Prompt-Robust Language Models: Which Training Strategies Work?

Anonymous ACL submission

Abstract

Despite their strong performance, large language models remain highly sensitive to prompt formulation. Prior work has addressed this issue through refined data construction or more complex robustness-oriented training methods. In this work, we reproduce and compare these strategies under controlled conditions, evaluating their effects on worst-case prompt sensitivity and average performance. We find that conventional instruction fine-tuning offers limited reductions in prompt sensitivity: across methods, the average gap between the best- and worst-performing prompts remains 40–57% of performance. Moreover, recent robustness-enhancing methods often fail to outperform simpler data construction strategies; in worst-case settings, at least one simple method performs better in 5 of 8 evaluations. Overall, our findings show that prompt sensitivity remains an open problem, and motivate practitioners to prioritize simpler data construction techniques before adopting more sophisticated robustness methods.

1 Introduction

Large language models (LLMs) are increasingly deployed in real-world decision-making pipelines, yet their performance remains highly sensitive to prompt formulation (Inger et al., 2025). Semantically equivalent prompts can yield drastically different outputs, making systems brittle and unreliable in practice (Cao et al., 2024). Addressing this at test-time through manual prompt engineering is labor-intensive and fundamentally does not scale (Schulhoff et al., 2025).

Train-time strategies, striving to create more robust models right from the training, offer a more principled solution: rather than patching sensitivity post-hoc, they build invariance directly into the model. Prior work has shown that tuning on multiple prompt formulations already outperforms standard single-prompt instruction finetuning (Wei et al., 2025, 2022), yet the design space remains poorly understood — it is unclear which data construction strategies matter, and whether more sophisticated robustness-enhancing methods provide meaningful gains on top.

We address this gap through a systematic study of train-time robustness methods improving on standard Instruction Finetuning (SIFT) (Wei et al., 2022). We analyze methods using multiple prompt formulations, specifically (i) data construction strategies keeping the original IFT objective (Wei et al., 2025), (ii) consistency regularization (Qiang et al., 2024), and (iii) contrastive approaches (Yan et al., 2024). With the last two adding computational overhead through additional loss factors. Our main contributions are:

- In controlled settings, we compare the effects of data construction techniques applied in IFT;
- We provide a reproduction study of training techniques aiming to improve robustness and critically compare their effectivity to cheaper methods.

Our work provides guidance for engineers and researchers, optimizing for prompt robustness, on which methods to prioritize when training production models.

2 Related Work

A large body of recent work documents that LLM performance **remain highly sensitive to a prompt formulation** regardless of model size and tasks (Habba et al., 2025; Sun et al., 2024; Zhu et al., 2024; Sclar et al., 2024; Mizrahi et al., 2024), with prompt paraphrases or minor structural changes causing substantial performance swings (Chatterjee et al., 2024; He et al., 2024; Zhuo et al., 2024). Critically, SIFT raises mean performance but leaves prompt sensitivity largely intact (Chatterjee et al.,

2024; Wei et al., 2025), indicating that sensitivity is not an artifact that is trivial to address.

Inference-time methods address brittleness through prompt optimization, model calibration, or model-based self-correction (Zhao et al., 2021; Hu et al., 2024; Shi et al., 2025; Zhan et al., 2024). While effective, these approaches add algorithmic overhead at deployment and do not generalize, requiring re-running at test time (Hu et al., 2024; Shi et al., 2025). **Train-time methods** instead build invariance into the model directly. Towards this goal, previous work experiments with multi-prompt instruction finetuning (MIFT) (Wei et al., 2025) and methods adding explicit robustness objectives such as consistency regularization (Qiang et al., 2024) and contrastive alignment (Yan et al., 2024) showing promise. However, these methods have not been evaluated together under controlled conditions.

We identify two recent studies most closely related to ours. Seleznyov et al. (2025) evaluate PPCL and MIFT, but restrict perturbations to minor structural changes (e.g. separators) and primarily assess inference-time approaches. Agrawal et al. (2025) include one train-time consistency regularization method (Sun et al., 2024) but still center on local, mechanistic prompt perturbations and primarily on test-time methods. Our study is the first to compare robustness-enhancing training techniques from three different categories on a benchmark of major prompt perturbations.

