

UC Davis

UC Davis Electronic Theses and Dissertations

Title

Towards More Performant Large Language Models via Data-Model Synergy

Permalink

<https://escholarship.org/uc/item/29r0z4zn>

ISBN

9798189226731

Author

Askari, Hadi

Publication Date

2026-06-25

Peer reviewed|Thesis/dissertation

Towards More Performant Large Language Models via Data-Model Synergy

By

HADI ASKARI
DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Computer Science

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

Muhao Chen, Chair

Anshuman Chhabra

Zhe Zhao

Committee in Charge

2026

To my parents, Syeda Aijaz Fatima and Hasan Askari.

Contents

List of Figures	vi
List of Tables	viii
Abstract	xi
Acknowledgments	xiii
Chapter 1. Introduction	1
1.1. The Rise of Large Language Models and the Post-Training Challenge	1
1.2. Data-Model Synergy as a Unifying Principle	2
1.3. Chapter II, Utility: Data-Driven Layer Quality Estimation	3
1.4. Chapter III, Generalization: Influence-Guided Demonstration Selection	4
1.5. Chapter IV, Robustness: Characterizing Failure Modes in LLM Summarization	6
1.6. Chapter V, Reliability: Training Robust Tool Planners	8
1.7. Contributions and Dissertation Outline	9
Chapter 2. Utility: Data-Driven Layer Quality Estimation	11
2.1. Introduction	11
2.2. Related Work	12
2.3. The LayerIF Framework	14
2.4. Application I: Expert Allocation in LoRA-MoE Fine-Tuning	18
2.5. Application II: Layer-Wise Sparsity Allocation for Pruning	19
2.6. Experimental Setup	20
2.7. Results on Expert Allocation in LoRA-MoE Fine-Tuning	22
2.8. Results on Layer-Wise Sparsity Allocation for Pruning	24
2.9. Summary of Findings	25

2.10.	Analysis and Discussion	26
2.11.	Limitations	30
2.12.	Broader Implications	30
Chapter 3.	Generalization: Influence-Guided Demonstration Selection	32
3.1.	Introduction	32
3.2.	Related Work	34
3.3.	Preliminaries	36
3.4.	The Indirect ICL Framework	39
3.5.	Experimental Setup	43
3.6.	Results on Mixture-of-Tasks	44
3.7.	Results on Noisy ICL	48
3.8.	Analysis and Discussion	49
3.9.	Limitations	57
3.10.	Conclusion	59
Chapter 4.	Robustness: Characterizing Failure Modes in LLM Summarization	60
4.1.	Introduction	60
4.2.	Background and Related Work	61
4.3.	Shared Problem Setting: Zero-Shot Abstractive Summarization	62
4.4.	Study I: Position Bias in Zero-Shot Summarization	63
4.5.	Study II: Robustness Through Relevance Paraphrasing	65
4.6.	Experimental Setup	67
4.7.	Results: Study I: Position Bias in Zero-Shot Summarization	68
4.8.	Results: Study II: Robustness Through Relevance Paraphrasing	71
4.9.	Discussion	74
4.10.	Additional Validating Analyses	78
4.11.	Limitations	79
4.12.	Conclusion	79
Chapter 5.	Reliability: Avoidance Learning of Failure Modes for Robust Tool Planning	80

5.1. Introduction	80
5.2. Related Work	83
5.3. Method	85
5.4. Experiments	92
5.5. Discussion	96
5.6. Qualitative Analysis	98
5.7. Limitations	100
5.8. Conclusion	101
Chapter 6. Concluding remarks and future work	102
Bibliography	106

List of Figures

2.1	Overview of the LAYERIF framework. LAYERIF computes task-specific layer-quality estimates by localizing influence-function analysis to individual Transformer blocks. These scores are then mapped to downstream post-training decisions such as expert allocation in LoRA-MoE fine-tuning and layer-wise sparsity allocation for pruning.	15
2.2	Comparison of ALPHALORA, MoLA, and LayerIF under varying total expert budgets (80, 160, and 224) on Mistral-7B-v0.1.	23
2.3	Mean accuracy across sparsity levels from 20% to 50% for Mistral-7B-v0.1 pruned with SparseGPT using different layer-wise sparsity allocation methods.	25
2.4	Heatmap of expert allocations across Transformer layers for Mistral-7B-v0.1 with a total of 160 experts. Unlike ALPHALORA and MoLA, which apply a fixed allocation pattern across tasks, LayerIF produces dataset-specific allocation profiles. Darker shades indicate higher expert allocation to a given layer.	27
3.1	Example showcasing demonstration selection for Indirect ICL using Influence Functions (IFs). Consider web corpora with many tasks (different from the end-task) and noisy data– Indirect ICL can be formalized as: <i>Mixture of Tasks</i> (Section 3.4.2) and <i>Noisy</i> (Section 3.4.3) ICL, respectively. In MoT, for a given target task (e.g. <i>Medical Genetics</i>), we first filter from this (indirect) pool of candidate demonstrations using BertScore and Cosine Similarity, then re-rank with IFs to select suitable demonstrations (e.g. <i>High-School Biology</i>). For Noisy ICL, we leverage IFs to filter out the Noisy ICL before conducting ICL with the remaining clean demonstrations.	40
3.2	Average performance of demonstration-selection methods across Llama-2-13B-Chat, Zephyr-7B-Beta, and Mistral-7B-v0.3 for $k = 3$ shots in the Mixture-of-Tasks setting.	45

3.3	Average performance of demonstration-selection methods across the four Noisy ICL datasets.	49
4.1	An example of <i>position bias</i> where gold summary is <i>tail biased</i> and model summary is <i>lead biased</i>	64
4.2	An example showcasing <i>relevance paraphrasing</i> . When sentences <i>relevant</i> to generating the summary are <i>paraphrased</i> to create a minimally perturbed article, we find that summarization performance drops as the model uses other sentences instead to craft the summary, leading to a loss of salient information.	65
4.3	Visualizing positional distributions of gold and model-generated summaries for multiple datasets. The more these distributions differ for a given dataset and model, the greater the model’s position bias.	69
4.4	Measuring summarization performance (R^1 score) and position bias (Wasserstein distance between gold and generated summaries’ positional distributions). Lower Wasserstein distance values correspond to lower position bias.	70
4.5	Additional results for R^2 and R^L metrics.	71
4.6	Paraphrasing results in different summaries.	73
4.7	Paraphrasing results in different summaries for temperature set to 0.	77
5.1	The FORTE training framework.	82
5.2	Effect of FORTE’s contrastive negatives.	97

List of Tables

2.1	Accuracy across datasets using ALPHALORA, MoLA, and LayerIF variants. We report scores under three LayerIF aggregation strategies: Top 25% positively influential samples, all positively influential samples (+ve), and all samples. Best average performance is shown in bold.	22
2.2	Accuracy across datasets using ALPHALORA, MoLA, and LAYERIF (+ve) for 160 experts on Gemma-7B.	23
2.3	Comparison of average accuracies across different pruning strategies (Magnitude, Wanda, and SparseGPT) and layer-wise sparsity allocation methods at 50% pruning on Mistral-7B-v0.1. Best results are shown in bold.	24
2.4	Comparison of average accuracies across different pruning strategies (Magnitude, Wanda, and SparseGPT) and layer-wise sparsity allocation methods at 50% pruning on Gemma-7B.	25
2.5	Comparison of accuracies across pruning strategies at 70% sparsity on Mistral-7B-v0.1 using LayerIF versus a reversed allocation strategy.	27
2.6	Spearman correlation between layer allocations produced by the Hessian-based DataInf estimator and the Hessian-free TracIn estimator across datasets.	29
3.1	Performance across datasets and demonstration-selection methods with $k = 5$ shots for Llama-2-13B-Chat in the Mixture-of-Tasks setting.	46
3.2	Performance comparison between zero-shot inference (No ICL) and $k = 3$ demonstrations selected via the surrogate-model approach on the closed-source model GPT-4.1-nano in the Mixture-of-Tasks setting.	47
3.3	Performance comparison between MOICL, DETAIL, INFICL, CONE, and SUR _[D,BSR] for Llama-2-13B-Chat with $k = 3$ shots in the Mixture-of-Tasks setting.	47

3.4	ICL accuracy across MRPC, QNLI, SST2, and QQP using different methods for Noisy ICL with 20% label noise.	48
3.5	Performance comparison between BSR, $\text{SUR}_{[\text{D},\text{BSR}]}$, and $\text{PRE}_{[\text{D},\text{BSR}]}$ for Qwen2.5-3B with $k = 3$ shots in the Mixture-of-Tasks setting.	51
3.6	Performance comparison between BSR, $\text{SUR}_{[\text{D},\text{BSR}]}$, $\text{PRE}_{[\text{D},\text{BSR}]}$, COS, $\text{SUR}_{[\text{D},\text{Cos}]}$, and $\text{PRE}_{[\text{D},\text{Cos}]}$ for Llama-3-70B with $k = 3$ shots in the Mixture-of-Tasks setting. . .	51
3.7	Performance comparison between BSR and $\text{SUR}_{[\text{D},\text{BSR}]}$ when DeBERTa-v3-Large is used as the surrogate model on Llama-2-13B-Chat with $k = 3$ shots.	52
3.8	Percentage of noisy samples among the top 100 examples ranked by influence in the Noisy ICL setting.	52
3.9	Defense results on AddSent and Style backdoor attacks. ASR denotes attack success rate; lower is better.	56
3.10	Worst-case per-test-sample time complexity of the main demonstration-selection methods. Here N is the number of candidate demonstrations, p is the number of model parameters, d is embedding size, K is the maximum candidate length, L is the test-input length, and Z is the number of n-grams.	58
3.11	Time required for extracting test gradients and computing IF across different model sizes using DataInf on MMLU moral-disputes with 200 test samples.	58
3.12	Time required for extracting test gradients and computing IF across different sample sizes using DataInf.	58
4.1	Performance change (%) observed after relevance paraphrasing across datasets/LLMs.	72
4.2	Effect of paraphrasing summary-relevant sentences versus randomly selected non-relevant sentences.	75
4.3	Effect of paraphrasing a larger proportion of relevant source sentences.	76
4.4	Performance change (%) observed after prompting the LLM with the same article-summary pair twice.	76
4.5	Performance change (%) observed after relevance paraphrasing across datasets/LLMs with temperature set to 0.	77
4.6	Robustness degradation after relevance paraphrasing for BART and Pegasus. . . .	78

5.1	FORTE’s 9-error taxonomy versus prior frameworks.	84
5.2	Main results on ToolHop and NESTFUL.	94
5.3	Closed-source frontier reference.	96

Abstract

Large language models have demonstrated strong capabilities across a wide range of language tasks, yet their downstream performance depends not only on what is acquired during pretraining, but also on how that capability is elicited, adapted, and evaluated at post-training and inference time. This dissertation studies that problem through the lens of *data-model synergy*: the hypothesis that LLMs internalize useful inductive biases during large-scale pretraining, and that these biases can be identified, quantified, and leveraged to improve downstream performance. Across four chapters, this dissertation develops data-driven methods for improving the utility, generalization, robustness, and reliability of LLMs by analyzing and, where useful, reshaping the relationship between training data, model internals, and prediction behavior.

The first contribution addresses *utility* through **LayerIF**, a framework for estimating task-specific layer quality using layer-localized influence analysis. Rather than relying solely on model-centric signals such as parameter or activation statistics, LayerIF uses sensitivity to training data as a signal of how well individual Transformer layers are specialized for a downstream task. This perspective shows that layer quality is often task dependent and provides a principled basis for guiding post-training decisions at the level of individual layers.

The second contribution addresses *generalization* through **Indirect In-Context Learning (Indirect ICL)**, a framework for demonstration selection when candidate examples are noisy, heterogeneous, or only indirectly related to the target task. In contrast to standard in-context learning assumptions, this work studies inference-time adaptation under imperfect supervision and argues that semantic similarity alone is often insufficient for selecting useful demonstrations. By using influence functions to identify examples whose inductive bias is better aligned with the target task, Indirect ICL shows how demonstration selection can be made more effective in challenging settings such as task mismatch and label corruption.

The third contribution addresses *robustness* in LLM summarization through two complementary studies. The first characterizes **position bias** in zero-shot abstractive summarization by analyzing whether models rely on the same regions of a source document as human-written summaries. The second studies **relevance paraphrasing robustness**, examining whether summaries remain stable when source sentences most central to the gold summary are paraphrased without changing their

meaning. Together, these studies show that strong aggregate performance does not necessarily imply reliable source grounding, and they provide more direct ways of evaluating robustness than standard summary-quality metrics alone.

The fourth contribution addresses *reliability* in agentic tool use through **FORTE**, a framework that trains multi-step tool planners to avoid structural failure modes by construction. In contrast to the analytic methods of the earlier chapters, FORTE synthesizes its own supervision: a nine-error taxonomy of how tool plans fail is used to generate contrastive negative plans, which train a structural judge that in turn supplies a dense reward for reinforcement-learning fine-tuning and serves as a verifier for Best-of- N inference. Across two multi-step tool-planning benchmarks, this failure-mode-aware training produces planners that match or exceed substantially larger closed-source models on structural reliability, using only modest open-source backbones.

Taken together, the work in this dissertation advances a view of LLM post-training as a data-driven science. By relating training data to internal model structure and downstream behavior, it demonstrates how post-training can be made more principled and how LLM systems can be improved along four critical dimensions: utility, generalization, robustness, and reliability.

Acknowledgments

To my parents, Syeda Aijaz Fatima and Hasan Askari, without whom I would not be the person I am today. Their unwavering support, countless sacrifices, and emphasis on education from a young age provided the foundation on which I stand. To my siblings, Hamza Askari, Shayan Askari, and Alina Askari, who have constantly motivated me to continue my journey toward excellence.

To my advisor, Muhao Chen, for supporting me from my third year onward and providing the guidance and support that allowed me to complete my degree. To Anshuman Chhabra, for being a constant friend and collaborator throughout my Ph.D., and for helping me through some of my lowest points. To Zainab Batool, for motivating me to pursue graduate school, helping me apply, and supporting me through the first half of my Ph.D. To Zubair Shafiq, for seeing potential in me and accepting me into the Ph.D. program. To Muhammad Haroon, who spent many hours of his own precious time helping me at the start of my Ph.D.

To the close friends I made during graduate school, Kartik Patwari, Yash Vekaria, Shaoor Munir, and Hari Venugopalan. To my soccer teams, The Microbes, Velocity FC, and The Papis, for providing invaluable stress relief and giving me a social life in Davis. To the great friends I made through soccer, including Eduardo Ruiz, Augustine Arredondo, Abbey Ragan, and many others whose names I cannot fully list here. To my current and former labmates at LUKA, who helped shape many of my ideas, and to all of my co-authors, without whom I would not have been able to write this dissertation.

Finally, to my partner, Anna Vernier, for being my rock, the person I turn to whenever I feel stressed, and for helping me finish my Ph.D.

CHAPTER 1

Introduction

1.1. The Rise of Large Language Models and the Post-Training Challenge

Over the past few years, large language models (LLMs) have emerged as a revolutionary technology in artificial intelligence, demonstrating remarkable capabilities across a broad range of natural language understanding and generation tasks such as question answering [1, 2], natural language inference [3], reasoning [4, 5], and code generation [6]. Trained on vast corpora of text spanning books, code, scientific literature, and web data, these models acquire rich representations of language, world knowledge, and reasoning patterns during pretraining [7, 8, 9]. The scale and diversity of this pretraining data endow LLMs with an extraordinary breadth of internalized knowledge, which, in principle, could support strong performance across many downstream applications.

Although large language models are trained on broad and diverse corpora, their downstream performance remains highly task dependent [10]. Scaling has enabled models to exhibit increasingly strong capabilities across a wide range of tasks, including forms of reasoning that are weak or absent in smaller models [11]. However, realized performance depends not only on what is acquired during pretraining, but also on how that capability is elicited, adapted, and evaluated at post-training and inference time. Indeed, prompting and decoding strategies such as chain-of-thought prompting and self-consistency can substantially change model performance by eliciting latent reasoning behaviors more effectively [12, 13].

These observations suggest that pretrained capability alone is not a sufficient explanation for downstream behavior. Instead, the performance of LLMs depends critically on how pretrained knowledge is activated, aligned, and operationalized for specific tasks and deployment settings. This motivates a central question for this dissertation: can we systematically leverage the inductive biases that LLMs internalize during large-scale pretraining in order to improve post-training performance?

This question gives rise to several more specific challenges. First, if different parts of a model contribute unevenly to downstream behavior, can we identify which internal components are already well specialized and which remain more sensitive to task-specific adaptation? Second, when labeled demonstrations are limited, noisy, or only indirectly related to a target task, how should examples be selected so that they best elicit the model’s useful latent behavior? Third, when an LLM produces fluent outputs, how can we determine whether those outputs are robust and fairly grounded in the relevant input evidence rather than artifacts of positional or stylistic bias? Fourth, when an LLM acts as an agent and produces a multi-step plan over external tools, how can we train it to avoid the structural mistakes that silently break such plans?

Taken together, these challenges point to a common scientific theme: the relationship between training data, internal model structure, and downstream behavior. This dissertation argues that post-training can be made more principled by studying that relationship directly, rather than treating the model as a black box and the training data as a static resource.

1.2. Data-Model Synergy as a Unifying Principle

This dissertation is organized around the hypothesis that large language models internalize useful inductive biases during large-scale pretraining, and that these biases can be identified, quantified, and leveraged to improve downstream performance. We refer to this perspective as **data-model synergy**: the systematic use of data-driven analysis to understand how training data interacts with model internals and how that interaction can be used to improve adaptation and evaluation.

A primary technical instrument for this perspective is the *influence function* (IF) [14, 15], a classical statistical tool that has been adapted in machine learning to estimate the effect of individual training examples on model predictions [16]. Influence functions provide a principled mechanism for relating model behavior back to the data that shaped it. IFs have been utilized with great success in a number of application scenarios (e.g. classification [17, 16], generative models [18, 19, 20] and active learning [21, 22]). In this dissertation, influence functions are the primary instrument of the first two contributions, where they bridge training data and model internals at the level of layers (Chapter 2) and demonstrations (Chapter 3). The data-model synergy perspective is not tied to a single instrument, however. The robustness study of Chapter 4 uses distributional and

perturbation-based diagnostics to ask how models use source evidence, and the reliability study of Chapter 5 uses contrastive data synthesis, grounded in a taxonomy of failure modes, to train better behavior directly. What unifies the chapters is not the tool but the move they share: to treat the relationship between training data, model internals, and downstream behavior as something that can be measured and acted upon.

The chapters that follow apply this perspective to four distinct post-training challenges. The first concerns **utility**: how to improve downstream accuracy and effectiveness by identifying which layers of a model are better or worse prepared for a task. The second concerns **generalization**: how to better elicit useful model behavior in low-resource or imperfect-supervision settings by selecting demonstrations that align with the model’s latent inductive biases. The third concerns **robustness**: how to diagnose whether model outputs are stably and fairly grounded in the input evidence rather than driven by incidental biases in phrasing or position. The fourth concerns **reliability**: how to train agentic tool planners that avoid the structural failure modes that derail multi-step plans, by synthesizing contrastive training data from a taxonomy of those failures.

