x} \in \mathcal{D}$, it is composed of the question Q and the options M, *i.e.*, $\boldsymbol{x} = Q \cup M$. Given a prompt template T, we make minor changes to T and obtain a new prompt template $\hat{T}$, *e.g.*, $T = \{\textit{Answers:}, A., B., C., D., \textit{Answer:}\}$ and $\hat{T} = \{\textit{ANSWERS:}, A., B., C., D., \textit{ANSWER:}\}$. By applying the pair of prompt templates T and $\hat{T}$ to the same input $Q \cup M$, we can construct two prompts. The LLM is supposed to extract similar salient interactions from $Q \cup M$ when processing these two prompts because T and $\hat{T}$ have the same semantic meaning. Thus, let $\Omega_{\text{salient}}(\boldsymbol{x}|T) = \{S \in \Omega(\boldsymbol{x}) \mid |\widetilde{I}_S(\boldsymbol{x}|T)| > \tau\}$ represent the set of **salient interactions** extracted from $\boldsymbol{x}$ given the prompt template T. Here τ is a threshold used to distinguish salient interactions from noise patterns. We set τ to 0.1 for all experiments in this paper. Similarly, let $\Omega_{\text{salient}}(\boldsymbol{x}|\hat{T}) = \{S \in \Omega(\boldsymbol{x}) \mid |\widetilde{I}_S(\boldsymbol{x}|\hat{T})| > \tau\}$ represent the set of **salient interactions** extracted from $\boldsymbol{x}$ given the prompt template $\hat{T}$. Therefore, on the dataset $\mathcal{D}$, we define the LLM's **Interaction-based Prompt Sensitivity** as ***IPS***.

$$\textit{IPS} \triangleq \mathbb{E}_{\boldsymbol{x}} \left[\mathbb{E}_{T,\hat{T}} \left[\frac{1}{|\Omega_{\text{union}}|} \sum_{S \in \Omega_{\text{union}}} \frac{|\widetilde{\mathcal{I}}_S(\boldsymbol{x}|T) - \widetilde{\mathcal{I}}_S(\boldsymbol{x}|\hat{T})|}{\frac{|\widetilde{I}_S(\boldsymbol{x}|T)| + |\widetilde{I}_S(\boldsymbol{x}|\hat{T})|}{2}} \right] \right], \tag{3}$$

where $\Omega_{\text{union}} = \Omega_{\text{salient}}(\boldsymbol{x}|T) \cup \Omega_{\text{salient}}(\boldsymbol{x}|\hat{T})$ is a unified set by taking the union of the two salient sets $\Omega_{\text{salient}}(\boldsymbol{x}|T)$ and $\Omega_{\text{salient}}(\boldsymbol{x}|\hat{T})$; the outer expectation, $\mathbb{E}_{\boldsymbol{x}}$, represents an averaging over all inputs $\boldsymbol{x} \in \mathcal{D}$, and the inner expectation, $\mathbb{E}_{T,\hat{T}}$, represents an averaging over all pairs of prompt templates $(T, \hat{T})$. This metric evaluates the prompt sensitivity of LLMs by calculating the symmetric mean absolute percentage error of salient interactions over all samples in the dataset $\mathcal{D}$.

Models and Datasets. We conduct experiments on 50 open-source LLMs from 6 model families. This diverse set includes 10 LLMs from the Llama family: Llama-2, Llama-3, Llama-MoE (Touvron et al., 2023; Grattafiori et al., 2024; Zhu et al., 2024b; Qu et al., 2024); 4 LLMs from the Mistral family: Mistral, Mixtral (Jiang et al., 2023; Jiang et al., 2024); 25 LLMs from the Qwen family: Qwen2, Qwen2.5, Qwen3, Qwen1.5-MoE, Qwen3-30B-A3B (Team, 2024; Yang et al., 2024; 2025); 9 LLMs from the OLMo family: OLMo, OLMo-2, OLMoE (Groeneveld et al., 2024; OLMo et al., 2024; Muennighoff et al., 2024); 2 LLMs from the InternLM family: InternLM2 (Cai et al., 2024). We evaluate all LLMs on two widely used multiple choice question (MCQ) benchmarks: ARC (Clark et al., 2018) and MMLU (Hendrycks et al., 2020). For each input from these datasets, we apply five distinct prompt templates to it. These templates maintain the same core content and differ only in minor formatting details, such as letter case and separators. For masking words in the input sentences, we follow the approach of Cheng et al. (2025) and utilize certain [MASK] token for each LLM. A comprehensive list of all LLMs, along with the specific prompt templates, mask tokens used in experiments and why we choose MCQ datasets, is provided in Appendix E.

We apply the IPS metric to evaluate the prompt sensitivity of 50 open-source LLMs. As shown in Figure 5, we observe a wide variance in IPS, ranging from a low of 1.338 for Qwen2-72B-Instruct to a high of 1.752 for Mistral-7B-v0.3. However, no series or families of LLMs have achieved a complete victory. The distribution indicates that an LLM's prompt sensitivity is influenced by multiple underlying factors. Identifying these influencing factors is of great significance for the robustness research of LLMs, which will be analyzed in the following section.

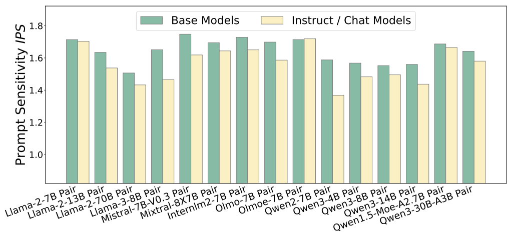


Figure 6: A comparison of the prompt sensitivity between instruct/chat models and base models. Results show that instruct/chat models are less sensitive than corresponding base models.

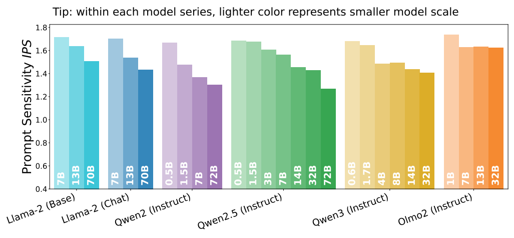


Figure 7: A comparison of prompt sensitivity across different model scales. As the model scale increases, the prompt sensitivity within a model series systematically decreases.

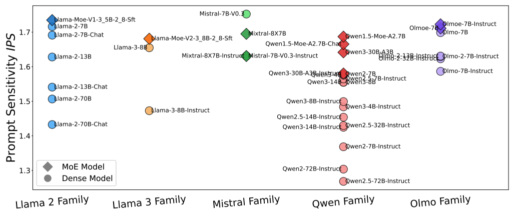


Figure 8: A Comparison of prompt sensitivity between MoE models and dense models. Generally, MoE models tend to be more sensitive than dense models in the same model family.

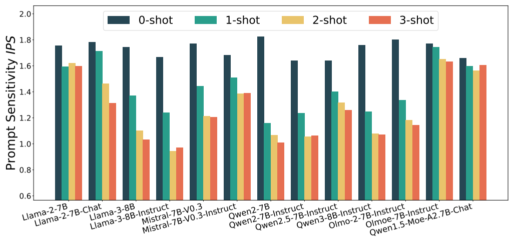


Figure 9: A comparison of prompt sensitivity between 0-shot learning and few-shot learning. The drop in prompt sensitivity is substantial from 0-shot to 1-shot.

4.2 Analyzing the Factors impacting Prompt Sensitivity

In this section, we investigate factors that might influence the prompt sensitivity of LLMs, including (1) supervised fine-tuning, (2) model scales, (3) model architectures, and (4) prompting methods.

Factor 1: instruct/chat models vs. base models. Supervised fine-tuning (*e.g.*, instruction tuning) is now a common practice to align base models with human preferences for certain tasks, yielding models often referred to as instruct or chat models (for different tasks or purposes). Therefore, we investigate the impact of supervised fine-tuning by comparing the prompt sensitivity of instruct/chat models with base models. Results[5] on the ARC dataset in Figure 6 show that almost all instruct/chat models exhibit lower prompt sensitivity than their corresponding base models. This demonstrates that supervised fine-tuning enables the LLM to encode more stable interactions. A plausible explanation is that base models are pre-trained on unstructured and raw texts, but instruct/chat models are further fine-tuned on instruction-response datasets or dialogue datasets. Thus, instruct/chat models can precisely understand the function of prompt templates and focus on the task-relevant inputs.

Factor 2: model scales. We investigate the relationship between the model scale (*i.e.*, the number of parameters) and the prompt sensitivity. Results[5] on the ARC dataset in Figure 7 show that within the same model series, as the model scale increases, the overall prompt sensitivity systematically decreases. This indicates that larger LLMs encode more stable interactions, making them less susceptible to superficial changes in the prompt template.

Factor 3: dense models vs. MoE models. MoE models scale up model capacity with minimal computational cost by dynamically activating different subsets of "expert" sub-networks when processing inputs (Cai et al., 2025). In contrast, all parameters in dense models participate in every computation. We aim to investigate the impact of model architectures by comparing the prompt sensitivity of dense models with MoE models. Results[5] on the ARC dataset in Figure 8 show that in model families including Llama-2, Llama-3, Qwen and Olmo, all MoE models exhibit higher prompt sensitivity than dense models. This suggests that MoE models encode more unstable interactions. In conclusion, while MoE models achieve impressive performance with reduced computational overhead, this benefit comes at the cost of weaker stability.

Factor 4: prompting methods (few-shot learning vs. 0-shot learning). Few-shot learning is utilized to improve LLMs' performance by providing in-context examples to better specify the task

[5]Results on the MMLU dataset in Appendix F exhibit the same conclusion.

(Wang et al., 2020). We investigate the impact of prompting methods on prompt sensitivity by comparing 0-shot learning with few-shot learning[6]. Results[5] on the ARC dataset in Figure 9 show that incorporating in-context examples leads to a significant reduction in prompt sensitivity across all tested LLMs. For most of the LLMs, the most substantial drop occurs when moving from 0-shot to 1-shot, while adding more in-context examples (2-shot, 3-shot) yields slower reductions. This suggests that even a single example is sufficient to establish the LLM's understanding of the task, leading it to ignore superficial template variations and focus on the core input.

4.3 Explore the Underlying Mechanisms of Improved Stability

In this section, we aim to explore **whether there exists a common reason to explain the underlying mechanisms by which the four aforementioned factors reduce the prompt sensitivity of LLMs**. Specifically, we analyze the prompt sensitivity of different types of interactions, so as to reveal the source of the LLM's prompt sensitivity. To this end, we analyze the sensitivity of interactions with different complexities, which are defined as the orders of interactions. The order of an interaction S is defined as the number of input variables included, *i.e.*, $|S|$. The order reflects the complexity of an interaction. An interaction with high order indicates an intricate relationship including many input variables, while an interaction with low order represents a simple relationship including few input variables. We further define three types of prompt sensitivity metrics corresponding to different types of interaction orders. Specifically, we partition interactions in Ω_{union} in Eq. (3) into three distinct groups based on their orders: low-order, mid-order, and high-order. For detail, given an input $\boldsymbol{x}$ with n words, $\Omega^{low}_{\text{union}} \triangleq \{S \in \Omega_{\text{union}} \mid 1 \leq |S| \leq \lfloor \frac{1}{3}n \rfloor\}, \Omega^{mid}_{\text{union}} \triangleq \{S \in \Omega_{\text{union}} \mid \lfloor \frac{1}{3}n \rfloor < |S| \leq \lfloor \frac{2}{3}n \rfloor\}, \Omega^{high}_{\text{union}} \triangleq \{S \in \Omega_{\text{union}} \mid \lfloor \frac{2}{3}n \rfloor < |S| \leq n\}$. Then we calculate prompt sensitivity for low-order, mid-order, and high-order interactions, as IPS^{low}, IPS^{mid}, and IPS^{high}.

Figure 10 presents the prompt sensitivity of different order types on the ARC dataset. Results[4] show that the prompt sensitivity of low-order interactions is the lowest, followed by mid-order, while high-order interactions exhibit the highest prompt sensitivity. This indicates that low-order interactions encoded by LLMs are relatively stable when faced with subtle changes to prompt templates, *i.e.*, simple interaction patterns are more robust. Conversely, the high sensitivity of high-order interactions reveals that the LLMs' internal representation of complex patterns is highly unstable.

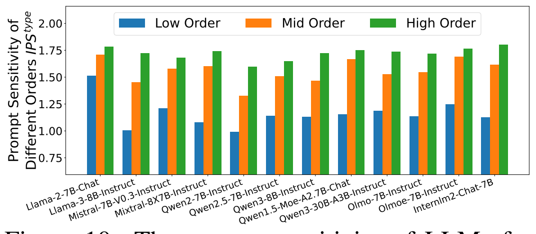


Figure 10: The prompt sensitivity of LLMs for different order types.

Explaining why the above four factors can reduce prompt sensitivity. Inspired by the results in Figure 10, we now investigate how the four aforementioned factors influence the prompt sensitivity of low-, mid-, and high-order interactions encoded by the LLM. This order-level analysis aims to reveal the common mechanism by which these factors reduce the LLM's prompt sensitivity. For each factor, we quantify its effect on low-, mid-, and high-order interactions by computing the relative change in IPS between the LLM with the factor and its counterpart without it. Specifically, given $type \in \{low, mid, high\}$, the relative change is defined as $\Delta IPS^{type} = (IPS^{type}_{\text{A}} - IPS^{type}_{\text{B}})/IPS^{type}_{\text{B}}$. For **Factor 1** (fine-tuned vs. base), A and B are fine-tuned and base models, respectively. For **Factor 3** (dense vs. MoE), A and B are dense and MoE models, with the final ΔIPS^{type} being the average over all pairs of a specific dense model and a specific MoE model within a model family. For **Factor 4** (few-shot vs. 0-shot), A and B are x-shot ($x \in \{1, 2, 3\}$) and 0-shot learning. The relative change metric is unsuitable for **Factor 2** (model scales), as scale is a continuous variable. Instead, we directly analyze the trend of IPS values as model parameters increase.

The results presented in Figure 11 and Figure 12 converge on a common explanation for how the four aforementioned factors reduce prompt sensitivity. **The most significant reduction in prompt sensitivity is consistently observed in low-order interactions.** An obvious, though less pronounced, decrease is also seen at the mid-order level. In stark contrast, the change of the prompt sensitivity of high-order interactions is relatively minimal, remaining at a high level. It indicates that the stability of low-order interactions is critical to the overall robustness of LLMs.

[6]See Appendix E for the prompt templates and settings of few-shot learning.

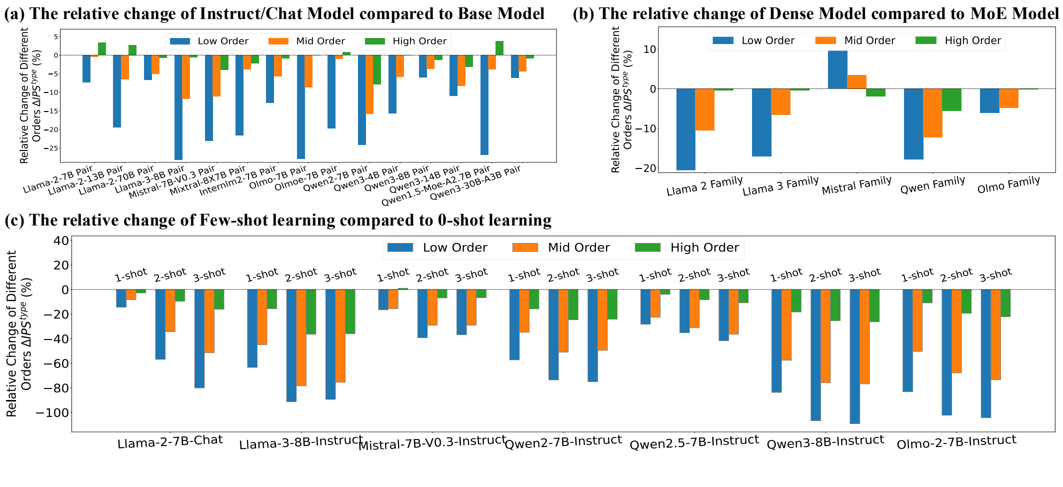


Figure 11: Comparing the relative change in the prompt sensitivity of low-, mid-, and high-order interactions for different factors.

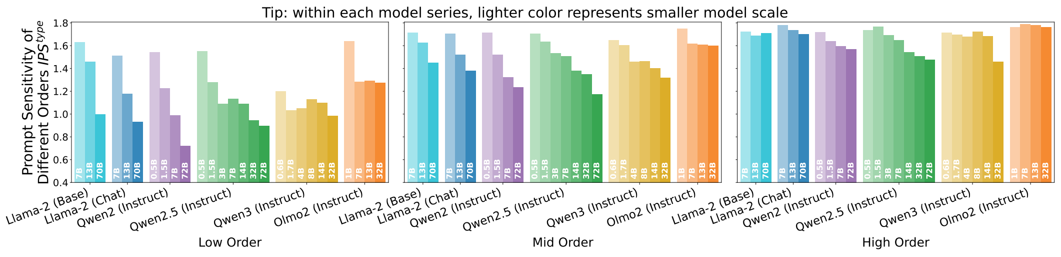


Figure 12: A comparison of prompt sensitivity at the order-level across different model scales.

This phenomenon is unexpected. Although results in Figure 10 show that low-order interactions are naturally more robust than other types of interactions, the four factors above still significantly reduce the prompt sensitivity of low-order interactions. Instead, they fail to reduce the prompt sensitivity of high-order interactions, which are inherently the most sensitive. This phenomenon indicates that stable low-order interactions are much easier for LLMs to learn, while it is more difficult for LLMs to make those complex high-order interactions more stable.

Detailed discussion on the impact of model architecture. We notice that in Figure 11 (b), the behavior of Mistral family is different from the other family: the dense models are more sensitive than the MoE models at low-order and mid-order level. We attribute this to the number of activated parameters, extending our finding from Factor 2 that larger models are less sensitive. While most MoE models (e.g., in Llama and Qwen families) are more sensitive due to fewer activated parameters, the Mistral case is reversed: Mixtral-8x7B activates more parameters than its dense counterpart Mistral-7B-V0.3 (13B vs. 7B), resulting in lower prompt sensitivity.

To more rigorously isolate the influence of model architecture on the prompt sensitivity from model size or other factors, we conduct a controlled variable analysis. We select specific pairs from the Qwen and OLMo families that share the most similar model scales (*i.e.*, activated parameters) and analogous training paradigms (*i.e.*, base vs. base, instruct/chat vs. instruct/chat). This targeted comparison enables us to minimize confounding factors and focus directly on the architectural impact.

Results in Table 1 consistently show that MoE models exhibit higher prompt sensitivity than dense models. This suggests that the increased prompt sensitivity of MoE architectures is not merely a consequence of smaller number of activated parameters. Instead, it further strengthens our conclusion that the MoE architecture inherently increases the prompt sensitivity of LLMs.