3 Experimental setup

Problem definition Given a set of datasets: $\{\mathcal{D}_1, ..., \mathcal{D}_n\}$ each containing a set of samples $\mathcal{D}_i = \{(X_{i,1}, Y_{i,1}), ..., (X_{i,m}, Y_{i,m})\}$ and each associated with a set of verbalization templates $\{\mathcal{T}_{i,1}, ..., \mathcal{T}_{i,k}\}$: $\mathcal{T}(X_{i,j}, Y_{i,j}) \rightarrow (x_{i,j}, y_{i,j})$ we generate an instruction fine-tuning dataset $X_{\text{IFT}} = \{(x_{1,1}, y_{1,1}), ..., (x_{1,m}, y_{1,m}), ..., (x_{n,m}, y_{n,m})\}$ composed of textual instructions x and expected responses y by applying dataset templates $\mathcal{T}_i$ on all X_i. During training, we iteratively update the trained model $\mathcal{M}$ according to batches of samples $\mathcal{B} : \{(x_1, y_1), ..., (x_{|\mathcal{B}|}, y_{|\mathcal{B}|})\}$ drawn from X_{IFT}. Previous work chooses different methods of constructing $\mathcal{B}$ or regularizing $\mathcal{M}$ with additional constraints.

The goal of our experiments is to compare the effectivity of these methods for the prompt robustness of resulting models. We have a separate train $\mathcal{D}_{\text{tr}}$ and test $\mathcal{D}_{\text{te}}$ dataset with disjunct tasks. We employ each training method with $\mathcal{D}_{\text{tr}}$ and evaluate its resulting model on $\mathcal{D}_{\text{te}}$, for each $\mathcal{D}_i \in \mathcal{D}_{\text{te}}$ reporting performance with (i) the best-performing template $\mathcal{T}_{i,\text{best}}$, (ii) the worst-performing template $\mathcal{T}_{i,\text{worst}}$, and (iii) as average performance across all templates $\mathcal{T}_i$.

3.1 Training Methods

We study three categories of train-time methods to improve prompt robustness, all making use of the availability of multiple $\mathcal{T}$s for each dataset.

(1) Data construction We test four strategies for how formulations are scheduled across training:

- SINGLE — Train on one template per dataset mirroring SIFT in FLAN (Wei et al., 2022)
- ALL SHUFFLED — all formulations for each example are shuffled randomly in the training
- ALL-IN-ONE-BATCH — all formulations for a given example are used in the same batch
- ONE-AT-A-TIME — All templates from each dataset are used in training, but only one per batch, mirroring Wei et al. (2025).

With these strategies we can test how varying prompt diversity per batch affects the models downstream performance and assess the optimality of strategies applied in previous work.

(2) Consistency regularization — PPCL Consistency regularization methods add an auxiliary loss that penalizes divergence between the model's output probabilities on semantically equivalent prompts. We select PPCL (Qiang et al., 2024) to represent this category, as it operates directly on the supervised finetuning objective and does not induce other covariates such as soft prompt infrastructure (Sun et al., 2024) or unsupervised label construction (Zhou et al., 2022; Hejabi et al., 2025). PPCL augments the standard cross-entropy loss, calculated for both prompt formulations, with a Jensen-Shannon divergence term computed over the average token probabilities of outputs across prompt formulations:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{CE\_orig} + \lambda_2 \mathcal{L}_{CE\_par} + \lambda_3 \cdot \mathcal{L}_{JSD} \quad (1)$$

where the λs control the trade-off between instruction following and instruction robustness.

(3) Contrastive alignment — CoIN Contrastive approaches go further by not only aligning equivalent prompts but also explicitly separating syntactically similar but semantically different ones. We select CoIN (Yan et al., 2024) as it operates on hidden representations and forms the basis of several subsequent approaches (Liu et al., 2025; Aissi et al., 2025). CoIN adds a contrastive loss over the model's internal representations:

$$\mathcal{L} = \mathcal{L}_{CE} + \lambda \cdot \mathcal{L}_{CoIN} \quad (2)$$

where $\mathcal{L}_{CoIN}$ is the contrastive objective over hidden states as defined in Yan et al. (2024).