1.3. Chapter II, Utility: Data-Driven Layer Quality Estimation

The first chapter of this dissertation concerns the *utility* of large language models: their ability to achieve strong downstream performance after adaptation. A central challenge in this setting is that the layers of a pretrained transformer are not uniform in their degree of specialization or contribution to downstream behavior [23, 24, 25]. Some layers appear to encode relatively stable and transferable representations, whereas others remain more sensitive to task-specific data and adaptation. This heterogeneity matters in practice because many post-training decisions, such as where to allocate adaptation capacity or where to prune parameters, implicitly treat the model’s layers as if they were equally well suited to the target task, even though different layers may play substantially different roles across tasks. Existing approaches to characterizing layer quality are predominantly model-centric [26, 27]. They typically rely on properties such as parameter magnitudes, gradient statistics, or activation patterns to infer which layers are more important or more redundant. While these signals can be informative, they largely omit a key source of evidence: the training data that shaped the model’s behavior. From the perspective of this dissertation, that omission is important.

A layer should not be regarded as high quality solely because it exhibits a desirable internal statistic; it should also be evaluated in terms of how stably and effectively it mediates the relationship between training data and downstream performance.

To address this gap, the first contribution of this dissertation is **LayerIF**, a framework for estimating layer quality in LLMs through layer-localized influence analysis. The central idea is to use sensitivity to training data as a signal of how mature and task-ready a layer is. Layers that behave stably with respect to relevant training examples can be viewed as better specialized for a downstream task, whereas layers that remain highly sensitive may require additional adaptation or different treatment during post-training.

An important finding of this work is that layer quality is not a fixed property of a model, but often a task-specific one. The same pretrained LLM can exhibit different layer-wise profiles depending on the downstream dataset and objective. This observation challenges the assumption that a single global ranking of layer importance is sufficient for guiding adaptation. Instead, it suggests that post-training decisions should take into account how the model’s internal structure interacts with the task at hand.

This perspective proves useful in two practical settings. The first is expert allocation in LoRA-based mixture-of-experts (LoRA-MoE) fine-tuning, where the question is how to distribute LoRA-MoE adapters across layers. The second is structured pruning, where the question is how to remove parameters without disproportionately degrading downstream performance. In both cases, LayerIF provides a data-driven signal for deciding where capacity should be added, preserved, or reduced. More broadly, this chapter argues that decisions about how to allocate adaptation resources should be guided not only by model-internal heuristics, but also by the interaction between training data and model components.

1.4. Chapter III, Generalization: Influence-Guided Demonstration Selection

The second chapter of this dissertation concerns *generalization*: the ability of a large language model to adapt effectively at inference time even when task-specific, high-quality demonstrations are limited, noisy, or altogether unavailable. A central mechanism for such adaptation is *in-context learning* (ICL), in which an LLM is conditioned on a small set of input-output exemplars placed in

the prompt and then asked to solve a new test instance without gradient-based parameter updates [28]. In its standard formulation, ICL has shown strong performance across a range of reasoning, classification, and generation tasks. However, much of the ICL literature assumes a favorable setting in which candidate demonstrations are drawn from the same task distribution as the test query and are accurately labeled [29, 28]. More recent work has begun to examine in-context learning under noisy or distribution-mismatched demonstrations, reflecting the fact that these idealized assumptions often fail in practice [30, 31, 32].

In many practical settings, however, adaptation must proceed from a *sub-optimal* pool of candidate demonstrations. Available examples may come from heterogeneous tasks whose relationship to the target task is unclear; they may be only indirectly relevant due to distribution shift; or they may contain annotation errors introduced by human labeling, weak supervision pipelines, or other language models. This dissertation treats such settings not as exceptional cases, but as an important and realistic formulation of inference-time adaptation under indirect supervision. To capture this broader regime, the second contribution of this dissertation introduces **Indirect In-Context Learning (Indirect ICL)**, a generalized demonstration-selection framework in which the candidate pool is not assumed to be task-aligned or label-clean. Rather than relying solely on semantic similarity between candidate demonstrations and the test query, Indirect ICL uses influence functions to estimate which examples are most likely to support correct target-task behavior, thereby selecting demonstrations whose inductive bias is better aligned with the model’s needs even under task mismatch or label corruption.

This framework is studied in two concrete and practically important settings. The first is the *Mixture-of-Tasks* setting, in which the demonstration pool contains examples from many unrelated tasks, only a small fraction of which may bear meaningful relationship to the target task. The second is the *Noisy ICL* setting, in which the candidate pool comes from the correct task distribution but contains corrupted labels or otherwise unreliable supervision. These settings represent two distinct barriers to generalization: task mismatch and supervision corruption. Indirect ICL provides a framework for addressing both.

The central claim of this chapter is that semantic similarity alone is often insufficient in such indirect settings. A semantically similar example may still be misleading if it is mislabeled or if it

encourages the wrong latent task structure, while an example from a different task may still be useful if it induces the right kind of reasoning pattern. Indirect ICL therefore reframes demonstration selection as a problem of *inductive bias alignment*: which candidate examples best align the model’s internal behavior with the demands of the target task?

More broadly, this chapter extends the data-model synergy perspective from parameter-space adaptation to inference-time adaptation. It argues that generalization in LLMs can be improved not only by updating model parameters, but also by more intelligently choosing the evidence presented to the model at test time. In this sense, prompt construction becomes another site where the relationship between data and model internals can be exploited in a principled way.

1.5. Chapter IV, Robustness: Characterizing Failure Modes in LLM Summarization

The third chapter concerns the *robustness* of LLMs and, more specifically, whether their outputs remain reliably grounded in source evidence under benign variation in the input. Even models that achieve strong aggregate performance can exhibit systematic biases or reliability failures that are not revealed by summary metrics alone [33, 34]. In generative tasks such as abstractive summarization, models may achieve competitive overlap-based scores while still producing outputs that are unfaithful to the source document or otherwise insufficiently grounded in the input evidence [35]. This chapter therefore argues that robustness in summarization should be studied directly, rather than inferred indirectly from aggregate summary quality alone.

We investigate these issues in the context of *zero-shot abstractive summarization*, a setting that is well suited for robustness analysis because the model must compress long, information-dense documents without task-specific fine-tuning. In this regime, summary quality depends not only on whether relevant information is represented in the model, but also on whether the model can identify and preserve that information reliably under benign variation in the input. This chapter studies two complementary failure modes. The first is **position bias**: the tendency of a summarization model to rely disproportionately on content from particular regions of a source document, regardless of whether those regions are the ones humans rely on when writing summaries. The second is **paraphrasing sensitivity**: the tendency of a model to produce meaningfully different summaries when salient source sentences are rewritten in a semantically equivalent manner. Together, these failure modes

capture two related threats to robust summarization: *where* the model draws information from in the source document, and *how stably* it behaves when the wording of that information changes.

The first study in this chapter revisits zero-shot abstractive summarization through the lens of position bias. Prior work in summarization has often focused on *lead bias*, that is, the tendency to overuse the opening portion of a document, especially in news summarization where early sentences often contain salient information [36, 37]. However, lead bias is only one instance of a broader problem. A model may also exhibit undue reliance on the middle or tail of a document, and whether such reliance is problematic depends on the behavior of the human-written summaries themselves. This dissertation therefore treats position bias as a mismatch between the regions of the source document emphasized by the model and those emphasized by gold summaries. Under this view, the question is not simply whether a model prefers the beginning of a document, but whether its source allocation is aligned with how humans identify salient content.

The second study examines a complementary robustness failure: whether LLM summaries remain stable when the most important source sentences are paraphrased without changing their meaning. Whereas the position bias study asks whether models rely on the right *positions* in the source document, this study asks whether they preserve the right *content* under minimal semantics-preserving perturbations. To investigate this question, the dissertation introduces a relevance-paraphrasing framework that selectively paraphrases source sentences most central to the gold summary and then measures how much model behavior changes. If a summarizer is robust, its outputs should remain largely consistent under such benign reformulations.

Together, these two studies provide a unified view of summarization robustness. The position bias study diagnoses whether a model draws on the appropriate *positions* of a document relative to human summaries. The relevance-paraphrasing study diagnoses whether the model preserves its content-selection behavior under ordinary linguistic variation. The former is a structural diagnostic of source allocation; the latter is a behavioral diagnostic of selection stability. Both expose aspects of model behavior that are not captured by standard summary-quality metrics alone, and both reinforce the broader point that average task performance is an incomplete proxy for reliability [35].

More broadly, this chapter extends the data-model synergy perspective into the evaluation of deployed LLM systems. In the earlier chapters, data-model interactions are used constructively

to improve utility and generalization. Here, they are used diagnostically to identify where model behavior remains brittle. In both studies, the central question is not merely whether the model produces a plausible output, but whether that output is grounded in source evidence in a stable, human-aligned, and semantically consistent manner.

1.6. Chapter V, Reliability: Training Robust Tool Planners

The fourth chapter concerns the *reliability* of large language models when they act as agents rather than predictors. In a growing range of deployments, an LLM is asked not to produce a single answer but to accomplish a goal by orchestrating external tools: search engines, calculators, databases, and application programming interfaces. To do so, it must decompose a request into sub-goals, select an appropriate tool for each, bind the arguments of every call with valid types, route the output of one call into the input of the next, and order the whole procedure so that every dependency is satisfied before it is used. The artifact it produces is therefore not an answer but a *tool plan*: a structured, multi-step program over an external toolset. In this regime, the property that determines whether the system is usable is not the fluency of any individual step but the structural correctness of the plan as a whole.

This raises a failure mode that the earlier chapters do not encounter. In multi-step tool planning no constraint is independent: a single local mistake, such as a near-miss tool choice, a mis-typed argument, or a reference threaded to the wrong step, propagates silently through dependent steps and breaks the entire trajectory. Such plans are locally well-formed yet globally broken, and the resulting brittleness persists even in frontier models. The dominant responses in the literature either search harder at inference time or impose dependency structure as an external scaffold, but neither changes the planner’s underlying tendency to make these errors.

To address this challenge, the fourth contribution of this dissertation is **FORTE**, a framework for improving the structural reliability of multi-step tool planners without relying on tool execution. FORTE begins with a systematic characterization of plan failures, organizing nine concrete failure modes into argument-level, dependency-level, and plan-structure categories. This taxonomy is then operationalized as supervision: for each correct plan, FORTE synthesizes minimally edited negative plans corresponding to specific failure modes, yielding contrastive examples that isolate distinct

sources of structural error. These contrastive pairs are used to train a lightweight judge that estimates the structural correctness of a candidate plan. The judge is then used in two complementary ways: as a dense reward model for reinforcement-learning fine-tuning of the planner, and as a verifier for selecting among diverse candidate plans during Best-of- N inference.

This chapter completes the dissertation’s arc by moving from behavioral diagnosis to intervention. The robustness chapter characterizes failure modes to identify and analyze unreliable behavior; FORTE characterizes failure modes to construct targeted supervision that reduces the likelihood of such behavior during planning. Similarly, whereas the utility and generalization chapters use influence functions to study how existing data shapes model behavior, FORTE constructs new training data, grounded in an explicit structural account of planning failures, to guide model behavior toward more reliable plan generation. The methodological instrument is different, but the underlying principle is consistent across the dissertation: downstream behavior improves when the relationship between data and model behavior is explicitly measured, modeled, and engineered rather than left implicit. Empirically, FORTE shows that targeted exposure to structurally defined failure modes can produce tool planners with reliability comparable to, or exceeding, substantially larger models.

1.7. Contributions and Dissertation Outline

This dissertation makes the following primary contributions, united by the data-model synergy perspective:

- (1) **LayerIF** (Chapter 2): A data-driven framework for estimating task-specific layer quality in LLMs through layer-localized influence-function analysis. This framework provides a principled basis for guiding post-training decisions at the level of individual Transformer blocks.
- (2) **Indirect ICL** (Chapter 3): A generalized framework for demonstration selection in in-context learning when candidate examples are task-mismatched, heterogeneous, or label-corrupted. This framework shows how influence-guided selection can improve inference-time adaptation under imperfect supervision.
- (3) **Position Bias Characterization** (Chapter 4): A framework for analyzing positional bias in zero-shot abstractive summarization by comparing the source regions emphasized

by model-generated and human-written summaries. This analysis provides a more direct account of source allocation behavior than standard summary-quality metrics alone.

- (4) **Relevance Paraphrasing Robustness** (Chapter 4): A model-agnostic evaluation framework for abstractive summarization that perturbs only the source sentences most relevant to the gold summary through semantics-preserving paraphrasing. This framework reveals the extent to which summary generation remains stable under minimal changes to salient source content.
- (5) **FORTE** (Chapter 5): A framework that trains multi-step tool planners to reduce structural failure modes through failure-aware supervision and verification. A nine-error taxonomy of tool-plan failures is used to synthesize contrastive training data; a lightweight judge trained on that data supplies a dense reward for reinforcement-learning fine-tuning; and the same judge serves as a verifier for Best-of- N inference. This contribution extends the data-model synergy perspective from analyzing and diagnosing model behavior to constructing targeted training data that systematically reduces unreliable planning behavior.

The remainder of the dissertation is organized as follows. Chapter 2 presents the LayerIF framework and its role in data-driven layer quality estimation. Chapter 3 introduces the Indirect ICL framework for demonstration selection under imperfect supervision. Chapter 4 examines robustness in LLM summarization through the complementary lenses of position bias and paraphrasing sensitivity. Chapter 5 turns from diagnosing failure modes to training them out, presenting the FORTE framework for reliable multi-step tool planning. Chapter 6 discusses broader implications and future directions, and concludes the dissertation.

Taken together, the work presented in this dissertation advances a view of LLM post-training as a data-driven methodology for post-training in which the relationship between training data and model behavior can be systematically analyzed and used to improve utility, generalization, robustness, and reliability in downstream deployment.

CHAPTER 2

Utility: Data-Driven Layer Quality Estimation

2.1. Introduction

Large language models achieve strong performance across a broad range of downstream tasks, yet their internal structure is far from uniform. Prior work has shown that different layers within a pretrained transformer can vary substantially in their degree of specialization, transferability, and contribution to downstream behavior [23, 38, 39, 24]. This heterogeneity matters in practice because many post-training decisions, including where to allocate adaptation capacity and where to prune parameters, implicitly treat the model’s layers as if they were equally well suited to a target task. In reality, however, different layers often play markedly different roles across tasks, suggesting the need for more principled methods for estimating layer quality [23, 24, 25].

Existing approaches to characterizing layer quality are predominantly model-centric. They typically rely on properties of learned weights, activations, or gradient-derived statistics to infer which layers are more important or more redundant [26, 27, 40, 41, 42, 43]. While such signals can be informative, they largely omit a central source of evidence: the training data that shaped the model’s behavior. From the perspective of this dissertation, that omission is consequential. A layer should not be evaluated solely in terms of its internal statistics, but also in terms of how it mediates the relationship between training data and downstream performance.

This chapter introduces **LayerIF**, a data-driven framework for estimating task-specific layer quality through layer-localized influence analysis. Influence Functions (IFs) provide a natural mechanism for this purpose because they quantify how perturbations to individual training examples affect model behavior, thereby offering a more explicitly data-centric perspective than model-only diagnostics [14, 15, 16, 18, 20, 44]. Building on this perspective, LayerIF uses a layer’s sensitivity to training data as a signal of how well it is specialized for a downstream objective. Layers whose behavior is relatively stable with respect to task-relevant training examples can be interpreted as

better prepared for the target task, whereas layers that remain highly sensitive may require additional adaptation or may be better candidates for more aggressive compression.

A central finding of this chapter is that layer quality is often task-specific rather than fixed. The same pretrained LLM can exhibit different layer-wise profiles depending on the dataset and downstream objective, challenging the assumption that a single global ranking of layer importance is sufficient for post-training. To demonstrate the practical utility of this perspective, the chapter studies two settings in which accurate layer-quality estimation is especially valuable. The first is expert allocation in LoRA-based mixture-of-experts fine-tuning, where prior work has questioned uniform expert allocation and explored heuristic or model-driven alternatives [45, 46, 47, 48, 49]. The second is layer-wise sparsity allocation for model pruning, where recent methods have moved beyond uniform sparsity but still rely primarily on weight statistics, activations, or spectral heuristics [50, 51, 26, 27]. Across both settings, LayerIF provides a principled signal for deciding where capacity should be added, preserved, or reduced.

More broadly, this chapter advances the dissertation’s central argument that post-training should be guided not only by model-internal heuristics, but also by the interaction between training data and model components. By localizing influence analysis to individual Transformer blocks, LayerIF provides one concrete framework for making that interaction measurable and actionable.

2.2. Related Work

Layer Quality Estimation. A substantial body of prior work has examined the representational differences that emerge across layers in deep neural networks and pretrained language models [52, 53, 54]. Probing methods have been widely used to analyze the semantics and functional role of intermediate representations [55, 56], revealing that different layers often capture distinct forms of linguistic and task-relevant information. Other work has attempted to quantify layer quality more directly. For example, [57] apply random matrix theory to the weight matrices of deep neural networks and show that heavy-tailed empirical spectral properties can serve as indicators of layer quality. More recently, Shapley-value-based approaches have been used to estimate the contribution of different layers to overall model capability [58, 59], while representation-based studies suggest that middle layers often provide especially useful features for downstream tasks [60]. Together, these

works show that layers are not equally informative or equally well specialized. However, most existing approaches remain primarily model-centric, relying on properties of weights or representations rather than explicitly accounting for the training data that shaped those layers.

Model Pruning and Layer-Wise Sparsity Allocation. Pruning has long been studied as a means of reducing model size by removing redundant or non-essential parameters [61, 62]. More recent work has extended these ideas to large language models, where the question is not only which parameters to remove, but also how sparsity should be distributed across layers. Several approaches use representation similarity or layer-wise redundancy as a signal for pruning, finding that some layers, particularly deeper ones, can be removed with relatively limited performance degradation [63, 64]. Other approaches rely on gradients, curvature, weights, or activations to identify removable structure, as in LLM-Pruner [65], SparseGPT [50], and Wanda-style pruning methods [51]. Layer-wise sparsity allocation has also been explored in earlier pruning work [66] and in related compression settings such as quantization [67]. Among recent adaptive approaches, AlphaPruning uses heavy-tailed spectral statistics to derive layer-specific sparsity ratios [27], while OWL allocates non-uniform sparsity based on the prevalence of outlier activations [26]. These methods move beyond uniform sparsity, but they remain fundamentally model-driven. In contrast, our work investigates whether training-data influence can provide a complementary basis for layer-wise sparsity allocation.

