When Punctuation Matters: A Large-Scale Comparison of Prompt Robustness Methods for LLMs

Mikhail Seleznyov[1,2], **Mikhail Chaichuk**[1,5], **Gleb Ershov** [3], **Alexander Panchenko**[1,2], **Elena Tutubalina**[1,6,7], **Oleg Somov**[1,4]
[1]AIRI, [2]Skoltech, [3]Yandex, [4]MIPT
[5]HSE University, [6]Sber AI, [7]ISP RAS Research Center for Trusted AI
Correspondence: seleznev@airi.net, tutubalina@airi.net, somov@airi.net

Abstract

Large Language Models (LLMs) are highly sensitive to subtle, non-semantic variations in prompt phrasing and formatting. In this work, we present the first systematic evaluation of 5 methods for improving prompt robustness within a unified experimental framework. We benchmark these techniques on 8 models from Llama, Qwen and Gemma families across 52 tasks from Natural Instructions dataset. Our evaluation covers robustness methods from both fine-tuned and in-context learning paradigms, and tests their generalization against multiple types of distribution shifts. Finally, we extend our analysis to GPT-4.1 and DeepSeek V3 to assess frontier models' current robustness to format perturbations. Our findings offer actionable insights into the relative effectiveness of these robustness methods, enabling practitioners to make informed decisions when aiming for stable and reliable LLM performance in real-world applications. Code: https://github.com/AIRI-Institute/when-punctuation-matters.

1 Introduction

Large Language Models (LLMs) today excel across a wide range of tasks in both in-context learning (ICL) and supervised fine-tuning (SFT) paradigms (Brown et al., 2020; Gao et al., 2021; Dong et al., 2024; Le Scao et al., 2023; Yang et al., 2024a; Wu et al., 2024; Mosbach et al., 2023; Yang et al., 2024b; Chen et al., 2024; Somov and Tutubalina, 2025).

However, a critical yet often overlooked challenge is the high sensitivity of LLMs to prompt formatting. Many large-scale task-rich benchmarks rely on a single instruction format to evaluate all language models on a wide-range of tasks, implicitly assuming that performance is independent of prompt format (Hendrycks et al., 2020; Srivastava et al., 2023). Recent work shows that even *semantically neutral* variations in prompt structure can lead to substantial changes in model predictions, often exceeding the variability introduced by model architecture or inference method (Voronov et al., 2024a; Mizrahi et al., 2023).

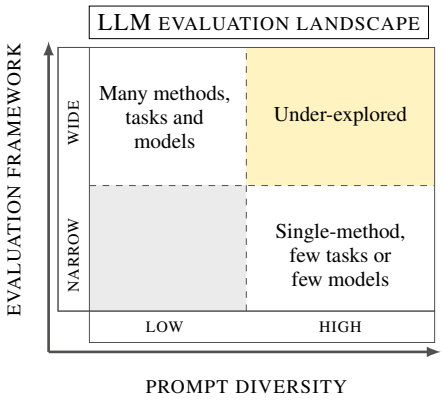


Figure 1: Most existing robustness methods are evaluated in isolation and in disparate settings, disallowing apples-to-apples comparison. Our work targets the under-explored upper-right quadrant by evaluating multiple prompt robustness techniques across a wide range of prompt formats, LLM families, learning paradigms, and distribution shifts under a unified framework.

Prompt format (e.g. spacing, capitalization, punctuation) can dramatically influence performance, leading to inconsistent or unreliable outputs (Zhao et al., 2021; Min et al., 2022). This phenomenon, known as **prompt sensitivity**, can be mitigated using specialized **robustness methods**.

A number of robustness techniques have been proposed to address this issue, including Template Ensembling (Voronov et al., 2024b), Sensitivity-Aware Decoding (Lu et al., 2024), Batch Calibration (Zhou et al., 2024), and Consistency Learning (Qiang et al., 2024b). However, these methods have primarily been evaluated in isolation, making it difficult for practitioners to assess their relative strengths or determine which method is best suited to a given scenario.

Our work addresses this gap through a compre-

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 20370–20385
November 4-9, 2025

hensive, systematic evaluation of prompt robustness techniques under a unified experimental framework. Specifically, we benchmark four widely cited robustness methods against standard few-shot prompting and fine-tuning with prompt format augmentation as baselines.

We conduct experiments using a carefully curated version of GSM8K dataset (Vendrow et al., 2025) and a representative subset of 52 tasks from the well-known Natural Instructions dataset, covering domains such as mathematics, logic, and text comprehension. As backbone models, we evaluate three modern LLM families — GEMMA (Gemma Team, 2024), LLAMA (Dubey et al., 2024), and QWEN (Qwen et al., 2025) — with sizes from 1.5B to 9B parameters. We additionally include closed-source models to study format sensitivity at scale. Within our framework, we answer the following research questions:

RQ1: How do existing robustness methods compare in effectiveness across various models?

RQ2: How distribution shifts affect the effectiveness of SFT-based and ICL-based methods?

RQ3: How do sampling strategies such as greedy decoding, temperature sampling, top-p and min-p sampling affect robustness?

RQ4: How sensitive are frontier models to prompt format perturbations, and what methods can be applied in black-box setting to improve their robustness?

To the best of our knowledge, this is the first study to offer a side-by-side comparison of multiple prompt robustness methods under a unified, large-scale evaluation protocol spanning diverse prompt formats, model families, learning paradigms, distribution shifts and inference strategies. By bridging previously disconnected lines of work, our findings provide actionable insights for both practitioners and researchers interested in building more stable and reliable LLM-based systems. We also release our code to encourage systematic evaluation in the field of prompt sensitivity mitigation.

2 Related Work

Recent work has highlighted the sensitivity of language models to subtle prompt variations, but current research remains fragmented (Zhuo et al., 2024; Pei et al., 2025). Adversarial-focused studies (Zhu et al., 2024; Zou et al., 2023) expose vulnerabilities to malicious or perturbed prompts, emphasizing safety but targeting directed threat models rather than benign formatting inconsistencies.

Other works propose robustness-enhancing methods such as Consistency Learning (Qiang et al., 2024b), Batch Calibration (Zhou et al., 2024), and Template Ensembles (Voronov et al., 2024a), which improve stability either during training or inference. However, these approaches are evaluated in isolation, making it difficult to assess their relative effectiveness.

Complementary studies (Lu et al., 2024; Zhao et al., 2021; Sclar et al., 2024) analyze prompt components and formatting artifacts, showing that even innocuous design choices (e.g., whitespace, punctuation) can introduce large performance shifts. This further underscores the need for unified, standardized evaluation protocols.

In summary, while prior research has addressed different aspects of prompt sensitivity, there is a lack of systematic, comparative evaluation across tasks, models, and learning paradigms. Our work fills this gap by benchmarking four robustness methods under a unified framework across 52 diverse tasks, multiple LLM families and distribution shift scenarios, resulting in actionable takeaways for practitioners.

3 Experimental Setup

To answer our research questions, we use a subset of Natural Instructions with a parametrized set of formats (Section 3.1) and implement the methods from Section 3.2. We evaluate performance and robustness using the metrics defined in Section 3.3.