4.4 Verifying the generalizability of the methods and conclusions

Verifying the generalizability of the methods and conclusions beyond the MCQ format. We conduct an additional set of experiments on open-ended generation tasks. We utilize the Dolly-15k dataset (Ouyang et al., 2022), which contains a diverse range of non-MCQ tasks, including creative

Model Name	Architecture	Act. Params.	Type	IPS (ARC) ↓	IPS (MMLU) ↓
Qwen1.5-moe-a2.7b-chat	MoE	2.7B	Chat	1.665	1.640
Qwen2.5-3b-instruct	Dense	3B	Instruct	**1.606**	**1.625**
Olmoe-7b	MoE	1B	Base	1.714	1.717
Olmo-1b	Dense	1B	Base	**1.632**	**1.658**
Qwen3-30b-a3b	MoE	3B	Base	1.642	1.680
Qwen3-4b	Dense	4B	Base	**1.568**	**1.599**
Qwen3-30b-a3b-instruct	MoE	3B	Instruct	1.580	1.617
Qwen3-4b-instruct	Dense	4B	Instruct	**1.483**	**1.551**

Table 1: Controlled variable analysis of prompt sensitivity between MoE and dense models.

writing, open Q&A, classification, *etc*. The results of this analysis, detailed in Appendix I, consistently verify our main conclusions drawn from the MCQ experiments. This replication strongly suggests that the four factors and the underlying mechanisms of prompt sensitivity we have uncovered can be generalized to open-ended questions and are inherent characteristics of LLMs.

Verifying the generalizability of the methods and conclusions beyond prompt template modifications to scenarios more aligned with real-world user interactions. In this experimental setup, we use the Dolly-15k dataset and introduce more complex prompt perturbations, specifically semantic paraphrases and instruction reordering, to test the robustness of our conclusions. The results presented in Appendix J consistently affirm our conclusions drawn from the template-based experiments. This validation indicates that the identified four factors and their shared mechanism are fundamental aspects of prompt sensitivity, rather than being confined to template alterations.

Verifying the generalizability of the methods and conclusions on different threshold τ. In Sections 4.2 and 4.3, we set the threshold τ to 0.1 to distinguish salient interactions from noise. Li & Zhang (2023) conducted experiments which show that conclusions are not sensitive to the choice of τ. To ensure the robustness of our findings, we conducted hyperparameter experiments with $\tau = 0.05$ and $\tau = 0.15$. Our main conclusions remain consistent across different threshold values. Please see Appendix F for the details of hyperparameter experiments.

Solutions for reducing the computational cost of the methods. A limitation of the framework is its exponential computational cost with respect to the number of input variables, making it challenging for long text inputs. To address this issue, we propose and validate several strategies: **(1) Select informative words as input variables while treating uninformative ones (e.g., stop words) as fixed background context. (2) Merge related words into combined phrases as input variables**. In our experiments on long-form open-ended questions, we apply these two methods to effectively control the number of input variables. The specific selection strategies are detailed in Appendix I and Appendix J. It demonstrates that these techniques can substantially reduce computational complexity without affecting the key conclusions. For future work, **(3) Approximation Techniques offer a promising avenue.** Methods specifically designed for efficiently computing interaction indices, such as the fast Möbius transform that leverages the sparse, low-degree nature of interactions (Kang et al., 2024), are directly applicable here. Further details are provided in Appendix H.

5 CONCLUSION AND DISCUSSION

In this paper, we propose an interaction-based metric to evaluate the prompt sensitivity of LLMs. We empirically find that employing supervised fine-tuning, increasing model scale, using dense over MoE architectures, and applying few-shot learning all serve to reduce the prompt sensitivity of LLMs. Our findings offer novel insights into both model designs and prompting methods for improving the robustness of LLMs. More crucially, we find that the aforementioned factors achieve lower prompt sensitivity primarily by reducing the sensitivity of low-order interactions, while the prompt sensitivity of high-order interactions remains at a relatively high level.

Our research aims to analyze the instability of LLM interactions resulting from minor prompt modifications, specifically concerning prompt templates. Subsequent efforts will focus on implementing this framework in more realistic, open-ended scenarios, such as semantic paraphrasing (Cao et al., 2024), thereby augmenting its practical applicability in mitigating prompt sensitivity.

REPRODUCIBILITY STATEMENT

We are committed to ensuring the reproducibility of our research. To this end, we provide detailed documentation for all aspects of our work. **Theoretical Framework**: The formal definitions, derivations, and proofs for all theoretical concepts and formulas presented in this paper, including the Universal Matching Property, the Sparsity Property and our interaction-based sensitivity metric (IPS), are detailed in Section 3.1, Section 4.1, Appendix B, Appendix C, and Appendix D. **Experimental Setup**: All experimental settings are thoroughly documented. This includes comprehensive lists of the LLMs and datasets used, the exact prompt templates, and the specifications of the computing infrastructure, which can be found in Section 4.1 and Appendix E. **Code and Datasets**: The code and datasets for data preprocessing, along with part of other implementation details necessary to replicate our experiments, are provided in the Supplementary Material. The full code and datasets will be publicly released upon the acceptance of this paper.

REFERENCES

Norah Alzahrani, Hisham Alyahya, Yazeed Alnumay, Sultan AlRashed, Shaykhah Alsubaie, Yousef Almushayqih, Faisal Mirza, Nouf Alotaibi, Nora Al-Twairesh, Areeb Alowisheq, M Saiful Bari, and Haidar Khan. When benchmarks are targets: Revealing the sensitivity of large language model leaderboards. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13787–13805, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.744. URL `https://aclanthology.org/2024.acl-long.744/`.

Marco Ancona, Cengiz Oztireli, and Markus Gross. Explaining deep neural networks with a polynomial time algorithm for shapley value approximation. In *International conference on machine learning*, pp. 272–281. PMLR, 2019.

Yadagiri Annepaka and Partha Pakray. Large language models: A survey of their development, capabilities, and applications. *Knowledge and Information Systems*, pp. 1–56, 2024.

Steven Bird. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL 2006 interactive presentation sessions*, pp. 69–72, 2006.

Weilin Cai, Juyong Jiang, Fan Wang, Jing Tang, Sunghun Kim, and Jiayi Huang. A survey on mixture of experts in large language models. *IEEE Transactions on Knowledge and Data Engineering*, 2025.

Zheng Cai, Maosong Cao, Haojiong Chen, Kai Chen, Keyu Chen, Xin Chen, Xun Chen, Zehui Chen, Zhi Chen, Pei Chu, et al. Internlm2 technical report. *arXiv preprint arXiv:2403.17297*, 2024.

Bowen Cao, Deng Cai, Zhisong Zhang, Yuexian Zou, and Wai Lam. On the worst prompt performance of large language models. *Advances in Neural Information Processing Systems*, 37: 69022–69042, 2024.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. A survey on evaluation of large language models. *ACM transactions on intelligent systems and technology*, 15(3):1–45, 2024.

Anwoy Chatterjee, H S V N S Kowndinya Renduchintala, Sumit Bhatia, and Tanmoy Chakraborty. POSIX: A prompt sensitivity index for large language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 14550–14565, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.852. URL `https://aclanthology.org/2024.findings-emnlp.852/`.

Lu Chen, Siyu Lou, Benhao Huang, and Quanshi Zhang. Defining and extracting generalizable interaction primitives from dnns. *arXiv preprint arXiv:2401.16318*, 2024.

Lei Cheng, Junpeng Zhang, Qihan Ren, and Quanshi Zhang. Revisiting generalization power of a dnn in terms of symbolic interactions. *arXiv preprint arXiv:2502.10162*, 2025.

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

Ian Covert, Scott Lundberg, and Su-In Lee. Feature removal is a unifying principle for model explanation methods. *arXiv preprint arXiv:2011.03623*, 2020.

Piotr Dabkowski and Yarin Gal. Real time image saliency for black box classifiers. *Advances in neural information processing systems*, 30, 2017.

Huiqi Deng, Na Zou, Mengnan Du, Weifu Chen, Guocan Feng, Ziwei Yang, Zheyang Li, and Quanshi Zhang. Unifying fourteen post-hoc attribution methods with taylor interactions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(7):4625–4640, 2024.

Federico Errica, Davide Sanvito, Giuseppe Siracusano, and Roberto Bifulco. What did I do wrong? quantifying LLMs' sensitivity and consistency to prompt engineering. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *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)*, pp. 1543–1558, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.73. URL `https://aclanthology.org/2025.naacl-long.73/`.

Ruth Fong, Mandela Patrick, and Andrea Vedaldi. Understanding deep networks via extremal perturbations and smooth masks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 2950–2958, 2019.

Ruth C Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE international conference on computer vision*, pp. 3429–3437, 2017.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, et al. Olmo: Accelerating the science of language models. *arXiv preprint arXiv:2402.00838*, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7B. *arXiv e-prints*, art. arXiv:2310.06825, October 2023. doi: 10.48550/arXiv.2310.06825.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Justin Kang, Yigit Efe Erginbas, Landon Butler, Ramtin Pedarsani, and Kannan Ramchandran. Learning to understand: Identifying interactions via the möbius transform. *Advances in Neural Information Processing Systems*, 37:46160–46202, 2024.

Mingjie Li and Quanshi Zhang. Does a neural network really encode symbolic concepts? In *International conference on machine learning*, pp. 20452–20469. PMLR, 2023.

Sheng Lu, Hendrik Schuff, and Iryna Gurevych. How are prompts different in terms of sensitivity? In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 5833–5856, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.325. URL `https://aclanthology.org/2024.naacl-long.325/`.

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

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, et al. Olmoe: Open mixture-of-experts language models. *arXiv preprint arXiv:2409.02060*, 2024.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, et al. 2 olmo 2 furious. *arXiv preprint arXiv:2501.00656*, 2024.

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

Xiaoye Qu, Daize Dong, Xuyang Hu, Tong Zhu, Weigao Sun, and Yu Cheng. Llama-moe v2: Exploring sparsity of llama from perspective of mixture-of-experts with post-training. *arXiv preprint arXiv:2411.15708*, 2024.

Amirhossein Razavi, Mina Soltangheis, Negar Arabzadeh, Sara Salamat, Morteza Zihayat, and Ebrahim Bagheri. Benchmarking prompt sensitivity in large language models. In *European Conference on Information Retrieval*, pp. 303–313. Springer, 2025.

Jie Ren, Mingjie Li, Qirui Chen, Huiqi Deng, and Quanshi Zhang. Defining and quantifying the emergence of sparse concepts in dnns. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 20280–20289, 2023a.

Jie Ren, Xinhao Zheng, Jiyu Liu, Andrew Lizarraga, Ying Nian Wu, Liang Lin, and Quanshi Zhang. Monitoring primitive interactions during the training of dnns. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(19):20183–20191, Apr. 2025a. doi: 10.1609/aaai.v39i19.34223. URL `https://ojs.aaai.org/index.php/AAAI/article/view/34223`.

Qihan Ren, Huiqi Deng, Yunuo Chen, Siyu Lou, and Quanshi Zhang. Bayesian neural networks avoid encoding complex and perturbation-sensitive concepts. In *International Conference on Machine Learning*, pp. 28889–28913. PMLR, 2023b.

Qihan Ren, Jiayang Gao, Wen Shen, and Quanshi Zhang. Where we have arrived in proving the emergence of sparse interaction primitives in DNNs. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=3pWSL8My6B`.

Qihan Ren, Junpeng Zhang, Yang Xu, Yue Xin, Dongrui Liu, and Quanshi Zhang. Towards the dynamics of a dnn learning symbolic interactions. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2025b. Curran Associates Inc. ISBN 9798331314385.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. *arXiv preprint arXiv:2310.11324*, 2023.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022.

Pascal Sturmfels, Scott Lundberg, and Su-In Lee. Visualizing the impact of feature attribution baselines. *Distill*, 5(1):e22, 2020.

Jiuding Sun, Chantal Shaib, and Byron C Wallace. Evaluating the zero-shot robustness of instruction-tuned language models. *arXiv preprint arXiv:2306.11270*, 2023.

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

Qwen Team. Qwen2 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Ian Tenney, James Wexler, Jasmijn Bastings, Tolga Bolukbasi, Andy Coenen, Sebastian Gehrmann, Ellen Jiang, Mahima Pushkarna, Carey Radebaugh, Emily Reif, et al. The language interpretability tool: Extensible, interactive visualizations and analysis for nlp models. *arXiv preprint arXiv:2008.05122*, 2020.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3):1–34, 2020.

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

Qwen An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxin Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yi-Chao Zhang, Yunyang Wan, Yuqi Liu, Zeyu Cui, Zhenru Zhang, Zihan Qiu, Shanghaoran Quan, and Zekun Wang. Qwen2.5 technical report. *ArXiv*, abs/2412.15115, 2024. URL `https://api.semanticscholar.org/CorpusID:274859421`.

Quanshi Zhang, Xin Wang, Ruiming Cao, Ying Nian Wu, Feng Shi, and Song-Chun Zhu. Extraction of an explanatory graph to interpret a cnn. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):3863–3877, 2020.

Huilin Zhou, Hao Zhang, Huiqi Deng, Dongrui Liu, Wen Shen, Shih-Han Chan, and Quanshi Zhang. Concept-level explanation for the generalization of a dnn. *arXiv e-prints*, pp. arXiv–2302, 2023a.

Huilin Zhou, Hao Zhang, Huiqi Deng, Dongrui Liu, Wen Shen, Shih-Han Chan, and Quanshi Zhang. Explaining generalization power of a dnn using interactive concepts. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 17105–17113, 2024.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023b.

Kaijie Zhu, Jindong Wang, Jiaheng Zhou, Zichen Wang, Hao Chen, Yidong Wang, Linyi Yang, Wei Ye, Yue Zhang, Neil Gong, and Xing Xie. Promptrobust: Towards evaluating the robustness of large language models on adversarial prompts. In *Proceedings of the 1st ACM Workshop on Large AI Systems and Models with Privacy and Safety Analysis*, LAMPS '24, pp. 57–68, New York, NY, USA, 2024a. Association for Computing Machinery. ISBN 9798400712098. doi: 10.1145/3689217.3690621. URL `https://doi.org/10.1145/3689217.3690621`.

Tong Zhu, Xiaoye Qu, Daize Dong, Jiacheng Ruan, Jingqi Tong, Conghui He, and Yu Cheng. Llama-moe: Building mixture-of-experts from llama with continual pre-training. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 15913–15923, 2024b.

Jingming Zhuo, Songyang Zhang, Xinyu Fang, Haodong Duan, Dahua Lin, and Kai Chen. ProSA: Assessing and understanding the prompt sensitivity of LLMs. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 1950–1976, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.108. URL `https://aclanthology.org/2024.findings-emnlp.108/`.

APPENDIX

A MASKING STRATEGIES OF INPUT VARIABLES

In attribution method research, it is common to employ a specific token or embedding to mask the input variables of a deep neural network (DNN) (Lundberg & Lee, 2017; Ancona et al., 2019; Fong et al., 2019) and use changes in network outputs on the masked samples to estimate attributions of different input variables. The selection of a masking approach is complex, as each method has its weakness. For example, replacing input variables with the mean baseline value (the average of all samples) or the zero baseline value can introduce out-of-distribution signals, thereby providing the model with artificial information, such as uniform grey or black dots in an image (Dabkowski & Gal, 2017; Ancona et al., 2019; Sundararajan et al., 2017). Additionally, blurring image pixels using a Gaussian kernel (Fong & Vedaldi, 2017; Fong et al., 2019) as the masked state removes high-frequency signals but fails to eliminate low-frequency signals (Covert et al., 2020; Sturmfels et al., 2020).

Given these challenges, we adopt a token replacement strategy, which is standard for the text domain. This involves substituting the target input word with a dedicated [MASK] token at the embedding level. For example, to mask the word ”green” in the input ”He is a green hand,” we would provide the LLM with the modified input ”He is a [MASK] hand.” This approach effectively nullifies the specific semantic contribution of the target word without introducing out-of-distribution artifacts, ensuring a clean and consistent baseline for our interaction analysis. For the specific [MASK] token for each LLM, please refer to Section 5 for details.

B PROOF OF THEOREM

B.1 PROOF OF UNIVERSAL MATCHING PROPERTY

In the main body of the paper, for the sake of simplicity and clarity, we introduced the Universal Matching Property (Theorem 1) primarily through the lens of AND interactions. However, our empirical analysis and the underlying theoretical framework are built upon a more comprehensive AND-OR interaction framework. This extended framework, which incorporates both AND and OR interaction patterns, also adheres to the Universal Matching Property.

In this section, we provide the formal proof for the Universal Matching Property of the complete AND-OR interaction framework. This proof is more general and naturally subsumes the proof for the AND interaction framework presented as Theorem 1 in the main text. We will demonstrate that the output of the surrogate logical model, which is the sum of all AND-OR interaction effects, can perfectly match the output of the Deep Neural Network (DNN) for any masked sample.

The **surrogate logical model** $\phi(\cdot)$ is defined as follows:

$$\phi(\boldsymbol{x}_T) \triangleq \phi(\boldsymbol{x}_\emptyset) + \sum_{S\subseteq N, S\neq\emptyset} \mathbb{1}_{\text{AND}}(S \mid \boldsymbol{x}_T) \cdot I_S^{\text{AND}} + \sum_{S\subseteq N, S\neq\emptyset} \mathbb{1}_{\text{OR}}(S \mid \boldsymbol{x}_T) \cdot I_S^{\text{OR}}, \tag{4}$$

where the AND trigger function $\mathbb{1}_{\text{AND}}(S \mid \boldsymbol{x}_T) \in \{0, 1\}$ represents an **AND relationship** between input variables in S, which can also be termed **AND interaction pattern**; the OR trigger function $\mathbb{1}_{\text{OR}}(S \mid \boldsymbol{x}_T) \in \{0, 1\}$ represents an **OR relationship** between input variables in S, which can also be termed **OR interaction pattern**. The scalar weight I_S^{AND} quantifies the effect of an AND relationship, which can also be termed **AND interaction effect**; the scalar weight I_S^{OR} quantifies the effect of an OR relationship, which can also be termed **OR interaction effect**. An AND relationship is activated only by the joint presence of all input variables in the set S, *i.e.*, all input variables in S are not masked. For instance, given the input sentence $\boldsymbol{x}$ = *“He is a green hand,”* the co-occurrence of the input variables in the set $S = \{green, hand\}$ contributes a numerical effect I_S^{AND} that pushes the surrogate logical model’s inference towards the semantic meaning of *“beginner.”* If an AND interaction S is triggered, *i.e.*, $\mathbb{1}_{\text{AND}}(S \mid \boldsymbol{x}_T) = 1$, the corresponding interaction effect I_S^{AND} is added to the output of the logical model. Otherwise, if any word in S is masked and the AND interaction is not triggered, *i.e.*, $\mathbb{1}_{\text{AND}}(S \mid \boldsymbol{x}_T) = 0$, the interaction effect I_S^{AND} is not added to the output of the logical model. An OR relationship is activated by the presence of any of all input variables in the set S, *i.e.*, any input variables in S are not masked. For instance, given the input sentence $\boldsymbol{x}$ = *"The service was terrible and the food was awful,"* the presence of any input variables in the set $S = \{$*terrible*, *awful*$\}$ contributes a numerical effect I_S^{OR} that pushes the surrogate logical model's inference towards a negative sentiment classification. If an OR interaction S is triggered, *i.e.*, $\mathbb{1}_{\text{OR}}(S \mid \boldsymbol{x}_T) = 1$, the corresponding interaction effect I_S^{OR} is added to the output of the logical model. Otherwise, if all words in S are masked and the OR interaction is not triggered, *i.e.*, $\mathbb{1}_{\text{OR}}(S \mid \boldsymbol{x}_T) = 0$, the interaction effect I_S^{OR} is not added to the output of the logical model. $\boldsymbol{x}_\emptyset$ represents that all input variables in N are masked.

Definition of universal matching property for AND-OR interactions. When the scalar weights in the surrogate logical model $\phi(\cdot)$ are set to $I_S^{\text{AND}} = \sum_{T\subseteq S}(-1)^{|S|-|T|}v_{\text{and}}(\boldsymbol{x}_T)$ and $I_S^{\text{OR}} = -\sum_{T\subseteq S}(-1)^{|S|-|T|}v_{\text{or}}(\boldsymbol{x}_{N\setminus T})$, the output of $\phi(\cdot)$ can always match the output score of the DNN $v(\cdot)$, *i.e.*, $\forall T \subseteq N, v(\boldsymbol{x}_T) = \phi(\boldsymbol{x}_T)$. Here $v_{\text{and}}(\boldsymbol{x}_T) + v_{\text{or}}(\boldsymbol{x}_T) = v(\boldsymbol{x}_T)$.