Expert Allocation in LoRA-MoE Architectures. Combining LoRA-based fine-tuning [68] with Mixture-of-Experts (MoE) modeling [69] has emerged as a promising strategy for increasing adaptation capacity while maintaining parameter efficiency. A growing body of work has proposed LoRA-MoE variants in which multiple low-rank experts are trained with routing mechanisms, often allocating the same number of experts to every Transformer block [70, 71, 72, 73, 74]. More recent studies question this uniform design and suggest that expert demand may vary across depth. For example, some approaches allocate experts using group-based heuristics [48], while others rely on heavy-tailed self-regularization signals to adapt expert counts across layers [49]. Related work has also studied how LoRA modules can be composed more effectively across tasks [75]. These studies collectively suggest that expert allocation should be layer-aware, but existing approaches largely rely on heuristics or model-only statistics. Our work instead explores whether a data-driven measure of layer quality can provide a more principled signal for expert allocation.

Influence Functions. Influence Functions (IFs) provide a classical framework for estimating the effect of individual training points on model behavior [15, 76, 77, 78, 79]. In modern machine learning, IFs are commonly approximated through gradient and curvature-based methods rather than explicit leave-one-out retraining [16, 80, 81, 21, 82]. These methods have been applied to problems such as mislabeled data detection, subset selection, model interpretation, data attribution, and bias analysis. However, classical formulations are often difficult to apply directly to large language models because of their computational cost and underlying assumptions [83, 84]. More recent work has therefore proposed LLM-scalable approximations, including DataInf [18], Arnoldi-based methods [19], alternative Hessian formulations [20, 85], self-influence methods [86, 87], ensemble approaches [44, 88], and more efficient Hessian-free approximations [84, 89, 90, 91]. Our work builds on this line of research, but differs in its objective: rather than using IFs to analyze the influence of training examples alone, we use them to estimate the quality of individual layers. Importantly, the proposed LayerIF framework is agnostic to the particular IF estimator used, allowing future advances in influence estimation to be directly incorporated into our approach.

Taken together, prior work suggests that layer quality matters for both interpretation and post-training optimization, but existing diagnostics remain largely model-centric. At the same time, influence-function methods provide a principled way to connect model behavior to training data, yet have not been systematically formulated for layer-wise quality estimation in LLMs. This chapter addresses that gap by introducing LayerIF, a data-driven framework that uses layer-localized influence analysis to estimate task-specific layer quality and apply those estimates to expert allocation and pruning.

2.3. The LayerIF Framework

This section presents the proposed LayerIF framework for estimating layer quality in large language models using layer-localized influence analysis. We first review the influence function machinery that underlies the method, and then show how it can be adapted from training-sample attribution to layer-wise quality estimation. We subsequently describe how the resulting scores are used in two downstream settings: expert allocation in LoRA-MoE fine-tuning and layer-wise sparsity allocation for pruning.

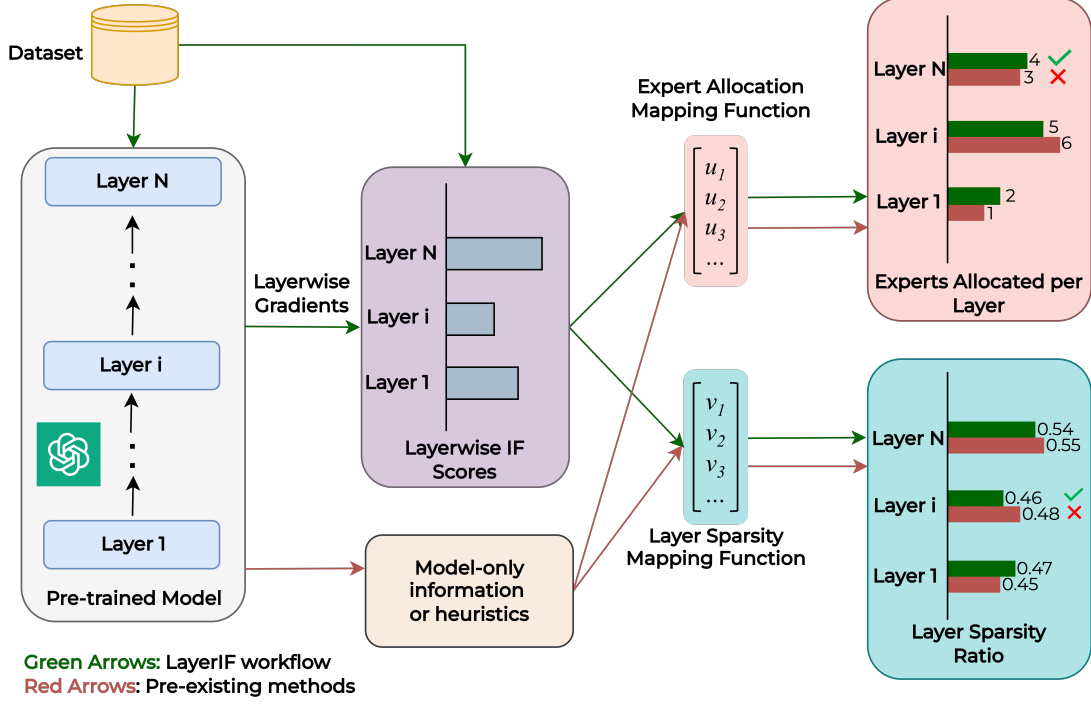


FIGURE 2.1. Overview of the LAYERIF framework. LAYERIF computes task-specific layer-quality estimates by localizing influence-function analysis to individual Transformer blocks. These scores are then mapped to downstream post-training decisions such as expert allocation in LoRA-MoE fine-tuning and layer-wise sparsity allocation for pruning.

Figure 2.1 provides an overview of the framework. At a high level, LayerIF takes as input a pretrained LLM together with a training set and validation set for a target task. It then computes layer-restricted influence scores that quantify how strongly the model’s validation behavior depends on the training data through each Transformer block. These scores serve as task-specific estimates of layer quality and can be mapped to different post-training decisions, such as how many LoRA-MoE experts to allocate to each layer or how aggressively each layer should be pruned.

2.3.1. Preliminaries on Influence Functions. Let the training dataset be $D^{\text{train}} = \{z_i\}_{i=1}^n$ and the validation dataset be $D^{\text{val}} = \{z_j^{\text{val}}\}_{j=1}^m$. Let f_{θ} denote a pretrained large language model with parameters θ , and let $\ell(z, \theta)$ denote the task loss for example z .

Influence Functions (IFs) provide a classical way to estimate how the model would change if a particular training example were infinitesimally upweighted in the empirical risk objective [14, 15, 92]. Following the modern formulation of [16], the influence of a training point z_i on the

learned parameters can be written as

$$I_{\theta^*}(z_i) = -H(\theta^*)^{-1} \nabla_{\theta} \ell(z_i, \theta),$$

where $H(\theta^*)$ denotes the Hessian of the training objective at the learned parameters.

The quantity most relevant for our purposes is not the change in parameters alone, but the effect of a training point on validation behavior. This yields the standard influence of a training example on the validation loss:

$$I(z_i) = - \sum_{j=1}^m \nabla_{\theta} \ell(z_j^{\mathcal{Y}}, \theta)^{\top} H(\theta)^{-1} \nabla_{\theta} \ell(z_i, \theta).$$

Intuitively, this score measures whether a given training example is beneficial or detrimental for the target validation set. A training point with more beneficial influence contributes more strongly to reducing validation loss, whereas a detrimental point pushes the model in the opposite direction. In prior work, influence functions have primarily been used to study training examples. In this chapter, we adapt them to study *layers*.

2.3.2. Layer-Localized Influence Analysis. In this dissertation, the term *layer* refers to a Transformer block, including its attention and feed-forward submodules. The core idea of LayerIF is to localize influence computation to a single Transformer block and use the resulting signal to estimate how strongly that block mediates the relationship between training data and downstream performance.

Let $\theta^{(l)}$ denote the parameters associated with layer l . We define the layer-specific influence of a training example z_i by restricting both gradients and the curvature approximation to that layer:

$$I^{(l)}(z_i) = - \sum_{j=1}^m \nabla_{\theta^{(l)}} \ell(z_j^{\mathcal{Y}}, \theta)^{\top} \left(H^{(l)}(\theta) \right)^{-1} \nabla_{\theta^{(l)}} \ell(z_i, \theta),$$

where $H^{(l)}(\theta)$ denotes the Hessian approximation restricted to layer l .

This localized score isolates the extent to which training example z_i affects validation performance *through layer l* . By computing this quantity separately for each layer while holding the task and data fixed, LayerIF produces a layer-wise view of training-data sensitivity.

2.3.3. From Layer-Wise Influence to Layer Quality Scores. Layer-specific influence scores are still defined at the level of individual training examples. To obtain a single scalar quality score per layer, LayerIF aggregates these example-level scores across the training set.

For each layer l , we compute

$$S^{(l)} = \sum_{i=1}^n \mathbb{I}[I^{(l)}(z_i) > 0] \cdot I^{(l)}(z_i),$$

where $\mathbb{I}[\cdot]$ is the indicator function. In other words, we sum only the positively influential training examples after sign normalization, so that larger positive values correspond to greater sensitivity of the validation loss to the training data through layer l .

This produces a vector

$$S = (S^{(1)}, S^{(2)}, \dots, S^{(L)}),$$

where L is the number of Transformer blocks. Each component summarizes the cumulative positive influence of the training data on validation behavior through a specific layer.

2.3.4. Interpretation of LayerIF Scores. The interpretation of these scores is central to the method. LayerIF is based on the hypothesis that well-trained or well-specialized layers should exhibit relatively low sensitivity to individual training examples on a target task. If a layer has already learned stable and transferable structure relevant to the validation objective, then small perturbations to the weights of specific training examples should have limited effect on validation loss through that layer. By contrast, if a layer remains highly sensitive to training examples, this suggests that its contribution is less stable and potentially less mature for the task.

Under this interpretation, lower aggregate positive influence corresponds to higher layer quality, while higher aggregate positive influence indicates a layer that may benefit from additional adaptation or may be less costly to modify. Importantly, these scores are task-specific: the same pretrained model can produce different layer-wise profiles depending on the dataset and downstream objective. LayerIF therefore does not assume that layer quality is a fixed property of a model in isolation; rather, it treats layer quality as a property of the interaction between model, task, and data.

2.4. Application I: Expert Allocation in LoRA-MoE Fine-Tuning

2.4.1. Motivation. A natural application of layer-quality estimation is expert allocation in LoRA-based mixture-of-experts (LoRA-MoE) fine-tuning. Existing LoRA-MoE methods often allocate the same number of experts to every Transformer block. However, if layers differ in how well they are already specialized for a task, then a uniform allocation may be suboptimal. Layers that are less mature for the target task may benefit from additional adaptation capacity, whereas layers that are already well specialized may require fewer experts.

2.4.2. Mapping LayerIF Scores to Expert Budgets. To allocate a fixed total budget of experts across layers, we first compute a scalar LayerIF score for each layer. These raw scores are then transformed into an allocation vector that assigns more experts to layers with lower estimated quality. Following the paper’s implementation, the scores are inverted and passed through a power transformation controlled by a hyperparameter β , which determines how sharply expert allocations vary across layers. The transformed scores are then normalized so that the total number of experts matches a predefined budget, while ensuring that each layer receives at least one expert.

This mapping produces a non-uniform expert allocation tailored to the layer-wise profile of the model on the target task. In this way, LayerIF translates data-driven estimates of layer quality into a concrete resource-allocation policy.

2.4.3. LoRA-MoE Setup and Baselines. Once the expert allocation vector is determined, each targeted module in a Transformer block is augmented with the corresponding number of LoRA experts. Following prior LoRA-MoE implementations, each expert is parameterized by a pair of low-rank trainable matrices, and a routing network selects a subset of experts for each input using top- K routing. Load-balancing losses are used to prevent expert collapse and promote more even utilization.

In the experiments, we compare LayerIF-based allocations against both uniform and adaptive baselines. These include uniform expert-allocation strategies as well as recent layer-adaptive methods such as MoLA and ALPHALoRA. This comparison allows us to evaluate whether a data-driven notion of layer quality provides a more effective allocation signal than architecture-level heuristics or model-only statistics.

2.5. Application II: Layer-Wise Sparsity Allocation for Pruning

2.5.1. Motivation. A second application of layer-quality estimation is layer-wise sparsity allocation in model pruning. Standard pruning methods often apply the same sparsity ratio to every layer. This uniform strategy is simple, but it ignores the fact that some layers may be more essential to downstream behavior than others. If layer quality varies across depth, then pruning budgets should also vary accordingly.

From the perspective of LayerIF, layers with lower estimated quality are more plausible candidates for aggressive pruning, whereas layers that appear better specialized for the task should be preserved more carefully. This yields a data-driven alternative to recent model-centric approaches for non-uniform sparsity assignment.

2.5.2. Mapping LayerIF Scores to Sparsity Ratios. To convert layer-wise LayerIF scores into sparsity budgets, we first post-process the raw scores to improve stability. In the implementation used here, absolute score magnitudes are normalized across layers and then smoothed to reduce high-frequency fluctuations introduced by influence estimation. The resulting smoothed profile is passed through a layer-wise mapping function that assigns each layer a sparsity ratio while satisfying a global sparsity constraint for the entire model.

This procedure produces a non-uniform sparsity distribution in which layers estimated to be less critical receive higher pruning ratios, while layers estimated to be better trained are pruned more conservatively.

2.5.3. Pruning Setup and Baselines. We integrate the resulting layer-wise sparsity budgets into several standard pruning pipelines, including Magnitude pruning, Wanda, and SparseGPT. In each case, the base pruning method remains unchanged except for the replacement of uniform sparsity with the LayerIF-derived layer-specific ratios.

We compare this approach against both uniform-sparsity baselines and recent adaptive methods such as ALPHAPRUNING and OWL. This setup allows us to isolate the contribution of the layer-allocation policy itself and assess whether a data-driven quality estimate improves the performance-sparsity trade-off.

2.5.4. A Walk-Through Example for LoRA-MoE. To make the framework concrete, we provide a concrete step-by-step walk-through of our method for greater clarity.

Concrete walk-through for expert allocation (CommonQ dataset, 32-layer LLM, T=160 experts, $\beta = 3$)

Below we provide the exact numbers produced by our implementation. For brevity only the first 5 layers are shown; the remaining layers follow identically.

Layer i	Raw IF score v_i	Inverted $\tilde{v}_i$	Power-scaled $\hat{v}_i$	Fractional f_i	Final experts s_i
0	-5.01×10^{11}	5.01×10^{11}	1.26×10^{35}	1.12	2
1	-8.36×10^{11}	8.36×10^{11}	5.84×10^{35}	5.21	6
2	-7.99×10^{11}	7.99×10^{11}	5.10×10^{35}	4.55	6
3	-8.46×10^{11}	8.46×10^{11}	6.05×10^{35}	5.40	6
4	-8.16×10^{11}	8.16×10^{11}	5.43×10^{35}	4.84	6
...	...	...	...	...	...

- **Step1 (sign inversion).** Invert the negative values to positive.
- **Step2 (power transform).** Amplify disparities by raising to the power β to get $\hat{v}$.
- **Step3 (normalise to budget).** Scale $\hat{v}$ so that the fractional allocations $\sum f_i$ equal $\mathbf{T} - \mathbf{m} = 128$ extra experts ($m = 32$ layers).
- **Step4 (floor + redistribute).** Set $s_i = \lfloor f_i \rfloor + 1$, guaranteeing ≥ 1 expert/layer, then distribute the remaining 24 units to the layers with the largest fractional parts.

Finally, the complete allocation for **CommonQ** becomes:

$$[2, 6, 6, 6, 6, 5, 7, 6, 8, 7, 6, 7, 6, 6, 6, 5, 6, 6, 5, 4, 5, 5, 4, 2, 4, 5, 3, 4, 2, 3, 4, 3]$$

Interpretation. The middle layers receive the most experts, indicating they are comparatively under-trained for CommonQ; the first and very deep layers receive lower number of experts, reflecting higher estimated quality.

2.6. Experimental Setup

This section describes the experimental settings used to evaluate LayerIF in the two downstream applications introduced above: expert allocation in LoRA-MoE fine-tuning and layer-wise sparsity allocation for pruning.

2.6.1. Datasets and Models. For the LoRA-MoE expert-allocation experiments, we evaluate on a mixture of language understanding and reasoning tasks, following prior work on adaptive LoRA allocation. The benchmark suite includes MRPC [93], CoLA [94], ScienceQA [95], CommonsenseQA [96], and OpenBookQA [97].

For the pruning experiments, we evaluate post-pruning zero-shot performance on a broader set of language understanding and reasoning benchmarks, including BoolQ [98], HellaSwag [99], Winogrande [100], ARC Easy and ARC Challenge [101], RTE [94], and OpenBookQA [97].

We conduct experiments on two open-weight LLMs, Mistral-7B-v0.1 [102] and Gemma-7B [103], in order to assess whether LayerIF generalizes across model families.

2.6.2. LayerIF Configurations and Baselines. For each target task, LayerIF computes layer-wise influence scores using a training split and a validation split. We consider multiple aggregation strategies when converting training-example influences into a scalar score per layer. These include summing over all examples, summing only positively influential examples, and summing only the top 25% most influential examples. This allows us to study whether downstream performance depends on the choice of aggregation strategy.

For the LoRA-MoE experiments, we compare three LayerIF variants: LAYERIF (ALL), LAYERIF (+VE), and LAYERIF (TOP 25%). These are evaluated against both adaptive and heuristic baselines, including ALPHALORA and multiple MOLLA configurations.

For the pruning experiments, we use LayerIF-derived layer-wise sparsity ratios within Magnitude pruning [104], Wanda [51], and SparseGPT [50]. We compare these against the original uniform versions of each pruning method, as well as adaptive sparsity-allocation baselines including ALPHAPRUNING [27] and OWL [26].

2.6.3. Evaluation Protocol. For the LoRA-MoE experiments, models are fine-tuned for three epochs and evaluated by zero-shot accuracy on the downstream tasks. We report per-dataset accuracy as well as average performance across tasks.

For the pruning experiments, we evaluate zero-shot performance after pruning at 50% global sparsity unless otherwise specified. In addition to the main 50% pruning results, we also study

TABLE 2.1. Accuracy across datasets using ALPHALoRA, MoLA, and LayerIF variants. We report scores under three LayerIF aggregation strategies: Top 25% positively influential samples, all positively influential samples (+ve), and all samples. Best average performance is shown in bold.

Dataset	ALPHALoRA	MoLA			LAYERIF		
		2468	5555	8642	+ve	TOP 25%	ALL
MRPC	75.30	82.02	80.64	80.70	82.49	82.63	81.63
CoLA	82.07	82.26	85.14	84.56	83.60	83.80	84.47
Text Science Q	78.52	64.65	72.75	76.39	75.04	79.59	77.61
Common Q	82.06	80.91	79.93	81.00	80.26	80.75	80.34
OpenBook Q	84.60	64.00	83.80	82.20	87.40	84.60	85.20
Average	80.51	74.77	80.45	80.97	81.76	82.27	81.85

performance across a range of sparsity levels in order to assess the robustness of LayerIF-based layer-wise allocation.