3.1 Data & Format

We use a subset of 52 tasks from Natural Instructions (Wang et al., 2022) with diverse human-written formats and instructions, comprising 19 multiple-choice tasks and 33 classification tasks with 2, 3 or 4 answer options. Given that there are more than 1600 tasks we select relevant and socially impact tasks following Sclar et al. (2024) task selection criteria (refer to Appendix D for details). Resulting tasks cover math and logic problems, text comprehension, detection of harassment and racial stereotypes. To evaluate the performance, we use a subset of 1000 random examples from each task.

To answer RQ3 (how do sampling strategies

Descriptor transformation	Separator	Space	Text & option separator	Option item style	Option item wrapper
.title()	': '	' '	' '	A, B, C, ...	{})
.uppercase()	'- '	'\n'	'\t'	1, 2, 3, ...	{}.
.lowercase()	'\n'	'; \n'	''	I, II, III, ...	[{}]

Table 1: Format components, with some example values. Descriptor transformation correspond to Python command making first character upper case (title), all letters upper case (uppercase) or all letters lower case (lowercase). For option item wrapper, {} is used as a placeholder for option item (e.g. A or 1).

such as greedy decoding, top-p and min-p sampling affect robustness), we also consider multi-step math reasoning on a carefully curated version of GSM8K dataset (Vendrow et al., 2025). It contains 1209 grade school math problems along with step-by-step solutions.

Prompt format. We consider 6 types of format components, following Sclar et al. (2024). They are listed in Table 1. For each component there are between 4 and 16 possible values. To construct a format, we select a specific value of each component. For example, default prompt for a task could be structured as following: **`question:{}A){}B){}answer:{}`**, where **`{}`** denotes the placeholders for the task instruction and multiple-choice answer options (following (Wang et al., 2022) formatting). Then choosing the first values from first row of Table 1 to modify original prompt design results in

```
Question: {} A) {} B) {} Answer: {}
```

whereas taking values from second row forms another prompt design:

```
QUESTION- {}\n1.\t{}\n2.\t{}\nANSWER- {}
```

For some tasks there are no multiple choice options — in this case the format is defined only by descriptor transformation, separator and space. Complete list of format components is available in Appendix G.

3.2 Methods

We consider five representative approaches for improving robustness to prompt formatting. These span both ICL and SFT paradigms. Below, we briefly describe each method.

Few-shot (FS). As a baseline, we use a standard 2-shot prompting strategy. Since the selection and order of demonstration examples can significantly influence results (Lu et al., 2022), we fix the in-context examples and their order across all models and test samples. Demonstration examples are also formatted in the same way as the test sample, hinting to the model that formatting should not influence the prediction.

Batch Calibration (BC). Batch Calibration (Zhou et al., 2024) is a post-hoc correction technique. It estimates contextual bias across a batch and adjusts predicted log-probabilities by subtracting the bias. While simple and efficient, it is limited to classification tasks.

Template Ensembles (TE). Template Ensembling (Voronov et al., 2024b) improves robustness by averaging predicted class probabilities across N prompt formats. This reduces variance caused by formatting changes, but increases inference cost linearly with N.

Sensitivity-Aware Decoding (SAD). This approach is inspired by Lu et al. (2024). It penalizes predictions that are sensitive to synthetic input perturbations. In our implementation, we use random token substitutions to estimate sensitivity. This approach helps to stabilize outputs but requires multiple forward passes per input.

LoRA with format augmentations (LoRA). We apply parameter-efficient fine-tuning (PEFT) using LoRA on an instruction-following dataset augmented with formatting variations. This method exposes the model to diverse prompt styles during training with the aim of mitigating spurious correlations between answers and format components.

LoRA with consistency loss (LoRA-JS). Following Qiang et al. (2024a), we add a Jensen-Shannon consistency loss between outputs of different prompt variants, encouraging format-invariant predictions. The total loss combines standard cross-entropy with a divergence term.

Full implementation details for each method are provided in Appendices C, I.

3.3 Metrics & Inference Approach

Evaluation Metrics. To evaluate model performance, we use *accuracy* as the primary metric. To assess sensitivity to prompt formatting, we report two measures: *spread* and *standard deviation*. Spread is defined as the difference between the maximum and minimum accuracy across a set of prompt formats (Sclar et al., 2024), providing a simple measure of output variability due to prompt variation. We also consider a class-imbalance setting. Since accuracy is often misleading on imbalanced tasks, we report *Matthews correlation coefficient (MCC)*. We choose MCC instead of F1 score since the latter puts emphasis on the positive class and may change dramatically after a permutation of classes. MCC treats classes symmetrically and accounts for all four confusion matrix components, including true negatives. Since our tasks usually do not have a distinguished positive class, MCC is better suited for our evaluations.

Inference Strategies. To obtain answers from the language model in multiple-choice and classification tasks, we use two common inference strategies: *greedy decoding* and *probability ranking*. Greedy decoding generates the answer token-by-token, selecting the most likely token at each step. The result string is normalized and evaluated to gold answer with exact match. In contrast, probability ranking computes the probability of each option from a predefined set of answers, and selects the highest-ranked one. Since all answer options are known in advance, this method is implemented using teacher forcing.

For long-form generation on GSM8K, we employ greedy decoding, temperature sampling, top-p (Holtzman et al., 2020) and min-p (Minh et al., 2025) sampling.

4 Results

Our experiments address the four research questions outlined in Section 1: methods comparison in default setting (**RQ1**, Section 4.1), effect of distribution shifts on fine-tuning and ICL-based methods (**RQ2**, Section 4.2), effect of inference strategy (**RQ3**, Section 4.3) and evaluations of frontier models (**RQ4**, Section 4.4).

4.1 Robustness methods comparison

RQ1: How existing robustness methods compare in effectiveness across different LLM families and sizes? To answer **RQ1**, we apply all methods under the default conditions (without a distribution shift) to evaluate their impact on accuracy and robustness. We assess accuracy to check whether robustness methods negatively affect performance. Figure 2 plots accuracy of each method, averaged across 52 tasks, along with standard deviation over formats[1].

To compare methods' effectiveness in improving robustness, we use the following procedure. For a fixed model M_0 (e.g. Llama 3.1 8B) and each task t we estimate the spreads of baseline few-shot approach and the competing method X over 10 formats, where $X \in \{\text{BC}, \text{TE}, \text{SAD}, \text{LoRA}\}$. Then, we consider differences

$$\begin{gathered}\text{SpreadDiffs}_{M_0,X} = \\ \left\{\text{spread(FS)}_{t,M_0} - \text{spread(X)}_{t,M_0}\right\} \\ \forall t \in \mathbf{T}, |\mathbf{T}| = 52,\end{gathered} \tag{1}$$

where $\mathbf{T}$ is our selected 52 tasks from Natural Instructions. Finally, we run Student's t-test with H_0: mean of $\text{SpreadDiffs}_{M_0,X}$ is equal to zero. If H_0 is rejected, the method with lower mean spread is considered more robust on model M_0. Otherwise, X ties with few-shot. Such tests are run for each of 8 open-source models we evaluate, and the results are shown in Figure 3.

Batch Calibration improves both accuracy and robustness across the board. From Figure 2 we can see that Batch Calibration achieves higher average accuracy compared to few-shot for all 8 open-source models. Meanwhile, Figure 3 shows that BC delivers statistically significant reduction of spread for 6/8 models. Since calibration methods do not require training data and have near-zero inference time overhead, these strong results put Batch Calibration as a clear leader in terms of format robustness enhancement in absence of distribution shifts.