We need to prove that given an input sample $\boldsymbol{x}$, for each masked sample $\{\boldsymbol{x}_T | T \subseteq N\}$, the network output score $v(\boldsymbol{x}_T) \in \mathbb{R}$ can be well matched by the surrogate logical model $\phi(\boldsymbol{x}_T)$. The surrogate logical model $\phi(\boldsymbol{x}_T)$ uses the sum of AND interactions and OR interactions to accurately explain/match the network output score $v(\boldsymbol{x}_T)$.

$$\begin{aligned}
&\forall T \subseteq N, v(\boldsymbol{x}_T) = \phi(\boldsymbol{x}_T).\\
&\phi(\boldsymbol{x}_T) = \phi(\boldsymbol{x}_\emptyset) + \sum_{S\subseteq N, S\neq\emptyset} \mathbb{1}_{\text{AND}}(S \mid \boldsymbol{x}_T)\cdot I_S^{\text{AND}} + \sum_{S\subseteq N, S\neq\emptyset} \mathbb{1}_{\text{OR}}(S \mid \boldsymbol{x}_T)\cdot I_S^{\text{OR}},\\
&\qquad\quad = \underbrace{v(\boldsymbol{x}_\emptyset) + \sum\nolimits_{S\subseteq T, S\neq\emptyset} I_S^{\text{AND}}}_{v_{\text{and}}(\boldsymbol{x}_T)} + \underbrace{\sum\nolimits_{S\subseteq N, S\cap T\neq\emptyset} I_S^{\text{OR}}}_{v_{\text{or}}(\boldsymbol{x}_T)}
\end{aligned} \tag{5}$$

Proof. **(1) Universal matching property of AND interactions.** For all 2^n masked samples $\{\boldsymbol{x}_T \mid T \subseteq N\}$, what we need to prove is that the output $v_{\text{and}}(\boldsymbol{x}_T)$ of a DNN can be universally explained by all the interactions in $T \subseteq N$, *i.e.*, $\forall S \subseteq T, S \neq \emptyset, v_{\text{and}}(\boldsymbol{x}_T) = \sum_{S\subseteq T, S\neq\emptyset} I_S^{\text{AND}}(\boldsymbol{x}) = v(\boldsymbol{x}_\emptyset) + \sum_{S\subseteq T,S\neq\emptyset} I_S^{\text{AND}}$. Here, $v(\boldsymbol{x}_\emptyset) = v_{\text{and}}(\boldsymbol{x}_\emptyset)$.

According to the definition of the AND interaction, $I_S^{\text{AND}}(\boldsymbol{x}) = \sum_{L\subseteq S}(-1)^{|S|-|L|}v_{\text{and}}(\boldsymbol{x}_L)$. To simplify the computation of the sum of AND interactions $\sum_{S\subseteq T,S\neq\emptyset} I_S^{\text{AND}}(\boldsymbol{x}) = \sum_{S\subseteq T,S\neq\emptyset}\sum_{L\subseteq S}(-1)^{|S|-|L|}v_{\text{and}}(\boldsymbol{x}_L)$, we exchange the order of summation of the set $L \subseteq S \subseteq T$ and the set $S \supseteq L$. Given a set of input variables L, we compute all linear combinations of all sets S containing L with respect to the model outputs $v_{\text{and}}(\boldsymbol{x}_S)$, *i.e.*, $\sum_{S:L\subseteq S\subseteq T}(-1)^{|S|-|L|}v_{\text{and}}(\boldsymbol{x}_L)$. Then, we compute all summations over the set $L \subseteq T$ as $\sum_{S\subseteq T,S\neq\emptyset} I_S^{\text{AND}}(\boldsymbol{x}) = \sum_{L\subseteq T}\sum_{S:L\subseteq S\subseteq T}(-1)^{|S|-|L|}v_{\text{and}}(\boldsymbol{x}_L)$. Then, we can compute different cases of $L \subseteq S \subseteq T$ as follows:

(1) When $L = T = S$, $\sum_{S:L\subseteq S\subseteq T}(-1)^{|S|-|L|}v_{\text{and}}(\boldsymbol{x}_L) = (-1)^{|T|-|T|}v_{\text{and}}(\boldsymbol{x}_L) = v_{\text{and}}(\boldsymbol{x}_L)$.

(2) When $L \subseteq S \subseteq T, L \neq T$, let us consider the linear combinations of all sets S with number $|S|$ for the model output $v_{\text{and}}(\boldsymbol{x}_L)$, respectively. Let $m := |S| - |L|$, $(0 \leq m \leq |T| - |L|)$, then there are a total of $C^m_{|T|-|L|}$ combinations of all sets S of order $|S|$. Given L, accumulating the model outputs $v_{\text{and}}(\boldsymbol{x}_L)$ corresponding to all $S \supseteq L$, we can get $\sum_{S:L\subseteq S\subseteq T}(-1)^{|S|-|L|}v_{\text{and}}(\boldsymbol{x}_L) = v_{\text{and}}(\boldsymbol{x}_L)\cdot \underbrace{\sum\nolimits_{m=0}^{|T|-|L|} C^m_{|T|-|L|}(-1)^m}_{=0} = 0$.

Considering all the cases, the complete derivation of the sum of AND interactions is as follows.

$$
\begin{aligned}
&\sum\nolimits_{S\subseteq T, S\neq\emptyset} I_S^{\text{AND}} \\
=&\sum\nolimits_{S\subseteq T, S\neq\emptyset} \sum\nolimits_{L\subseteq S} (-1)^{|S|-|L|} v_{\text{and}}(\boldsymbol{x}_L) \\
=&\sum\nolimits_{L\subseteq T} \sum\nolimits_{S: L\subseteq S\subseteq T} (-1)^{|S|-|L|} v_{\text{and}}(\boldsymbol{x}_L) - v_{\text{and}}(\boldsymbol{x}_\emptyset) \\
=&\underbrace{v_{\text{and}}(\boldsymbol{x}_T)}_{L=T} + \sum\nolimits_{L\subseteq T, L\neq T} v_{\text{and}}(\boldsymbol{x}_L)\cdot \underbrace{\sum\nolimits_{m=0}^{|T|-|L|} C_{|T|-|L|}^{m} (-1)^m}_{=0} - v_{\text{and}}(\boldsymbol{x}_\emptyset) \\
=&v_{\text{and}}(\boldsymbol{x}_T) - v(\boldsymbol{x}_\emptyset)
\end{aligned}
\tag{6}
$$

Therefore, we have proved that $\forall\emptyset \neq T\subseteq N, v_{\text{and}}(\boldsymbol{x}_T) = v(\boldsymbol{x}_\emptyset) + \sum_{S\subseteq T, S\neq\emptyset} I_S^{\text{AND}}$.

(2) Universal matching theorem of OR interactions. What we need to prove is that $\forall T \subseteq N, v_{\text{or}}(\boldsymbol{x}_T) = \sum_{S\in\{S: S\cap T\neq\emptyset\}\cup\{\emptyset\}} I_S^{\text{OR}} = \sum_{S: S\cap T\neq\emptyset} I_S^{\text{OR}}$. Here $I_\emptyset^{\text{OR}} = v_{\text{or}}(\boldsymbol{x}_\emptyset) = 0$.

According to the definition of the OR interaction, $I_S^{\text{OR}} := -\sum_{L\subseteq S}(-1)^{|S|-|L|} v_{\text{or}}(\boldsymbol{x}_{N\setminus L})$. To simplify the computation of the sum of OR interactions $\sum_{S: S\cap T\neq\emptyset} I_S^{\text{OR}} = \sum_{S: S\cap T\neq\emptyset}\left[-\sum_{L\subseteq S}(-1)^{|S|-|L|} v_{\text{or}}(\boldsymbol{x}_{N\setminus L})\right]$, we also exchange the order of summation of the set $L\subseteq S\subseteq N$ and the set $S: S\cap T=\emptyset$. Given a set of input variables L, we compute all linear combinations of all sets S containing L with respect to the model outputs $v_{\text{or}}(\boldsymbol{x}_{N\setminus L})$, *i.e.*, $\sum_{S: S\cap T\neq\emptyset, N\supseteq S\supseteq L}(-1)^{|S|-|L|} v_{\text{or}}(\boldsymbol{x}_{N\setminus L})$. Then, we compute all summations over the set $L\subseteq N$ as $\sum_{S: S\cap T\neq\emptyset} I_S^{\text{OR}} = -\sum_{L\subseteq N}\sum_{S: S\cap T\neq\emptyset, N\supseteq S\supseteq L}(-1)^{|S|-|L|} v_{\text{or}}(\boldsymbol{x}_{N\setminus L})$. Then, we can compute different cases of $L\subseteq S\subseteq N, S\cap T\neq\emptyset$ as follows:

(1) When $L=N$ (then $S=N$), $\sum_{S: S\cap T\neq\emptyset, S\supseteq L}(-1)^{|S|-|L|} v_{\text{or}}(\boldsymbol{x}_{N\setminus L}) = (-1)^{|N|-|N|} v_{\text{or}}(\boldsymbol{x}_\emptyset) = v_{\text{or}}(\boldsymbol{x}_\emptyset) = 0$, Here $I_\emptyset^{\text{OR}} = v_{\text{or}}(\boldsymbol{x}_\emptyset) = 0$.

(2) When $L = N\setminus T$, for all sets $S: S\supseteq L, S\cap T\neq\emptyset$ (then $S\neq N\setminus T, S\neq L$), let us consider the linear combinations of all sets S with number $|S|$ for the model output $v_{\text{or}}(\boldsymbol{x}_T)$, respectively. Let $|S'| := |S|-|L|$, $(1\leq |S'|\leq |T|)$, then there are a total of $C_{|T|}^{|S'|}$ combinations of all sets S of order $|S|$. Thus, $\sum_{S: S\cap T\neq\emptyset, S\supseteq L}(-1)^{|S|-|L|} v_{\text{or}}(\boldsymbol{x}_{N\setminus L}) = v_{\text{or}}(\boldsymbol{x}_T)\cdot \underbrace{\sum_{|S'|=1}^{|T|} C_{|T|}^{|S'|}(-1)^{|S'|}}_{=-1} = -v_{\text{or}}(\boldsymbol{x}_T)$.

(3) When $L\cap T\neq\emptyset, L\neq N$, for all sets $S: S\supseteq L, S\cap T\neq\emptyset$, let us consider the linear combinations of all sets S with number $|S|$ for the model output $v_{\text{or}}(\boldsymbol{x}_T)$, respectively. Let us split $|S|-|L|$ into $|S'|$ and $|S''|$, *i.e.*,$|S|-|L| = |S'|+|S''|$, where $S' = \{i | i\in S, i\notin L, i\in N\setminus T\}$, $S'' = \{i|i\in S, i\notin L, i\in T\}$ (then $0\leq |S''|\leq |T|-|T\cap L|$) and $S'+S''+L = S$. Thus, there are a total of $C_{|T|-|T\cap L|}^{|S''|}$ combinations of all sets S'' of order $|S''|$. Thus, $\sum_{S: S\cap T\neq\emptyset, S\supseteq L}(-1)^{|S|-|L|} v_{\text{or}}(\boldsymbol{x}_{N\setminus L}) = v_{\text{or}}(\boldsymbol{x}_{N\setminus L})\cdot \sum_{S'\subseteq N\setminus T\setminus L}\underbrace{\sum_{|S''|=0}^{|T|-|T\cap L|} C_{|T|-|T\cap L|}^{|S''|}(-1)^{|S'|+|S''|}}_{=0} = 0$.

(4) When $L\cap T=\emptyset, L\neq N\setminus T$, let us split $|S|-|L|$ into $|S'|$ and $|S''|$, *i.e.*,$|S|-|L| = |S'|+|S''|$, where $S' = \{i|i\in S, i\notin L, i\in N\setminus T\}$, $S'' = \{i|i\in S, i\in T\}$ (then $0\leq |S''|\leq |T|$) and $S'+S''+L=S$. Thus, there are a total of $C_{|T|}^{|S''|}$ combinations of all sets S'' of order $|S''|$. Thus, $\sum_{S: S\cap T\neq\emptyset, S\supseteq L}(-1)^{|S|-|L|} v_{\text{or}}(\boldsymbol{x}_{N\setminus L}) = v_{\text{or}}(\boldsymbol{x}_{N\setminus L})\cdot \sum_{S'\subseteq N\setminus T\setminus L}\underbrace{\sum_{|S''|=0}^{|T|} C_{|T|}^{|S''|}(-1)^{|S'|+|S''|}}_{=0} = 0$.

Considering all the cases, the complete derivation of the sum of OR interactions is as follows.

$$\begin{aligned}
\sum\nolimits_{S:S\cap T\neq\emptyset} I_S^{\text{OR}} &= \sum\nolimits_{S:S\cap T\neq\emptyset}\left[-\sum\nolimits_{L\subseteq S}(-1)^{|S|-|L|}v_{\text{or}}(\boldsymbol{x}_{N\setminus L})\right]\\
&= -\sum\nolimits_{L\subseteq N}\sum\nolimits_{S:S\cap T\neq\emptyset, N\supseteq S\supseteq L}(-1)^{|S|-|L|}v_{\text{or}}(\boldsymbol{x}_{N\setminus L})\\
&= -\left[\sum_{|S'|=1}^{|T|}C_{|T|}^{|S'|}(-1)^{|S'|}\right]\cdot\underbrace{v_{\text{or}}(\boldsymbol{x}_T)}_{L=N\setminus T}-\underbrace{v_{\text{or}}(\boldsymbol{x}_\emptyset)}_{L=N}\\
&\quad-\sum_{L\cap T\neq\emptyset, L\neq N}\left[\sum_{S'\subseteq N\setminus T\setminus L}\left(\sum_{|S''|=0}^{|T|-|T\cap L|}C_{|T|-|T\cap L|}^{|S''|}(-1)^{|S'|+|S''|}\right)\right]\cdot v_{\text{or}}(\boldsymbol{x}_{N\setminus L})\\
&\quad-\sum_{L\cap T=\emptyset, L\neq N\setminus T}\left[\sum_{S'\subseteq N\setminus T\setminus L}\left(\sum_{|S''|=0}^{|T|}C_{|T|}^{|S''|}(-1)^{|S'|+|S''|}\right)\right]\cdot v_{\text{or}}(\boldsymbol{x}_{N\setminus L})\\
&= -(-1)\cdot v_{\text{or}}(\boldsymbol{x}_T)-v_{\text{or}}(\boldsymbol{x}_\emptyset)-\sum_{L\cap T\neq\emptyset, L\neq N}\left[\sum_{S'\subseteq N\setminus T\setminus L}0\right]\cdot v_{\text{or}}(\boldsymbol{x}_{N\setminus L})\\
&\quad-\sum_{L\cap T=\emptyset, L\neq N\setminus T}\left[\sum_{S'\subseteq N\setminus T\setminus L}0\right]\cdot v_{\text{or}}(\boldsymbol{x}_{N\setminus L})\\
&= v_{\text{or}}(\boldsymbol{x}_T)-v_{\text{or}}(\boldsymbol{x}_\emptyset)\\
&= v_{\text{or}}(\boldsymbol{x}_T)
\end{aligned} \tag{7}$$

Therefore, we have proved that $\forall T\subseteq N, v_{\text{or}}(\boldsymbol{x}_T)=\sum_{S:S\cap T\neq\emptyset} I_S^{\text{OR}}$.

(3) Universal matching theorem of AND-OR interactions. With the universal matching property of AND interactions and the universal matching property of OR interactions, we can easily get $v(\boldsymbol{x}_T)=\phi(\boldsymbol{x}_T)=v_{\text{and}}(\boldsymbol{x}_T)+v_{\text{or}}(\boldsymbol{x}_T)=v(\boldsymbol{x}_\emptyset)+\sum_{S\subseteq T,S\neq\emptyset} I_S^{\text{AND}}+\sum_{S\subseteq N,S\cap T\neq\emptyset} I_S^{\text{OR}}$, thus, we obtain the universal matching property of AND-OR interactions. □

B.2 Proof of Sparsity Property

Given all the masked samples $\{\boldsymbol{x}_T \mid T\subseteq N\}$, the surrogate logical model $\phi(\boldsymbol{x}_T)$ only utilizes a small set of salient AND interactions in Ω^{AND} and salient OR interactions in Ω^{OR} to approximate the network output score $v(\boldsymbol{x}_T)$. That is, the network's output can be well approximated by a small set of AND-OR interactions.

$$v(\boldsymbol{x}_T)=\phi(\boldsymbol{x}_T)\approx v(\boldsymbol{x}_\emptyset)+\sum\nolimits_{S\subseteq T,S\neq\emptyset,S\in\Omega_{\text{AND}}} I_S^{\text{AND}}+\sum\nolimits_{S\subseteq T,S\neq\emptyset,S\in\Omega_{\text{OR}}} I_S^{\text{OR}} \tag{8}$$

Proof. It has been proven by Ren et al. (2024) that under three common conditions[7], the output score $v_{\text{and}}(\boldsymbol{x}_T)$ of a well-trained DNN on all 2^n masked samples $\{\boldsymbol{x}_T|T\subseteq N\}$ could be universally estimated by a small number of AND interactions $T\in\Omega^{\text{AND}}$ with salient interaction effects I_S^{AND}, *s.t.*, $|\Omega^{\text{AND}}|\ll 2^n$, *i.e.*, $\forall T\subseteq N, v_{\text{and}}(\boldsymbol{x}_T)=\sum_{S\subseteq T,S\neq\emptyset} I_S^{\text{AND}}\approx\sum_{S\subseteq T,S\neq\emptyset,S\in\Omega^{\text{AND}}} I_S^{\text{AND}}$. According to Eq. (6), $v_{\text{and}}(\boldsymbol{x}_T)=v(\boldsymbol{x}_\emptyset)+\sum_{S\subseteq T,S\neq\emptyset} I_S^{\text{AND}}$. Therefore, $v_{\text{and}}(\boldsymbol{x}_T)\approx v(\boldsymbol{x}_\emptyset)+\sum_{S\subseteq T,S\neq\emptyset,S\in\Omega^{\text{AND}}} I_S^{\text{AND}}$.

Besides, as proven in Section C, the OR interaction can be considered as a special AND interaction. Thus, the confidence score $v_{\text{or}}(\boldsymbol{x}_T)$ of a well-trained DNN on all 2^n masked samples $\{\boldsymbol{x}_T|T\subseteq N\}$ could be universally estimated by a small number of OR interactions $T\in\Omega^{\text{OR}}$ with salient interaction effects I_S^{OR}, *s.t.*, $|\Omega^{\text{OR}}|\ll 2^n$. Similarly, $v_{\text{or}}(\boldsymbol{x}_T)=\sum_{S\subseteq T,S\neq\emptyset} I_S^{\text{OR}}\approx\sum_{S\subseteq T,S\neq\emptyset,S\in\Omega^{\text{OR}}} I_S^{\text{OR}}$

[7]Here are the three conditions: (1) The DNN doesn't encode extremely high-order AND interactions. (2) The DNN performs effectively on masked samples and exhibits greater confidence as the input sample is less masked. (3) When we increase the number of masked input variables, the confidence of the DNN does not drop significantly.