All experiments were run on 8×NVIDIA RTX 6000 Ada GPUs.

2.7. Results on Expert Allocation in LoRA-MoE Fine-Tuning

2.7.1. Main Results. Table 2.1 compares LayerIF against ALPHALoRA and three MoLA variants on the expert-allocation benchmark suite. Across datasets, LayerIF consistently achieves the strongest average performance. In particular, all three LayerIF variants outperform the strongest baseline on average, with LAYERIF (TOP 25%) achieving the best overall result.

These gains indicate that influence-based layer quality estimation provides a more effective signal for expert allocation than uniform or heuristic layer-wise strategies. Whereas prior methods allocate experts according to fixed patterns or model-centric statistics, LayerIF uses the relationship between training data and validation behavior to identify which layers are likely to benefit most from additional adaptation capacity.

2.7.2. Performance Across Different Expert Budgets. To assess whether these gains persist under different resource budgets, we also vary the total number of experts on Mistral-7B-v0.1. Figure 2.2 reports average performance for 80, 160, and 224 total experts, together with additional MoLA configurations designed to reflect strong expert-allocation patterns from the prior literature.

LayerIF remains consistently competitive across all three budgets and outperforms the baselines on average. This result suggests that the value of the method does not depend on a single expert

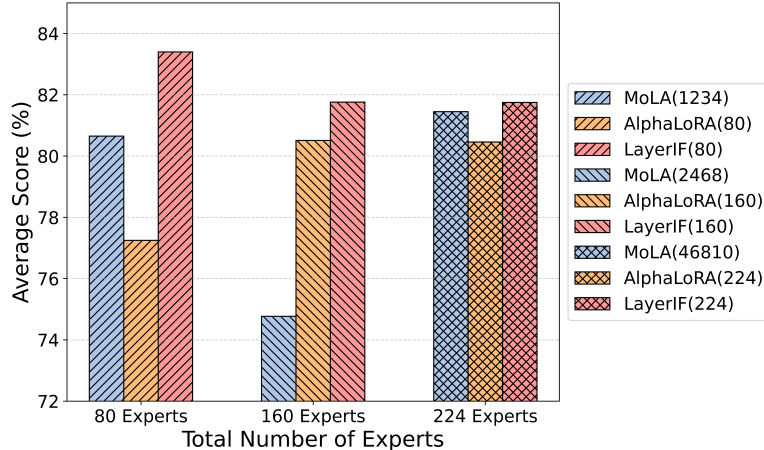


FIGURE 2.2. Comparison of ALPHALoRA, MoLA, and LayerIF under varying total expert budgets (80, 160, and 224) on Mistral-7B-v0.1.

TABLE 2.2. Accuracy across datasets using ALPHALoRA, MoLA, and LAYERIF (+VE) for 160 experts on Gemma-7B.

Dataset	ALPHALoRA	MoLA(3,5,7,8)	LAYERIF (+VE)
MRPC	82.61	82.89	84.75
CoLA	86.09	86.86	86.96
Text Science Q	94.15	93.88	93.66
Common Q	84.11	83.94	83.46
OpenBook Q	87.80	88.80	88.20
Average	86.55	87.47	87.81

budget or a narrow tuning regime. Instead, the layer-quality signal provided by LayerIF remains useful across a range of adaptation-capacity settings.

2.7.3. Results on Gemma-7B. To assess whether the gains generalize beyond a single model family, we also evaluate expert allocation on Gemma-7B using a total of 160 experts. Table 2.2 shows that LayerIF again achieves the strongest average performance, exceeding both ALPHALoRA and MoLA. Although individual datasets vary, the overall pattern remains consistent with the Mistral results: influence-guided layer quality estimation provides a useful signal for allocating adaptation capacity across layers.

TABLE 2.3. Comparison of average accuracies across different pruning strategies (Magnitude, Wanda, and SparseGPT) and layer-wise sparsity allocation methods at 50% pruning on Mistral-7B-v0.1. Best results are shown in bold.

Method	Magnitude	Wanda	SparseGPT	Average
UNIFORM	56.41	58.71	60.45	58.52
ALPHAPRUNING	56.74	58.48	60.07	58.43
OWL	56.16	58.75	59.99	58.30
LAYERIF	56.89	58.94	60.61	58.81

2.8. Results on Layer-Wise Sparsity Allocation for Pruning

2.8.1. Main Results at 50% Sparsity. Table 2.3 reports average accuracies across Magnitude pruning, Wanda, and SparseGPT when using different layer-wise sparsity allocation strategies at 50% global sparsity on Mistral-7B-v0.1. LayerIF achieves the best performance in all three pruning pipelines and also obtains the highest average across methods.

These results suggest that LayerIF provides a useful signal not only for allocating adaptation capacity, but also for deciding where capacity can be removed with minimal harm to downstream performance. Compared with both uniform sparsity and model-centric adaptive methods such as ALPHAPRUNING and OWL, the LayerIF allocation leads to a more favorable performance-sparsity trade-off.

2.8.2. Performance Across Multiple Sparsity Ratios. We further evaluate the robustness of LayerIF across multiple pruning ratios. Figure 2.3 shows mean accuracy over sparsity levels ranging from 20% to 50% for Mistral-7B-v0.1 pruned using SparseGPT. LayerIF maintains the strongest average performance across all sparsity levels, while the ranking of the baseline methods varies more substantially.

This stability is important because it suggests that LayerIF is not narrowly tuned to a single pruning regime. Instead, the layer-quality estimates remain informative even as the global sparsity constraint changes, supporting the broader claim that training-data sensitivity is a useful signal for structured model compression.

2.8.3. Results on Gemma-7B. Finally, we evaluate layer-wise sparsity allocation on Gemma-7B at 50% global sparsity. As shown in Table 2.4, LayerIF again achieves the highest average performance across pruning methods. Although the absolute performance levels differ from those of

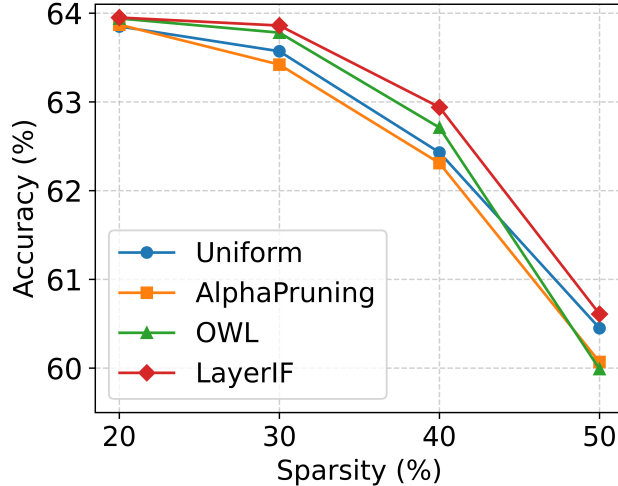


FIGURE 2.3. Mean accuracy across sparsity levels from 20% to 50% for Mistral-7B-v0.1 pruned with SparseGPT using different layer-wise sparsity allocation methods.

TABLE 2.4. Comparison of average accuracies across different pruning strategies (Magnitude, Wanda, and SparseGPT) and layer-wise sparsity allocation methods at 50% pruning on Gemma-7B.

Method	Magnitude	Wanda	SparseGPT	Average
UNIFORM	32.21	51.19	49.92	44.44
ALPHA PRUNING	34.21	50.76	51.39	45.45
OWL	33.27	47.15	47.48	42.63
LAYERIF	32.94	53.01	50.90	45.62

Mistral, the same qualitative conclusion holds: a data-driven estimate of layer quality provides a competitive and often superior signal for non-uniform sparsity allocation.

2.9. Summary of Findings

Across both downstream applications, the results support three main conclusions. First, LayerIF consistently improves over strong heuristic and model-centric baselines in both expert allocation and pruning. Second, the gains generalize across multiple model families, including both Mistral-7B and Gemma-7B. Third, the usefulness of the method persists across varying expert budgets and sparsity levels, suggesting that LayerIF captures a stable and practically meaningful notion of task-specific layer quality.

Taken together, these findings support the chapter’s central claim: training-data-aware layer quality estimation provides a more effective signal for post-training resource allocation than approaches based solely on model-internal heuristics.

2.10. Analysis and Discussion

This section analyzes the behavior of LayerIF in greater detail and discusses the design choices underlying the framework. In particular, we examine how LayerIF differs from prior allocation strategies, how its layer-sensitivity estimates should be interpreted, and how practical implementation choices affect performance.

2.10.1. Task-Specific Layer Allocations. A key difference between LayerIF and prior approaches is that LayerIF produces task-specific allocation profiles. Figure 2.4 compares the layer-wise expert allocations produced by LayerIF with those of ALPHALORA and MOLA. Unlike these baselines, which use the same allocation pattern across all tasks, LayerIF adapts the allocation to the target dataset.

This behavior reflects one of the central claims of the chapter: layer quality is not a fixed property of a pretrained model, but depends on the interaction between model, task, and data. By using influence-based estimates of training-data sensitivity, LayerIF can distinguish between tasks that rely on different parts of the model and allocate resources accordingly. This task specificity is one of the main advantages of the framework over purely model-centric heuristics.

2.10.2. Interpreting Layer Sensitivity. LayerIF is based on the hypothesis that better-trained layers should be less sensitive to perturbations in individual training examples. Under this view, a layer with lower aggregate influence is interpreted as more stable and more mature for the target task, whereas a layer with higher aggregate influence is interpreted as requiring more adaptation or being less costly to modify.

To test this interpretation, we perform an ablation in which the allocation logic is reversed for pruning: instead of preserving layers with lower influence sensitivity more carefully, we prune them more aggressively. Table 2.5 shows that this reversed strategy consistently degrades performance across Magnitude pruning, Wanda, and SparseGPT at 70% sparsity on Mistral-7B-v0.1. This result

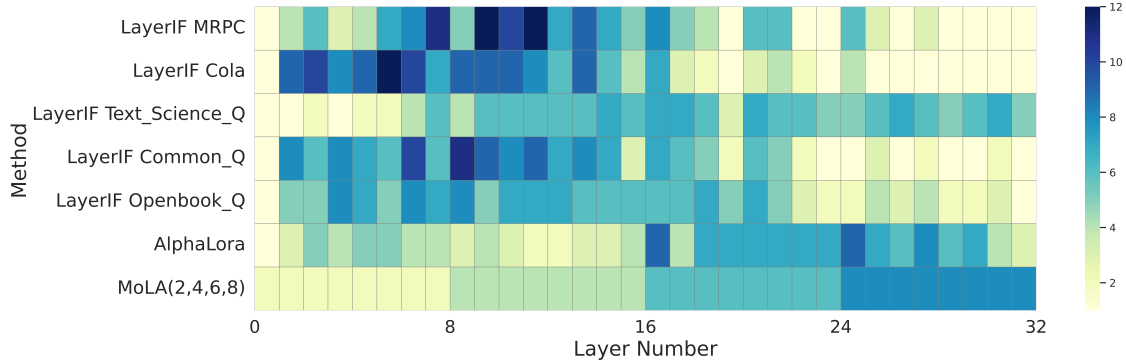


FIGURE 2.4. Heatmap of expert allocations across Transformer layers for Mistral-7B-v0.1 with a total of 160 experts. Unlike ALPHALORA and MoLA, which apply a fixed allocation pattern across tasks, LayerIF produces dataset-specific allocation profiles. Darker shades indicate higher expert allocation to a given layer.

TABLE 2.5. Comparison of accuracies across pruning strategies at 70% sparsity on Mistral-7B-v0.1 using LayerIF versus a reversed allocation strategy.

Method	Magnitude	Wanda	SparseGPT	Average
LayerIF	33.44	32.50	41.14	35.69
Reversed Allocation	33.10	32.02	38.45	34.52

supports the intended interpretation of LayerIF scores: layers with lower influence sensitivity are more important to preserve, not less.

2.10.3. Influence Sample Selection. An important design choice in LayerIF is how to aggregate example-level influence scores into a single scalar score for each layer. Throughout the main experiments, the strongest results are obtained by aggregating only positively influential training samples, and in several cases by restricting further to the top 25% of these positive influences.

This choice can be understood as a form of noise filtering. Influence estimates in deep models depend on stochastic gradient and curvature approximations, and low-magnitude scores may therefore reflect weak or noisy training-validation interactions. Restricting aggregation to the upper tail of positive influences increases the signal-to-noise ratio and focuses the estimate on the training examples that contribute most strongly to the target validation behavior. This interpretation is

also consistent with prior observations that influence values in deep models often exhibit heavy-tailed behavior [20], suggesting that a relatively small subset of training examples accounts for a disproportionately large share of the constructive signal.

In practice, these results suggest that influence aggregation should not necessarily use all training samples uniformly. Instead, selectively focusing on the most beneficial training examples can yield more reliable estimates of layer quality.

2.10.4. Per-Block Versus Per-Matrix Estimation and Smoothing Versus No-smoothing.

In the main experiments, influence scores are computed at the level of Transformer blocks rather than for each individual weight matrix within a block. That is, the attention and MLP subcomponents of a layer are assigned a shared layer-quality score rather than being scored independently. This choice follows the granularity used by prior work in adaptive pruning [27, 26] and reflects the fact that many post-training decisions, such as expert allocation or sparsity assignment, are naturally made at the block level.

To assess this design choice empirically, we conducted an ablation on LLaMA2-7B at 70% sparsity across all three pruning methods. We found that per-block estimation produced an average improvement of 2.14% over more fine-grained alternatives. This suggests that block-level aggregation provides a useful balance between expressivity and stability, capturing meaningful differences across model depth while avoiding the noise introduced by overly fine-grained estimates.

We also examined the effect of smoothing the layer-wise influence profile before mapping scores to sparsity ratios. Smoothing produced more stable layer-allocation patterns and modestly improved pruning performance, suggesting that it helps suppress local noise in raw influence estimates without changing the broader layer-quality trend.

2.10.5. Comparison with Alternative Influence Estimators. LayerIF is agnostic to the specific influence-function estimator used, but in practice the choice of estimator matters. One alternative class consists of Hessian-free methods such as TracIn [89], which are computationally efficient but often less faithful than Hessian-based approaches. To assess whether such methods are suitable for layer-quality estimation, we compare the layer-wise allocations obtained from TracIn

TABLE 2.6. Spearman correlation between layer allocations produced by the Hessian-based DataInf estimator and the Hessian-free TracIn estimator across datasets.

Dataset	Correlation
MRPC	-0.22
CoLA	0.44
OpenBookQA	0.09
Text Science Q	-0.69
Common Q	0.12

with those produced by the Hessian-based DataInf approximation [18], which is used in the main experiments.

Table 2.6 reports the Spearman correlation between the resulting allocations across datasets. The correlations are generally weak and inconsistent, suggesting that Hessian-free approximations fail to recover the same layer-quality structure as the Hessian-based method. In other words, although Hessian-free methods are attractive from a computational standpoint, they appear less suitable for the type of layer-sensitive diagnostic required by LayerIF.

We also examined Gauss-Newton-based alternatives such as EK-FAC [20]. In our experiments, these methods were substantially more expensive than DataInf, making them difficult to use at the scale required for the main chapter experiments. Taken together, these observations make DataInf the most practical choice among the estimators considered here.

2.10.6. Computational Considerations. A practical limitation of influence-function methods is computational cost, especially when second-order information is required. In LayerIF, this cost arises primarily from the curvature approximation used to estimate layer-wise influence. However, two considerations mitigate this concern.

First, LayerIF is a training-time diagnostic rather than an inference-time method. Its purpose is to guide structural adaptation decisions such as expert allocation and pruning, not to be executed during deployment. Second, the framework is modular with respect to the influence estimator: as more efficient approximations become available, they can be incorporated directly into LayerIF without changing the overall method.

These considerations suggest that the additional computation introduced by LayerIF should be understood as an offline cost for improved post-training decisions, rather than a runtime burden at inference.

2.11. Limitations

Several limitations of LayerIF should be noted. First, the theoretical foundations of classical influence functions rely on assumptions such as local convexity that do not strictly hold in modern large language models. As with prior IF-based work in deep learning, the use of influence estimates here is therefore best understood as an approximate but empirically useful diagnostic rather than an exact characterization.

Second, the method is inherently data dependent. Because LayerIF estimates layer quality with respect to a training and validation set, the resulting scores depend on how representative those data are of the target task. If the data used for influence estimation are biased or unrepresentative, the resulting layer-quality estimates may be less reliable.

Third, influence estimation remains computationally nontrivial for large models. Although practical approximations such as DataInf make the method feasible, the cost is still higher than that of purely model-centric heuristics. This trade-off is justified when more accurate layer-quality estimates lead to better post-training decisions, but it remains an important practical consideration.

Finally, LayerIF estimates quality at the level of Transformer blocks rather than finer subcomponents. While this granularity is appropriate for the applications studied here, it may not capture more localized structure within a block. Future work could explore hierarchical or submodule-level variants of the framework.

2.12. Broader Implications

The broader significance of LayerIF lies in its demonstration that training-data-aware diagnostics can guide structural decisions in large language models more effectively than model-centric heuristics alone. In this chapter, that principle is instantiated in two settings: allocating adaptation capacity in LoRA-MoE fine-tuning and distributing sparsity budgets in pruning. More generally, however, the same perspective may be useful for other forms of post-training model surgery in which different parts of a network should be treated differently.

LayerIF also contributes to interpretability in a practical sense. Rather than asking only which layers appear important according to internal statistics, it asks how strongly each layer mediates the effect of the training data on downstream behavior. This provides a different lens on model structure: one grounded not just in representations or parameters, but in the interaction between data and model components.

At a broader level, the chapter supports the dissertation’s central thesis that post-training can be made more principled by studying the relationship between training data and model internals directly. LayerIF is one concrete example of how that relationship can be quantified and translated into actionable design choices.

Generalization: Influence-Guided Demonstration Selection

3.1. Introduction

In-context learning (ICL) has emerged as a powerful mechanism for adapting large language models (LLMs) at inference time. By conditioning a model on a small set of input-output demonstrations placed in the prompt, ICL enables a single pretrained model to perform new tasks without gradient-based parameter updates [28, 105]. This flexibility has made ICL one of the most important forms of inference-time adaptation for modern LLMs, with demonstrated benefits across reasoning, classification, and generation tasks [106, 12, 107]. At the same time, prior work has shown that ICL is highly sensitive to the demonstrations used in the prompt: small changes in the selected examples, their ordering, or their phrasing can lead to large differences in downstream performance [108, 106, 95].