Template Ensembles improve robustness at cost of reducing accuracy. Ensembling method proposed by Voronov et al. (2024a) also reduces spread, with statistically significant reductions for 4/8 models. However, it results in lower accuracy

[1]We also tested LoRA with consistency loss. However, the results were less favorable compared to the other methods, especially LoRA with format augmentations. Some of these findings are available in Appendix B.

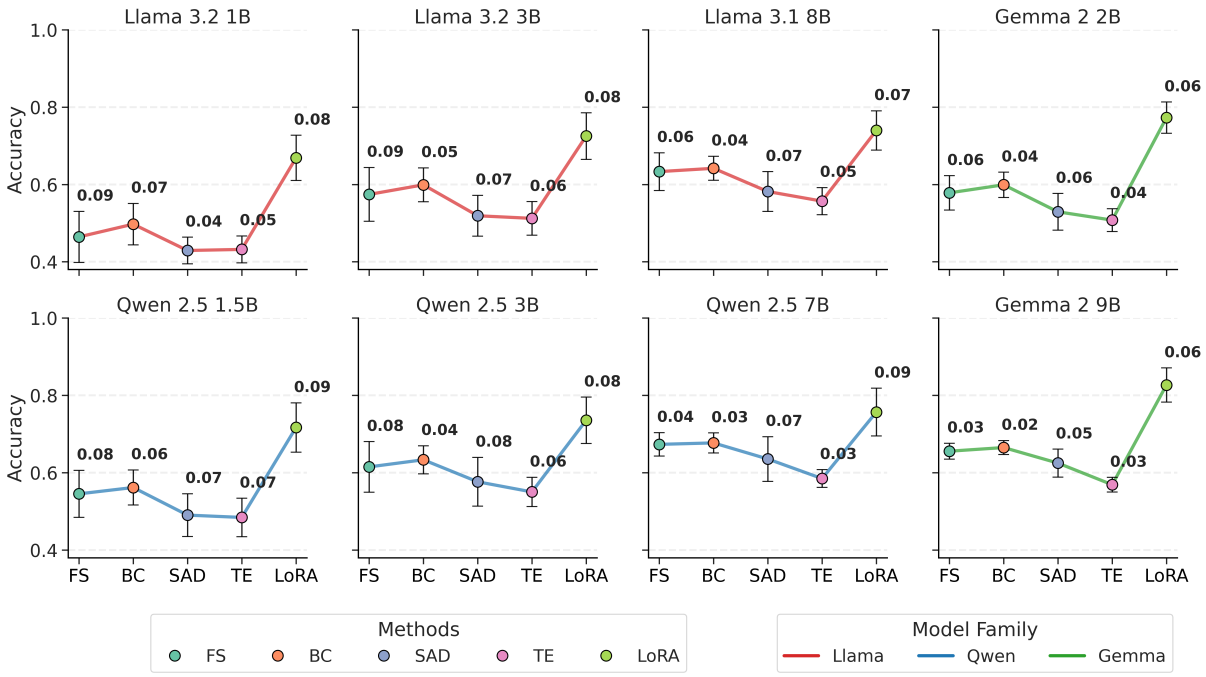


Figure 2: Comparing format sensitivity mitigation methods in terms of their effect on accuracy and standard deviation over prompt formats. To aggregate accuracy, we first compute median accuracy over formats for each task, and then average over 52 tasks. Error bars are 2 × (standard deviation over formats, averaged across tasks).

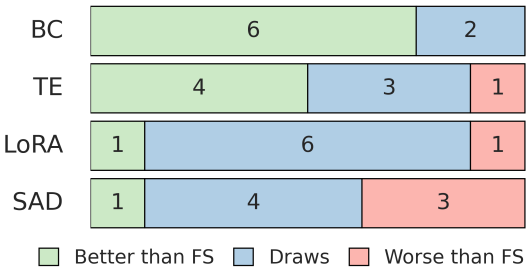


Figure 3: Comparing format sensitivity mitigation methods against regular few-shot in terms of spread on 8 language models. Method wins against few-shot for a given model if it has statistically significantly lower spread than few-shot on 52 tasks.

compared to few-shot baseline. To investigate the causes of the drop in performance, we inspected predictions of individual ensemble members. It turned out that for one format in the ensemble the accuracy sometimes underperforms, affecting average probabilities. This aligns with the original findings of Voronov et al. (2024a), which note that a single suboptimal template may make the ensemble perform noticeably worse. Together, it suggests that logit averaging is sometimes a brittle strategy, sensitive to outliers.

LoRA with augmentations enhances accuracy, but struggles to consistently improve robustness. On Figure 2 we can see that LoRA with augmentations achieves much higher average accuracy compared to ICL-based approaches. This is expected, since LoRA is the only SFT-based method on the plot, and has access to training labels. Perhaps more surprisingly, augmentations have almost no impact on robustness: Figure 3 shows that LoRA improves spread compared to few-shot only on a single model out of 8, with 6 ties and 1 loss.

Takeaways:

1. In absence of distribution shifts, calibration-based approach shows promise in improving robustness to prompt formats due to its ability to significantly reduce spread, positive effect on accuracy and low overhead.

2. While naive parameter-efficient finetuning with augmentations significantly improves accuracy, it turns ineffective in mitigating sensitivity to format changes.

3. Probability averaging strategy used in Template Ensemble helps to reduce spread, but may suffer from sensitivity to especially poor-performing formats.

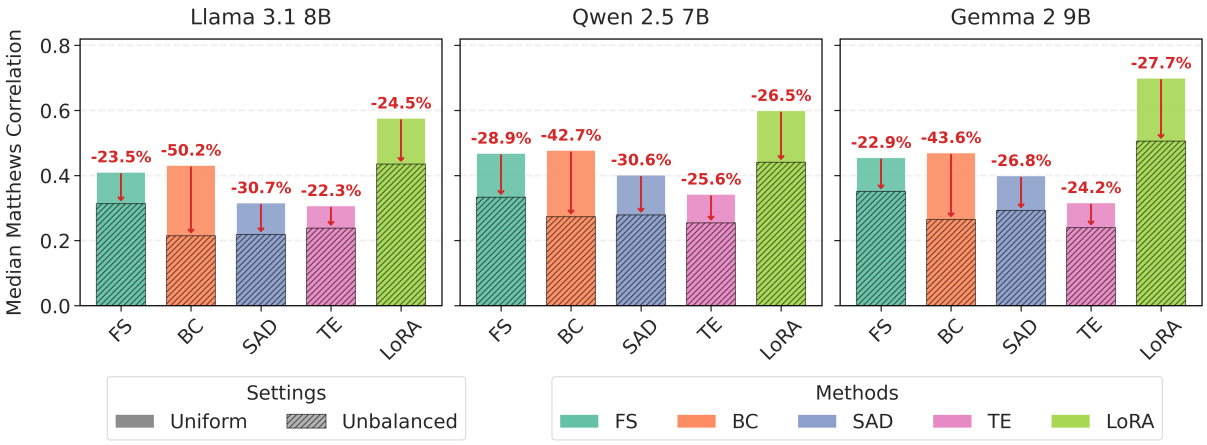


Figure 4: Median Matthews Correlation of robustness methods in uniform vs. unbalanced settings for LLaMA 3.1 8B, Qwen 2.5 7B, and Gemma 2 9B. Red values indicate the drop in performance under the unbalanced setting relative to the uniform case.