Thus, for each randomly masked sample $\boldsymbol{x}_T, T \subseteq N$, the surrogate logical model $\phi(\boldsymbol{x}_T)$ can use a small number of salient AND-OR interactions to approximate the network output score $v(\boldsymbol{x}_T)$, *i.e.*, $v(\boldsymbol{x}_T) = \phi(\boldsymbol{x}_T) = v_{\text{and}}(\boldsymbol{x}_T) + v_{\text{or}}(\boldsymbol{x}_T) \approx (\boldsymbol{x}_\emptyset) + \sum_{S \subseteq T, S \neq \emptyset, S \in \Omega_{\text{AND}}} I_S^{\text{AND}} + \sum_{S \subseteq T, S \neq \emptyset, S \in \Omega_{\text{OR}}} I_S^{\text{OR}}$.

□

C OR Interactions Can Be Considered as Special AND Interactions

If we reverse the definition of the masked state and the unmasked state of the input variable, the OR interaction I_S^{OR} can be considered as a special kind of AND interaction I_S^{AND}.

Given an input sample $\boldsymbol{x} \in \mathbb{R}^n$ and the output score of a DNN as $v(\cdot)$, if we randomly mask input variables in $\boldsymbol{x}$, we can get all 2^n masked samples. Let $\boldsymbol{x}_S$ denote the certain masked input sample when input variables in $N \setminus S$ are all masked and input variables in S are kept unchanged.

$$(\boldsymbol{x}_S)_i = \begin{cases} x_i, & i \in S \\ b_i, & i \in N \setminus S \end{cases} \tag{9}$$

where $\mathbf{b} \in \mathbb{R}^n$ are baseline values to represent the masked state of input variables.

If we reverse the definition of the masked state and the unmasked state of an input variable, *i.e.*, we consider $\mathbf{b}$ as the input sample and consider $\boldsymbol{x}$ as the masked state, then the masked sample $\widetilde{\boldsymbol{x}}_S$ can be defined as follows.

$$(\widetilde{\boldsymbol{x}}_S)_i = \begin{cases} b_i, & i \in S \\ x_i, & i \in N \setminus S \end{cases} \tag{10}$$

Thus, we can get $\boldsymbol{x}_{N \setminus S} = \widetilde{\boldsymbol{x}}_S$. To simplify the analysis, let us assume $v_{\text{and}}(\boldsymbol{x}_S) = v_{\text{or}}(\boldsymbol{x}_S) = 0.5v(\boldsymbol{x}_S)$, then the OR interaction I_S^{OR} can be regarded as a specific AND interaction $I_S^{\text{AND}}(\widetilde{\boldsymbol{x}})$ as follows.

$$\begin{aligned} I_S^{\text{OR}}(\mathbf{x}) &= -\sum_{T \subseteq S} (-1)^{|S|-|T|} v_{\text{or}}(\boldsymbol{x}_{N \setminus T}), \\ &= -\sum_{T \subseteq S} (-1)^{|S|-|T|} v_{\text{or}}(\widetilde{\boldsymbol{x}}_T), \\ &= -\sum_{T \subseteq S} (-1)^{|S|-|T|} v_{\text{and}}(\widetilde{\boldsymbol{x}}_T), \\ &= -I_S^{\text{AND}}(\widetilde{\boldsymbol{x}}). \end{aligned} \tag{11}$$

Now we have proved that OR interactions can be considered as special AND interactions.

D Details of Extracting the Sparsest AND-OR Interactions

We follow Li & Zhang (2023) to extract AND-OR interactions. Given a masked sample $\boldsymbol{x}_T$, the output score of the network $v(\boldsymbol{x}_T)$ can be decomposed into a combination of AND interaction and OR interaction, *i.e.*, $v(\boldsymbol{x}_T) = v_{\text{and}}(\boldsymbol{x}_T) + v_{\text{or}}(\boldsymbol{x}_T)$. Specifically, $v_{\text{and}}(\boldsymbol{x}_T) = 0.5 \cdot v(\boldsymbol{x}_T) + \gamma_T$ and $v_{\text{or}}(\boldsymbol{x}_T) = 0.5 \cdot v(\boldsymbol{x}_T) - \gamma_T$, where $\{\gamma_T \mid T \subseteq N\}$ is a set of learnable parameters. The parameters $\{\gamma_T\}$ were trained through minimizing the following LASSO-like loss to obtain sparse interactions:

$$\min_{\{\gamma_T\}} \sum_{S \subseteq N} |I_S^{\text{AND}}(\boldsymbol{x})| + |I_S^{\text{OR}}(\boldsymbol{x})|, \tag{12}$$

where $I_S^{\text{AND}}(\boldsymbol{x}) = \sum_{T \subseteq S} (-1)^{|S|-|T|} v_{\text{and}}(\boldsymbol{x}_T) = \sum_{T \subseteq S} (-1)^{|S|-|T|} (0.5 \cdot v(\boldsymbol{x}_T) + \gamma_T)$ and $I_S^{\text{OR}}(\boldsymbol{x}) = -\sum_{T \subseteq S} (-1)^{|S|-|T|} v_{\text{or}}(\boldsymbol{x}_{N \setminus T}) = -\sum_{T \subseteq S} (-1)^{|S|-|T|} (0.5 \cdot v(\boldsymbol{x}_T) - \gamma_T)$. Thus, we can extract the sparsest set of AND-OR interactions.

E Experimental Details

E.1 Why We Choose MCQ Datasets

In this study, we use multiple-choice question (MCQ) datasets for evaluating prompt sensitivity of LLMs. The primary reason is that the prompt templates of MCQs are more structured and compli-

Model Family	Model Series	Model	Type	Architecture	Scale
Llama (10 models)	Llama 2 (6 models)	Llama-2-7b	Base	Dense	7B
		Llama-2-7b-chat	Chat	Dense	7B
		Llama-2-13b	Base	Dense	13B
		Llama-2-13b-chat	Chat	Dense	13B
		Llama-2-70b	Base	Dense	70B
		Llama-2-70b-chat	Chat	Dense	70B
	Llama 3 (2 models)	Llama-3-8b	Base	Dense	8B
		Llama-3-8b-instruct	Instruct	Dense	8B
	Llama MoE (2 models)	Llama-moe-v1-3_5b-2_8-sft	Instruct	MoE	3.5B (Activated)
		Llama-moe-v2-3_8b-2_8-sft	Instruct	MoE	3.8B (Activated)
Mistral (4 models)	Mistral (2 models)	Mistral-7b-v0.3	Base	Dense	7B
		Mistral-7b-v0.3-instruct	Instruct	Dense	7B
	Mixtral (2 models)	Mixtral-8x7b	Base	MoE	13B (Activated)
		Mixtral-8x7b-instruct	Instruct	MoE	13B (Activated)
Qwen (25 models)	Qwen 1.5 MoE (2 models)	Qwen1.5-moe-a2.7b-chat	Chat	MoE	2.7B (Activated)
		Qwen1.5-moe-a2.7b	Base	MoE	2.7B (Activated)
	Qwen 2 (5 models)	Qwen2-7b	Base	Dense	7B
		Qwen2-0.5b-instruct	Instruct	Dense	0.5B
		Qwen2-1.5b-instruct	Instruct	Dense	1.5B
		Qwen2-7b-instruct	Instruct	Dense	7B
		Qwen2-72b-instruct	Instruct	Dense	72B
	Qwen 2.5 (7 models)	Qwen2.5-0.5b-instruct	Instruct	Dense	0.5B
		Qwen2.5-1.5b-instruct	Instruct	Dense	1.5B
		Qwen2.5-3b-instruct	Instruct	Dense	3B
		Qwen2.5-7b-instruct	Instruct	Dense	7B
		Qwen2.5-14b-instruct	Instruct	Dense	14B
		Qwen2.5-32b-instruct	Instruct	Dense	32B
		Qwen2.5-72b-instruct	Instruct	Dense	72B
	Qwen 3 (9 models)	Qwen3-0.6b-instruct	Instruct	Dense	0.6B
		Qwen3-1.7b-instruct	Instruct	Dense	1.7B
		Qwen3-4b	Base	Dense	4B
		Qwen3-4b-instruct	Instruct	Dense	4B
		Qwen3-8b	Base	Dense	8B
		Qwen3-8b-instruct	Instruct	Dense	8B
		Qwen3-14b	Base	Dense	14B
		Qwen3-14b-instruct	Instruct	Dense	14B
		Qwen3-32b-instruct	Instruct	Dense	32B
	Qwen 3 A3B (2 models)	Qwen3-30b-a3b-instruct	Instruct	MoE	3B (Activated)
		Qwen3-30b-a3b	Base	MoE	3B (Activated)
Olmo (9 models)	Olmo v1 (3 models)	Olmo-1b	Base	Dense	1B
		Olmo-7b	Base	Dense	7B
		Olmo-7b-instruct	Instruct	Dense	7B
	Olmo v2 (4 models)	Olmo-2-1b-instruct	Instruct	Dense	1B
		Olmo-2-7b-instruct	Instruct	Dense	7B
		Olmo-2-13b-instruct	Instruct	Dense	13B
		Olmo-2-32b-instruct	Instruct	Dense	32B
	OlmoE (2 models)	Olmoe-7b	Base	MoE	1B (Activated)
		Olmoe-7b-instruct	Instruct	MoE	1B (Activated)
InternLM (2 models)	InternLM 2 (2 models)	Internlm2-7b	Base	Dense	7B
		Internlm2-chat-7b	Chat	Dense	7B

Table 2: A comprehensive list and characteristics of LLMs, grouped by model family and model series.

cated, offering a wide range of modifiable components that do not alter the semantic meaning of the prompt template, including letter case (*e.g.*,“Answer” vs. “ANSWER”), separators (*e.g.*, “Answer:” vs. “Answer::”), option markers (*e.g.*, “A.” vs. “A)”), spacing, and some other components (Sclar et al., 2023).

This high degree of structural variability allows us to introduce a diverse set of superficial perturbations to the prompt templates while keeping the input (*i.e.*, the question and the options) identical. This setting is ideal for our core objective: to precisely measure whether the LLM’s internal representation of the input, captured by its interaction patterns, remains stable.

In contrast, the prompt templates for open-ended generation tasks are often simple, with fewer components to change. This limited diversity of template variations would have constrained our ability to conduct a comprehensive analysis on the prompt sensitivity of LLMs. We chose MCQ datasets because they provide the most effective and controlled testbed for modifying the structure of prompt templates.

Application to open-ended generation. It is crucial to note that not choosing open-ended generation tasks is not a limitation of the interaction framework itself. The framework can be readily applied to open-ended generation tasks, provided there is a ground truth answer for each input. Specifically, we set the output of the DNN as $v(\boldsymbol{x}) = \log\frac{p(y=y^*|\boldsymbol{x})}{1-p(y=y^*|\boldsymbol{x})} \in \mathbb{R}$, where $p(y = y^*|\boldsymbol{x})$ represents the probability of generating the ground truth token y^* given the input $\boldsymbol{x}$, then we can apply this interaction framework to open-ended generation tasks.

E.2 Computing Infrastructure

We conducted all our experiments on four NVIDIA Tesla V100-DGXS GPUs, each with 32 GB of VRAM. The software environment consisted of NVIDIA Driver version 570.133.07 and CUDA 12.8.

To ensure a balance between computational feasibility and model fidelity, we employed mixed-precision loading for all LLMs. For the majority of the evaluated LLMs, we used a torch.float16 data type, which provides a standard level of precision for inference tasks.

E.3 Model Details

We conduct experiments on 50 open-source LLMs from 6 major model families. A comprehensive list of all evaluated models is provided in Table 2. To facilitate a controlled analysis of the factors influencing prompt sensitivity, we group these models into specific subsets for each comparison, as detailed below.

(1) **Instruct/Chat vs. Base Models.** To investigate the impact of the alignment process, we form pairs of instruct/chat models and their corresponding base models. This comparison includes models from the Llama, Mistral, Qwen, InternLM, and Olmo families. The main LLMs used for this comparison are:

- *Llama Family:*
  - `llama-2-7b-chat` vs. `llama-2-7b`
  - `llama-2-13b-chat` vs. `llama-2-13b`
  - `llama-2-70b-chat` vs. `llama-2-70b`
  - `llama-3-8b-instruct` vs. `llama-3-8b`
- *Mistral Family:*
  - `mistral-7b-v0.3-instruct` vs. `mistral-7b-v0.3`
  - `mixtral-8x7b-instruct` vs. `mixtral-8x7b`
- *Qwen Family:*
  - `qwen3-4b-instruct` vs. `qwen3-4b`
  - `qwen3-8b-instruct` vs. `qwen3-8b`
  - `qwen3-14b-instruct` vs. `qwen3-14b`
  - `qwen1.5-moe-a2.7b-chat` vs. `qwen1.5-moe-a2.7b`

  - `qwen3-30b-a3b-instruct` vs. `qwen3-30b-a3b`
- *Olmo Family:*
  - `olmo-7b-instruct` vs. `olmo-7b`
  - `olmoe-7b-instruct` vs. `olmoe-7b`
- *InternLM Family:*
  - `internlm2-chat-7b` vs. `internlm2-7b`

(2) **Dense vs. MoE Models.** To analyze the effect of architecture, we compare dense and Mixture-of-Experts (MoE) models, primarily within the same model family to control for other variables. The main LLMs used for this comparison are:

- *Llama Family:*
  - Dense: `llama-2-7b`, `llama-2-7b-chat`, `llama-2-13b`, `llama-2-13b-chat`, `llama-2-70b`, `llama-2-70b-chat`, `llama-3-8b`, `llama-3-8b-instruct`.
  - MoE: `llama-moe-v1-3_5b-2_8-sft`, `llama-moe-v2-3_8b-2_8-sft`.
- *Mistral Family:*
  - Dense: `mistral-7b-v0.3`, `mistral-7b-v0.3-instruct`.
  - MoE: `mixtral-8x7b`, `mixtral-8x7b-instruct`.
- *Qwen Family:*
  - Dense: `qwen2-7b`, `qwen2-7b-instruct`, `qwen2-72b-instruct`, `qwen2.5-7b-instruct`, `qwen2.5-14b-instruct`, `qwen2.5-32b-instruct`, `qwen2.5-72b-instruct`, `qwen3-4b`, `qwen3-4b-instruct`, `qwen3-8b`, `qwen3-8b-instruct`, `qwen3-14b`, `qwen3-14b-instruct`, `qwen3-32b-instruct`.
  - MoE: `qwen1.5-moe-a2.7b`, `qwen1.5-moe-a2.7b-chat`, `qwen3-30b-a3b`, `qwen3-30b-a3b-instruct`.
- *Olmo Family:*
  - Dense: `olmo-7b`, `olmo-7b-instruct`, `olmo-2-7b-instruct`, `olmo-2-13b-instruct`, `olmo-2-32b-instruct`.
  - MoE: `olmoe-7b`, `olmoe-7b-instruct`.

(3) **Model Scale.** To study the impact of model scale, we analyze a series of LLMs from the same family and with the same training paradigm but with varying parameter counts. The main LLMs used for this comparison are:

- *Llama-2 (Base):* `7b`, `13b`, `70b`.
- *Llama-2 (Chat):* `7b`, `13b`, `70b`.
- *Qwen2 (Instruct):* `0.5b`, `1.5b`, `7b`, `72b`.
- *Qwen2.5 (Instruct):* `0.5b`, `1.5b`, `3b`, `7b`, `14b`, `32b`, `72b`.
- *Qwen3 (Instruct):* `0.6b`, `1.7b`, `4b`, `8b`, `14b`, `32b`.
- *Olmo-2 (Instruct):* `1b`, `7b`, `13b`, `32b`.

E.4 Generation Configuration of LLMs

To ensure reproducible results, we employ a greedy search strategy for all LLMs. This is achieved by setting the "*do_sample*" parameter to "*False*" in our generation configuration. When "*do_sample=False*", the LLM selects the token with the highest probability as the next token in the sequence. By adopting this greedy approach, we eliminate the randomness inherent in sampling-based methods. The configuration ensures that for a given input, the same LLM will generate the exact same output every time, which is a critical requirement for the replicability of our experiments.

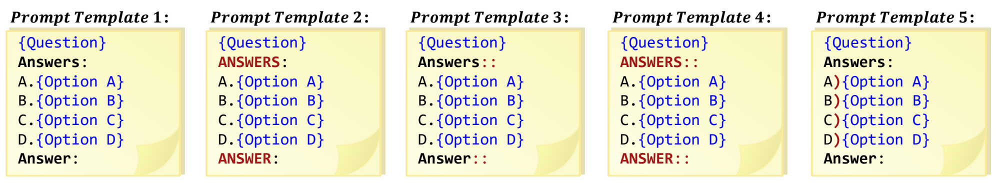
E.5 How to Mask Input Words For Different LLMs

To compute interactions, we follow the approach of Cheng et al. (2025) and mask the words in $N \setminus S$ by replacing them with a LLM-specific [MASK] token. Our selection of this token follows a prioritized strategy: (1) We preferentially use the LLM's designated unknown (`<unk>`) token. (2) If an unknown token is not available or suitable, we use the padding (`<pad>`) token as a fallback. Since the specific token strings and their corresponding IDs vary across different LLMs, the exact mask token used for each LLM is detailed below:

- For `llama-2-7b`, `llama-2-7b-chat`, `llama-moe-v1-3.5b-2.8-sft`, `mistral-7b-v0.3`, `mistral-7b-v0.3-instruct`, `mixtral-8x7b`, `mixtral-8x7b-instruct`, `internlm2-7b`, `internlm2-chat-7b`, we use the `<unk>` token (ID: `0`) to mask words.
- For `llama-2-13b`, `llama-2-13b-chat`, `llama-2-70b`, `llama-2-70b-chat`, we use the `<|pad_token|>` token (ID: `0`) to mask words.
- For `llama-3-8b`, `llama-3-8b-instruct`, we use the `<|pad_token|>/<|reserved_special_token_250|>` token (ID: `128255`) to mask words.
- For `llama-moe-v2-3.8b-2.8-sft`, we use the `<|pad_token|>/<|eot_id|>` token (ID: `128009`) to mask words.
- For `qwen2-7b`, we use the `<|PAD_TOKEN|>` token (ID: `151646`) to mask words.
- For a large group of Qwen models, including `qwen2-0.5b-instruct`, `qwen2-1.5b-instruct`, `qwen2-7b-instruct`, `qwen2-72b-instruct`, `qwen2.5` series, `qwen3-0.6b-instruct`, `qwen3-1.7b-instruct`, `qwen3-4b` series, `qwen3-32b-instruct`, `qwen1.5-moe` series, and `qwen3-30b-a3b` series, we use the `<|pad_token|>/<|endoftext|>` token (ID: `151643`) to mask words.
- For `qwen3-8b`, `qwen3-8b-instruct`, `qwen3-14b`, `qwen3-14b-instruct`, we use the `<|pad_token|>/<|vision_pad|>` token (ID: `151654`) to mask words.
- For the Olmo V1 series models, including `olmo-1b`, `olmo-7b`, `olmo-7b-instruct`, `olmoe-7b`, and `olmoe-7b-instruct`, we use the `<|padding|>` token (ID: `1`) to mask words.
- For the Olmo V2 series models, including `olmo-2-1b-instruct`, `olmo-2-7b-instruct`, `olmo-2-13b-instruct`, and `olmo-2-32b-instruct`, we use the `<|pad_token|>/<|endoftext|>` token (ID: `100257`) to mask words.

E.6 Prompt Templates

To systematically evaluate the prompt sensitivity of LLMs, we designed a set of five distinct prompt templates. As illustrated in Figure 13, these templates are derived from a base prompt template (i.e., Prompt Template 1) through a series of subtle, semantically irrelevant modifications. These variations include changes in letter case, e.g., "Answers" vs. "ANSWERS" and alterations to separators, e.g., ":" vs. "::" or the format of option markers, e.g., "A." vs. "A)". Crucially, these changes only affect the superficial formatting while preserving the core semantic meaning of the prompt template.