Models We evaluate all methods on four models spanning two model families and two scales: Llama3.2-1B and Llama3.1-8B (Grattafiori et al., 2024) and Qwen3-0.6B and Qwen3-8B (Yang et al., 2025). Using two families allows us to assess whether findings generalize across architectures, while the two scales test whether robustness effects are consistent across model capacity. For all models we use the base variants, ensuring that observed robustness differences are attributable to our training methods rather than prior instruction tuning. For the 8B models we apply LoRA (Hu et al., 2021) to reduce computational cost. For each model and loss function we sweep a separate learning rate; full hyperparameter details are in Appendix B.

Datasets We replicate the data setup from Sanh et al. (2022) testing for strict generalization by training on 48 datasets and testing on 11 dataset of strictly unseen tasks. As a collection of prompt formulations we use the PromptSource (Bach et al., 2022), which covers paraphrasing and structural changes for all our datasets — the perturbation types models are most sensitive to (Chatterjee et al., 2024). We manually checked all our applied templates and filtered the ones that were inapplicable or changed the task type. Datasets with only a single remaining template were removed (see Appendix A). Training dataset sizes are capped at 10,240 examples to balance contributions across tasks. CoIN and PPCL require all prompt formulations to share the same label, so we sample the largest subset of templates whose expected labels agree. We call this the majority templates. To measure how much this constraint affects performance, we also train the ALL-SHUFFLED method using only the majority templates.

Metrics Following Wang et al. (2022), we use Rouge-L (Lin, 2004) as our main metric. Each method is assessed on three metrics relative to the single-template IFT baseline: average, worst-case and best-case template performance (Sec. 3). Statistical significance between methods is assessed using pairwise bootstrapped confidence intervals at $\alpha = 0.05$.

4 Results

Model robustness Figure 1 reveals that smaller models are disproportionately sensitive to prompt formulation. For Llama-1B, single-template IFT yields a worst-to-best spread of 41.89 Rouge-L percentage points, indicating substantial instability. The larger models present a more nuanced picture: while Qwen3-8B exhibits a comparable variance span (33.69 vs. 31.02 percentage points), Llama3.1-8B displays considerably lower sensitivity to prompt formulation after single-template training than all other models examined.

Across models, the evaluated methods generally improve worst-case, average, and best-case performance relative to the single-template baseline. This is expected as the model sees more varianced data and is trained for more steps through the multiple formulations increasing the dataset size. Still, the gains are modest relative to the remaining performance variance reaching 40–57% across methods (see Figure 2 in Appx. C). The exception is Llama3.1-8B, where several methods produce statistically significant *decreases* in worst-case performance. We attribute this to two factors: (1) the model's already strong single-template baseline reduces the ceiling for robustness gains, and (2) as indicated by training logs, the majority of performance improvement occurs within the first 1,000 steps, suggesting that additional prompt variation yields diminishing returns for this model.

RQ1: How does data construction affect robustness? Prior work constructs multi-prompt training data using a sequential per-formulation schedule (ONE-AT-A-TIME; Wei et al. 2025), but it is unclear whether this specific construction is necessary or optimal. Figure 1 shows that for smaller models, ONE-AT-A-TIME indeed yields consistent improvements, achieving statistically best or equivalent-to-best performance across worst-case, average, and best-case metrics.

ALL-IN-ONE-BATCH narrows the performance spread across formulations compared to ONE-AT-