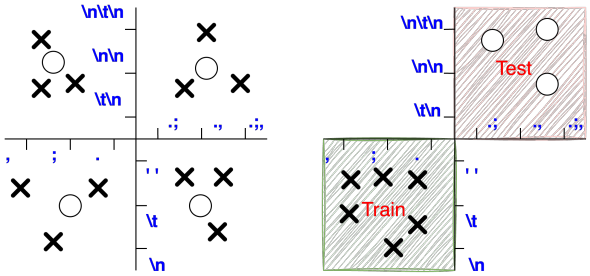


Figure 5: Without distribution shift (left), train and test formats are sampled uniformly. Under the compositional distribution shift (right), the test set contains novel combinations of known components, requiring systematic generalization. Cross (×) stands for train samples, circle (○) for test.

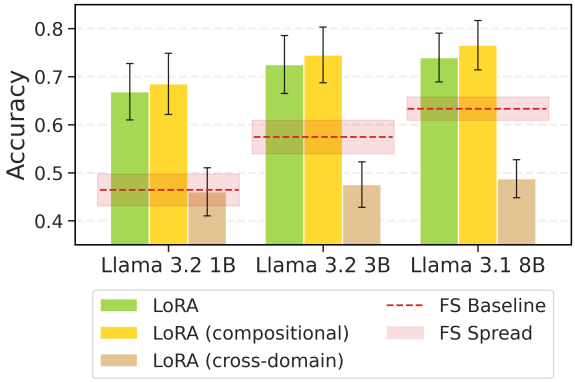


Figure 6: LoRA method under distribution shifts. To aggregate accuracy, we first compute median accuracy over formats for each task, and then average over 52 tasks. Error bars are 2 × (standard deviation over formats, averaged over tasks).

Method	BC	FS	LoRA	SAD	TE
Default	2.6	2.9	1.7	4.3	3.5
Unbalanced	$3.2_{+0.6}$	$2.7_{-0.2}$	1.7	$4.0_{-0.3}$	$3.3_{-0.2}$

Table 2: Rankings of methods across models by Matthews correlation coefficient (1 is best). Rankings are averaged across models and tasks.

4.2 Impact of distribution shifts

RQ2: How do distribution shifts affect the effectiveness of SFT-based and ICL-based methods? As we see in Section 4.1, Batch Calibration helps to improve robustness while LoRA significantly stands out in terms of accuracy. To answer **RQ2**, we zoom in and inspect these methods in more detail to understand their limitations.

Covariate shift (class imbalance). In Batch Calibration, predicted probabilities of each class are adjusted by subtracting mean probability of this class over batch. Naturally, predicted probabilities for classes that occur more often are adjusted more. They in turn are less often selected by argmax, which leads to a more uniform predictive distribution. Thus, Batch Calibration implicitly assumes more uniform class distribution compared to baseline few-shot approach. While reasonable in balanced tasks, this assumption may lead to calibration errors under class imbalance. To investigate this, we construct an artificial imbalanced dataset by downsampling each of our 52 tasks such that the most frequent class constitutes 90% of the examples, with the remaining classes evenly splitting the remaining 10%.

In Figure 4, we observe that all methods are affected by the unbalanced setting, but Batch Calibration suffers the most due to the model's inductive

bias toward a uniform class distribution. Table 3 confirms this finding: Batch Calibration exhibits the largest change in average ranking. LoRA-fine-tuned models also degrade, as they were trained assuming a uniform class distribution and thus are also subject to covariate shift.

Compositional and cross-domain shifts. To evaluate the robustness of the LoRA method to distribution shifts, we consider two scenarios: **compositional** and **cross-domain**.

Compositional shift. Inspired by the notion of systematic compositionality (Hupkes et al., 2020), this setting tests the model's ability to generalize by recombining known elements in novel ways. In default scenario in Section 4.1, train and test formats are sampled uniformly. However, under compositional shift test formats contain new combinations of previously seen format components. An illustration of compositional train/test format split is shown in Figure 5.

Cross-domain shift. To evaluate robustness to domain changes, this setting uses training data from an external dataset (see Appendix I). The prompt formats remain uniformly distributed during both training and testing like in Section 4.1. This setup probes the model's and method's ability to disentangle semantics from format and generalize beyond the training domain.

Analysis. Compositional shift with respect to formats does not affect accuracy and robustness much. We hypothesize this is due to the fact that LoRA with augmentations does not consistently improve robustness even in default scenario considered in **RQ1**, Section 4.1, and the complexity of compositional shift remains hidden.

Cross-domain transfer is a challenging setup. While it is possible that with another configuration of training data and hyperparameters it might perform better, in our experiments cross-domain fine-tuning with augmentations decreases accuracy below the few-shot baseline.

Takeaways:

1. Batch Calibration implicitly assumes a more uniform prior on classes compared to baseline few-shot approach. This inductive bias backfires when the class distribution is skewed.

2. Cross-domain experiments confirm that high accuracy achieved by LoRA approach substantially relies on training dataset.

Inference Strategy	Greedy Decoding	Probability Ranking
Gemma 2 2B	$0.48 \pm \mathbf{0.28}$	0.58 ± 0.06
Gemma 2 9B	$0.55 \pm \mathbf{0.32}$	0.66 ± 0.03
Llama 3.1 8B	$0.63 \pm \mathbf{0.11}$	0.63 ± 0.06
Llama 3.2 1B	$0.46 \pm \mathbf{0.11}$	0.46 ± 0.09
Llama 3.2 3B	$0.56 \pm \mathbf{0.13}$	0.57 ± 0.09
Qwen 2.5 1.5B	$0.49 \pm \mathbf{0.12}$	0.55 ± 0.08
Qwen 2.5 3B	$0.59 \pm \mathbf{0.12}$	0.61 ± 0.08
Qwen 2.5 7B	$0.63 \pm \mathbf{0.14}$	0.67 ± 0.04

Table 3: Comparison of inference strategies: greedy decoding vs. probability ranking. To aggregate accuracy, we first compute median accuracy and standard deviation over formats for each task, and then average over 52 tasks. We report averaged median accuracy $\pm 2\times$ averaged std. Higher standard deviation is in bold.

4.3 Impact of sampling strategies

We split **RQ3** into two questions.

RQ3-1: How does greedy decoding affect robustness compared to choosing highest-probability answer option from a predefined answer set? To answer **RQ3-1**, we run experiments with two inference strategies, described in Section 3.3. Table 3 demonstrates, that generation is always less robust to format choice, with Gemma models exhibiting especially large instability.

Generation approach is widely used in practical applications (chat-bots, API calls), so the problem of format sensitivity can be even more acute there.