***Prompt Template* 1:**
```
{Question}
Answers:
A.{Option A}
B.{Option B}
C.{Option C}
D.{Option D}
Answer:
```

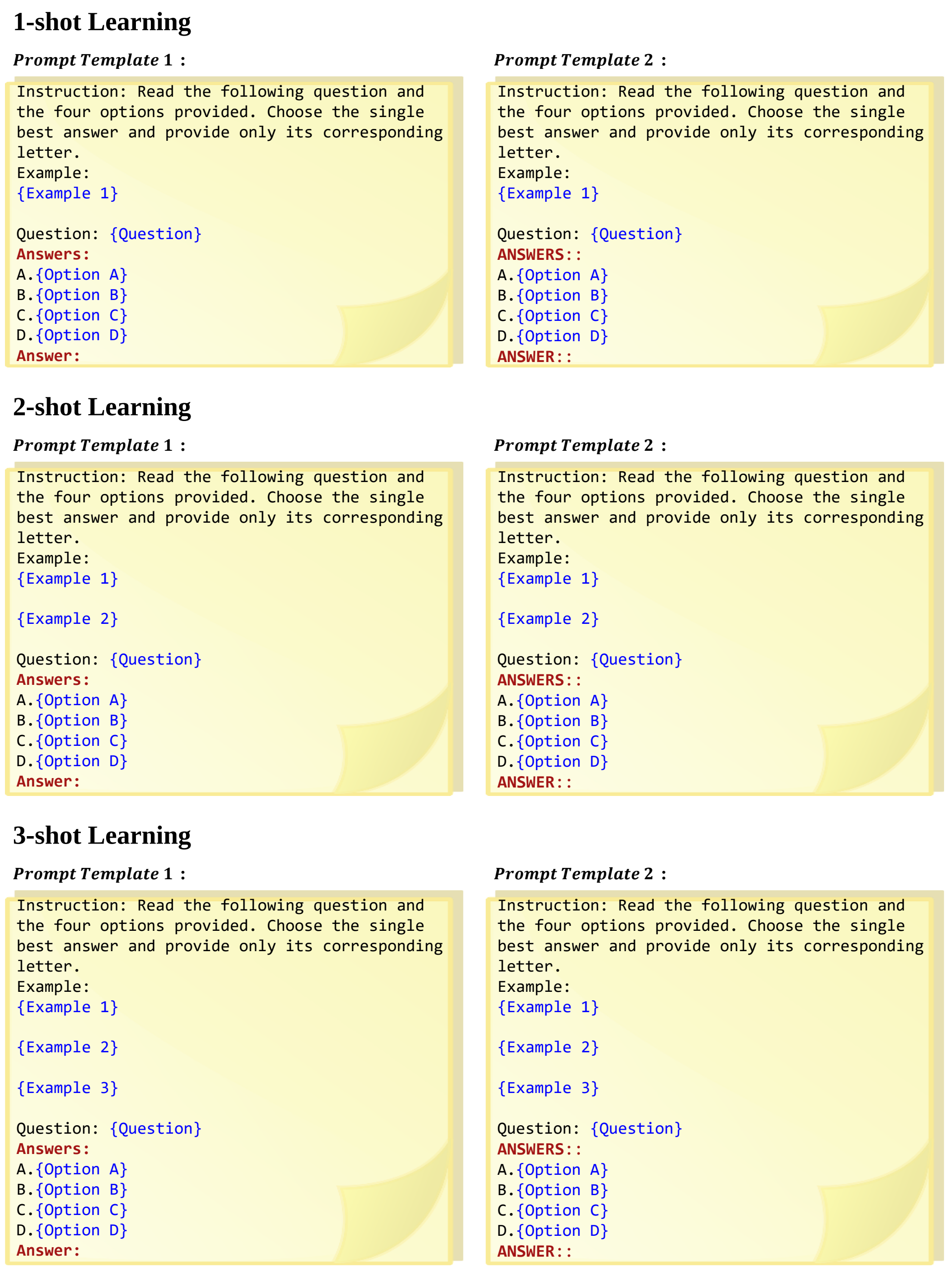
***Prompt Template* 2:**
```
{Question}
ANSWERS:
A.{Option A}
B.{Option B}
C.{Option C}
D.{Option D}
ANSWER:
```

***Prompt Template* 3:**
```
{Question}
Answers::
A.{Option A}
B.{Option B}
C.{Option C}
D.{Option D}
Answer::
```

***Prompt Template* 4:**
```
{Question}
ANSWERS::
A.{Option A}
B.{Option B}
C.{Option C}
D.{Option D}
ANSWER::
```

***Prompt Template* 5:**
```
{Question}
Answers:
A){Option A}
B){Option B}
C){Option C}
D){Option D}
Answer:
```

Figure 13: Different prompt templates. Red parts show the difference between the current prompt template with the first prompt template, *i.e.*, Prompt Template 1.



1-shot Learning

***Prompt Template* 1 :**

```
Instruction: Read the following question and
the four options provided. Choose the single
best answer and provide only its corresponding
letter.
Example:
{Example 1}

Question: {Question}
Answers:
A.{Option A}
B.{Option B}
C.{Option C}
D.{Option D}
Answer:
```

***Prompt Template* 2 :**

```
Instruction: Read the following question and
the four options provided. Choose the single
best answer and provide only its corresponding
letter.
Example:
{Example 1}

Question: {Question}
ANSWERS::
A.{Option A}
B.{Option B}
C.{Option C}
D.{Option D}
ANSWER::
```

2-shot Learning

***Prompt Template* 1 :**

```
Instruction: Read the following question and
the four options provided. Choose the single
best answer and provide only its corresponding
letter.
Example:
{Example 1}

{Example 2}

Question: {Question}
Answers:
A.{Option A}
B.{Option B}
C.{Option C}
D.{Option D}
Answer:
```

***Prompt Template* 2 :**

```
Instruction: Read the following question and
the four options provided. Choose the single
best answer and provide only its corresponding
letter.
Example:
{Example 1}

{Example 2}

Question: {Question}
ANSWERS::
A.{Option A}
B.{Option B}
C.{Option C}
D.{Option D}
ANSWER::
```

3-shot Learning

***Prompt Template* 1 :**

```
Instruction: Read the following question and
the four options provided. Choose the single
best answer and provide only its corresponding
letter.
Example:
{Example 1}

{Example 2}

{Example 3}

Question: {Question}
Answers:
A.{Option A}
B.{Option B}
C.{Option C}
D.{Option D}
Answer:
```

***Prompt Template* 2 :**

```
Instruction: Read the following question and
the four options provided. Choose the single
best answer and provide only its corresponding
letter.
Example:
{Example 1}

{Example 2}

{Example 3}

Question: {Question}
ANSWERS::
A.{Option A}
B.{Option B}
C.{Option C}
D.{Option D}
ANSWER::
```

Figure 14: Prompt templates of few-shot learning.

In our experimental procedure, for a given input, which consists of a question and options, we apply each of the five prompt templates to generate five prompts. For every unique pair of these five prompts, we then calculate the prompt sensitivity by quantifying the change in the interactions among the input variables (*i.e.*, words within the question and options). This procedure allows us to precisely measure how much the LLM's interaction patterns of the core input are perturbed by superficial changes in the prompt template, thus evaluating the prompt sensitivity of LLMs.

E.7 Few-shot Learning Templates

For this experiment, we selected the pair of prompt templates that exhibited the highest average prompt sensitivity in the 0-shot setting, aiming to test if few-shot learning could help the most severe situation. To investigate whether few-shot learning can mitigate high prompt sensitivity, we conducted a follow-up experiment. We selected **Prompt Template 1** and **Prompt Template 4** from Figure 13 for this analysis, as this pair exhibited the highest average prompt sensitivity in our 0-shot setting. This allowed us to test the efficacy of few-shot learning in the most challenging scenario.

Based on these two base templates, we constructed few-shot learning prompts with one, two, and three in-context examples (*i.e.*, 1-shot, 2-shot, and 3-shot learning), as illustrated in Figure 14. The examples were formulated using certain questions and their corresponding answers, randomly selected from a set of datasets that are not included in the test set. The structure of each example is related to its corresponding prompt template. For instance, the first example (`Example 1`) is formatted differently for each template:

- **Example 1 For Prompt Template 1:**

```
Question: Which type of precipitation consists of frozen
   rain drops?
Answers:
A.sleet
B.hail
C.snow
D.fog
Answer: A
```

- **Example 1 For Prompt Template 4 (Note the different format):**

```
Question: Which type of precipitation consists of frozen
   rain drops?
ANSWERS::
A.sleet
B.hail
C.snow
D.fog
ANSWER:: A
```

The other two examples (`Example 2` and `Example 3`) are presented below:

- **Example 2 For Prompt Template 1:**

```
Question: Decayed prehistoric plants have helped in the
   formation of
Answers:
A.coal, shale, and quartz.
B.coal, oil, and gas.
C.shale, quartz, and coal.
D.oil, shale, and granite.
Answer: B
```

- **Example 2 For Prompt Template 4:**

```
Question: Decayed prehistoric plants have helped in the
   formation of
ANSWERS::
A.coal, shale, and quartz.
B.coal, oil, and gas.
C.shale, quartz, and coal.
D.oil, shale, and granite.
ANSWER:: B
```

- **Example 3 For Prompt Template 1:**

```
Question: Which describes a material that is not a food?
Answers:
```

```
A.It stores energy but not nutrients.
B.It does not store energy or nutrients.
C.It stores energy and nutrients.
D.It does not store energy but stores nutrients.
Answer: B
```

- **Example 3 For Prompt Template 4:**

```
Question: Which describes a material that is not a food?
ANSWERS::
A.It stores energy but not nutrients.
B.It does not store energy or nutrients.
C.It stores energy and nutrients.
D.It does not store energy but stores nutrients.
ANSWER:: B
```

F More Experimental Results

F.1 More Results on the Verification of the Sparsity of Interactions

Here are more results on the verification of the sparsity of interactions. As illustrated in Figure 15, the results verify that only a small set of interactions have salient effects, while most of the interactions have negligible effects and can be considered as noise patterns.

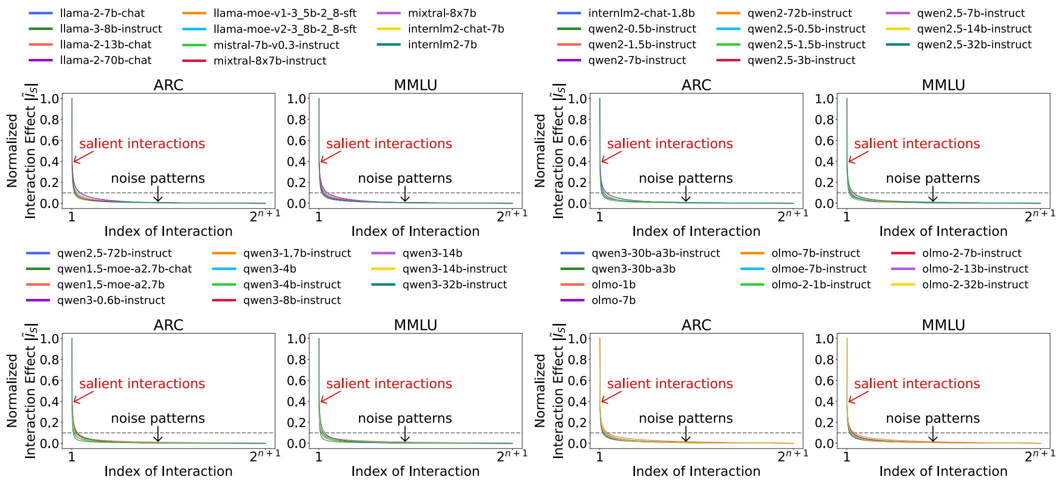


Figure 15: Verifying the sparsity of interactions. We show absolute values of normalized interactions in a descending order. LLMs all encode a small number of salient interactions, while most of the interaction effects are negligible.

F.2 More Results on the Verification of the Sparsity of Interactions

Here are more results on the verification of quality of universal matching. Figure 16 compares the LLM's true output $v(\boldsymbol{x}_T)$ for all masked inputs against the logical model using only the most salient interactions. Even when using just the top 3% or top 5% of all interactions, the matching error is minimal. This empirically demonstrates that the LLM's output can be faithfully approximated by a small, sparse set of salient interactions.

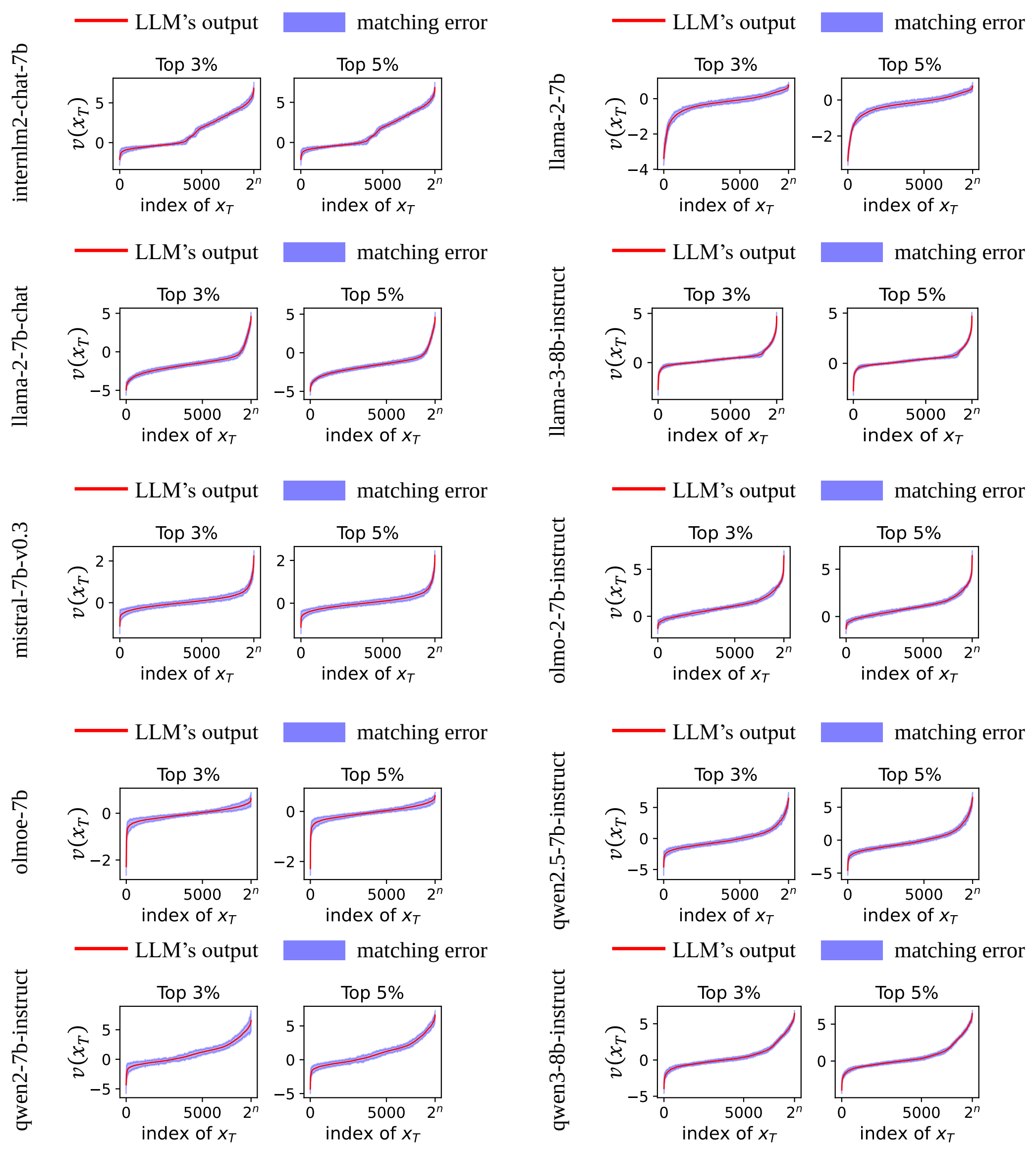


Figure 16: Verifying the quality of universal matching for any 2^n masked inputs. The red line plots outputs of the LLM in an ascending order.

F.3 Detailed case study

Figure 17 is the detailed case study of how to use our interaction-based analytical tool. It offers preliminary evidence that semantically irrelevant alterations to the prompt template can lead to significant changes in the salient interaction patterns, even when the input and output remains unchanged. This reveals the existence of unstable interactions, which we propose as the underlying cause of prompt sensitivity.