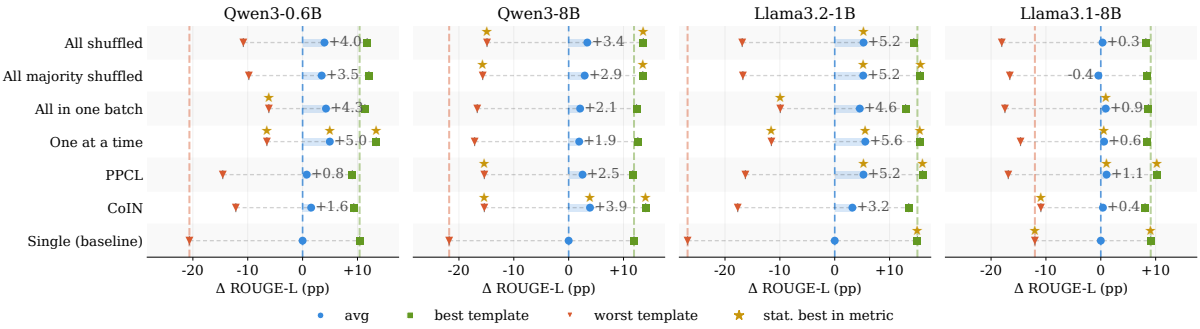

Figure 1: Performance of each method relative to SIFT in percentage points across models. The circle, square, and triangle mark the average, best, and worst template score respectively; the dashed span reflects sensitivity to prompt variance. Vertical lines extending from SIFT serve as reference. A star above a marker indicates membership in the statistically best group. See Table 5 for the raw scores.

A-TIME, but it does so primarily by degrading best-case performance rather than lifting the worst case. We hypothesize that the counterintuitive success of ONE-AT-A-TIME may be caused by that training on all formulations simultaneously leads to interfering gradients rather than the emergence of prompt-agnostic structure. This effect is less pronounced in larger models, where the difference between ONE-AT-A-TIME and fully shuffled schedules is not statistically significant.

Comparing ALL-SHUFFLED against ALL-MAJORITY-SHUFFLED reveals no meaningful performance difference. Although one might expect broader formulation coverage to improve robustness, Appendix A shows that for most datasets the majority-template subset constitutes the bulk of available templates, making the practical data difference minor. Overall, the shuffled variants perform comparably to the structured scheduling approaches, with the structured approaches retaining a modest but consistent advantage on worst-case performance (Figure 2).

RQ2: Is the additional complexity of addressed robustness methods justified? PPCL and COIN both yield only modest gains compared to the baseline (Figure 2). These gains are consistently matched or exceeded by simpler data construction strategies (Figure 1). The sole exception is COIN on Qwen3-8B, where it achieves the highest average performance of all evaluated methods; on all remaining models, COIN and PPCL are either statistically indistinguishable from or inferior to the best data augmentation approach. In 5 out of 8 cases, there is always at least one simple data construction method providing more robust model.

This finding is noteworthy given the higher implementation cost of these methods. PPCL requires computing a Jensen-Shannon divergence term across each label at each training step, while COIN imposes strict batch composition constraints requiring two examples with three different prompt formulations. Against this overhead, the marginal robustness benefit — absent for three of four models — is difficult to justify.

5 Conclusion

This paper provides a systematic comparison of train-time strategies for improving prompt robustness of LLMs. Our results show all multi-prompt methods improve robustness over single-template IFT, yet the gains remain modest relative to the underlying variance caused by prompt formulation.

We find that the differences among data construction strategies are subtle, but ONE-AT-A-TIME scheduling provides the most reliable worst-case gains, particularly for smaller models. On the other hand, our reproductions and comparisons of recent robustness-enhancing methods shows that these methods are mostly matched or are outperformed by simple data augmentation, making their additional complexity difficult to justify. Our results advise practitioners seeking prompt-robust models to prioritize simple ONE-AT-A-TIME or ALL-SHUFFLED multi-prompt training schedule, depending on data availability, before investing in consistency regularization or contrastive objectives.

Taken together, our findings presents the prompt robustness as an open challenge, where even the best train-time methods leave ample space for improvement. We hope that our results will motivate future work towards this challenge and towards more cautious reproductions of prior findings.

Limitations

Model scale and family coverage. Our experiments are limited to models up to 8B parameters across two model families (Llama and Qwen), due to computational constraints. It remains an open question whether the observed patterns — in particular the advantage of ONE-AT-A-TIME scheduling and the marginal benefit of auxiliary loss objectives — hold at larger scales where models may be more capable of leveraging richer training signals. Expanding the evaluation to additional architectures would further strengthen the generalizability of our conclusions.