Much of the literature on ICL studies this problem under a relatively favorable set of assumptions. Candidate demonstrations are typically assumed to be drawn from the same task distribution as the test query and to be accurately labeled [29, 109]. Under these conditions, demonstration selection can often be treated as a retrieval problem, where the main goal is to identify examples that are semantically similar or otherwise relevant to the test input [110, 111]. However, these assumptions are often difficult to satisfy in realistic deployment settings. In low-resource, highly specialized, or noisy domains, task-aligned demonstrations may be scarce, unavailable, or unreliable. More generally, the pool of accessible labeled examples may contain demonstrations drawn from heterogeneous tasks, shifted distributions, or corrupted labels, making direct retrieval from a clean task-specific pool impossible or suboptimal [30, 31, 32].

This chapter studies ICL under precisely these less idealized conditions. It introduces **Indirect In-Context Learning (Indirect ICL)**, a generalized formulation in which the candidate pool of demonstrations is not assumed to be task aligned, label clean, or even directly suitable for the end

task. Instead, the goal is to select demonstrations from a broader and potentially imperfect pool in a way that still elicits useful behavior from the model. In the terminology of this dissertation, Indirect ICL treats inference-time adaptation as a problem of *indirect supervision*: the available evidence may not match the target task exactly, but it may still contain examples whose interaction with the model helps induce the right behavior. This reframing is central to the chapter and reflects the observation that many practical ICL settings are better characterized by imperfect or incidental supervision [112, 113].

The chapter focuses on two concrete and practically important settings within this broader formulation. The first is the *Mixture-of-Tasks* setting, in which the demonstration pool contains examples from many unrelated tasks and only a small fraction may bear meaningful relation to the end task. The second is the *Noisy ICL* setting, in which the pool comes from the correct task distribution but includes corrupted or otherwise unreliable labels. These settings represent two distinct obstacles to effective generalization at inference time: task mismatch and supervision corruption. Traditional retrieval signals such as semantic similarity can be useful in these settings, but they are often insufficient on their own. A semantically similar example may be harmful if it is mislabeled or induces the wrong latent task structure, whereas an example from a different task may nevertheless be helpful if it elicits a compatible reasoning pattern.

To address this problem, this chapter proposes an influence-guided framework for demonstration selection. The central idea is to use *Influence Functions* (IFs) as a measure of affinity between candidate demonstrations and the target behavior to be elicited. Unlike standard retrieval metrics, which operate primarily on surface or embedding-space similarity, IFs provide a data-centric mechanism to estimate how strongly individual examples support or hinder model behavior on a validation objective [14, 15, 16]. Recent advances have made it increasingly practical to estimate influence in large generative models [18, 19, 20]. Building on this line of work, the chapter argues that influence can serve as a more informative signal than semantic similarity alone when demonstrations are noisy, heterogeneous, or indirectly related to the target task.

The resulting framework treats demonstration selection not simply as a problem of lexical or semantic matching, but as a problem of *inductive bias alignment*: which candidate examples are most likely to activate the model’s latent capabilities in a way that supports correct target-task

behavior? In the Mixture-of-Tasks setting, this leads to influence-based re-ranking strategies that operate after an initial semantic filtering stage. In the Noisy ICL setting, it leads to influence-based pruning and weighting strategies that distinguish helpful examples from corrupted ones. Across both settings, the broader methodological claim is the same: the effectiveness of a demonstration depends not only on how similar it looks to the query, but on how it interacts with the model’s internal decision process.

More broadly, this chapter extends the dissertation’s data-model synergy perspective from parameter-space adaptation to inference-time adaptation. In the previous chapter, influence analysis was used to estimate the quality of internal model components and guide structural post-training decisions. Here, the same general perspective is applied to prompt construction. The chapter shows that generalization in LLMs can be improved not only by changing model parameters, but also by selecting evidence that better aligns the model’s internal inductive biases with the target task. In this sense, Indirect ICL provides a data-driven framework for making inference-time adaptation more principled under realistic conditions of task mismatch and noisy supervision.

3.2. Related Work

In-Context Learning and Demonstration Selection. With the scaling of model size and training data, large language models have acquired strong inference-time adaptation capabilities through in-context learning (ICL) [28, 114]. In the standard ICL setting, a model is conditioned on a small set of input-output demonstrations and then applied to a new test instance without gradient-based parameter updates. A substantial body of work has shown, however, that ICL performance is highly sensitive to the choice of demonstrations [115, 106, 116]. This has led to a growing literature on demonstration selection, where the goal is to identify examples that are most helpful for eliciting correct behavior from the model.

One major line of work treats demonstration selection as a retrieval problem and scores candidate examples using model-agnostic criteria such as perplexity [117], mutual information [118], or semantic similarity [106, 119]. Another line of work emphasizes structural properties of the selected prompt, such as diversity [120, 121], coverage [111], or other desirable prompt-level features [110, 122]. These approaches have shown that demonstration choice matters substantially, but most of them assume a

relatively favorable setting in which the candidate pool is already aligned with the target task and contains reliable labels.

ICL Under Distribution Shift and Noisy Supervision. More recent work has begun to relax these assumptions by studying ICL under noisier and less idealized conditions. Prior research has investigated how in-context learning behaves when demonstrations are mislabeled, noisy, or otherwise unreliable [32], and has explored methods for mitigating the effect of corrupted demonstrations [30]. Other work has examined more general forms of distribution mismatch, showing that the utility of demonstrations may degrade when the example pool is not cleanly aligned with the target task or domain [31, 123]. These studies help motivate the setting considered in this chapter: in many realistic deployment scenarios, the available demonstration pool may be heterogeneous, weakly related to the task of interest, or partially corrupted.

Our work builds on this line of research but differs in both formulation and objective. Rather than viewing these conditions as special cases of otherwise standard ICL, we treat them as part of a broader setting of *indirect supervision*, in which useful demonstrations need not come from the same task distribution as the test query. This motivates the Indirect ICL framework introduced in this chapter, which unifies the *Mixture-of-Tasks* and *Noisy ICL* settings under a common demonstration selection perspective.

Influence Functions and Data Attribution. *Influence Functions* (IFs) originate in robust statistics [14, 124] and provide a principled way to relate model behavior back to the training examples that shaped it. In modern machine learning, IFs have been adapted as a scalable approximation to leave-one-out retraining for gradient-based models trained under empirical risk minimization [16]. Subsequent work has developed alternative formulations and improved approximations for deep learning, including Representer Point [80], Hydra [81], DataInf [18], Arnoldi-based methods [19], and Kronecker-factored curvature approximations [20]. Other approaches have explored simpler Hessian-free or gradient-based surrogates [89, 91], ensemble strategies [44, 88], and self-influence measures [86, 87].

Beyond methodological development, IFs have been used in a wide range of applications, including mislabeled data detection [16, 17], generative-model analysis [18, 19, 20], active learning [21, 22], and selective annotation for ICL [125]. These works establish IFs as a general-purpose tool for data

attribution. In this chapter, we build on that perspective and use IFs not to analyze the contribution of training data alone, but to select inference-time demonstrations whose interaction with the model is most likely to support correct target-task behavior.

Influence-Guided Demonstration Selection. A small but growing body of work has explored the use of influence for demonstration selection in direct ICL settings. For example, recent studies have used influence-based selection for bias mitigation, faster demonstration retrieval, or improved prompt construction when the demonstration pool is already task aligned [126, 127, 128]. These works show that influence can be a useful alternative to surface-level similarity for identifying helpful examples.

Our work differs from these approaches in two important respects. First, it focuses on *indirect* ICL settings in which the candidate pool is not assumed to be cleanly aligned with the target task. Second, it emphasizes the role of *inductive bias alignment*: the best demonstration is not necessarily the one that looks most similar to the test query, but the one that most effectively activates the model’s latent behavior for the target task. To our knowledge, this chapter is the first to formulate influence-guided demonstration selection explicitly in the Mixture-of-Tasks and Noisy ICL settings and to use influence as a mechanism for selecting demonstrations under indirect supervision.

Taken together, prior work has established three relevant facts: ICL is highly sensitive to demonstration choice; realistic demonstration pools are often noisy or distributionally mismatched; and influence functions provide a principled way to relate model behavior to individual examples. What remains underexplored is how these ideas can be combined to guide demonstration selection when the available examples are not directly task aligned. This chapter addresses that gap through the Indirect ICL framework, which uses influence-guided selection to improve inference-time adaptation under heterogeneous and imperfect supervision.

3.3. Preliminaries

This section introduces the formal setup for traditional in-context learning, the broader Indirect ICL setting studied in this chapter, and the influence-function machinery used to guide demonstration selection.

3.3.1. Traditional In-Context Learning. In-context learning (ICL) allows a large language model to solve a new test instance by conditioning on a small set of labeled demonstrations placed

directly in the prompt. Let $\{(x_i, y_i)\}_{i=1}^k$ denote a set of demonstration examples, let $T(\cdot)$ be a prompt template, and let x_{test} denote a test input. In the standard ICL setting, the language model is prompted to generate a prediction for the test example according to

$$y_{\text{test}} \sim P_{\text{LM}}(\cdot \mid T(x_1, y_1, \dots, x_k, y_k, x_{\text{test}})).$$

A central problem in ICL is *demonstration selection*. In practice, the available pool of labeled examples is usually much larger than the number of examples that can be placed in the prompt, due to both context-length constraints and inference cost [110, 111]. Given a candidate pool $\{(x_i, y_i)\}_{i=1}^N$ with $N \gg k$, the goal is to select a subset of size k that maximizes the probability of producing the correct output for the test input.

Most prior work studies this problem in a *direct* ICL setting, where the candidate demonstrations are assumed to come from the same task as the test example and to be correctly labeled [29, 109]. Under these assumptions, demonstration selection is often framed as a retrieval problem over a task-aligned pool.

3.3.2. Indirect In-Context Learning. This chapter studies a broader and more realistic formulation of inference-time adaptation, which we call *Indirect In-Context Learning*. In this setting, the candidate demonstration pool is not assumed to be task aligned, label clean, or directly suitable for the end task.

Formally, let

$$\mathcal{D} = \{(x_i, y_i, t_i)\}_{i=1}^N$$

denote a pool of candidate demonstrations, where $x_i \in \mathcal{X}$ is an input, $y_i \in \mathcal{Y}$ is a label, and $t_i \in \mathcal{T}$ denotes a task identity. Crucially, we do not assume that the task identities t_i are observed, that they match the test-time task, or that the labels are noise free.

Given a test input x_{test} and a prompt template $T(\cdot)$, the objective is to select a subset

$$S \subset \mathcal{D}, \quad |S| = k,$$

such that conditioning on S yields the strongest possible prediction for the target output:

$$S^* = \arg \max_{\substack{S \subset \mathcal{D} \\ |S|=k}} P_{\text{LM}}(y_{\text{test}} \mid T(S, x_{\text{test}})).$$

This formulation generalizes traditional ICL by removing two standard assumptions: first, that demonstrations belong to the same task distribution as the test example, and second, that their labels are correct. In this sense, Indirect ICL reframes demonstration selection as inference-time adaptation under imperfect supervision.

In this chapter, we study two concrete instantiations of this broader setting:

Mixture-of-Tasks. In this setting, the candidate demonstration pool contains examples drawn from a mixture of different tasks. Some of these examples may be only weakly related to the end task, while others may nonetheless prove useful if they induce a compatible reasoning pattern. The central challenge is therefore to identify examples from a heterogeneous pool that best support the target behavior.

Noisy ICL. In this setting, the candidate pool comes from the correct task distribution but contains corrupted or unreliable labels. The challenge is not task mismatch, but supervision quality: semantically relevant examples may still be harmful if they are mislabeled.

Taken together, these two settings capture the two main ways in which available demonstrations can fail to satisfy the assumptions of standard ICL: they may come from the wrong task, or they may come with the wrong label.

3.3.3. Influence Functions. To address these settings, this chapter uses *Influence Functions* (IFs) as a data-centric signal for demonstration selection. Let

$$D = \{(x_i, y_i)\}_{i=1}^n$$

be a training dataset, let ℓ denote a loss function, and let θ^* denote the parameters of a model trained by empirical risk minimization:

$$\theta^* = \arg \min_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^n \ell(y_i, f_{\theta}(x_i)).$$

Influence Functions estimate how the learned parameters would change if a particular training example were infinitesimally upweighted [14, 15, 92]. For a training point (x_k, y_k) , the influence on the learned parameters is given by

$$I_{\theta^*}(x_k, y_k) = -H(\theta^*)^{-1} \nabla_{\theta} \ell(y_k, f_{\theta}(x_k)),$$

where $H(\theta^*)$ is the Hessian of the empirical loss.

More relevant for our purposes is the influence of a training point on validation loss. Given a validation set

$$D^{\mathcal{V}} = \{(x_i^{\mathcal{V}}, y_i^{\mathcal{V}})\}_{i=1}^m,$$

the influence of (x_k, y_k) on the validation objective can be written as

$$I(x_k, y_k) = - \sum_{i=1}^m \nabla_{\theta} \ell(y_i^{\mathcal{V}}, f_{\theta}(x_i^{\mathcal{V}}))^{\top} H(\theta^*)^{-1} \nabla_{\theta} \ell(y_k, f_{\theta}(x_k)).$$

Intuitively, this score measures whether a training example helps or hurts performance on the validation set. In the context of this chapter, IFs are used to estimate whether a candidate demonstration is likely to support correct target-task behavior. A higher influence score indicates that the example is more beneficial with respect to the chosen validation objective.

As in prior IF-based work for deep models, we use these estimates as an approximate but empirically useful diagnostic, even though the convexity assumptions underlying classical IF theory do not strictly hold for LLMs [16, 20, 17, 129]. Recent work has nonetheless shown that influence-based methods can be effective in large generative models [18, 19, 130, 90], which motivates their use here.

3.4. The Indirect ICL Framework

This section describes how influence-based demonstration selection is instantiated in the two settings studied in this chapter: Mixture-of-Tasks and Noisy ICL. Across both settings, the guiding principle is the same: rather than selecting demonstrations solely on the basis of semantic similarity, we use influence to estimate which examples are most likely to support correct target-task behavior.

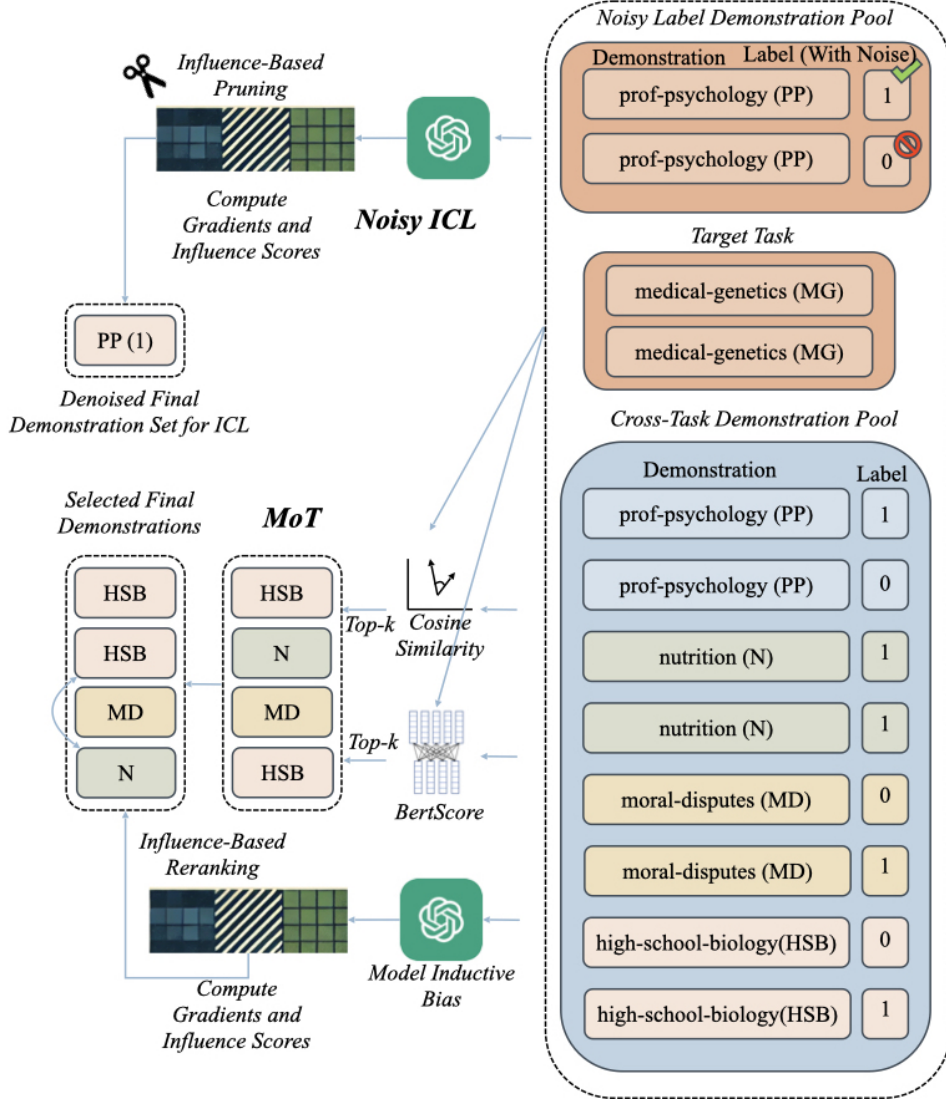


FIGURE 3.1. Example showcasing demonstration selection for Indirect ICL using Influence Functions (IFs). Consider web corpora with many tasks (different from the end-task) and noisy data— Indirect ICL can be formalized as: *Mixture of Tasks* (Section 3.4.2) and *Noisy* (Section 3.4.3) ICL, respectively. In MoT, for a given target task (e.g. *Medical Genetics*), we first filter from this (indirect) pool of candidate demonstrations using BertScore and Cosine Similarity, then re-rank with IFs to select suitable demonstrations (e.g. *High-School Biology*). For Noisy ICL, we leverage IFs to filter out the Noisy ICL before conducting ICL with the remaining clean demonstrations.

3.4.1. Overview. Indirect ICL treats demonstration selection as a problem of identifying examples whose interaction with the model is beneficial for the target task, even when the available demonstration pool is noisy, heterogeneous, or only indirectly related to the query. In practice, this

means using influence estimates either to re-rank candidate demonstrations, to filter out harmful ones, or to combine influence with more conventional similarity-based retrieval signals.

The chapter studies two broad strategies for influence estimation. The first is a *surrogate-model approach*, in which a lightweight encoder model such as RoBERTa [131] is fine-tuned on the available candidate demonstrations and then used to compute influence scores. The second is a *pretrained-gradient approach*, in which gradients are extracted directly from the LLM and influence is estimated in that parameter space. These alternatives allow us to examine whether influence is more effective when computed in a specialized surrogate model or in the underlying generative model itself.

3.4.2. Mixture-of-Tasks: Influence-Guided Demonstration Selection. In the Mixture-of-Tasks setting, the candidate pool contains examples from multiple tasks, and only a subset may be useful for the target problem. Here, the goal is to identify examples from a heterogeneous pool that are most helpful for the validation behavior associated with the target task.