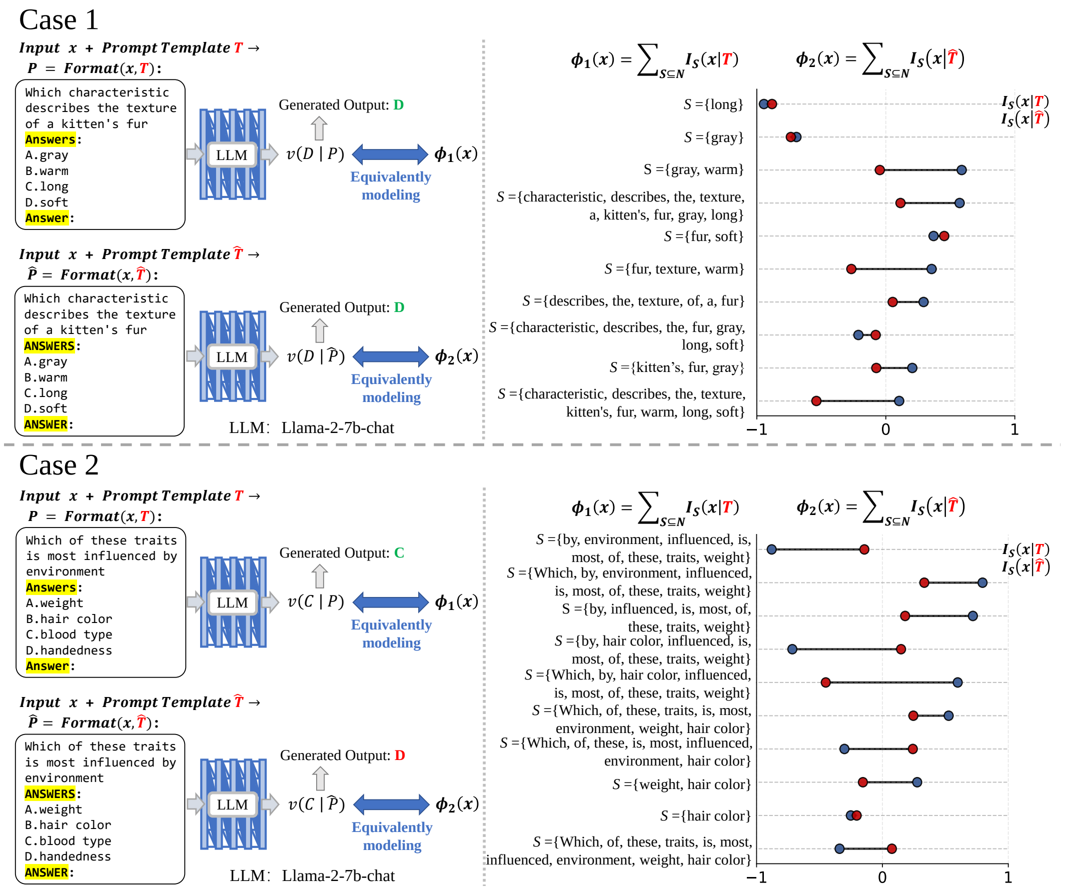

Case 1
Input x + Prompt Template T →
P = Format(x,T):
Which characteristic describes the texture of a kitten's fur
Answers:
A.gray
B.warm
C.long
D.soft
Answer:
LLM
Generated Output: D
v(D | P)
ϕ1(x)
Equivalently modeling
Input x + Prompt Template T̂ →
P̂ = Format(x,T̂):
ANSWERS:
ANSWER:
Generated Output: D
v(D | P̂)
ϕ2(x)
LLM: Llama-2-7b-chat
ϕ1(x) = ∑S⊆N IS(x|T)
ϕ2(x) = ∑S⊆N IS(x|T̂)
IS(x|T)
IS(x|T̂)
S ={long}
S ={gray}
S ={gray, warm}
S ={characteristic, describes, the, texture, a, kitten's, fur, gray, long}
S ={fur, soft}
S ={fur, texture, warm}
S ={describes, the, texture, of, a, fur}
S ={characteristic, describes, the, fur, gray, long, soft}
S ={kitten's, fur, gray}
S ={characteristic, describes, the, texture, kitten's, fur, warm, long, soft}
−1
0
1
Case 2
Which of these traits is most influenced by environment
Answers:
A.weight
B.hair color
C.blood type
D.handedness
Answer:
Generated Output: C
v(C | P)
ANSWERS:
ANSWER:
Generated Output: D
v(C | P̂)
S ={by, environment, influenced, is, most, of, these, traits, weight}
S ={Which, by, environment, influenced, is, most, of, these, traits, weight}
S ={by, influenced, is, most, of, these, traits, weight}
S ={by, hair color, influenced, is, most, of, these, traits, weight}
S ={Which, by, hair color, influenced, is, most, of, these, traits, weight}
S ={Which, of, these, traits, is, most, environment, weight, hair color}
S ={Which, of, these, is, most, influenced, environment, hair color}
S ={weight, hair color}
S ={hair color}
S ={Which, of, these, traits, is, most, influenced, environment, weight, hair color}

Figure 17: A case study of interaction-level analysis revealing latent instability. The same input $\boldsymbol{x}$ is formatted with two semantically identical templates, T and $\hat{T}$, differing only in letter case (e.g.,"Answer" vs. "ANSWER"). Although the LLM generates the same correct output ("D") in both cases, the composition of the interaction-based logical model $\phi(\boldsymbol{x})$ reveals significant internal divergence. Many interaction effects are highly unstable, changing in either sign or magnitude. This highlights a critical risk of prompt sensitivity that is invisible to output-leve

F.4 More Results on the Prompt Sensitivity of Different Orders

Here are more results on the prompt sensitivity of different orders on the ARC dataset. As illustrated in Figure 18, it shows that the prompt sensitivity of low-order interactions is the lowest, followed by mid-order, while high-order interactions exhibit the highest prompt sensitivity. This indicates that low-order interactions encoded by LLMs are highly stable when faced with subtle changes to prompt templates, *i.e.*, simple interaction patterns are more robust. Conversely, the high sensitivity of high-order interactions reveals that the LLMs' internal representation of complex patterns is highly unstable.

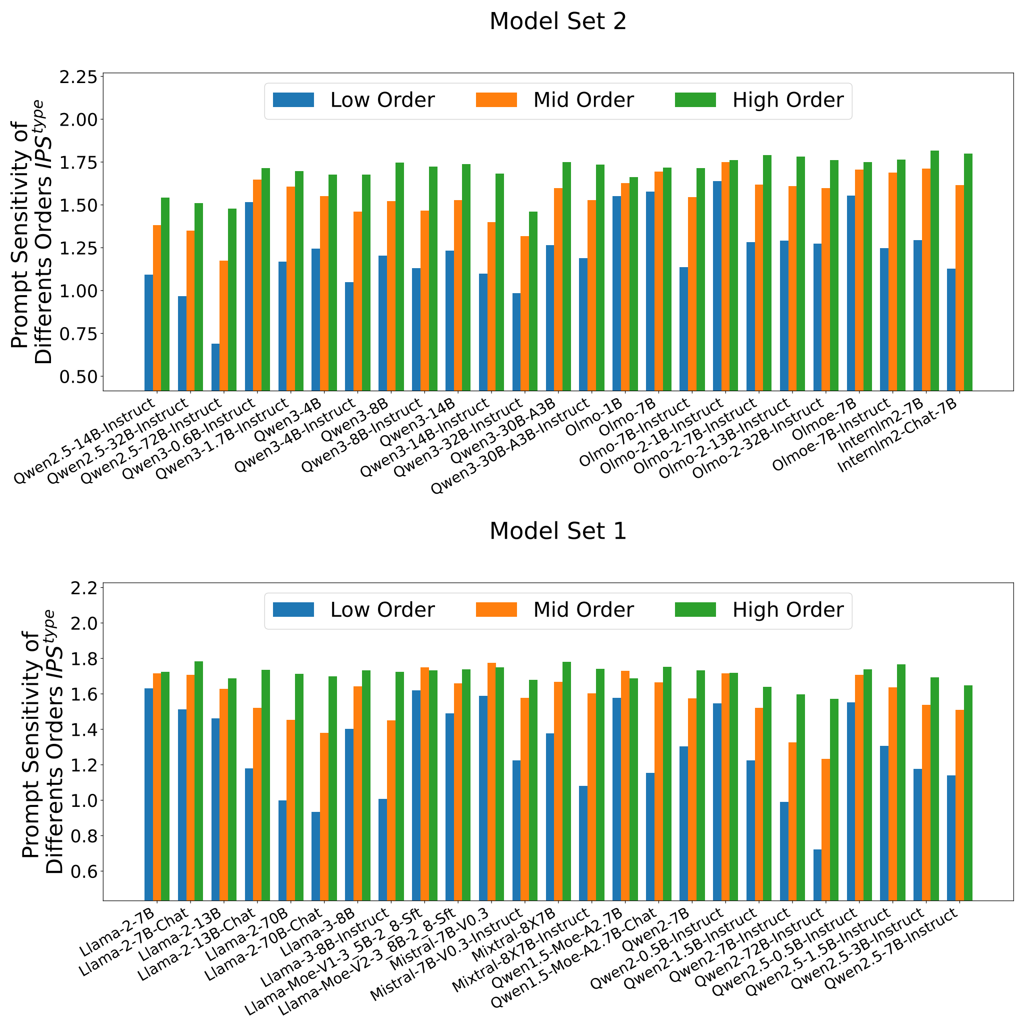


Figure 18: A comparison of the prompt sensitivity of three order types. Results show that low-order interactions are the least sensitive, while high-order interactions are the most sensitive.

F.5 More Results on the Prompt Sensitivity of Different Orders for Each Individual LLM when Applying Few-Shot Learning

As illustrated in Figure 19, we provide a detailed, per-model analysis of how few-shot learning impacts prompt sensitivity of different interaction orders. A consistent and powerful trend emerges across nearly all individual LLMs: the combination of in-context examples reduces the prompt sensitivity of all three levels of interaction order: low-, mid-, and high-order. The most substantial decrease is consistently observed in low-order sensitivity. Mid-order sensitivity follows with a significant, albeit smaller, reduction. While high-order interactions see the most modest decline, the downward trend is still clear and consistent for most models. This per-model analysis reinforces our earlier finding and demonstrates the exceptional effect of few-shot learning for reducing the prompt sensitivity of LLMs.

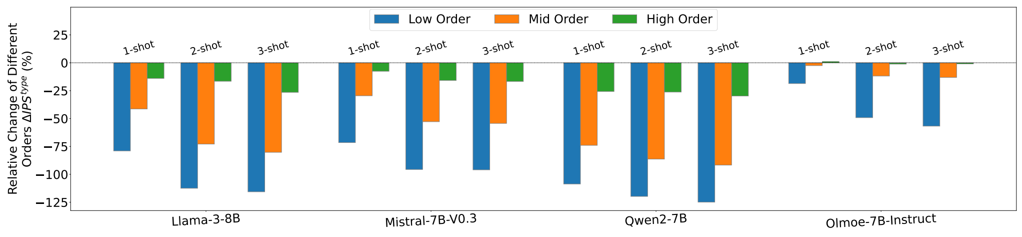


Figure 19: A comparison of prompt sensitivity of low-, mid-, and high-order interactions between 0-shot learning and few-shot learning. Prompt sensitivity at all three order levels shows an clear drop when applying few-shot learning.

F.6 Hyperparameter Experiments of the Threshold τ

In the full paper, the threshold τ is set as 0.1. We do hyperparameter experiments for $\tau = 0.15$ and $\tau = 0.05$. Results show that the conclusions remain the same for different τ

Here are the results for $\tau = 0.15$.

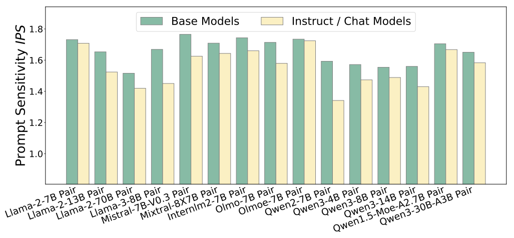


Figure 20: A comparison of the prompt sensitivity between instruct/chat models and base models. Results show that instruct/chat models are less sensitive than corresponding base models.

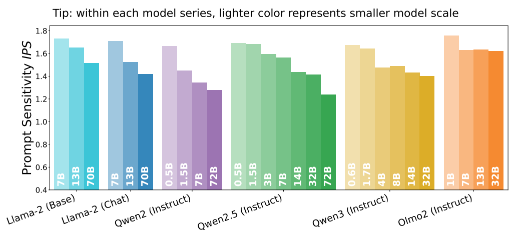


Figure 21: A comparison of prompt sensitivity across different model scales. As the model scale increases, the prompt sensitivity within a model series systematically decreases.

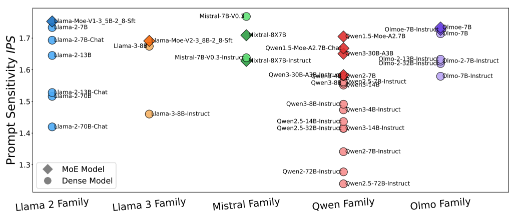


Figure 22: A Comparison of prompt sensitivity between MoE models and dense models. Generally, MoE models tend to be more sensitive than dense models in the same model family.

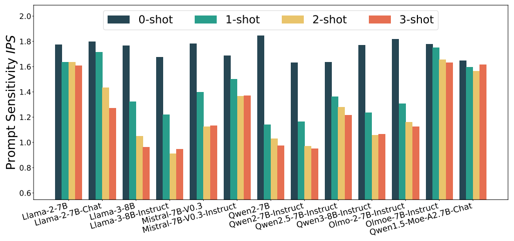


Figure 23: A comparison of prompt sensitivity between 0-shot learning and few-shot learning. The drop in prompt sensitivity is substantial from 0-shot to 1-shot.

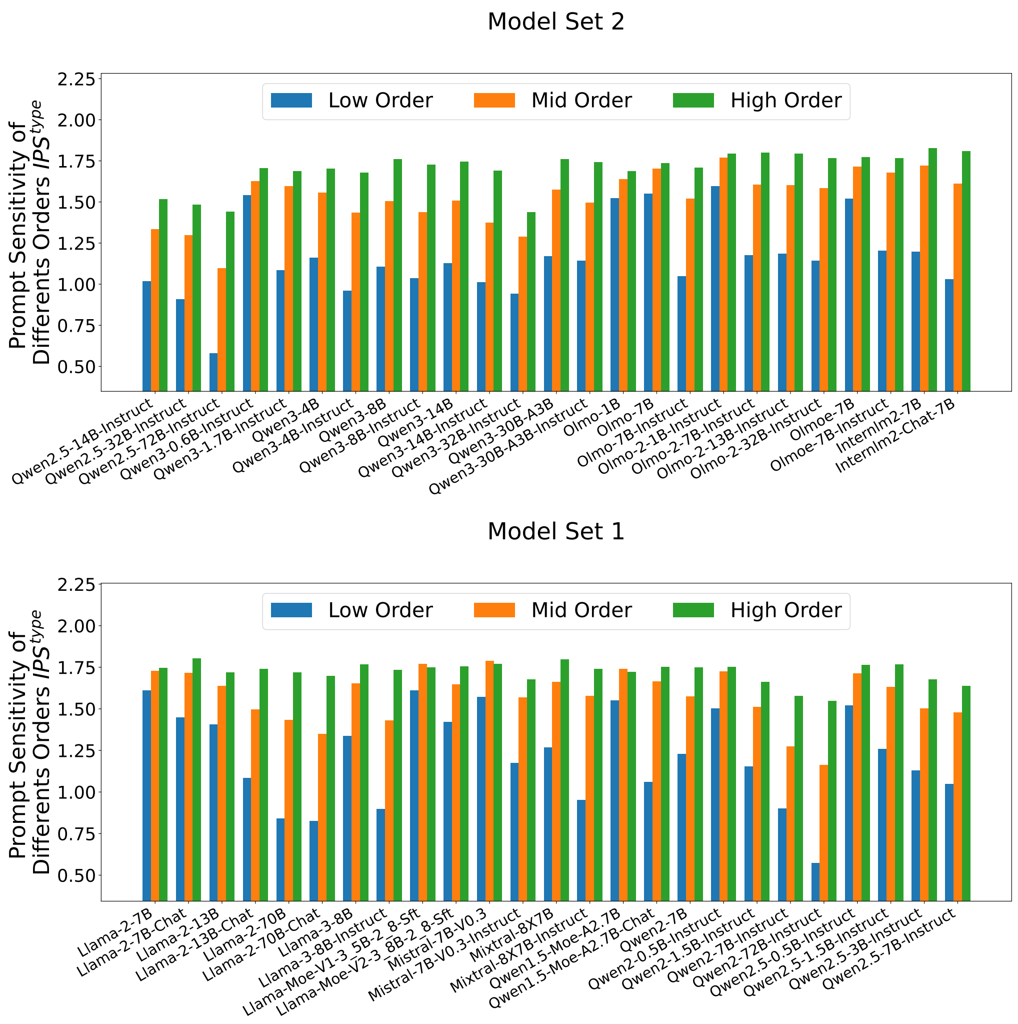


Figure 24: A comparison of the prompt sensitivity of three order types. Results show that low-order interactions are the least sensitive, while high-order interactions are the most sensitive.

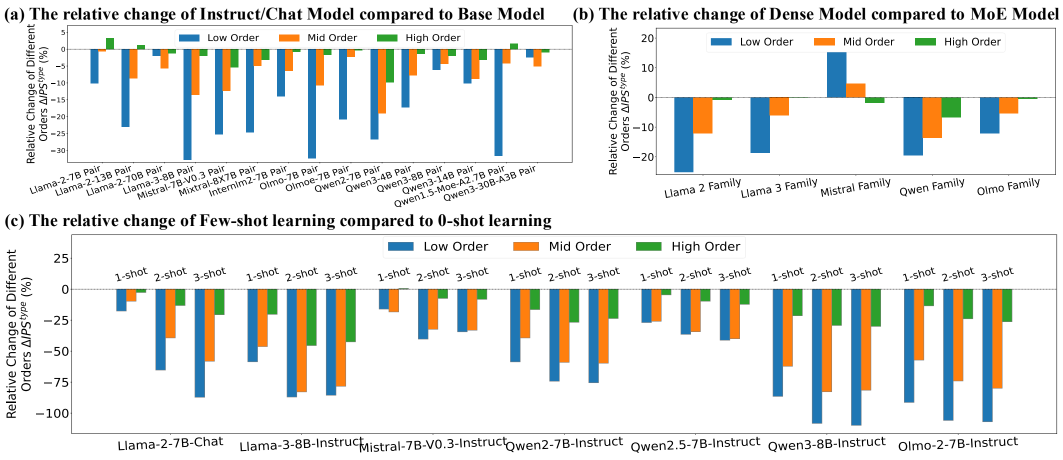


Figure 25: Comparing the relative change in the prompt sensitivity of low-, mid-, and high-order interactions for different factors.

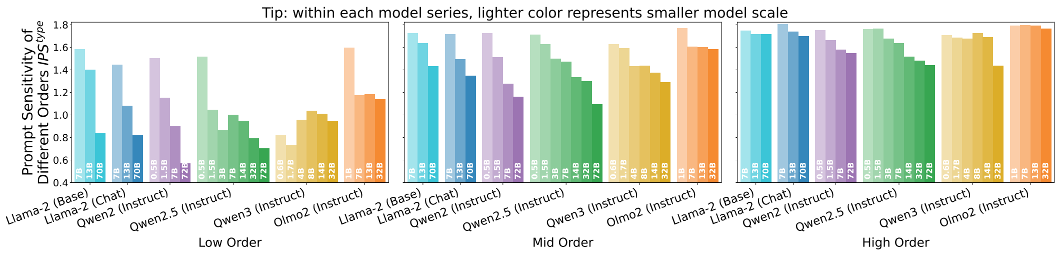


Figure 26: A comparison of prompt sensitivity at the order-level across different model scales.

Here are the results for $\tau = 0.05$.

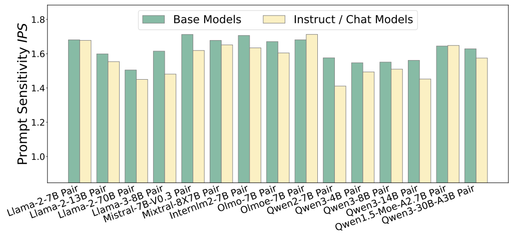


Figure 27: A comparison of the prompt sensitivity between instruct/chat models and base models. Results show that instruct/chat models are less sensitive than corresponding base models.

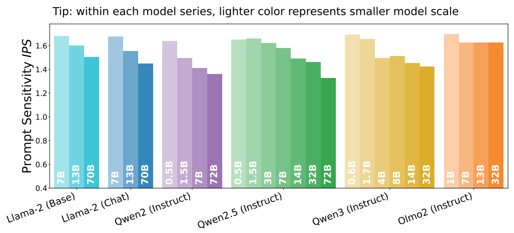


Figure 28: A comparison of prompt sensitivity across different model scales. As the model scale increases, the prompt sensitivity within a model series systematically decreases.

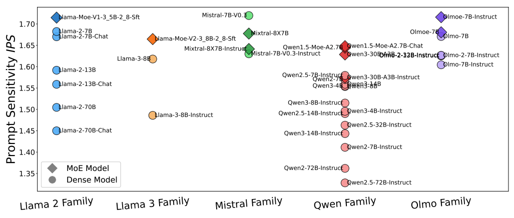


Figure 29: A Comparison of prompt sensitivity between MoE models and dense models. Generally, MoE models tend to be more sensitive than dense models in the same model family.

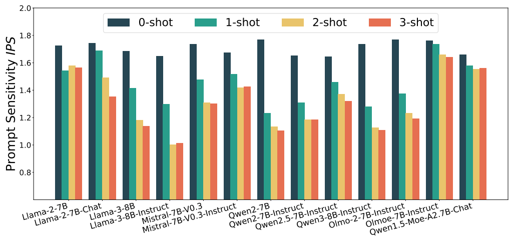


Figure 30: A comparison of prompt sensitivity between 0-shot learning and few-shot learning. The drop in prompt sensitivity is substantial from 0-shot to 1-shot.