RQ3-2: How do sampling strategies such as greedy decoding, temperature sampling, top-p and min-p sampling affect robustness in long-form generation tasks? For tasks with long-form generation, other sampling strategies than greedy decoding are used, and they've been reported to deliver superior performance (Holtzman et al., 2020, Minh et al., 2025). It's unclear however what is their effect on robustness to prompt formatting. On the one hand, sampling introduces stochasticity, which may increase sensitivity. On the other hand, over the course of long generation (e.g. in chain-of-thought) stochasticity may have a smoothing effect, allowing to recover from earlier mistakes. To explore this, we run additional experiments on GSM8k dataset. We evaluate models using 10 different formats, and compute spread to measure sensitivity. Results are given on Figure 7. Most of the time, greedy decoding delivers best or nearly best robustness. We also provide distribution of accuracy scores per each format in Appendix A,

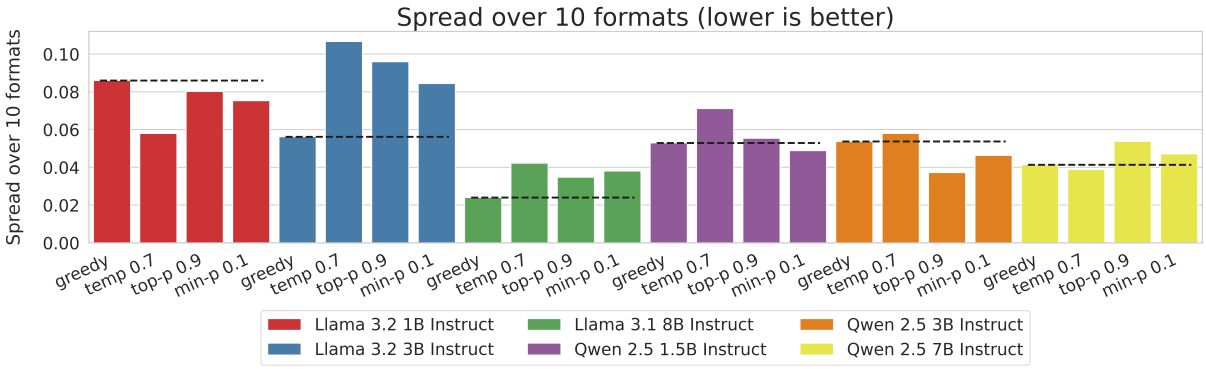
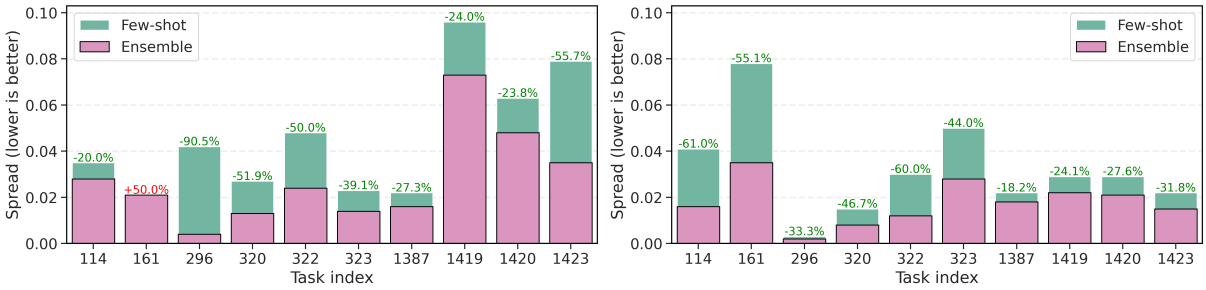


Figure 7: Comparing effect of sampling strategies on robustness to prompt format variations on GSM8K dataset.



(a) DeepSeek V3

(b) GPT-4.1

Figure 8: Spreads for selected tasks from Natural Instructions for two frontier models. Even at this scale, for some tasks spread still might reach 8-10 accuracy points. Using a modified version of Template Ensembles with majority voting instead of probability averaging, we are able to reduce the spread in 19/20 cases, in 9 of which the reduction is at least 44%.

finding out e.g. that low spread for Llama 3.2 1B Instruct is caused by a performance degradation rather than enhanced robustness.

Takeaways:

1. Greedy decoding exacerbates models' format sensitivity. In classification and multiple-choice probability ranking is preferable.

2. In long-form generation, greedy decoding might be the best available option. Temperature sampling has mixed effect on spread but consistently decreases accuracy, especially for smaller models. Top-p and min-p sampling have robustness and performance similar to greedy decoding.

4.4 Frontier models' robustness

We split **RQ4** into two questions.

RQ4-1: How sensitive are frontier models to format perturbations? To answer the first part of RQ4, we perform experiments with two frontier models, GPT-4.1 and DeepSeek V3. We evaluate them on a subset of 10 out of 52 tasks due to budget limitations. The results are presented in Table 4. We can see that large closed-source models show much better robustness. While DeepSeek V3 significantly outperforms GPT-4.1 in terms of accuracy, it is more sensitive to format changes.

However, Figure 8 shows that on individual tasks, even frontier models might have spread of 8-10 accuracy points.

RQ4-2: What methods can be applied in black-box setting to improve their robustness? To answer the second part of RQ4, we need to consider each method assumptions. Batch Calibration, Sensitivity-Aware Decoding and Template Ensembles require logit access, which is not always available for closed source models. With SFT-based methods the problem is that the user usually has no control over what exact method and hyperparameters are used, even if company provides fine-tuning as a service.

For a broadly applicable approach, we consider an adaptation of Template Ensembles which uses majority voting instead of probability averaging.

Method	Model	Accuracy ↑	Std accuracy ↓	Spread ↓
Few-shot	Llama 3.1 8B	0.563	0.052	0.161
	Qwen 2.5 7B	0.605	0.058	0.190
	DeepSeek V3 0324	**0.741**	0.015	0.045
	GPT-4.1	0.624	**0.010**	**0.032**
Template Ensembles (majority voting)	DeepSeek V3 0324	**0.742**	0.009	0.028
	GPT-4.1	0.625	**0.005**	**0.018**

Table 4: Evaluation of frontier models on a subset of 10 out of 52 tasks. For reference, we also include a couple of open-source models. To aggregate accuracy, we take median over formats and average over tasks. Standard deviation and spread are first computed over formats for each task individually and then averaged.

Figure 8 confirms that this strategy effectively reduces spread, and in Table 4 we even see a slight performance improvement, contrary to results of Template Ensembles in Section 4.1. We attribute this to the fact that mode, utilized in majority voting, is substantially more robust to outlier formats than the mean, used in original version.

Takeaways:

1. Frontier models are substantially more robust compared to small open-source models, suggesting that scaling improves robustness.

2. Occasionally, there are still cases where spread can reach 8-10 accuracy points. To deal with them, Template Ensembles with majority voting might be used.

5 Conclusion

To the best of our knowledge, we have conducted the first comprehensive comparison of existing prompt sensitivity mitigation methods across multiple model families, sizes and distribution shifts.

We provide actionable insights for practitioners. For example, excessive fragility of calibration-based methods to class imbalance underscores the consequences of implicit assumptions, and hints that a reliable estimate of prior is important. Meanwhile, ineffectiveness of light supervised finetuning with augmentations at improving robustness suggests that more research is needed to develop a strong baseline in this paradigm. Finally, our experiments on frontier models confirm that scale is positively correlated with robustness. Still, on some tasks even large models exhibit 8-10 accuracy point differences solely due to format changes. Version of Template Ensembles with majority voting helps to mitigate this instability.

We also release our code to facilitate research aimed to address the problem of format sensitivity.

Limitations

Our study provides deep insights into classification tasks, multiple-choice tasks and multi-step math reasoning, leaving some settings like text generation or summarization out of the scope.

Some of considered robustness methods are harder to apply to frontier models. Batch Calibration, Sensitivity-Aware decoding and Template Ensembles require access to logits, which are sometimes unavailable. Finetuning approaches might be expensive at large scale. Additionally, when using finetuning API, users usually have limited control over the finetuning procedure.