Benchmark scope and evaluation protocol. We adopt the training and evaluation split of Sanh et al. (2022), which, while well-established, consists predominantly of classification and short-answer tasks with closed label sets. In such settings, prompt sensitivity may manifest differently than in open-ended generation, so evaluating on more diverse benchmarks — including long-form generation and reasoning tasks — would provide a more complete picture of prompt robustness.

Method selection within each category. We evaluate a single representative method for each of the consistency regularization and contrastive alignment categories — PPCL and CoIN, respectively — selected based on their central standing in those categories. Alternative methods within these categories (Sun et al., 2024; Zhou et al., 2022; Hejabi et al., 2025; Liu et al., 2025; Aissi et al., 2025) may exhibit different trade-offs, and our conclusions should be interpreted as applying to the specific instantiations studied rather than to the broader methodological families.

Hyperparameter optimization. Although we performed a grid search over learning rates for all methods and tuned the PPCL regularization weight on one model, a more exhaustive search — particularly for the auxiliary loss weights of PPCL and CoIN — could yield stronger performance for these methods and may affect the relative comparisons reported here. The computational cost of such a search across all models and methods was prohibitive within our resource budget.

References

Aryan Agrawal, Lisa Alazraki, Shahin Honarvar, and Marek Rei. 2025. Enhancing LLM robustness to perturbed instructions: An empirical study. *ArXiv*, abs/2504.02733.

Mohamed Salim Aissi, Clément Romac, Thomas Carta, Sylvain Lamprier, Pierre-Yves Oudeyer, Olivier Sigaud, Laure Soulier, and Nicolas Thome. 2025. Reinforcement learning for aligning large language models agents with interactive environments: Quantifying and mitigating prompt overfitting. In *Findings of the association for computational linguistics: NAACL 2025*, pages 7030–7046, Albuquerque, New Mexico. Association for Computational Linguistics.

Stephen Bach, Victor Sanh, Zheng-Xin Yong, Albert Webson, Colin Raffel, Nihal V Nayak, Abheesht Sharma, Taewoon Kim, M Saiful Bari, Thibault Fevry, and others. 2022. Promptsource: An integrated development environment and repository for natural language prompts. In *Proceedings of the 60th annual meeting of the association for computational linguistics: System demonstrations*, pages 93–104.

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

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

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. ArXiv: 2407.21783 [cs.AI].

Eliya Habba, Ofir Arviv, Itay Itzhak, Yotam Perlitz, Elron Bandel, Leshem Choshen, Michal Shmueli-Scheuer, and Gabriel Stanovsky. 2025. DOVE: a large-scale multi-dimensional predictions dataset towards meaningful LLM evaluation. In *Findings of the association for computational linguistics: ACL 2025*, pages 11744–11763. Association for Computational Linguistics.

Jia He, Mukund Rungta, David Koleczek, Arshdeep Sekhon, Franklin X Wang, and Sadid Hasan. 2024. Does prompt formatting have any impact on LLM performance? ArXiv: 2411.10541 [cs.CL].

Parsa Hejabi, Elnaz Rahmati, Alireza S. Ziabari, and Morteza Dehghani. 2025. Flip-flop consistency: Unsupervised training for robustness to prompt perturbations in llms. ArXiv: 2510.14242 [cs.CL].

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-rank adaptation of large language models. ArXiv: 2106.09685 [cs.CL].

Wenyang Hu, Yao Shu, Zongmin Yu, Zhaoxuan Wu, Xiaoqiang Lin, Zhongxiang Dai, See-Kiong Ng, and Bryan Kian Hsiang Low. 2024. Localized zeroth-order prompt optimization. *Advances in Neural Information Processing Systems*, 37:86309–86345.

Nurit Cohen Inger, Yehonatan Elisha, Bracha Shapira, Lior Rokach, and Seffi Cohen. 2025. Forget what you know about llms evaluations-llms are like a chameleon. In *Proceedings of the 2025 conference on empirical methods in natural language processing*, pages 21675–21688.

Chin-Yew Lin. 2004. ROUGE: a package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Shuhan Liu, Xing Hu, Kerui Huang, Xiaohu Yang, David Lo, and Xin Xia. 2025. Improving code LLM robustness to prompt perturbations via layer-aware model editing. *ArXiv*, abs/2507.16407.