For each validation example $(x_{\text{val}}, y_{\text{val}})$, we compute the influence of every training example (x_i, y_i) in the candidate pool. This yields a score

$$I((x_i, y_i), (x_{\text{val}}, y_{\text{val}})),$$

which estimates how beneficial training example (x_i, y_i) is for the validation example. The simplest influence-only selection strategy is then to rank the candidate pool by influence and choose the top k examples as in-context demonstrations.

In practice, however, purely influence-based retrieval may be unnecessarily expensive or may ignore useful semantic filtering. We therefore also study a two-stage selection strategy. In the first stage, a semantic retrieval method such as BERTScore-Recall, cosine similarity, or BM25 is used to prune the candidate pool down to a smaller subset. In the second stage, this subset is re-ranked using influence scores, and the final top- k examples are selected. This hybrid strategy combines the coarse relevance signal of semantic retrieval with the finer-grained data-model signal provided by influence.

We compare two influence estimators in this setting. The first is DataInf [18], a Hessian-based approximation that is computationally practical for large models. The second is TracIn [89], a Hessian-free method based on gradient similarity across training trajectories. This comparison allows

us to assess the trade-off between computational efficiency and the quality of the resulting influence signal.

3.4.3. Noisy ICL: Influence-Guided Filtering and Reweighting. In the Noisy ICL setting, the candidate pool belongs to the correct task distribution but contains mislabeled or otherwise unreliable examples. Here, influence is used not to distinguish across tasks, but to identify which examples are likely to be harmful or helpful under corrupted supervision.

We study two strategies in this setting.

IF Pruning. The first strategy uses influence to identify harmful examples and remove them from the candidate pool before standard demonstration retrieval is applied. After this pruning step, the remaining examples are ranked using BERTScore-Recall or cosine similarity, and the top- k are selected as demonstrations. This approach uses influence as a filtering signal for noisy supervision.

IF Averaging. The second strategy combines influence values directly with similarity-based retrieval scores. Both signals are normalized and then combined linearly:

$$\text{Score}(x_i, y_i) = \alpha I((x_i, y_i), (x_{\text{val}}, y_{\text{val}})) + \beta S(x_i, y_i),$$

where $I(\cdot)$ is the influence score, $S(\cdot)$ is a semantic similarity score such as BERTScore or cosine similarity, and $\alpha + \beta = 1$. The top- k examples under this combined score are then selected as demonstrations.

This formulation reflects the central claim of the chapter: semantic relevance alone is not enough in noisy settings, but influence alone may also discard useful surface similarity. Combining the two allows the method to favor demonstrations that are both semantically relevant and empirically beneficial for target-task behavior.

In this setting, we use the surrogate-model approach for influence estimation and compare multiple IF algorithms, including DataInf and LiSSA [132, 16]. Although LiSSA is more computationally expensive, it provides a second-order approximation better suited to identifying mislabeled examples than simpler first-order methods.

3.4.4. Practical Instantiations and Design Choices. The framework studied in this chapter therefore includes three design dimensions. First, it varies in the *source of gradients*: a lightweight

surrogate model or the underlying LLM itself. Second, it varies in the *influence estimator*: DataInf, TracIn, or LiSSA. Third, it varies in the *selection strategy*: influence-only selection, two-stage re-ranking, pruning, or influence-similarity averaging.

These design choices are evaluated empirically in the experimental sections that follow. Taken together, they allow the chapter to test not only whether influence is useful for Indirect ICL, but also how it should be instantiated in practice under different forms of task mismatch and label corruption.

3.5. Experimental Setup

This section describes the evaluation settings used to assess Indirect ICL in the two settings studied in this chapter: Mixture-of-Tasks and Noisy ICL.

3.5.1. Tasks, Candidate Pools, and Evaluation. For the *Mixture-of-Tasks* setting, we construct a heterogeneous demonstration pool in which candidate input-output pairs need not correspond to the same task as the test input. The candidate pool contains three examples each from 28 tasks drawn from MMLU [133], BigBench [134], StrategyQA [135], and CommonsenseQA [96]. Performance is then evaluated on 12 held-out tasks from MMLU and BigBench using ICL accuracy.

For the *Noisy ICL* setting, we follow the noisy-data framework of [18]. Specifically, we use the binary classification GLUE datasets MRPC, QQP, QNLI, and SST2 [94], and introduce label corruption by randomly flipping 20% of the training labels. These noisy datasets serve as the candidate demonstration pools, and ICL accuracy is again used as the evaluation metric.

3.5.2. Base Models. In the Mixture-of-Tasks experiments with $k = 3$ shots, we evaluate on Llama-2-13B-Chat [136], Mistral-7B-v0.3 [137], Zephyr-7B-Beta [138], Qwen2.5-3B [139], and Llama-3-70B [140]. For $k = 5$ shots, we report results on Llama-2-13B-Chat. Following prior work on ICL selection [111, 119], the temperature is fixed to 0 during inference.

For Noisy ICL, experiments are conducted primarily on Llama-2-13B-Chat. All experiments were run on 8×NVIDIA RTX 6000 Ada GPUs.

3.5.3. Methods and Baselines.

Mixture-of-Tasks. We evaluate four IF-only variants. Two use a surrogate-model approach, denoted SUR, and two use pretrained gradients from the LLM itself, denoted PRE. For each approach, we compare DataInf and TracIn, yielding SUR_D, SUR_T, PRE_D, and PRE_T. The surrogate-model methods use RoBERTa [131], while the pretrained-gradient methods use the underlying LLM.

We compare these IF-based methods with strong retrieval baselines, including BERTScore-Recall (BSR), cosine similarity (Cos), and BM25. We also evaluate combined methods that first prune the candidate pool using BSR or Cos and then re-rank the remaining examples using influence scores.

Noisy ICL. In the noisy setting, we evaluate two influence-guided strategies: IF PRUNING, which removes a portion of the candidate pool before standard retrieval, and IF AVERAGING, which combines normalized influence scores with semantic similarity scores. Here we use only the surrogate-model approach and compare both DataInf and LiSSA [16]. Unless otherwise stated, we use 10% pruning and set the influence weight to $\alpha = 0.5$ for averaging. Baselines again include BSR, Cos, and BM25.

3.5.4. Implementation Details. For semantic retrieval, we follow the implementations of [111]. Cosine similarity is computed using *all-mpnet-base-v2* from SentenceBERT, BM25 uses Okapi BM25, and BERTScore-Recall uses *deberta-large-mnli* as the encoder. Random baselines are averaged over multiple seeds.

3.6. Results on Mixture-of-Tasks

3.6.1. Main Results. Figure 3.2 summarizes average performance across Llama-2-13B-Chat, Zephyr-7B-Beta, and Mistral-7B-v0.3 for $k = 3$ shots. The clearest pattern is that combining influence-based re-ranking with semantic retrieval yields the strongest overall performance. In particular, SUR_[D,BSR] achieves the best average results, indicating that DataInf combined with BERTScore-Recall provides the most effective selection signal in the main Mixture-of-Tasks regime.

This result supports the central claim of the chapter: semantic similarity alone is often insufficient when the demonstration pool is task heterogeneous, but influence alone is also not enough. The strongest performance arises when semantic retrieval is used to identify a small set of plausibly

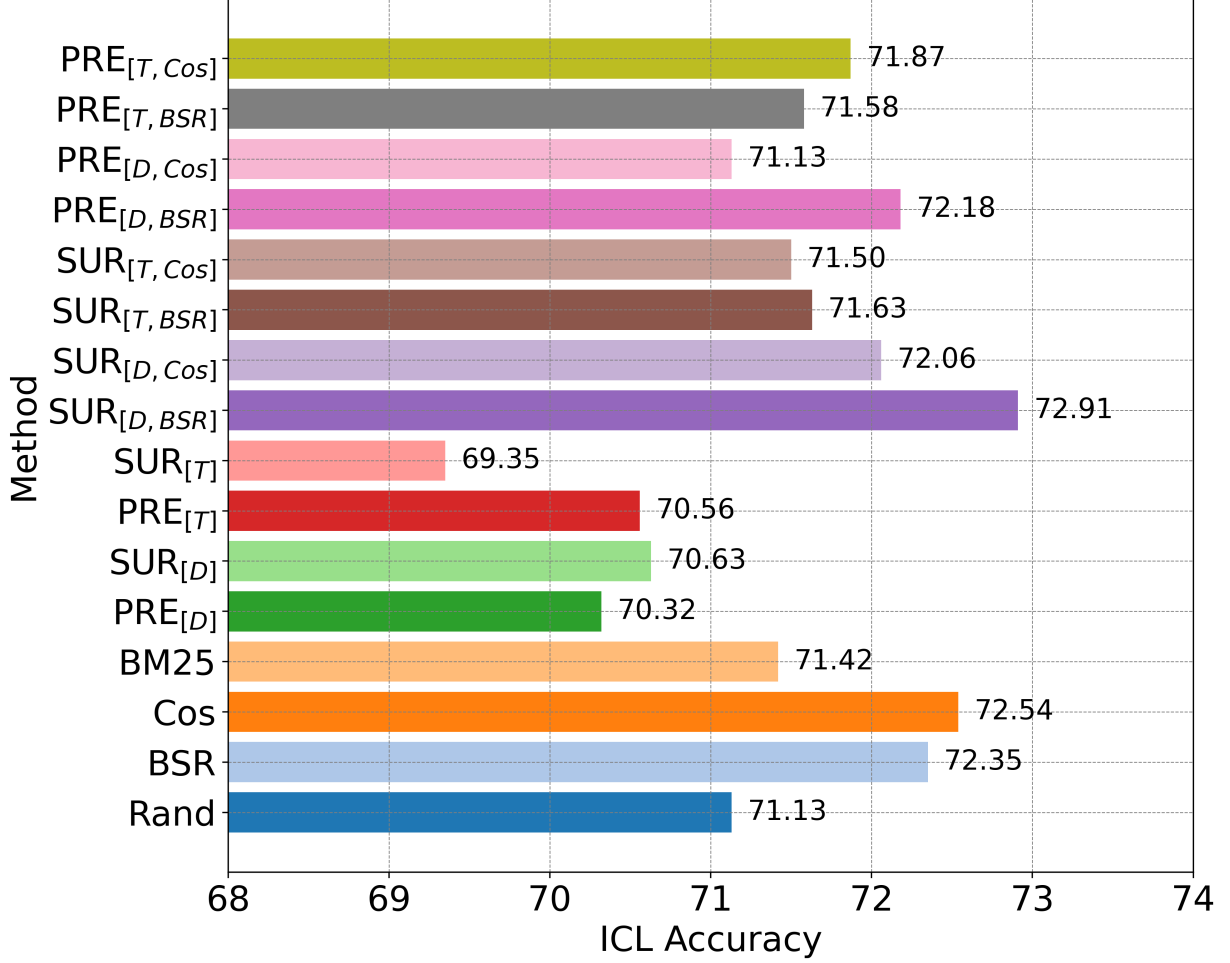


FIGURE 3.2. Average performance of demonstration-selection methods across Llama-2-13B-Chat, Zephyr-7B-Beta, and Mistral-7B-v0.3 for $k = 3$ shots in the Mixture-of-Tasks setting.

relevant demonstrations and influence is then used to re-rank them according to their likely benefit for target-task behavior.

3.6.2. Effect of Number of Shots. The same qualitative pattern holds when the number of demonstrations is increased. In the 5-shot setting, $\text{SUR}_{[D, \text{BSR}]}$ again achieves the strongest average performance, reinforcing the conclusion that influence-guided re-ranking remains useful beyond a single prompt budget. More broadly, these results suggest that the gains from Indirect ICL are not tied to a particular shot count, but reflect a more general improvement in demonstration selection under task mismatch.

TABLE 3.1. Performance across datasets and demonstration-selection methods with $k = 5$ shots for Llama-2-13B-Chat in the Mixture-of-Tasks setting.

Dataset	RAND	BSR	Cos	BM25	PRED	SURD	SUR _[D,BSR]	SUR _[D,Cos]
medical-genetics	80.00	86.00	81.00	81.00	84.00	80.00	84.00	83.00
prof-psychology	71.00	71.00	73.50	66.00	70.00	68.50	77.00	71.00
formal-logic	62.70	59.52	57.94	56.35	58.73	68.25	62.70	61.90
moral-disputes	79.50	77.50	81.00	81.00	82.00	81.00	81.50	81.50
public-relations	70.00	78.18	81.82	76.36	70.91	77.82	80.91	78.18
comp-security	75.00	78.00	82.00	77.00	76.00	77.00	81.00	77.00
astronomy	78.95	85.53	82.89	80.92	80.92	82.89	84.87	81.58
abstract-algebra	52.00	63.00	62.00	58.00	63.00	55.00	67.00	65.00
nutrition	71.50	81.00	80.00	78.00	76.00	78.00	79.00	81.00
high-school-biology	73.00	79.00	80.00	76.50	77.00	79.00	81.50	75.50
formal-fallacies	46.50	48.50	48.00	50.00	47.50	43.50	53.00	43.50
tracking-3	36.50	49.00	47.00	46.00	42.50	52.00	42.00	39.00
Average	66.38	71.35	71.42	68.93	69.04	70.24	72.87	69.84

3.6.3. Transfer to Closed-Source Models. An important question is whether the proposed framework remains useful when the final inference model is closed source and its internal weights are inaccessible. To test this, we evaluate the surrogate-model-based selection strategy on GPT-4.1-nano in the Mixture-of-Tasks setting. Because model internals are unavailable, demonstrations are selected entirely using the surrogate model, and inference is then performed in a prompt-only manner with the closed-source LLM.

As shown in Table 3.2, adding three demonstrations selected by the surrogate-based Indirect ICL pipeline substantially improves performance over zero-shot inference, raising average accuracy from 66.14 to 76.57. This result is important because it shows that the framework is not limited to open-weight models. The influence-guided selection mechanism can still improve prompt construction even when the final deployed model cannot itself be analyzed directly.

3.6.4. Comparison with Strong Baselines. To assess the strength of the proposed method more rigorously, we compare SUR_[D,BSR] against several recent demonstration-selection baselines, including DETAIL [128], CONE [141], INFICL [127], and MOICL [142]. These baselines span several approaches, including influence-based selection for direct ICL, non-influence-based retrieval, and training-based demonstration weighting.

TABLE 3.2. Performance comparison between zero-shot inference (No ICL) and $k = 3$ demonstrations selected via the surrogate-model approach on the closed-source model GPT-4.1-nano in the Mixture-of-Tasks setting.

Dataset	No ICL	$k = 3$ Shots
medical-genetics	86.00	91.00
prof-psychology	71.50	80.50
formal-logic	50.79	73.02
moral-disputes	70.50	85.00
public-relations	75.45	76.36
comp-security	81.00	77.00
astronomy	82.89	94.00
abstract-algebra	53.00	57.00
nutrition	80.00	87.50
high-school-biology	72.00	89.50
formal-fallacies	52.00	59.00
tracking-3	18.50	49.00
Average	66.14	76.57

TABLE 3.3. Performance comparison between MOICL, DETAIL, INFICL, CONE, and $\text{SUR}_{[\text{D}, \text{BSR}]}$ for Llama-2-13B-Chat with $k = 3$ shots in the Mixture-of-Tasks setting.

Dataset	MOICL	DETAIL	INFICL	CONE	$\text{SUR}_{[\text{D}, \text{BSR}]}$
medical-genetics	77.00	77.00	82.00	81.00	86.00
prof-psychology	70.00	66.50	68.00	71.00	72.00
formal-logic	57.14	61.90	65.08	61.90	57.14
moral-disputes	79.50	79.50	79.50	81.50	80.50
public-relations	68.18	70.00	75.45	75.45	79.09
comp-security	75.00	71.00	78.00	80.00	76.00
astronomy	77.63	77.63	78.29	80.92	80.26
abstract-algebra	53.00	71.00	54.00	62.00	72.00
nutrition	75.50	76.00	79.50	80.00	79.50
high-school-biology	75.00	75.50	78.50	76.50	80.50
formal-fallacies	46.00	48.00	46.00	47.50	47.50
tracking-3	27.00	34.00	36.50	48.00	43.50
Average	65.07	67.34	68.40	70.48	71.16

As shown in Table 3.3, $\text{SUR}_{[\text{D}, \text{BSR}]}$ achieves the highest average accuracy among all compared methods. This result is especially notable because the proposed method is designed for the more difficult Indirect ICL setting, where the demonstration pool is not cleanly aligned with the target task.

TABLE 3.4. ICL accuracy across MRPC, QNLI, SST2, and QQP using different methods for Noisy ICL with 20% label noise.

	Method	MRPC	QNLI	SST2	QQP
	RAND	70.4	69.6	86.2	70.9
	BSR	71.3	74.6	80.4	71.4
	Cos	72.3	68.2	82.6	73.2
	BM25	70.6	67.6	88.0	71.0
PRU-0.1	RAND _[Cos]	70.1	68.4	87.0	69.4
	SUR _[D,Cos]	69.4	69.8	84.0	71.8
	SUR _[L,Cos]	68.9	68.2	86.8	70.8
	RAND _[BSR]	71.1	65.2	86.4	68.4
	SUR _[D,BSR]	70.6	68.0	82.0	71.0
	SUR _[L,Cos]	70.1	67.4	88.8	68.2
AVG-0.5	SUR _[D,Cos]	75.5	74.8	89.8	67.8
	SUR _[L,Cos]	70.6	75.8	86.4	73.0
	SUR _[D,BSR]	74.3	69.6	90.6	73.8
	SUR _[L,BSR]	73.3	73.4	93.6	69.2

3.7. Results on Noisy ICL

3.7.1. Main Results. Figure 3.3 and Table 3.4 report results on MRPC, QNLI, SST2, and QQP with 20% label noise. Across datasets, the strongest methods are consistently those based on IF AVERAGING, which combine surrogate-model influence scores with either cosine similarity or BERTScore-Recall. At least one, and often both, of the strongest-performing methods on each dataset come from this averaging family.

These results support a central distinction between the Mixture-of-Tasks and Noisy ICL settings. In the heterogeneous-task setting, influence works best as a re-ranking signal after semantic pruning. In the noisy-label setting, by contrast, influence is most effective when it is integrated directly with semantic similarity through a weighted combination. This suggests that under label corruption, the best demonstrations are those that are not only semantically relevant, but also empirically beneficial according to the influence signal.

We also find that DataInf and LiSSA are comparably effective in this setting, though DataInf remains preferable from a practical standpoint due to its lower computational cost.