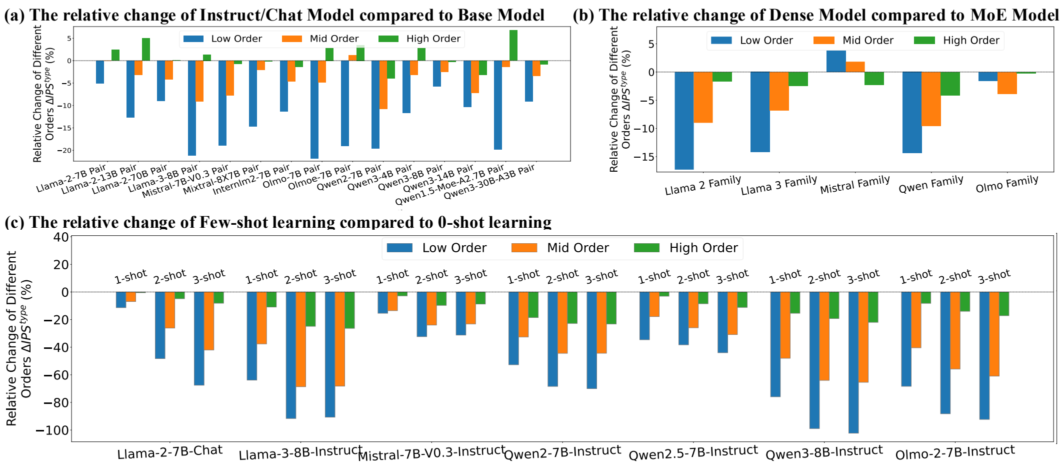


Figure 31: Comparing the relative change in the prompt sensitivity of low-, mid-, and high-order interactions for different factors.

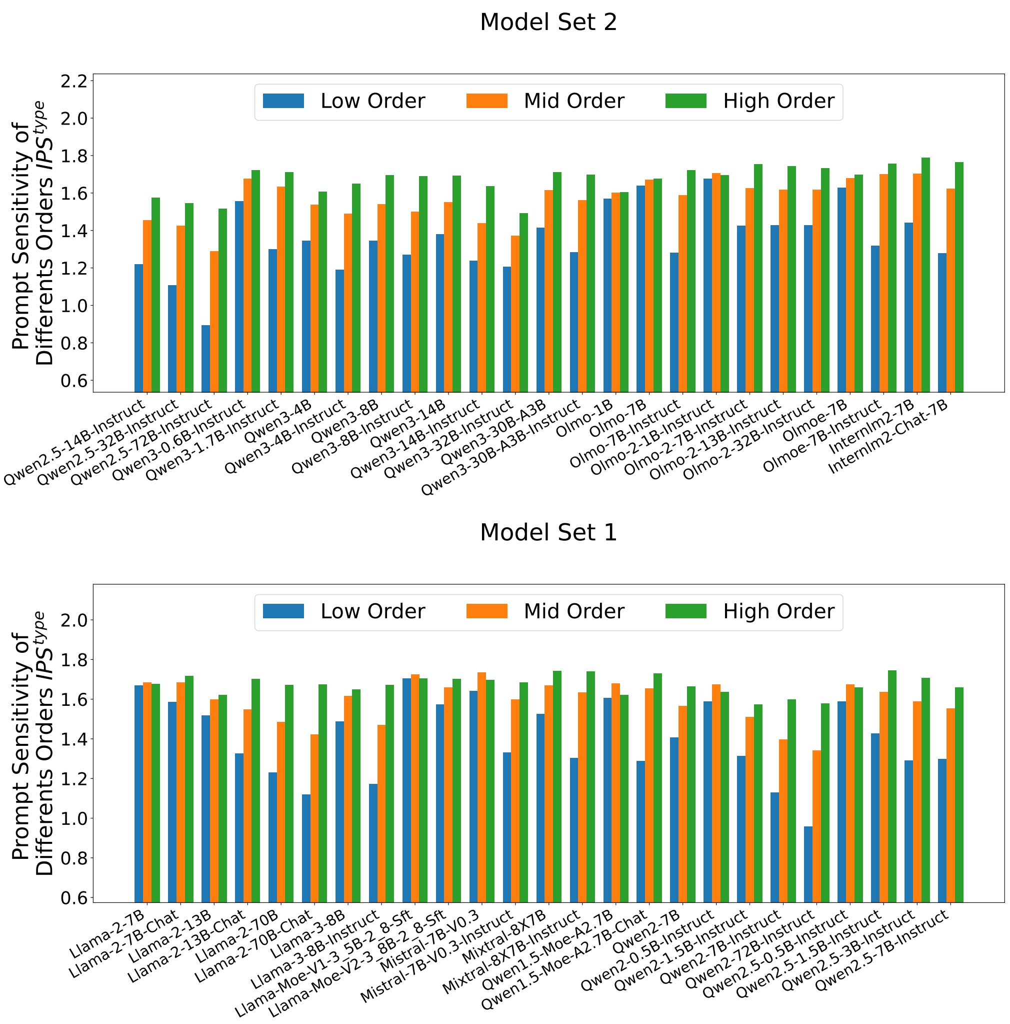


Figure 32: A comparison of the prompt sensitivity of three order types. Results show that low-order interactions are the least sensitive, while high-order interactions are the most sensitive.

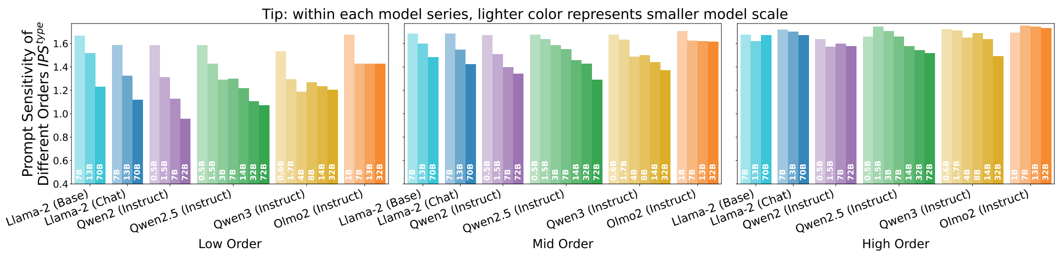


Figure 33: A comparison of prompt sensitivity at the order-level across different model scales.

F.7 Results on MMLU Dataset

Here are the results on the MMLU dataset and τ is set to 0.1, we can observe the same conclusions on this dataset.

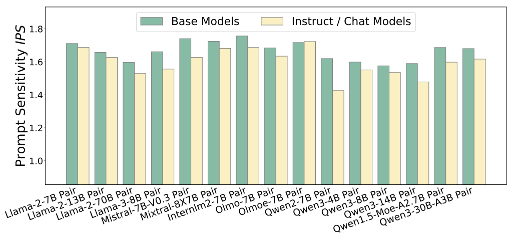


Figure 34: A comparison of the prompt sensitivity between instruct/chat models and base models. Results show that instruct/chat models are less sensitive than corresponding base models.

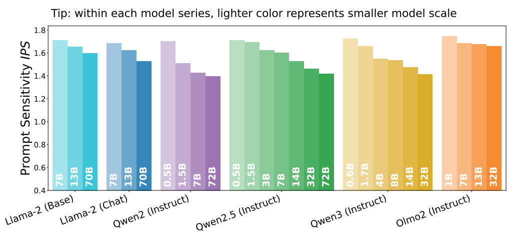


Figure 35: A comparison of prompt sensitivity across different model scales. As the model scale increases, the prompt sensitivity within a model series systematically decreases.

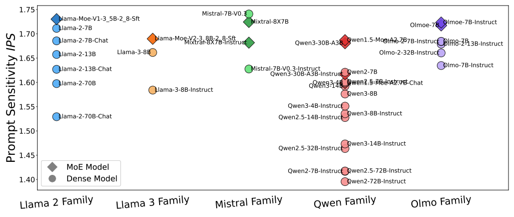


Figure 36: A Comparison of prompt sensitivity between MoE models and dense models. Generally, MoE models tend to be more sensitive than dense models in the same model family.

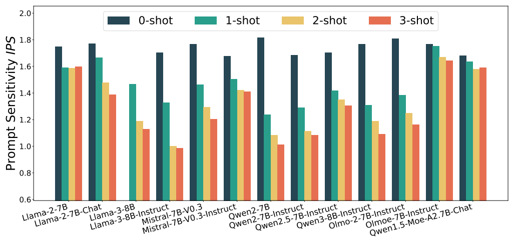


Figure 37: A comparison of prompt sensitivity between 0-shot learning and few-shot learning. The drop in prompt sensitivity is substantial from 0-shot to 1-shot.

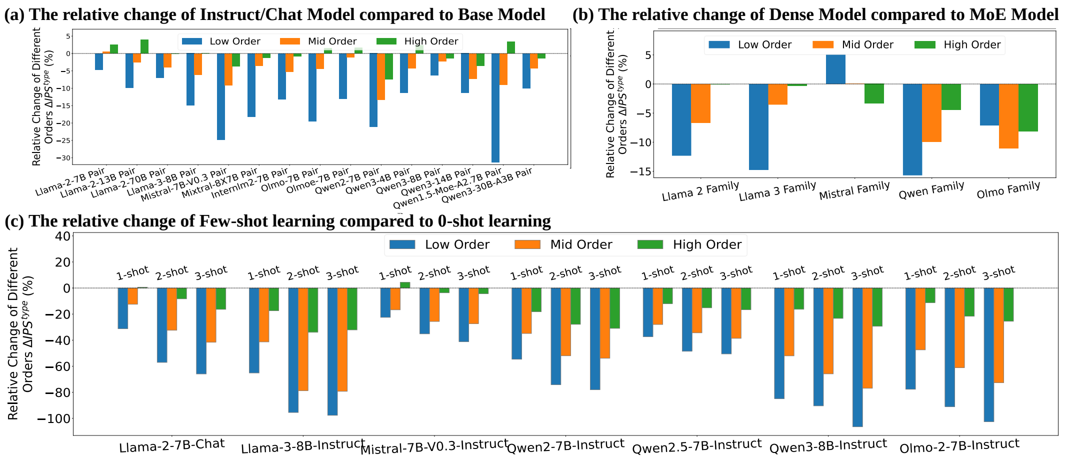


Figure 38: Comparing the relative change in the prompt sensitivity of low-, mid-, and high-order interactions for different factors.

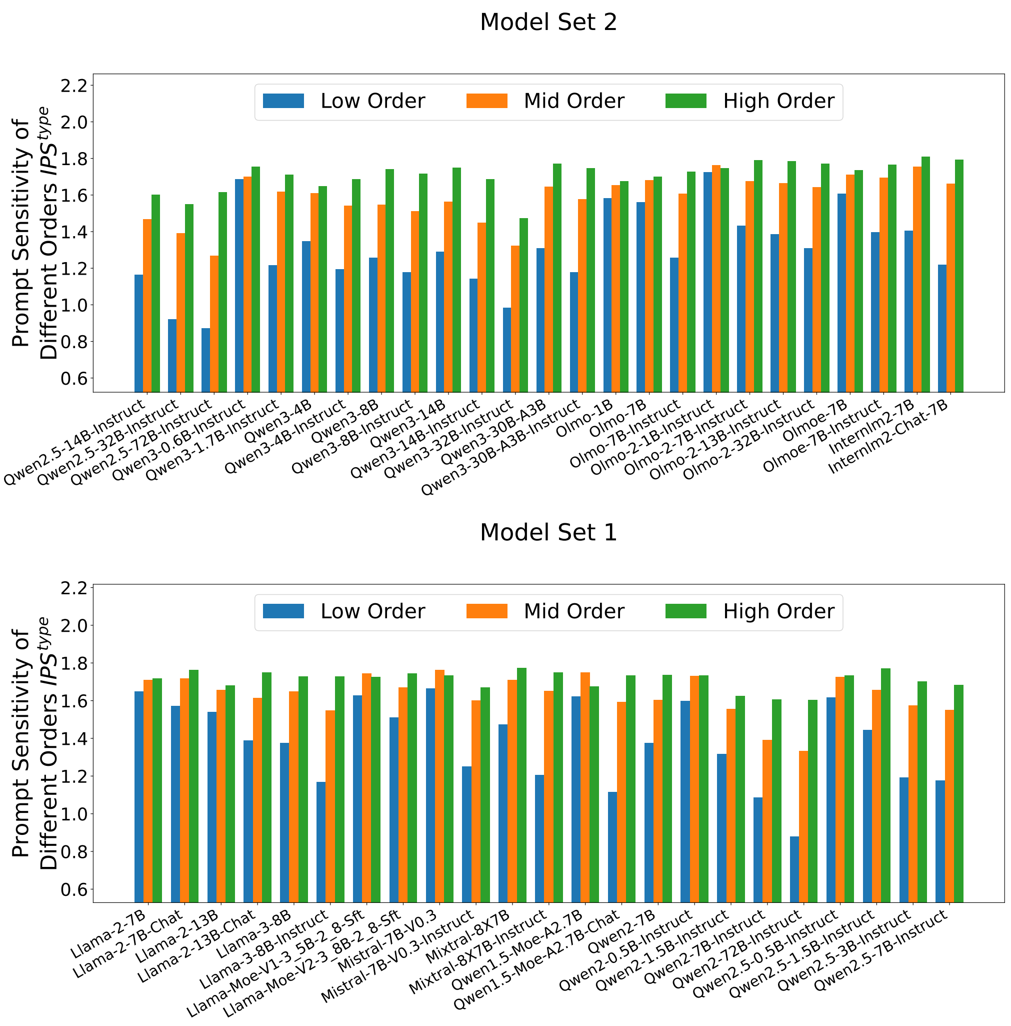


Figure 39: A comparison of the prompt sensitivity of three order types. Results show that low-order interactions are the least sensitive, while high-order interactions are the most sensitive.

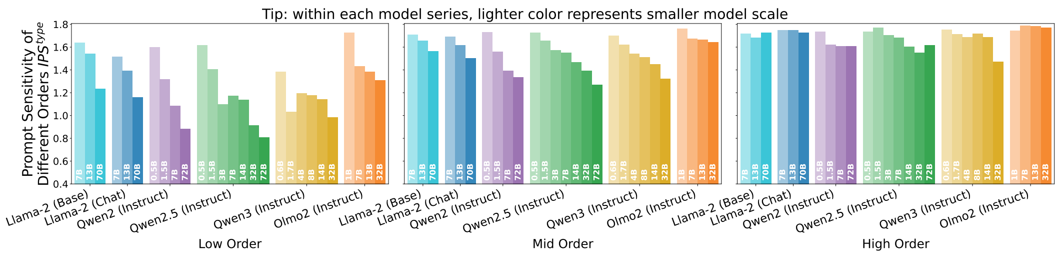


Figure 40: A comparison of prompt sensitivity at the order-level across different model scales.

G Prompt Sensitivity of altering tokens vs. adding tokens

Disaggregating prompt sensitivity by the type of perturbation offers deeper insights into model behavior. Following this direction, we conduct a fine-grained decomposition of our experimental results, comparing prompt sensitivity of two distinct categories to prompt alterations:

1. **Altering tokens:** This involves modifying the capitalization of words, such as from `"Answer"` to `"ANSWER"`.
2. **Adding tokens:** This involves adding symbolic components, for instance, changing a colon from `":"` to `"::"`.

Dense Model	**Altering Tokens**	**Adding Tokens**	**Difference**
internlm2-chat-7b	1.673	**1.676**	+0.003
llama-2-13b-chat	**1.520**	1.433	-0.087
llama-2-7b-chat	1.633	**1.710**	+0.077
llama-3-8b-instruct	1.450	**1.467**	+0.017
mistral-7b-v0.3-instruct	1.538	**1.635**	+0.097
olmo-2-7b-instruct	1.618	**1.655**	+0.037
olmo-7b-instruct	1.462	**1.496**	+0.034
qwen2-7b-instruct	1.366	**1.404**	+0.038
qwen2.5-7b-instruct	1.558	**1.567**	+0.009
qwen3-8b-instruct	1.505	**1.550**	+0.045

Table 3: IPS Scores of Dense Models on Different Perturbation Types. Higher scores indicate greater sensitivity. The more sensitive perturbation type for each model is highlighted in bold.

MoE Model	**Altering Tokens**	**Adding Tokens**	**Difference**
llama-moe-v1-3.5b-sft	**1.736**	1.724	-0.012
llama-moe-v2-3.8b-sft	**1.698**	1.655	-0.043
mixtral-8x7b-instruct	**1.658**	1.617	-0.041
olmoe-7b-instruct	**1.725**	1.699	-0.026
qwen1.5-moe-a2.7b-chat	**1.669**	1.634	-0.035
qwen3-30b-a3b-instruct	1.608	**1.638**	+0.030

Table 4: IPS Scores of MoE Models on Different Perturbation Types. The more sensitive perturbation type for each model is highlighted in bold.

Our results highlight a clear architectural divide: **(1) Dense models are more sensitive to adding tokens.** As shown in Table 3, 9 out of the 10 analyzed dense models exhibit greater sensitivity to the ”adding tokens” category. This suggests a strong, consistent trend where changes to the template's structure have a more pronounced impact on the internal interactions of dense architectures. **(2) MoE models are more sensitive to altering tokens.** In stark contrast, Table 4 shows that 5 out of the 6 MoE models are more sensitive to ”altering tokens”. This consistent pattern suggests that MoE architectures are more susceptible to variations in the change of word capitalization.

H Limitations and Future Work

The limitation of our current study lies in the computational cost of the interaction framework. The method's complexity scales exponentially with the number of input variables n, as it requires evaluating 2^n masked inputs. However, applying this method to very long text inputs would demand a high computational load.

Future work can address this scalability challenge through several promising avenues. These strategies aim to reduce the effective number of input variables without fundamentally changing the faithfulness of the analysis:

(1) **Selective Input Variable Analysis.** One approach is to analyze only a subset of informative input variables (*i.e.*, words) while treating uninformative ones (e.g., stop words) as fixed background context. Previous research has demonstrated that this selection does not significantly impair the faithfulness of the interaction framework (Chen et al., 2024).

(2) **Phrase-level Aggregation.** Instead of analyzing individual words, we can operate at a coarser perspective by merging related words into combined phrasal units. This reduces the total number of input variables while preserving key semantic meaning.

(3) **Approximation Methods.** The computational burden can also be alleviated by exploring approximation techniques. For instance, we propose adopting the **Sparse Möbius Transform (SMT)** algorithm (Kang et al., 2024). This method is directly applicable in our framework because the "interactions" we calculate are mathematically equivalent to the coefficients of the Möbius transform. While our original method needs to calculate the exhaustive 2^n interactions by filtering with a threshold τ, SMT achieves the objective of identifying the top-K salient interactions by directly selecting them, thereby bypassing the exhaustive 2^n calculation. This approach relies on the condition that the distribution of interactions is **sparse**, which aligns perfectly with our paper's empirical findings showing that only a small subset of interactions have salient effects.

In terms of methodology, SMT employs **structured subsampling**—grounded in group testing theory—to intentionally "alias" interactions into a reduced number of bins. Subsequently, a **peeling decoder** is utilized to efficiently disentangle these summed values and isolate the salient interactions. Consequently, this approach drastically reduces the computational complexity from exponential 2^n to $O(Kt\log n)$. Within the SMT framework, K serves as a tunable hyperparameter representing the expected sparsity (or the budget of salient interactions to be recovered), while t denotes the maximum interaction order. Given the low-degree nature of interactions observed in many real-world datasets (Kang et al., 2024), t can be set to a relatively small constant (e.g., $5 \leq t \leq 20$). Thus, the computational complexity only has a linear relationship with n, rendering the fine-grained analysis of long texts computationally feasible and rigorous. Furthermore, empirical evidence suggests that interactions generated via the sparse Möbius transform maintain a level of faithfulness comparable to that of the exhaustive method.