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

Yao Qiang, Subhrangshu Nandi, Ninareh Mehrabi, Greg Ver Steeg, Anoop Kumar, Anna Rumshisky, and Aram Galstyan. 2024. 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.

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, and 21 others. 2022. Multitask prompted training enables zero-shot task generalization. In *International conference on learning representations*.

Sander Schulhoff, Michael Ilie, Nishant Balepur, Konstantine Kahadze, Amanda Liu, Chenglei Si, Yinheng Li, Aayush Gupta, HyoJung Han, Sevien Schulhoff, Pranav Sandeep Dulepet, Saurav Vidyadhara, Dayeon Ki, Sweta Agrawal, Chau Pham, Gerson Kroiz, Feileen Li, Hudson Tao, Ashay Srivastava, and 12 others. 2025. The prompt report: a systematic survey of prompt engineering techniques. ArXiv: 2406.06608 [cs.CL].

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 *International conference on learning representations*, volume 2024, pages 25055–25083.

Mikhail Seleznyov, Mikhail Chaichuk, Gleb Ershov, Alexander Panchenko, Elena Tutubalina, and Oleg Somov. 2025. When Punctuation Matters: A Large-Scale Comparison of Prompt Robustness Methods for LLMs. *arXiv preprint*. ArXiv:2508.11383 [cs].

Zeru Shi, Zhenting Wang, Yongye Su, Weidi Luo, Hang Gao, Fan Yang, Ruixiang Tang, and Yongfeng Zhang. 2025. Robustness-aware Automatic Prompt Optimization. *arXiv preprint*. ArXiv:2412.18196 [cs].

Jiuding Sun, Chantal Shaib, and Byron Wallace. 2024. Evaluating the zero-shot robustness of instruction-tuned language models. In *International conference on learning representations*, volume 2024, pages 48103–48141.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Gary Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Maitreya Patel, and 21 others. 2022. Super-NaturalInstructions: Generalization via declarative instructions on 1600+ NLP tasks. ArXiv: 2204.07705 [cs.CL].

Chenxing Wei, Yao Shu, Mingwen Ou, Ying He, and Fei Yu. 2025. PAFT: Prompt-agnostic fine-tuning. In *Proceedings of the 2025 conference on empirical methods in natural language processing*, pages 694–717.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned Language Models Are Zero-Shot Learners. *arXiv preprint*. ArXiv:2109.01652 [cs].

Tianyi Yan, Fei Wang, James Y Huang, Wenxuan Zhou, Fan Yin, Aram Galstyan, Wenpeng Yin, and Muhao Chen. 2024. Contrastive instruction tuning. In *Findings of the association for computational linguistics: ACL 2024*, pages 10288–10302.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. ArXiv: 2505.09388 [cs.CL].

Pengwei Zhan, Zhen Xu, Qian Tan, Jie Song, and Ru Xie. 2024. Unveiling the lexical sensitivity of llms: Combinatorial optimization for prompt enhancement. In *Conference on empirical methods in natural language processing*.

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.

Chunting Zhou, Junxian He, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. Prompt consistency for zero-shot task generalization. In *Findings of the association for computational linguistics: EMNLP 2022*, pages 2613–2626, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Kaijie Zhu, Qinlin Zhao, Hao Chen, Jindong Wang, and Xing Xie. 2024. Promptbench: A unified library for evaluation of large language models. *Journal of Machine Learning Research*, 25(254):1–22.

Jingming Zhuo, Songyang Zhang, Xinyu Fang, Haodong Duan, Dahua Lin, and Kai Chen. 2024. ProSA: Assessing and understanding the prompt sensitivity of llms. In *Conference on empirical methods in natural language processing*.

A Dataset statistics

Table 1 shows an overview of all training datasets mirroring the T0 Sanh et al. (2022) training split. We replaced tydiqa with Zaid/coqa_expanded and gigaword with scitldr as the versions on huggingface where not compatible with most prompt templates in prompt source.

Afterward we manually filter out all templates which are not applicable to the examples or change the task type. For example, some templates reformulated HellaSwag examples as topic classification problems, which would cause train/test leakage.