3.7.2. Effect of Noise Level and Hyperparameters. We additionally examined the robustness of the Noisy ICL results under different noise levels and different hyperparameter settings for

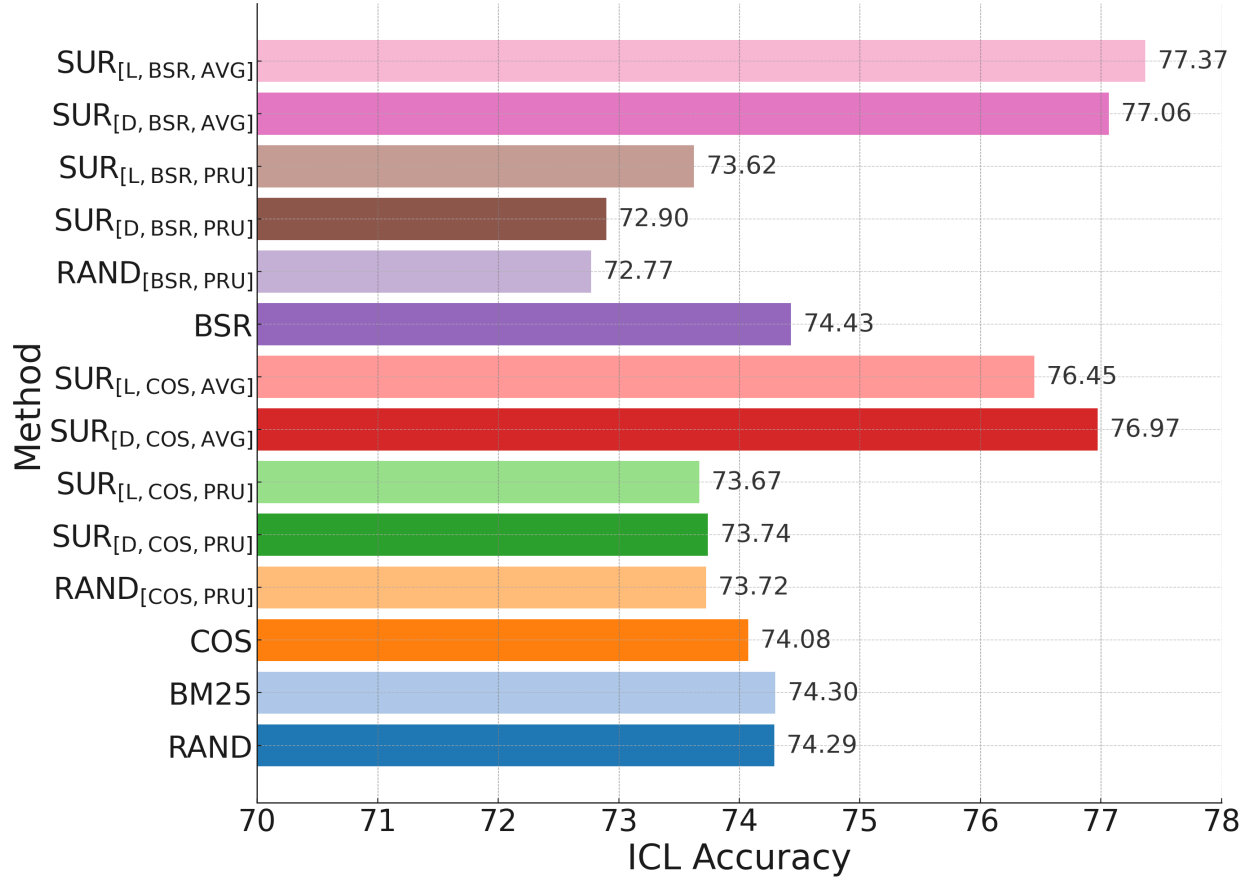


FIGURE 3.3. Average performance of demonstration-selection methods across the four Noisy ICL datasets.

pruning and averaging. The overall trends remain consistent: averaging-based methods are more reliable than pruning-based methods, and influence is most useful when used to refine rather than discard semantically relevant examples. For the purposes of the main chapter, we therefore focus on the 20% noise setting and the representative hyperparameters above.

3.8. Analysis and Discussion

3.8.1. Why Combining Influence with Semantic Retrieval Helps. A recurring pattern across the Mixture-of-Tasks experiments is that the best results do not come from semantic similarity alone or from influence alone, but from their combination. This can be understood by examining what each signal contributes. Semantic retrieval methods such as BERTScore-Recall and cosine similarity are effective at filtering the candidate pool down to examples that are at least broadly

relevant to the query. In our ablation, BERTScore-Recall selected demonstrations from the same task as the test example 33.90% of the time, while cosine similarity did so 36.03% of the time. Although these rates are far from perfect, they are sufficient to narrow the search space to semantically plausible candidates.

Influence-based re-ranking then provides a complementary signal. Rather than selecting the most semantically similar example, it identifies the example most likely to support correct target-task behavior given the model’s inductive biases. This explains why the two-stage procedure consistently outperforms either component in isolation.

3.8.2. Surrogate Models Versus Pretrained Gradients. Another consistent finding is that the surrogate-model approach outperforms the pretrained-gradient approach in the main Mixture-of-Tasks setting. A likely explanation is that the surrogate model, once fine-tuned on the candidate pool, becomes better aligned with the structure of the target selection problem, whereas gradients extracted directly from the pretrained LLM remain more influenced by broad language-modeling objectives.

At the same time, additional experiments on larger models suggest that pretrained-gradient methods become more competitive as model scale increases. On Qwen2.5-3B, $\text{PRE}_{[\text{D}, \text{BSR}]}$ slightly outperforms the surrogate-based variant on average, as shown in Table 3.5. A similar pattern appears on Llama-3-70B, where $\text{PRE}_{[\text{D}, \text{Cos}]}$ achieves the strongest average performance among the compared methods in Table 3.6. Taken together, these results suggest that surrogate models are preferable in the primary regime studied in this chapter, but that pretrained gradients may become more useful as the underlying LLM becomes stronger and more specialized.

3.8.3. Alternative Surrogate Models. To examine whether the effectiveness of the surrogate-based approach depends strongly on the particular encoder used, we replace RoBERTa-Large with DeBERTa-v3-Large and compare $\text{SUR}_{[\text{D}, \text{BSR}]}$ against the BSR baseline on Llama-2-13B-Chat with $k = 3$ shots. As shown in Table 3.7, the DeBERTa-based surrogate again improves over pure BERTScore retrieval on average, raising performance from 69.82 to 70.30. Although the margin is modest, it shows that the Indirect ICL framework is not tied to a single surrogate architecture.

TABLE 3.5. Performance comparison between BSR, $\text{SUR}_{[\text{D},\text{BSR}]}$, and $\text{PRE}_{[\text{D},\text{BSR}]}$ for Qwen2.5-3B with $k = 3$ shots in the Mixture-of-Tasks setting.

Dataset	BSR	$\text{SUR}_{[\text{D},\text{BSR}]}$	$\text{PRE}_{[\text{D},\text{BSR}]}$
medical-genetics	88.00	90.00	87.00
prof-psychology	80.00	82.50	81.50
formal-logic	71.43	66.67	68.25
moral-disputes	83.50	80.00	82.50
public-relations	79.09	79.09	80.91
comp-security	84.00	87.00	85.00
astronomy	85.53	88.16	88.82
abstract-algebra	64.00	63.00	63.00
nutrition	84.50	85.50	85.00
high-school-biology	87.50	87.00	89.00
formal-fallacies	49.00	50.00	50.50
tracking-3	48.00	48.00	49.00
Average	75.38	75.57	75.87

TABLE 3.6. Performance comparison between BSR, $\text{SUR}_{[\text{D},\text{BSR}]}$, $\text{PRE}_{[\text{D},\text{BSR}]}$, COS, $\text{SUR}_{[\text{D},\text{Cos}]}$, and $\text{PRE}_{[\text{D},\text{Cos}]}$ for Llama-3-70B with $k = 3$ shots in the Mixture-of-Tasks setting.

Dataset	BSR	$\text{SUR}_{[\text{D},\text{BSR}]}$	$\text{PRE}_{[\text{D},\text{BSR}]}$	COS	$\text{SUR}_{[\text{D},\text{Cos}]}$	$\text{PRE}_{[\text{D},\text{Cos}]}$
medical-genetics	95.00	94.00	97.00	97.00	98.00	98.00
prof-psychology	92.50	91.00	90.50	91.50	91.50	90.50
formal-logic	72.22	77.78	74.60	73.81	73.02	79.37
moral-disputes	89.00	91.00	90.00	87.00	90.50	91.00
public-relations	86.36	84.55	84.55	83.64	85.45	86.36
comp-security	86.00	86.00	86.00	87.00	88.00	87.00
astronomy	98.68	98.68	97.37	98.03	97.37	99.34
abstract-algebra	68.00	66.00	67.00	65.00	65.00	66.00
nutrition	95.50	93.50	93.50	95.50	95.50	94.00
high-school-biology	93.50	94.00	93.50	94.00	94.50	94.00
formal-fallacies	59.00	66.50	60.50	59.50	63.00	66.00
tracking-3	43.50	46.00	47.00	48.00	45.50	47.50
Average	81.605	82.418	81.793	81.665	82.278	83.256

3.8.4. DataInf Versus TracIn. Across the Mixture-of-Tasks experiments, DataInf consistently outperforms TracIn as an influence estimator. This is notable because TracIn is considerably simpler and more computationally efficient, but the performance gap suggests that the second-order information retained by DataInf is important for this task. A plausible explanation is that TracIn effectively treats the Hessian as the identity and therefore fails to capture more subtle aspects of the relationship between training examples and validation behavior. This interpretation is also consistent

TABLE 3.7. Performance comparison between BSR and $\text{SUR}_{[\text{D}, \text{BSR}]}$ when DeBERTa-v3-Large is used as the surrogate model on Llama-2-13B-Chat with $k = 3$ shots.

Dataset	BSR	$\text{SUR}_{[\text{D}, \text{BSR}]}$
medical-genetics	87.00	85.00
prof-psychology	70.00	72.50
formal-logic	59.52	59.52
moral-disputes	78.00	84.50
public-relations	79.09	76.36
comp-security	76.00	77.00
astronomy	80.26	80.26
abstract-algebra	58.00	59.00
nutrition	77.50	79.00
high-school-biology	76.50	79.00
formal-fallacies	52.00	45.50
tracking-3	44.00	46.50
Average	69.82	70.30

TABLE 3.8. Percentage of noisy samples among the top 100 examples ranked by influence in the Noisy ICL setting.

Dataset	DataInf	LiSSA
MRPC	83%	66%
QNLI	54%	86%
QQP	79%	95%
SST-2	90%	96%

with prior work showing that first-order influence approximations can be substantially weaker than Hessian-based methods [17].

3.8.5. Influence for Identifying Mislabeled Demonstrations. To better understand why influence helps in the Noisy ICL setting, we conducted a simple experiment measuring how often IF-based ranking places mislabeled examples among the top 100 most influential points. As shown in Table 3.8, both DataInf and LiSSA are highly effective at surfacing noisy examples. This provides a mechanistic explanation for the success of IF Averaging: influence scores help distinguish semantically relevant demonstrations that are still harmful because of corrupted labels.

3.8.6. Qualitative Evidence for Inductive-Bias Alignment. For MoT, to understand the merits of our method, we compare demonstrations selected via BSR and $\text{SUR}_{[\text{D}, \text{BSR}]}$ on the MMLU-abstract-algebra dataset:

Example 1:

Q: Compute the product in the given ring. $(2, 3)(3, 5)$ in $\mathbb{Z}_5 \times \mathbb{Z}_9$

Options: (B) (3,1) (C) (1,6)

BSR Shots.

- (1) **Q:** Statement 1 | Every element of a group generates a cyclic subgroup of the group. Statement 2 | The symmetric group S_{10} has 10 elements. **Options:** (A) True, True (C) True, False **Answer:** (C)
- (2) **Q:** Statement 1 | Every function from a finite set onto itself must be one-to-one. Statement 2 | Every subgroup of an abelian group is abelian. **Options:** (A) True, True (D) False, True **Answer:** (A)
- (3) **Q:** How many attempts should you make to cannulate a patient before passing the job on to a senior colleague, according to the medical knowledge of 2020? **Options:** (A) 4 (B) 3 (C) 2 (D) 1 **Answer:** (C)

SUR_[D,BSR]

- (1) **Q:** Statement 1 | Every function from a finite set onto itself must be one-to-one. Statement 2 | Every subgroup of an abelian group is abelian. **Options:** (A) True, True (D) False, True **Answer:** (A)
- (2) **Q:** Olivia used the rule "Add 11" to create the number pattern shown below: 10, 21, 32, 43, 54. Which statement about the number pattern is true? **Options:** (B) The number pattern will never have two even numbers next to each other. (D) If the number pattern started with an odd number, then the pattern would have only odd numbers in it. **Answer:** (B)
- (3) **Q:** Tomorrow is 11/12/2019. What is the date one year ago from today in MM/DD/YYYY format? **Options:** (B) 11/11/2018 (C) 08/25/2018 **Answer:** (B)

We can see that while BSR selects more semantically relevant samples, SUR_[D,BSR]'s selected shots guide the model toward the correct answer (C) instead of (B) by encouraging more structured reasoning.

Example 2:

Q: Statement 1 | If R is an integral domain, then $R[x]$ is an integral domain. Statement 2 | If R is a ring and $f(x)$ and $g(x)$ are in $R[x]$, then

$$\deg(f(x)g(x)) = \deg f(x) + \deg g(x).$$

Options: (C) True, False (B) False, False

BSR Shots

- (1) **Q:** Pence compares six different cases of reproduction, from natural twinning to SCNT. What conclusion does he draw from this comparison? **Options:** (A) SCNT is not a different kind of reproduction because there are no morally relevant differences between it and other permissible means of reproduction. (B) Because there is a low risk of harm for natural twinning, there will be a low risk of harm for SCNT. (C) Both A and B (D) Neither A nor B **Answer:** (A)
- (2) **Q:** Statement 1 | Every element of a group generates a cyclic subgroup of the group. Statement 2 | The symmetric group S_{10} has 10 elements. **Options:** (A) True, True (C) True, False **Answer:** (C)
- (3) **Q:** Statement 1 | Every function from a finite set onto itself must be one-to-one. Statement 2 | Every subgroup of an abelian group is abelian. **Options:** (A) True, True (D) False, True **Answer:** (A)

SUR_[D,BSR]

- (1) **Q:** Statement 1 | Every function from a finite set onto itself must be one-to-one. Statement 2 | Every subgroup of an abelian group is abelian. **Options:** (A) True, True (D) False, True **Answer:** (A)
- (2) **Q:** Select the best translation into predicate logic. George borrows Hector's lawnmower. (g : George; h : Hector; l : Hector's lawnmower; $Bxyx$: x borrows y from z). **Options:** (A) B_{lgh} (B) B_{hlg} (C) B_{glh} (D) B_{ghl} **Answer:** (C)
- (3) **Q:** Statement 1 | Every element of a group generates a cyclic subgroup of the group. Statement 2 | The symmetric group S_{10} has 10 elements. **Options:** (A) True, True (C) True, False **Answer:** (C)

Here again, BSR selects the more semantically relevant shots (with the top three shots ordered in ascending order of relevance), while $\text{SUR}_{[\text{D}, \text{BSR}]}$ selects less semantically similar but more influential shots, which ultimately improves model performance.

3.8.7. Integration into Practical Workflows. Indirect ICL is particularly relevant in settings where task-specific demonstration pools are unavailable or unreliable at inference time, but broader labeled resources are still accessible. One important use case is real-time prompt construction for general-purpose LLM systems, where the exact task may not be known in advance and a task-agnostic pool of examples must be searched on the fly. In such settings, the Mixture-of-Tasks formulation provides a practical way to improve prompt quality without requiring clean, directly task-matched demonstrations.

Additionally, we provide several use cases of Indirect ICL in real-world scenarios. Here are a few examples:

- **Medical diagnosis:** Indirect ICL can be used to diagnose rare medical conditions based on symptoms. Since such conditions are rare, demonstrations for these specific cases are often unavailable. However, the model can learn diagnostic reasoning patterns from more common conditions with overlapping symptoms, improving accuracy for the rare cases.
- **Code generation for obscure programming languages:** Indirect ICL can aid in generating code for rarely-used or proprietary programming languages. Demonstrations from code generation tasks in related languages with similar structures can be leveraged, enabling the model to generalize and perform well in these low-resource scenarios.
- **Ideology Estimation from Underrepresented Contexts:** We can use our paradigm to estimate political ideology, or any other sort of text classification, from text in an underrepresented cultural or linguistic context. We can use demonstrations from ideology estimation in well-represented contexts such as Western political texts. They can transfer learned associations between linguistic cues and ideological stances, adapting them to the new context.

These examples highlight just a few of the practical applications of indirect ICL, particularly in low-resource settings.

TABLE 3.9. Defense results on AddSent and Style backdoor attacks. ASR denotes attack success rate; lower is better.

SST-2	AddSent	Style
ASR		
No Defense	100.00	98.68
Task-Aware	69.41	32.02
SUR _[D,BSR]	58.22	10.86

3.8.8. Indirect ICL as a Backdoor Defense Mechanism. To probe the robustness of the framework under a different class of adversarial noise, we also evaluate influence-guided demonstration selection as a task-agnostic backdoor defense. Here, the setting again reflects imperfect supervision: clean task-specific labeled data may be unavailable at inference time, but a broader pool of related examples is still accessible.

We apply SUR_[D,BSR] to SST-2 [94] poisoned with two types of backdoor attacks: AddSent [143], which inserts a trigger sentence, and Style [144], which poisons data via style transfer. We compare against a task-aware defense baseline [145] that selects five demonstrations from a clean task-specific dataset. Results are shown in Table 3.9.

The influence-guided indirect strategy reduces attack success rate more effectively than the task-aware baseline in both attack settings. This result is noteworthy because it shows that even without direct task-specific supervision, demonstrations selected through inductive-bias alignment can provide a meaningful defense signal. More broadly, it suggests that Indirect ICL may be useful not only for improving standard accuracy, but also for improving robustness under adversarially corrupted supervision.

3.8.9. Computational Considerations. Table 3.10 summarizes the asymptotic inference-time complexity of the major selection strategies. Influence-based methods are comparable to other strong baselines and, in the combined setting, can be applied after semantic pruning to only a small subset of candidates.

In practice, the surrogate-based methods also have a clear memory advantage. In our experiments on 4 NVIDIA RTX 6000 Ada Generation GPUs, SUR_D required 7,998 MiB of memory compared

with 18,188 MiB for PRE_D . This difference makes the surrogate approach more practical when computational resources are limited.

We also measured the time required to extract test gradients and compute IF scores across model scales using DataInf on the MMLU moral-disputes dataset with 200 test samples. As shown in Table 3.11, runtime grows sublinearly in the tested regime when moving from RoBERTa-Large (125M) to Llama-2-13B-Chat and Llama-2-70B-Chat. Interestingly, although RoBERTa is far smaller, IF computation takes longer because the surrogate model is fine-tuned and influence is computed over its LoRA-adapted components, whereas in the larger LLaMA models no fine-tuning is performed and only LayerNorm weights produce non-zero gradients. This substantially reduces the effective parameter space for IF estimation. For comparison, TracIn on Llama-2-13B-Chat requires only 3.576×10^{-6} seconds, illustrating its substantial speed advantage, albeit at a clear cost in empirical performance.