Ethics

The models and datasets used in this study are publicly available for research purposes, with licenses detailed in Appendix F. All experiments were performed on NVIDIA A100 80GB GPUs. Each finetuning or evaluation run was conducted on a single GPU, and took between 1 and 24 hours, depending on the model size and method's efficiency. To optimize computation, we utilized up to 46 GPUs in parallel. In total, the experiments took approximately 15,000 GPU-hours. Our PyTorch/Hugging Face code will be released alongside the paper, and we expect no direct social or ethical concerns arising from this work.

Use of AI Assistants We utilize Grammarly to enhance and proofread the text of this paper, correcting grammatical, spelling, and stylistic errors, as well as rephrasing sentences. Consequently, certain sections of our publication may be identified as AI-generated, AI-edited, or a combination of human and AI contributions. We also used DeepSeek V3, Claude Sonnet 3.5 and ChatGPT to improve text fluency and implement some of the code for results visualization.

Acknowledgements

This work was supported by a grant, provided by the Ministry of Economic Development of the Russian Federation in accordance with the subsidy agreement (agreement identifier 000000C313925P4G0002) and the agreement with the Ivannikov Institute for System Programming of the Russian Academy of Sciences dated June 20, 2025 No. 139-15-2025-011.

We acknowledge the computational resources of the HPC facilities at HSE University.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Yunmo Chen, Tongfei Chen, Harsh Jhamtani, Patrick Xia, Richard Shin, Jason Eisner, and Benjamin Van Durme. 2024. Learning to retrieve iteratively for in-context learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 7156–7168. Association for Computational Linguistics.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. 2024. A survey on in-context learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1107–1128, Miami, Florida, USA. Association for Computational Linguistics.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online. Association for Computational Linguistics.

Gemma Team. 2024. Gemma.

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

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. 2020. Compositionality decomposed: How do neural networks generalise? *Journal of Artificial Intelligence Research*, 67:757–795.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2023. Bloom: A 176b-parameter open-access multilingual language model.

Sheng Lu, Hendrik Schuff, and Iryna Gurevych. 2024. How are prompts different in terms of sensitivity? In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5833–5856, Mexico City, Mexico. Association for Computational Linguistics.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered

prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098, Dublin, Ireland. Association for Computational Linguistics.

Sewon Min, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Noisy channel language model prompting for few-shot text classification. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5316–5330, Dublin, Ireland. Association for Computational Linguistics.

Nguyen Nhat Minh, Andrew Baker, Clement Neo, Allen G. Roush, Andreas Kirsch, and Ravid Shwartz-Ziv. 2025. Turning up the heat: Min-p sampling for creative and coherent LLM outputs. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Moran Mizrahi, Guy Kaplan, Dan Malkin, Rotem Dror, Dafna Shahaf, and Gabriel Stanovsky. 2023. State of what art? a call for multi-prompt llm evaluation. *arXiv preprint arXiv:2401.00595*.

Marius Mosbach, Tiago Pimentel, Shauli Ravfogel, Dietrich Klakow, and Yanai Elazar. 2023. Few-shot fine-tuning vs. in-context learning: A fair comparison and evaluation. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 12284–12314. Association for Computational Linguistics.

Aihua Pei, Zehua Yang, Shunan Zhu, Ruoxi Cheng, and Ju Jia. 2025. SelfPrompt: Autonomously evaluating LLM robustness via domain-constrained knowledge guidelines and refined adversarial prompts. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 6840–6854, Abu Dhabi, UAE. Association for Computational Linguistics.

Yao Qiang, Subhrangshu Nandi, Ninareh Mehrabi, Greg Ver Steeg, Anoop Kumar, Anna Rumshisky, and Aram Galstyan. 2024a. Prompt perturbation consistency learning for robust language models. In *Findings of the Association for Computational Linguistics: EACL 2024, St. Julian's, Malta, March 17-22, 2024*, pages 1357–1370. Association for Computational Linguistics.

Yao Qiang, Subhrangshu Nandi, Ninareh Mehrabi, Greg Ver Steeg, Anoop Kumar, Anna Rumshisky, and Aram Galstyan. 2024b. Prompt perturbation consistency learning for robust language models. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 1357–1370, St. Julian's, Malta. Association for Computational Linguistics.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi 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, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2024. Quantifying language models' sensitivity to spurious features in prompt design or: How I learned to start worrying about prompt formatting. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Oleg Somov and Elena Tutubalina. 2025. Confidence estimation for error detection in text-to-sql systems. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(23):25137–25145.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adri Garriga-Alonso, et al. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on machine learning research*.

Teknium. 2023. Openhermes 2.5: An open dataset of synthetic data for generalist llm assistants.

Joshua Vendrow, Edward Vendrow, Sara Beery, and Aleksander Madry. 2025. Do large language model benchmarks test reliability? *CoRR*, abs/2502.03461.

Anton Voronov, Lena Wolf, and Max Ryabinin. 2024a. Mind your format: Towards consistent evaluation of in-context learning improvements. *arXiv preprint arXiv:2401.06766*.

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

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

5085–5109, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Minghao Wu, Thuy-Trang Vu, Lizhen Qu, and Reza Haf. 2024. Mixture-of-skills: Learning to optimize data usage for fine-tuning large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 14226–14240, Miami, Florida, USA. Association for Computational Linguistics.

Haoran Yang, Yumeng Zhang, Jiaqi Xu, Hongyuan Lu, Pheng-Ann Heng, and Wai Lam. 2024a. Unveiling the generalization power of fine-tuned large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 884–899, Mexico City, Mexico. Association for Computational Linguistics.

Linyi Yang, Shuibai Zhang, Zhuohao Yu, Guangsheng Bao, Yidong Wang, Jindong Wang, Ruochen Xu, Wei Ye, Xing Xie, Weizhu Chen, and Yue Zhang. 2024b. Supervised knowledge makes large language models better in-context learners. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *International conference on machine learning*, pages 12697–12706. PMLR.

Han Zhou, Xingchen Wan, Lev Proleev, Diana Mincu, Jilin Chen, Katherine A. Heller, and Subhrajit Roy. 2024. Batch calibration: Rethinking calibration for in-context learning and prompt engineering. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Kaijie Zhu, Jindong Wang, Jiaheng Zhou, Zichen Wang, Hao Chen, Yidong Wang, Linyi Yang, Wei Ye, Yue Zhang, Neil Gong, and Xing Xie. 2024. 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 2024, Salt Lake City, UT, USA, October 14-18, 2024*, pages 57–68. ACM.

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

Andy Zou, Zifan Wang, J. Zico Kolter, and Matt Fredrikson. 2023. Universal and transferable adversarial attacks on aligned language models. *CoRR*, abs/2307.15043.

A Accuracy distribution with different sampling strategies

Figure 9 shows accuracy at various prompt formats for 6 models from Qwen and Llama families. Temperature sampling consistently decreases accuracy, while top-p and min-p sampling vary from being worse to being on par with greedy decoding. Thus, smaller spread of temperature sampling for Llama 3.2 1B Instruct is likely a consequence of lower performance.

B Consistency loss

Figure 10 provides some results for LoRA with consistency loss. Compared to LoRA with format augmentations, it shows higher spread and lower accuracy for all models.

C Extended Description of Methods

In this section, we provide a more detailed description of the methods used in the paper.