After removing these templates we needed to remove social_i_qa, riddle_sense, and wiki_bio as they did not have more than one template remaining.

B Hyperparameters

Table 3 lists the hyperparameters shared across all models and methods. Table 4 reports the learning rates selected per model and method via grid search over $\{10^{-5}, 3\times10^{-5}, 5\times10^{-5}, 7\times10^{-5}, 9\times10^{-5}, 10^{-4}\}$. The λ_3 hyperparameter for PPCL was tuned over $\{0.1, 1, 10, 100\}$ for Llama-1B, with the best value being **1**. The COIN hyperparameters τ and λ were set to the advised values from Yan et al. (2024) and not tuned further due to computational constraints. Due to compute constraints we only sweep for the first 2000 training steps and take the best value from there.

All experiments are run on H200, and we trained for roughly 2800 GPU hours.

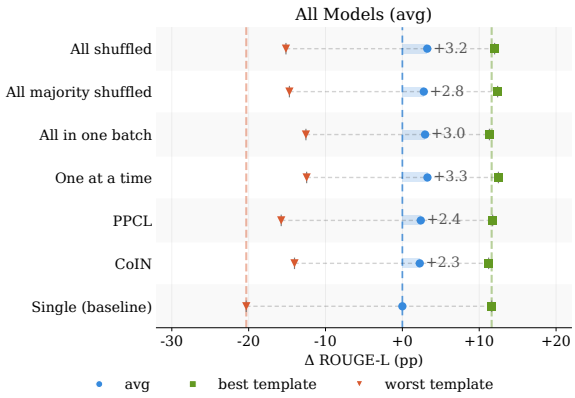

Figure 2: Performance of each method relative to SIFT in percentage points averaged across models. The circle, square, and triangle mark the average, best, and worst template score respectively; the dashed span reflects sensitivity to prompt variance. Vertical lines extending from SIFT serve as reference. See Table 5 for the raw scores

C Additional Results

D Statement on AI use and Risks

The authors used AI-assisted writing tools for language editing. All content was verified by the authors.

The authors do not see direct risks from publishing the paper as all used data is from well-known benchmarks used before.

Table 1: The 48 training datasets grouped by task type. *Templates* is the total number of PromptSource templates per dataset; *Filtered* is the count after removing manually identified off-task templates; *Majority* is the subset whose template applies to most examples.

Task	Dataset	Samples	Templates	Filtered	Majority
Paraphrase	GLUE (MRPC)	3,668	5	5	4
	GLUE (QQP)	10,240	6	6	4
	PAWS	10,240	11	10	10
Extractive QA	HotpotQA	10,240	5	5	5
	TriviaQA	10,240	5	4	1
	WebQuestions	3,778	5	5	1
	WikiQA	10,240	5	5	4
	AdvQA (BiDAF)	10,000	5	4	4
	AdvQA (BERT)	10,000	5	4	4
	AdvQA (RoBERTa)	10,000	5	4	4
	DuoRC (Self)	10,240	7	5	1
	DuoRC (Paraphrase)	10,240	7	5	1
	ROPES	10,240	12	12	12
	SQuAD v2	10,240	8	4	2
	SuperGLUE (ReCoRD)	10,240	20	7	0
	Quoref	10,240	11	10	1
	CoQA	10,240	7	7	6
MC QA	ARC-Challenge	1,119	6	5	3
	ARC-Easy	2,251	6	5	3
	CoS-E	9,741	11	6	1
	CosmosQA	10,240	13	8	4
	DREAM	6,116	5	4	2
	OpenBookQA	4,957	7	7	6
	PIQA	10,240	11	6	3
	QASC	8,134	8	6	6
	QuAIL	10,240	13	11	6
	QuaRel	1,941	5	5	5
	QuaRTz	2,696	8	8	8
	RACE (High)	10,240	8	5	3
	RACE (Middle)	10,240	8	5	3
	SciQ	10,240	5	5	5
	SuperGLUE (BoolQ)	9,427	10	5	2
	SuperGLUE (MultiRC)	10,240	10	10	10
	WIQA	10,240	8	4	2
Sentiment	Amazon Polarity	10,240	9	9	3
	App Reviews	10,240	4	4	1
	IMDB	10,240	11	11	7
	Rotten Tomatoes	8,530	10	10	7
	Yelp Reviews	10,240	7	7	5
Summarization	CommonGen	10,240	9	7	7
	CNN/DailyMail	10,240	9	7	7
	SciTLDR	1,992	6	5	5
	MultiNews	10,236	6	5	5
	SAMSum	10,240	7	6	5
	XSum	10,240	10	10	10
Topic Classification	AG News	10,240	7	7	4
	DBpedia	10,240	4	4	4
	TREC	5,452	7	5	5
Total (48 datasets)		**417,238**			