These strategies represent promising directions for extending the powerful capabilities of interaction-based analysis to a wider range of long-text NLP tasks.

I Details of the experiments on open-ended generation tasks

I.1 Experimental Setup

This section illustrates the detailed setup of open-ended question-answering tasks, which more closely resemble real-world user scenarios. We employed the **Databricks Dolly-15k** dataset, an open-source collection of instruction-following records. This dataset spans multiple behavioral categories as defined in the InstructGPT paper (Ouyang et al., 2022), including brainstorming, classification, closed QA, open QA, and summarization, providing a diverse and realistic dataset for evaluating model robustness. We test the results mainly on the Llama-2 family.

I.2 Selection of Input Variables

Given that the text length in open-ended instructions far exceeds that of MCQ tasks, a direct analysis of all words would lead to exponential computational costs. To address this challenge, we applied the optimization strategies discussed in our limitations section: **(1) Selective Input Variable Analysis** and **(2) Phrase-level Aggregation**.

Our approach is guided by a systematic procedure to choose a fixed number of key input variables from the full input. Specifically, for each input sentence, we select meaningful words or phrases to construct the set of input variables N. A word is considered "meaningful" if it is not an NLTK (Bird, 2006) stop word or a punctuation mark. The remaining parts of the text, such as generic instruction templates (e.g., "Below is an instruction...") and stop words, are treated as fixed background context. During the interaction analysis, only the variables within the set N are masked.

For a concrete example, consider the following prompt:

```
Below is an instruction that describes a task.  Write a
response that appropriately completes the request.

### Instruction:
Identify from the following list characters from The X-Files
who are bald or balding:  Walter Skinner, John Fitzgerald
Byers, Dana Scully, Melvin Frohike, Darius Michaud, Peter
Watts, Conrad Strughold, Queequeg

### Response:
```

Selection Process for Input Variables:

1. **Method (1) Application:** We designate the generic instruction template and functional stop words (e.g., "from", "the", "who", "are") as background context, excluding them from the input variable set.
2. **Method (2) Application:** We aggregate words forming core semantic concepts into single phrasal units, such as the key entity `"Walter Skinner"` and the critical condition `"bald or balding"`.

Final Input Variables:

```
[
  "Identify", "following list", "characters","X-Files",
  "bald or balding","Walter Skinner", "John Fitzgerald Byers",
  "Dana Scully","Melvin Frohike", "Darius Michaud",
  "Peter Watts","Conrad Strughold", "Queequeg"
]
```

I.3 Experimental Results

By applying the aforementioned input variable selection strategies (Methods 1 and 2) in our experiments on the Dolly dataset, we successfully managed the analytical complexity for each long-text input, leading to a substantial reduction in computational cost.

Crucially, the experimental outcomes derived from open-ended questions and the optimized setup remained **highly consistent** with the main conclusions drawn from our MCQ-based experiments.

Here are the results and τ is set to 0.1, we can observe the same conclusions in this experiment.

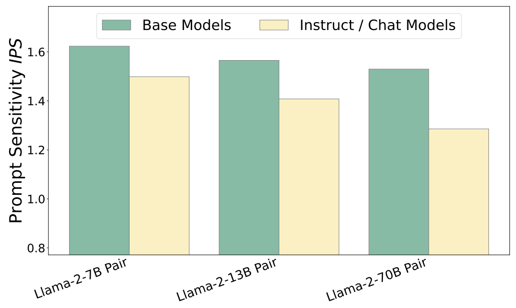


Figure 41: A comparison of the prompt sensitivity between instruct/chat models and base models. Results show that instruct/chat models are less sensitive than corresponding base models.

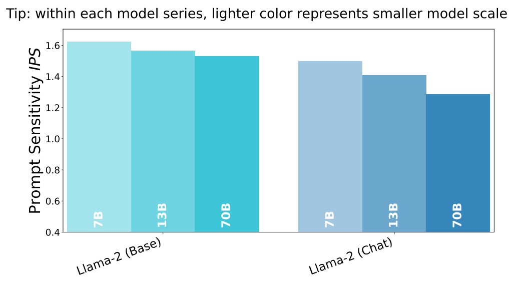


Figure 42: A comparison of prompt sensitivity across different model scales. As the model scale increases, the prompt sensitivity within a model series systematically decreases.

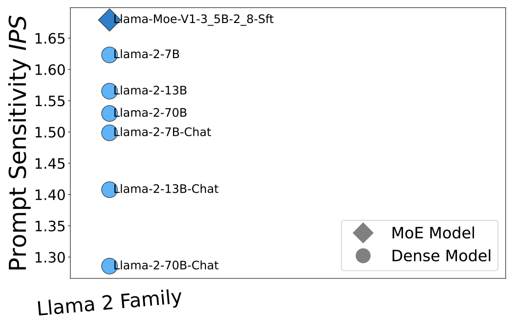


Figure 43: A Comparison of prompt sensitivity between MoE models and dense models. Generally, MoE models tend to be more sensitive than dense models in the same model family.

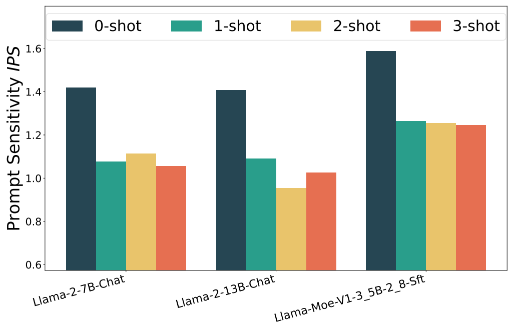


Figure 44: A comparison of prompt sensitivity between 0-shot learning and few-shot learning. The drop in prompt sensitivity is substantial from 0-shot to 1-shot.

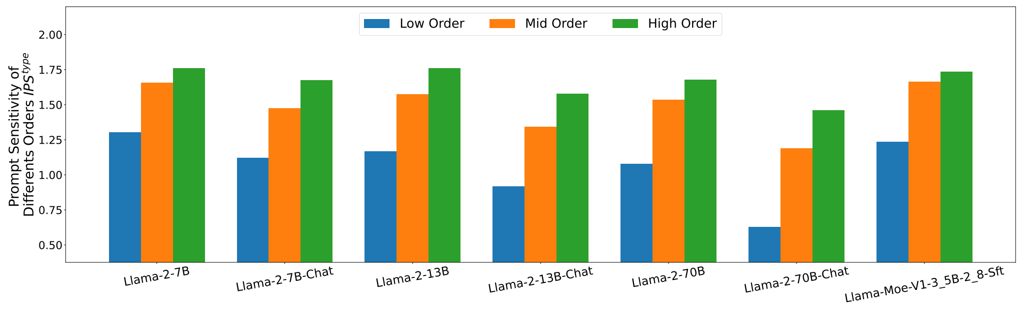


Figure 45: A comparison of the prompt sensitivity of three order types. Results show that low-order interactions are the least sensitive, while high-order interactions are the most sensitive.

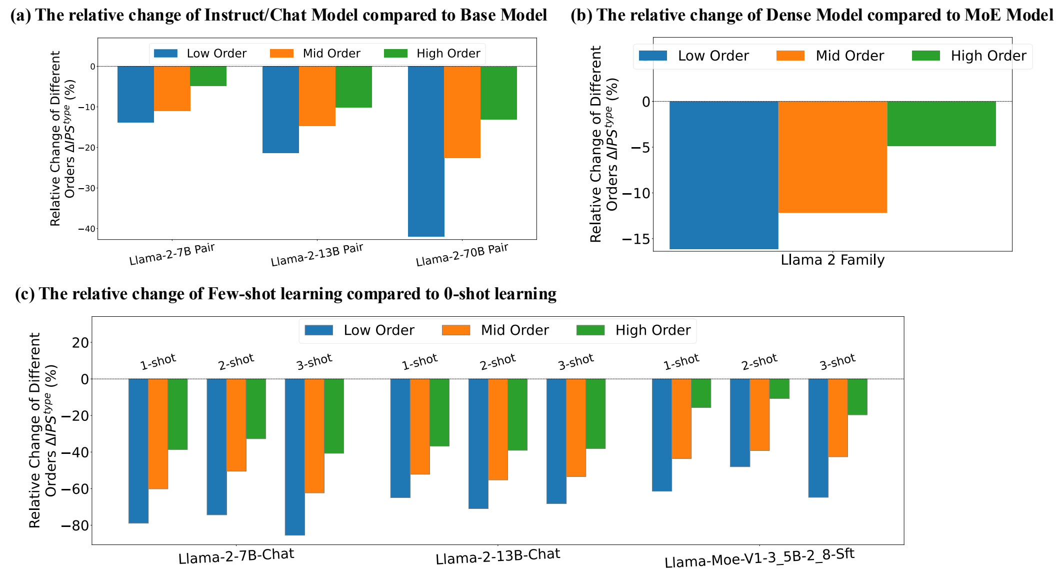


Figure 46: Comparing the relative change in the prompt sensitivity of low-, mid-, and high-order interactions for different factors.

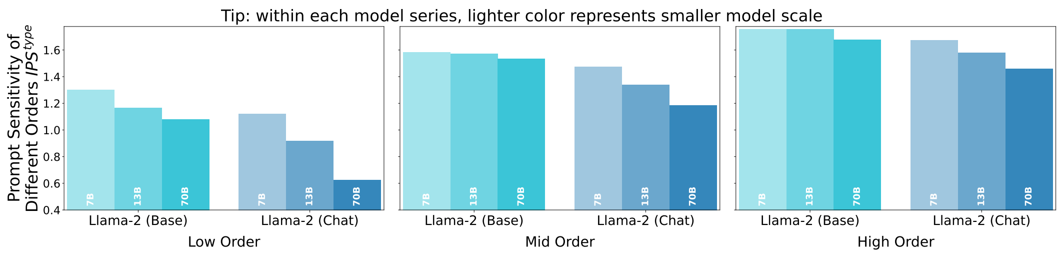


Figure 47: A comparison of prompt sensitivity at the order-level across different model scales.

J Details of the experiments beyond prompt template modifications

J.1 Experimental Setup

To assess the generalizability of our findings beyond superficial template modifications, we conducted an additional experiment on the **Dolly-15k** dataset. This setup introduces more realistic and complex prompt perturbations that better reflect authentic user interactions, specifically **semantic paraphrasing** and **instruction reordering**. We employed these techniques to test the robustness of our conclusions under more challenging conditions. An illustrative example of a semantic paraphrase used in our experiment is shown below:

An illustrative example of the full prompt structure and the applied perturbations is shown below.

Original Prompt:

```
Below is an instruction that describes a task. Write a
response that appropriately completes the request.

### Instruction:
Identify from the following list characters from The X-Files
who are bald or balding: Walter Skinner, John Fitzgerald
Byers, Dana Scully, Melvin Frohike, Darius Michaud,
Peter Watts, Conrad Strughold, Queequeg

### Response:
```

Perturbation 1: Semantic Paraphrase:

```
Below is an instruction that describes a task. Write a
response that appropriately completes the request.

### Instruction:
Select from the list below figures from The X-Files
that are losing their hair or bald: Walter Skinner,
John Fitzgerald Byers, Dana Scully, Melvin Frohike,
Darius Michaud, Peter Watts, Conrad Strughold, Queequeg

### Response:
```

Perturbation 2: Instruction Reordering:

```
Below is an instruction that describes a task. Write a
response that appropriately completes the request.

### Instruction:
From The X-Files, identify characters who are bald or balding
from the following list: Walter Skinner, John Fitzgerald
Byers, Dana Scully, Melvin Frohike, Darius Michaud,
Peter Watts, Conrad Strughold, Queequeg

### Response:
```

J.2 Selection of Input Variables

For this experiment, we employed the same input variable selection strategy as detailed in Appendix I.2. This approach combines Selective Input Variable Analysis and Phrase-level Aggregation to manage the computational complexity associated with longer, open-ended instructions while preserving the core semantic elements for interaction analysis.

J.3 Experimental Results

The experimental outcomes from this more challenging setup verify the main conclusions of our paper. Despite the increased complexity of the perturbations, we consistently observed that the four identified factors (supervised fine-tuning, increased model scale, dense architectures, and few-shot learning) reduce prompt sensitivity, primarily by stabilizing low-order interactions. This replication demonstrates that our conclusions are not confined to simple template variations but hold true in scenarios that more closely mirror authentic user interactions, thereby strengthening the generalizability of our work.

Here are the results and τ is set to 0.1, we can observe the same conclusions in this experiment.

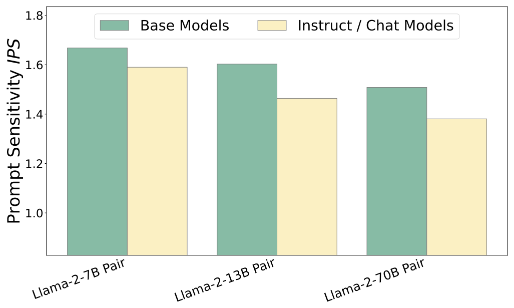


Figure 48: A comparison of the prompt sensitivity between instruct/chat models and base models. Results show that instruct/chat models are less sensitive than corresponding base models.

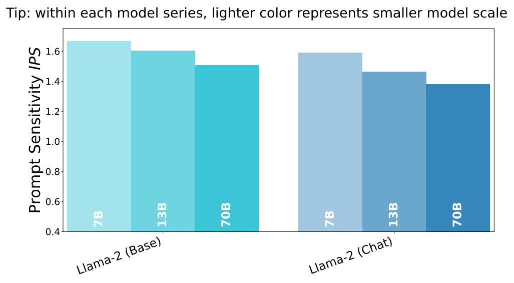


Figure 49: A comparison of prompt sensitivity across different model scales. As the model scale increases, the prompt sensitivity within a model series systematically decreases.

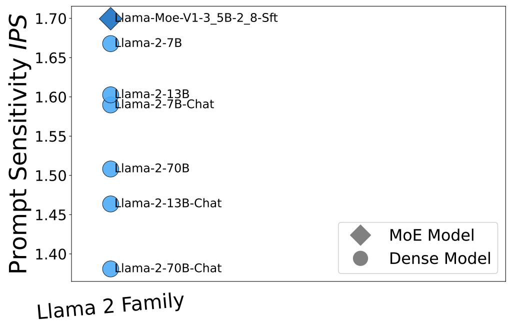


Figure 50: A Comparison of prompt sensitivity between MoE models and dense models. Generally, MoE models tend to be more sensitive than dense models in the same model family.

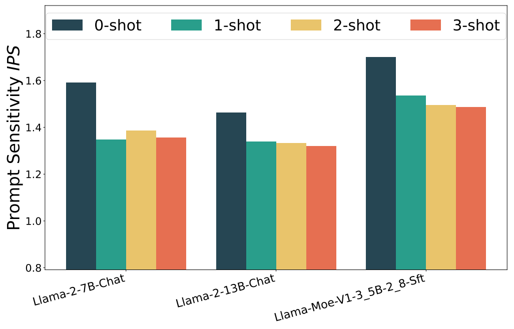


Figure 51: A comparison of prompt sensitivity between 0-shot learning and few-shot learning. The drop in prompt sensitivity is substantial from 0-shot to 1-shot.

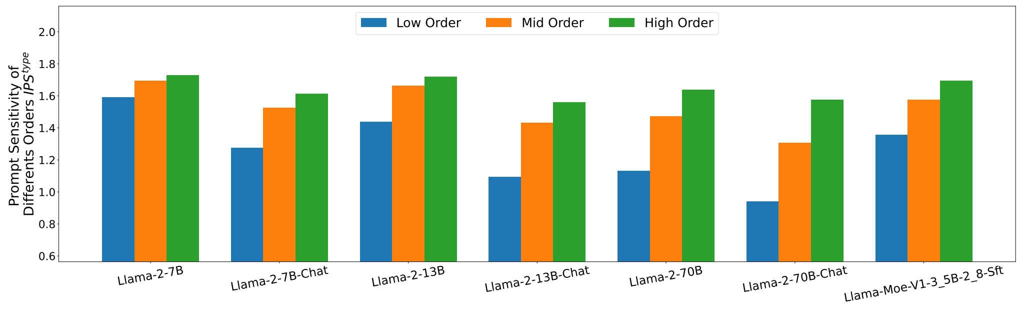


Figure 52: A comparison of the prompt sensitivity of three order types. Results show that low-order interactions are the least sensitive, while high-order interactions are the most sensitive.

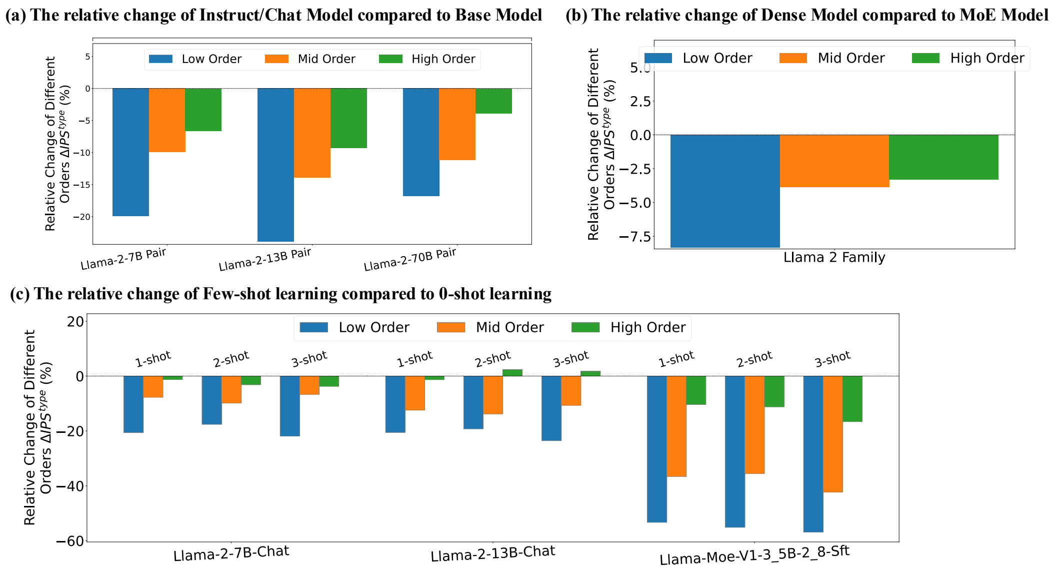


Figure 53: Comparing the relative change in the prompt sensitivity of low-, mid-, and high-order interactions for different factors.

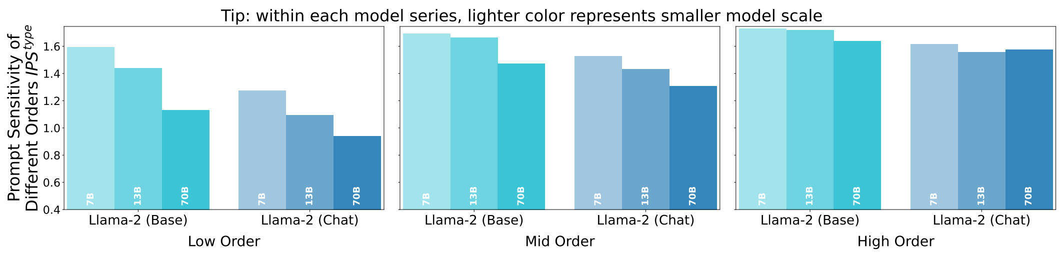


Figure 54: A comparison of prompt sensitivity at the order-level across different model scales.

K The Use of Large Language Models (LLMs)

We used large language models during the writing process of this paper exclusively for language editing tasks, such as grammar checks and rephrasing for clarity. The scientific contributions, including the core ideas, experiments, and analysis, were conceived and executed entirely by the authors.