Batch Calibration (BC). Batch Calibration (Zhou et al., 2024) is a post-hoc approach, which calibrates model prediction with the estimate of contextual bias term $p(y \mid C)$. The contextual bias for each class $p(y = y_j|C)$ is estimated from a batch of B samples by marginalizing the output scores over all samples within the batch. The calibrated probabilities $\hat{y}_i$ are derived by shifting the log-probability $\log p(y|x_i, C)$ by the corresponding estimated mean of each class:

$$\forall y_j \in Y: \tag{2}$$

$$\overline{\log p(y|C)_j} = \frac{1}{B}\sum_{i=1}^{B} \log p(y = y_j|x^{(i)}, C),$$

$$\hat{y}^{(i)} = \underset{y \in Y}{\operatorname{argmax}} \left(\log p(y|x^{(i)}, C) - \overline{\log p(y|C)} \right).$$

The method was originally designed exclusively for classification, and is inapplicable to other tasks.

Template Ensembles (TE). Template Ensembles (Voronov et al., 2024b) average the predictions probability across various prompt formats f_i and select the class with the largest probability.

$$\hat{y} = \underset{y \in Y}{\operatorname{argmax}} \frac{1}{N} \sum_{i=1}^{N} p(y \mid x, f_i) \tag{3}$$

The main drawback of this method is the need to run the model N times, where N is the ensemble size.

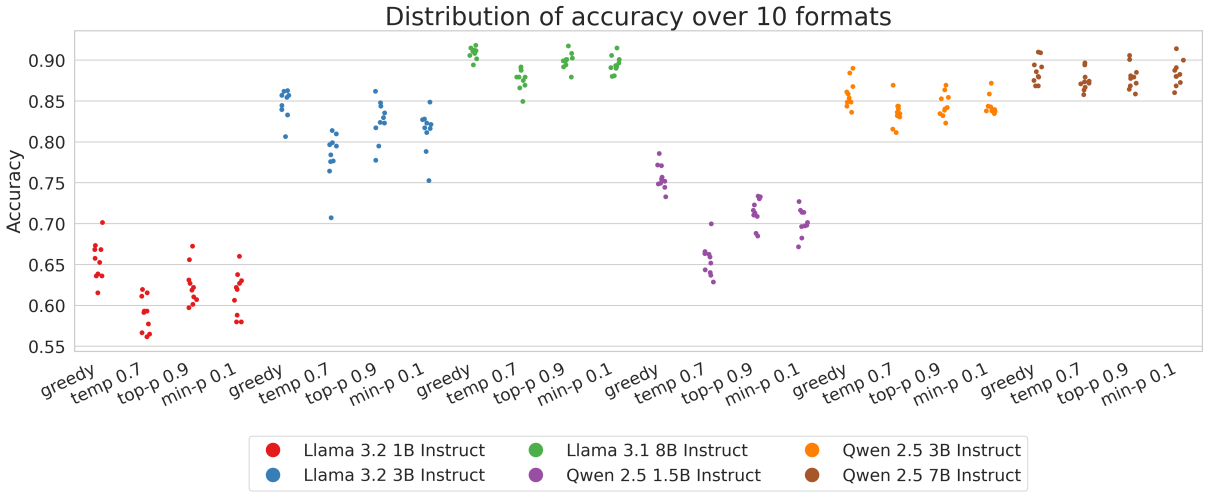


Figure 9: Comparing effect of sampling strategies on accuracy at various prompt formats on GSM8K dataset. Each dot is the accuracy of a single format computed over all GSM8K examples.

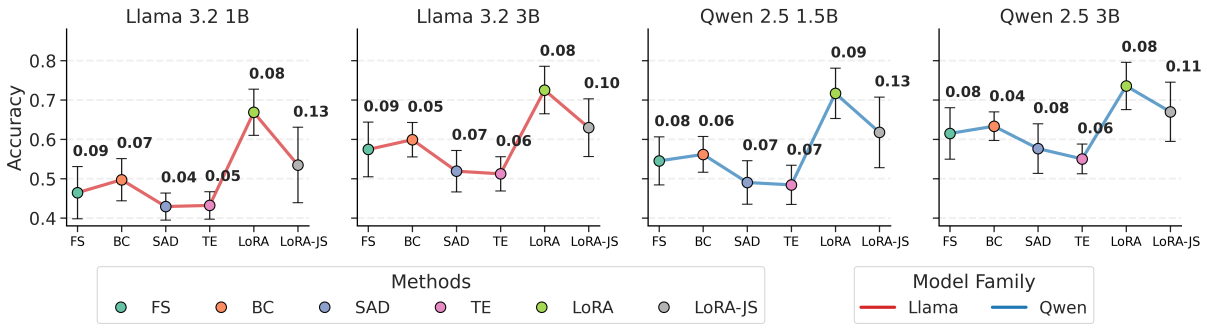


Figure 10: Comparing LoRA with consistency loss (LoRA-JS) with other methods in terms of their effect on accuracy and standard deviation over prompt formats. To aggregate accuracy, we first compute median accuracy over formats for each task, and then average over 52 tasks. Error bars are 2 × (standard deviation over formats, averaged across tasks).

Sensitivity-aware decoding (SAD). Sensitivity-aware decoding (Lu et al., 2024) assesses model sensitivity to input data by evaluating the variance of N predictions over modified inputs, using synthetic perturbations based on real data. The sensitivity value s is then used as a penalty in the greedy decoding process:

$$\hat{y} = \underset{y \in V}{\operatorname{argmax}} \left[\alpha P(y|x) - (1-\alpha)s \right], \qquad (4)$$

where $P(y|x)$ is the probability of an output y given x, α is the reweighting hyperparameter and V is vocabulary – the set of all tokens. In this paper we use a simplified version of sensitivity-aware decoding, using random token substitutions as a perturbation. This approach reduces model variance but also requires N times more runs.

LoRA with augmentations (LoRA). We fine-tune an instruction-finetuned model M on a small dataset D_{format} containing augmented samples. Here augmentation refers to changing the formatting while maintaining the same content. To build D_{format}, we select a subset of samples from a generic instruction-following dataset D_{source} and insert K augmented versions for each sample.

Parameter-efficient finetuning on D_{format} is conducted using a standard language modeling cross-entropy loss. Loss is only computed on answer tokens, while the prefix tokens are masked.

LoRA with consistency loss (LoRA-JS). One way to enforce consistent predictions is to use an auxiliary loss during fine-tuning. To test this approach, we reproduce *prompt perturbation consistency learning* (Qiang et al., 2024a).

The training objective contains supervised cross-entropy losses for pairs of augmented examples x_1, x_2 that share the same target label y, along with the consistency loss, based on Jensen-Shannon

divergence. Formally, the overall loss function is defined as:

$$\mathcal{L} = \text{CE}(\hat{y}_1, y) + \text{CE}(\hat{y}_2, y) + \beta \text{JS}(\hat{\bar{y}}_1 || \hat{\bar{y}}_2), \quad (5)$$

where CE denotes the cross-entropy loss, β is the coefficient controlling the contribution of the consistency loss, JS is Jensen-Shannon divergence, $\hat{y}_1, \hat{y}_2$ are the response token probability distributions, and $\hat{\bar{y}}_1$ and $\hat{\bar{y}}_2$ are corresponding distributions averaged over response length.

D Task selection

We selected tasks from Super-Natural Instructions following the criteria outlined in (Sclar et al., 2024).