Table 2: The 11 zero-shot evaluation datasets grouped by task type. Column definitions are the same as in Table 1. For datasets without public test labels we use the official validation split.

Task	Dataset	Samples	Templates	Filtered	Majority
NLI	SuperGLUE (CB)	56	15	15	8
	SuperGLUE (RTE)	277	10	10	9
	ANLI (R1)	1,000	15	15	8
	ANLI (R2)	1,000	15	15	8
	ANLI (R3)	1,200	15	15	8
WSD / Coref	SuperGLUE (WiC)	638	10	10	9
	SuperGLUE (WSC)	104	10	10	7
Commonsense	WinoGrande	1,767	6	6	5
	SuperGLUE (COPA)	100	8	8	8
	StoryCloze	1,871	6	5	5
	HellaSwag	10,003	11	5	3
Total (11 datasets)		**18,016**			

Table 3: Shared hyperparameters for all experiments. †LoRA is only applied to the 8B models.

Hyperparameter	Value
Optimizer	AdamW (fused)
LR schedule	Constant with linear warmup
Warmup steps	100
Effective batch size	256
Epochs	1
Precision	bfloat16
Max train samples / dataset	10 240
LoRA rank†	16
LoRA alpha†	32
LoRA dropout†	0.0
LoRA target modules†	`q_proj`, `v_proj`, `output_proj`, MLP
COIN τ	0.05
COIN λ	1000
PPCL λ_1, λ_2	1, 1
PPCL λ_3	1

Table 4: Learning rates selected per model and method. All template variants (single, all, etc.) share the same standard loss function, therefore we tune the learning rate only once.

Model	Standard Loss	COIN	PPCL
Llama-3.2-1B	5×10^{-5}	7×10^{-5}	3×10^{-5}
Llama-3.1-8B	5×10^{-5}	9×10^{-5}	10^{-4}
Qwen3-0.6B	3×10^{-5}	5×10^{-5}	10^{-5}
Qwen3-8B	10^{-5}	5×10^{-5}	10^{-5}

Table 5: Raw ROUGE-L scores on the test set. For each model we report the mean across all templates (*avg*), the best-template score (*best*), and the worst-template score (*worst*). **Bold** values belong to the statistically best group per column under pairwise bootstrapped testing.

	Qwen3-0.6B			Qwen3-8B			Llama3.2-1B			Llama3.1-8B		
Method	avg	best	worst	avg	best	worst	avg	best	worst	avg	best	worst
Single (baseline)	0.514	0.618	0.308	0.634	0.753	0.416	0.479	**0.629**	0.210	0.639	**0.730**	**0.519**
All shuffled	0.554	0.632	0.406	0.668	**0.770**	**0.485**	**0.531**	0.623	0.310	0.642	0.721	0.458
All majority shuffled	0.549	0.635	0.416	0.663	**0.769**	**0.477**	**0.531**	**0.635**	0.311	0.634	0.722	0.473
All in one batch	0.556	0.627	**0.452**	0.655	0.758	0.467	0.524	0.608	**0.379**	**0.648**	0.726	0.464
One at a time	**0.563**	**0.648**	**0.449**	0.653	0.760	0.462	**0.534**	**0.634**	**0.363**	**0.645**	0.723	0.493
PPCL	0.521	0.605	0.368	0.659	0.751	**0.480**	**0.531**	**0.639**	0.316	**0.649**	**0.741**	0.470
CoIN	0.529	0.607	0.392	**0.673**	**0.774**	**0.480**	0.511	0.613	0.302	0.643	0.720	**0.530**