A total of 52 evaluation tasks were selected from Super-Natural Instructions using several heuristics. First, datasets were required to contain at least 1000 samples. Then, tasks with long instructions (over 3000 characters) and inputs (over 2000 characters) were excluded to ensure scalability. Additionally, tasks with a predicted accuracy of 0% for LLaMA-2-7B 1-shot were removed, and no more than 4 tasks from the same dataset were included. Furthermore, socially significant tasks and formats for them were added if they were missing.

The selected tasks were the following 52: `task050`, `task065`, `task069`, `task070`, `task114`, `task133`, `task155`, `task158`, `task161`, `task162`, `task163`, `task213`, `task214`, `task220`, `task279`, `task280`, `task286`, `task296`, `task297`, `task316`, `task317`, `task319`, `task320`, `task322`, `task323`, `task325`, `task326`, `task327`, `task328`, `task335`, `task337`, `task385`, `task580`, `task607`, `task608`, `task609`, `task904`, `task905`, `task1186`, `task1283`, `task1284`, `task1297`, `task1347`, `task1387`, `task1419`, `task1420`, `task1421`, `task1423`, `task1502`, `task1612`, `task1678`, `task1724`.

`task190` fits all previous requirements, but was excluded due to labeling errors.

For evaluation of frontier models, we selected `task114`, `task161`, `task296`, `task320`, `task322`, `task323`, `task1387`, `task1419`, `task1420`, `task1423`, since they cover math, text compherension, simple tasks like counting the number of words with a given letter and socially significant topics like detecting racial stereotypes.

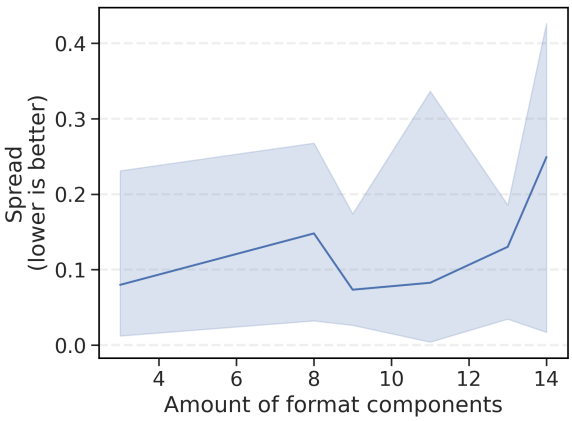


Figure 11: Empirical dependency between spread and amount of components in format. 90% confidence interval is based on percentiles.

E Dependence between spread and format complexity

Figure 11 shows the relation between spread and format complexity, measured as the amount of prompt components. While the dependence is quite noisy, maximal spread values occur at formats of maximal length.

F Use of scientific artifacts.

Artifact	License
Natural Instructions	Apache 2.0
Open Hermes 2.5	CC-BY-NC
Llama 3.1 8B	Llama 3.1 Community License Agreement
Llama 3.2 (1B, 3B)	Llama 3.2 Community License Agreement
Gemma 2 (2B, 9B)	Apache License 2.0
Qwen2.5 (1.5B, 3B, 7B)	Apache License 2.0

Table 5: Scientific artifacts used in this paper and their licenses.

In Table 5 we list the artifacts used in this paper along with their licenses. To the best of our knowledge, using these artifacts for research purposes is consistent with their intended use.

G Complete list of format components.

Complete list of format components is given in Table 6.

H Example of input prompt for frontier models

Example of input prompt used for frontier models in presented on Figure 12.

Modification	Values
Descriptor transformation	`lambda x: x.title(), lambda x: x.upper(), lambda x: x.lower(), lambda x: x`
Separator	”, ’::: ’, ’:: ’, ’: ’, ’ \n\t’, ’\n ’, ’ : ’, ’ - ’ , ’ ’, ’\n ’, ’\n\t’, ’:’, ’::’, ’- ’, ’\t’
Space	”, ’ ’, ’\n’, ’ \n’, ’ - ’, ’ ’, ’; \n’, ’ || ’, ’ <sep> ’, ’ - ’, ’, ’, ’ \n ’, ’ , ’, ’\n ’, ’. ’, ’ , ’
Text & option separator	”, ’ ’, ’ ’, ’\t’
Option item style	`1, 2, ...; A, B, ...; a, b, ...; I, II, ...; i, ii, ...`
Option item wrapper	({}); {}.; {}); {}); [{}]]; <{}>

Table 6: Descriptor transformation correspond to Python commands making first character upper case (title), all letters upper case (uppercase), all letters lower case (lowercase) or keeping input as is. Option item style includes Arabic numerals, uppercase and lowercase Latin letters and uppercase and lowercase Roman numerals. For option item wrapper, {} is used as a placeholder for option item (e.g. ‘a’ or ‘1’).

```
System: In this task, you need to answer 'Yes' if the given word is the longest
word (in terms of number of letters) in the given sentence, else answer 'No'. Note
that there could be multiple longest words in a sentence as they can have the same
length that is the largest across all words in that sentence. PAY ATTENTION TO
THE OUTPUT FORMAT - ONLY OUTPUT THE ANSWER WITHOUT ANY OTHER TEXT, LIKE IN EXAMPLES.

User: Sentence
'woman sitting on a chair holding three teddy bears'. Is 'a' the longest word in
the sentence?
Answer
No

Sentence
'a large green plant with leaves and spiky flowers'. Is 'flowers' the longest
word in the sentence?
Answer
Yes

Sentence
'a long white airplane covered with a lot pastel hears on it'. Is 'covered' the
longest word in the sentence?
Answer
```

Figure 12: Example of input prompt used for GPT-4.1 and DeepSeek V3 0324.

Learning rate	2e-4
LoRA α	16
LoRA Rank	16
Amount of epochs	1
Batch size	64
Weight decay	0.01

Table 7: LoRA training hyperparameters.

I Method hyperparameters

All LoRA models were trained with default hyperparameters, given in Table 7.

LoRA fine-tuning data. For experiments in Section 4.1 we construct D_{format} from our subset of Natural Instructions benchmark, choosing up to 1000 samples per task, disjoint with the test samples chosen before. For cross-domain experiments in 4.2 we use a custom fine-tuning dataset built from a subsample of the Open Hermes 2.5 dataset (Teknium, 2023). Open Hermes 2.5 contains synthetically generated tasks in the form of prompts for LLMs, covering various task types. Our research primarily focuses on classification tasks and multiple-choice questions. Although the dataset includes many such tasks, only a small portion has clearly defined labels. To find such examples, we selected those where the GPT response length does not exceed 20 symbols, as suitable tasks typically feature simple and concise answers. This threshold was determined empirically. Resulting dataset has approximately 50k samples.

Other methods' hyperparameters. In experiments with Template Ensembles and Sensitivity-aware decoding, we used ensembles of size 5, following (Voronov et al., 2024b).

The α parameter in Sensitivity-aware decoding was set to 0.7 based on Section A.7 in (Lu et al., 2024). To create synthetic inputs for sensitivity estimation, we replaced 15% of the original tokens with random tokens from the entire vocabulary.

Format 1	`Question: {} Answer: {}`
Format 2	`Question:: {} Answer:: {}`
Format 3	`QUESTION\n{}\nANSWER\n{}`
Format 4	`question - {} answer - {}`

Table 8: Some of augmentations used during LoRA finetuning.