Finally, we assess scalability with respect to dataset size in Table 3.12. Increasing the number of samples from 100 to 1000 leads to an approximately $10\times$ increase in runtime, indicating roughly linear scaling with sample size. In the Mixture-of-Tasks setting, this cost can be reduced further by computing IF only over the $2k$ candidates retained after semantic pruning rather than over the full candidate pool. Additional implementation improvements, such as moving DataInf gradient handling from nested Python dictionaries to tensorized GPU-based operations, could likely reduce runtime further.

Taken together, these results suggest that Hessian-based influence computation remains practical for offline demonstration selection, especially in the surrogate-model regime, while TracIn remains attractive only when speed is prioritized over selection quality.

3.9. Limitations

Several limitations of the empirical study should be noted. First, the chapter focuses on two specific forms of indirect supervision, task heterogeneity and label corruption, and does not exhaust the broader range of ways in which demonstration pools may be imperfect in practice. Second, the strongest methods rely on influence estimation, which introduces additional computational cost relative to purely similarity-based retrieval, even if this cost remains manageable offline. Third, the

TABLE 3.10. Worst-case per-test-sample time complexity of the main demonstration-selection methods. Here N is the number of candidate demonstrations, p is the number of model parameters, d is embedding size, K is the maximum candidate length, L is the test-input length, and Z is the number of n-grams.

Method	Time Complexity
BSR	$O(NLKd)$
Cos	$O(Nd)$
BM25	$O(NZ)$
PRED	$O(Np)$
SURD	$O(Np)$
PRET	$O(Np)$
SURT	$O(Np)$
BSR COMBINED METHODS	$O(NLKd) + O(Np)$
Cos COMBINED METHODS	$O(Nd) + O(Np)$

TABLE 3.11. Time required for extracting test gradients and computing IF across different model sizes using DataInf on MMLU moral-disputes with 200 test samples.

Model	Test Gradients (s)	Computing IF (s)
RoBERTa	7.447	35.72
Llama-2-13b-chat	68.5	4.69
Llama-2-70b-chat	257.64	8.81

TABLE 3.12. Time required for extracting test gradients and computing IF across different sample sizes using DataInf.

Model & Samples	Test Gradients (s)	Computing IF (s)
RoBERTa (100 Samples)	6.7	26.4
RoBERTa (200 Samples)	7.4	35.7
RoBERTa (1000 Samples)	67.8	273.7
Llama-2-13b-chat (100 Samples)	41.0	3.6
Llama-2-13b-chat (200 Samples)	68.5	4.7
Llama-2-13b-chat (1000 Samples)	410.8	35.4

relative strength of surrogate models versus pretrained-gradient methods appears to depend in part on model scale, suggesting that the best instantiation of Indirect ICL may vary across LLM families. Finally, although the main results establish that influence-guided selection improves performance, the precise mechanisms by which certain demonstrations activate beneficial latent reasoning patterns remain only partially understood.

3.10. Conclusion

This chapter introduced Indirect In-Context Learning as a generalized framework for demonstration selection under imperfect supervision. Across both the Mixture-of-Tasks and Noisy ICL settings, the results show that influence-guided selection improves over strong semantic retrieval baselines and recent demonstration-selection methods. The strongest performance arises when influence is combined with semantic similarity rather than used in isolation, indicating that the most useful demonstrations are those that are both relevant on the surface and beneficial with respect to the model’s internal behavior.

More broadly, these findings extend the dissertation’s data-model synergy perspective from parameter-space adaptation to prompt construction. They show that inference-time generalization can be improved by selecting evidence in a way that is informed not only by semantic relevance, but also by how individual examples interact with the model’s latent inductive biases.

CHAPTER 4

Robustness: Characterizing Failure Modes in LLM Summarization

4.1. Introduction

This chapter studies the *robustness* of large language models in zero-shot abstractive summarization. The central question is whether a summarization model remains reliably grounded in source evidence under benign variation in the input. This question is important because strong aggregate performance does not by itself imply reliable behavior: a model may achieve competitive ROUGE or related summary-quality scores while still drawing information from the wrong parts of a document, or while producing unstable summaries when salient source content is rephrased in a meaning-preserving way [35].

We examine this problem through two complementary studies. The first studies **position bias**, asking whether a model allocates attention to source regions in a way that is aligned with human-written summaries. Prior work has often discussed *lead bias*, especially in news summarization, but that framing is too narrow: models may over-rely on the beginning, middle, or end of a document, and such behavior is only problematic when it differs from the positional behavior of gold summaries [36, 37]. The second study examines **relevance paraphrasing**, asking whether a summarizer remains stable when the source sentences most relevant to the gold summary are paraphrased without changing their meaning. Together, these studies evaluate robustness along two dimensions: *where* the model draws information from, and *how stably* it behaves when that information is re-expressed.

More broadly, this chapter extends the dissertation’s data-model synergy perspective into the *diagnostic evaluation* of deployed LLM systems. The earlier chapters used data-model interactions constructively to improve utility and generalization. Here, the same perspective is used diagnostically to identify brittle behavior in zero-shot summarization. The goal is not only to determine whether a

model produces a plausible summary, but whether that summary is grounded in source evidence in a stable and human-aligned manner.

4.2. Background and Related Work

4.2.1. Robustness and Reliability in Large Language Models. Large language models have demonstrated strong performance across a wide range of natural language tasks, including abstractive summarization [146]. Yet robustness in LLMs has been studied primarily in discriminative, reasoning, or instruction-following settings, including shortcut-based analyses [147, 148, 149, 150], benchmark sensitivity [151, 152], and adversarial prompting [153, 154, 155, 156, 157, 158, 159, 160]. By contrast, relatively less work has examined robustness for the structured generative task of abstractive summarization.

This chapter focuses on *general robustness* rather than adversarial robustness. Instead of constructing worst-case, model-specific attacks, it studies whether summarization behavior remains stable under benign, meaning-preserving variation in the source document. This distinction matters because summarization quality depends not only on output fluency, but also on how the model identifies and uses salient source information.

4.2.2. Summarization Evaluation Beyond Overlap Metrics. Abstractive summarization is typically evaluated using overlap- or similarity-based metrics such as ROUGE [161] and BERTScore [162]. These metrics are useful for comparing generated summaries against reference summaries, but they provide only indirect evidence about the generation process itself. In particular, they do not reveal whether the model relied on appropriate source regions or whether its behavior would remain stable under benign paraphrasings of salient evidence.

This limitation is related to broader concerns about faithfulness and factuality in summarization. Prior work has shown that systems can achieve strong automatic scores while still hallucinating, distorting, or omitting key information [35]. Other work on non-LLM summarization has studied robustness through factuality-oriented architectures [163, 164, 165], distribution shift [166], temporal drift [167], and synthetic input corruption [168]. The present chapter asks a different question: whether a summarization model uses source evidence in a stable, human-aligned, and semantically consistent way.

4.2.3. Lead Bias, Position Bias, and Source Allocation. One established phenomenon in summarization is *lead bias*, the tendency to over-rely on the opening portion of the source document, especially in news summarization [169, 170, 171, 37]. Prior work has typically treated this as either a bias to mitigate or a prior to exploit. This chapter adopts the broader notion of *position bias*, under which lead bias is only one special case. A model may over-utilize the beginning, middle, or end of a document, and such behavior should be considered problematic only when it is misaligned with the behavior of human-written summaries. Position bias therefore requires a comparative distributional view of source allocation rather than a fixed preference for any one document region.

4.2.4. Robustness Under Semantic-Preserving Perturbations. A complementary line of work studies robustness under input changes that preserve meaning, including paraphrasing and lexical substitution [172]. Similar ideas have appeared in broader LLM evaluation through benchmark sensitivity and reformulation studies [151, 152], but they have rarely been applied to zero-shot abstractive summarization. This chapter introduces *relevance paraphrasing*, a framework that perturbs only the source sentences most relevant to the gold summary. Because the perturbation is localized and meaning-preserving, it provides a direct test of whether summarization behavior remains stable when salient evidence is expressed differently.

4.3. Shared Problem Setting: Zero-Shot Abstractive Summarization

4.3.1. Why Zero-Shot Summarization is a Useful Robustness Testbed. Both studies in this chapter are conducted in the setting of *zero-shot abstractive summarization*. In this setting, the model must summarize without task-specific fine-tuning, which exposes inference-time behavior more directly than supervised summarization pipelines do. The model must determine what information in a long document is most important, how to compress it, and how to express it coherently using only the capabilities acquired during pretraining and instruction tuning.

This makes zero-shot summarization a useful robustness testbed. If a model is robust, its summary quality should not depend excessively on superficial aspects of the input, such as where salient sentences appear or whether those sentences are phrased one way or another. Robust summarization therefore requires both appropriate source selection and stability under benign linguistic variation.

4.3.2. Datasets, Models, and Evaluation. Across the two studies, we evaluate multiple summarization datasets and model families, including both modern LLMs and pretrained encoder-decoder models. The datasets span different discourse structures and summary styles, allowing us to study whether the same robustness patterns hold across settings.

The evaluation perspective differs from standard summarization benchmarking. Rather than asking only whether a generated summary matches a reference summary, we ask whether the model’s generation process remains reliably grounded in source evidence. The first study operationalizes this through positional source distributions; the second through controlled paraphrastic perturbations of summary-relevant sentences.

4.3.3. A Unified View of Robustness in Summarization. The two studies target complementary dimensions of robustness. The first examines *structural robustness*: whether the model uses the appropriate regions of the source document. The second examines *behavioral robustness*: whether the model preserves its content-selection behavior when salient source evidence is re-expressed without changing its meaning. Together, they motivate a broader evaluation view in which robustness is assessed not only through output quality, but also through source-groundedness and stability of the generation process.

4.4. Study I: Position Bias in Zero-Shot Summarization

4.4.1. Motivation. Lead bias has long been discussed in summarization, especially for news, where important content often appears early in the article [169, 170, 171, 37]. But a general robustness analysis requires a broader framing. A model may over-rely on the beginning, middle, or end of a document, and whether that behavior is problematic depends on how human summaries themselves distribute attention across the source. This motivates the notion of *position bias*: systematic mismatch between the source regions emphasized by the model and those emphasized by gold summaries. An example of a *positionally biased* summary is shown in Figure 4.1.

Article: During a peaceful kayaking trip on a serene river, John found himself in a frantic situation when he realized he had lost his phone. His faithful dog, Max, was his only companion on this adventure..... Hours passed, and just when hope seemed to wane, John's perseverance paid off as he spotted a glimmer of his phone beneath the riverbank's mud. With his phone safe in hand, the kayaking trip became an unforgettable adventure filled with both despair and triumph.

Gold Summary: Amidst despair and triumph, a lost phone is finally recovered during a kayaking adventure with a loyal canine companion.

Model Summary: A man's kayaking trip with his dog takes a stressful turn when he loses his phone on a serene river.

FIGURE 4.1. An example of *position bias* where gold summary is *tail biased* and model summary is *lead biased*.

4.4.2. From Lead Bias to Position Bias. Let an article $x \in X$ contain N_x sentences, and let a generated summary $s \in S = \mathcal{A}(X)$ contain N_s sentences. Because we consider *abstractive* summarization, there is no exact one-to-one mapping between summary sentences and article sentences. We therefore assume a proxy mapping function ϕ that maps a summary sentence $s_i \in s$ to the article sentence $x_j \in x$ that contributed most to it. In practice, ϕ can be approximated using similarity-based methods such as TF-IDF or ROUGE-style matching.

To make positions comparable across documents of different lengths, each article x is divided into K approximately equal segments, where $K \leq \min_{x \in X} |x|$. Once summary sentences are mapped back to source sentences via ϕ , those sentences can be mapped to segments, producing a positional distribution over source regions for both generated summaries and gold summaries.

4.4.3. Estimating Positional Source Distributions. For a given dataset and model, we estimate positional distributions by applying ϕ to each summary sentence and recording the corresponding source segment. Aggregating across documents yields an empirical distribution that characterizes where the model, or the gold summaries, tend to draw information from. This makes it possible to study not only lead bias, but any systematic positional mismatch.

4.4.4. Measuring Bias via Distributional Mismatch. We quantify position bias as the distance between the gold and generated positional distributions, using Wasserstein distance [173].

Intuitively, this measures how much probability mass must be shifted across source regions to transform the model’s distribution into the gold distribution. Low distance indicates alignment with human source allocation; high distance indicates positional mismatch.

This provides a robustness metric complementary to ROUGE. A model may achieve strong overlap-based performance while still exhibiting substantial position bias, which means that summary quality alone does not fully capture how the model uses the source document.

4.5. Study II: Robustness Through Relevance Paraphrasing

4.5.1. Motivation. The second study asks whether a summarization model remains stable when the most important source content is paraphrased without changing its meaning. In the summarization setting, the most informative robustness test is not to perturb the entire document uniformly, but to perturb the source sentences most relevant to the reference summary. This motivates *relevance paraphrasing*: a framework that identifies summary-relevant source sentences, paraphrases them, and compares model performance on the original and perturbed articles. Thus, by measuring the change in performance on both the original and perturbed set of input articles, we can assess LLM summarization robustness. An example of *relevance paraphrasing* is shown in Figure 4.2.

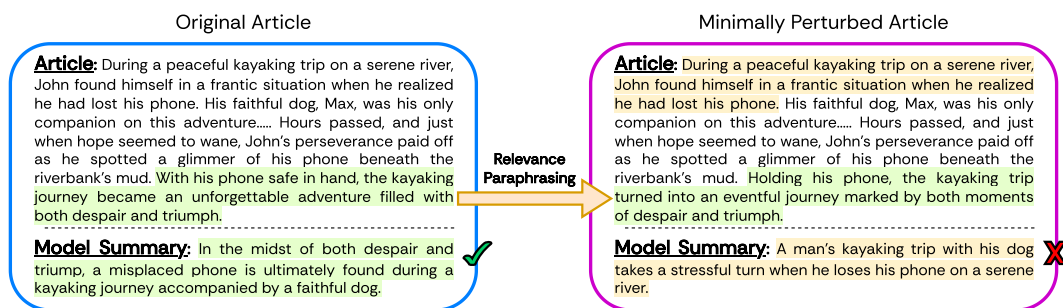


FIGURE 4.2. An example showcasing *relevance paraphrasing*. When sentences *relevant* to generating the summary are *paraphrased* to create a minimally perturbed article, we find that summarization performance drops as the model uses other sentences instead to craft the summary, leading to a loss of salient information.

4.5.2. Identifying Summary-Relevant Sentences. Let an article be denoted by $x \in X$ and its corresponding gold summary by $g \in G$. We assume a proxy mapping ψ that maps a gold-summary

Algorithm 1 Relevance Paraphrasing

- 1: **Input:** summarization model $\mathcal{M}$, dataset $T = (X, S^G)$, mapping function ψ , paraphrasing model θ , evaluation metric $\mathcal{E}$
 - 2: Initialize $X' \leftarrow \emptyset$
 - 3: **for** each article-summary pair (x, s) with $x \in X$ and $s \in S^G$ **do**
 - 4: Let $I_x = \{j \mid x_j = \psi(s_i), \forall s_i \in s\}$
 - 5: Construct x' by replacing each x_i , for $i \in I_x$, with $\theta(x_i)$
 - 6: Update $X' \leftarrow X' \cup \{x'\}$
 - 7: **end for**
 - 8: Measure $\mathcal{E}(S^G, \mathcal{M}(X))$ and $\mathcal{E}(S^G, \mathcal{M}(X'))$
-

sentence $g_i \in g$ to the source sentence $x_j \in x$ that contributed most to it [174, 175]. In practice, ψ is approximated using TF-IDF similarity. Applying ψ to all summary sentences yields the set

$$I_x = \{j \mid x_j = \psi(g_i), \forall g_i \in g\},$$

that is, the set of source sentences most relevant to the gold summary.

4.5.3. Constructing Minimally Perturbed Articles. Let θ denote a paraphrasing model that rewrites a sentence while preserving its meaning. In this study, an instruction-tuned Llama-2-13B model is used for this purpose. For each relevant source sentence x_i with index $i \in I_x$, we obtain a paraphrased version x'_i using θ and replace the original sentence in the article. Repeating this process yields a minimally perturbed article x' , and repeating over the dataset produces a perturbed collection X' .

Because only a small number of summary-relevant sentences are changed, and because the paraphrases preserve meaning, the perturbed articles remain semantically close to the originals while altering the surface form of the most salient source content.

4.5.4. Evaluating Stability Under Relevance Paraphrasing. The relevance-paraphrasing procedure is summarized in Algorithm 1. The summarization model is evaluated on both the original articles X and the perturbed articles X' . Robustness is then assessed by comparing the resulting summary-quality metrics.

If

$$\mathcal{E}(G, \mathcal{A}(X)) > \mathcal{E}(G, \mathcal{A}(X')),$$

then performance worsens on the perturbed inputs, indicating that the model is sensitive to minimal paraphrasings of summary-relevant evidence. Because the perturbation preserves meaning, this degradation reflects instability in how the model identifies and uses salient content rather than a genuine change in article semantics.

4.5.5. Why Relevance Paraphrasing is a Useful Robustness Probe. Relevance paraphrasing targets exactly the part of the source document the model ought to care about most. Unlike broad document-level perturbations or model-specific adversarial attacks, it is model-agnostic and explicitly tests stability under ordinary linguistic reformulation. In this sense, it complements the position-bias framework: position bias asks whether the model looks in the right *regions* of the source document, while relevance paraphrasing asks whether it preserves the right *content-selection behavior* when that evidence is expressed differently.

4.6. Experimental Setup

This chapter evaluates robustness in zero-shot abstractive summarization through two complementary studies: *position bias* and *relevance paraphrasing*. Across both studies, we consider four datasets with distinct discourse structures and summarization regimes: *CNN/DM* [176], *XSum* [177], *News* [178], and *Reddit* [174]. These datasets differ substantially in article length, summary length, and the positional structure of their gold summaries, making them suitable for robustness analysis.

We evaluate multiple model families, including instruction-tuned LLMs and pretrained encoder-decoder baselines. For the LLM-based experiments, we consider GPT-3.5_{Turbo}, Llama-2_{13B}, Mistral_{7B}, and Dolly-v2_{7B}. For the position-bias study, we additionally include Pegasus and BART as reference encoder-decoder summarization models. Unless otherwise stated, all results are obtained in the zero-shot setting.

For **Study I**, position bias is measured by mapping summary sentences back to source sentences using a similarity-based proxy function ϕ , converting those source sentences into article-segment positions, and comparing the resulting positional distributions of gold and model-generated summaries using Wasserstein distance. We report ROUGE-based performance alongside position bias in order to study the relationship between overall summary quality and source-allocation behavior.

For **Study II**, robustness is measured through relevance paraphrasing. We identify the source sentences most relevant to the gold summary using a proxy mapping ψ , paraphrase those sentences using Llama-2_{13B}, and compare summarization performance on the original and perturbed articles. We report relative performance change under relevance paraphrasing using ROUGE-1/2/L and BERTScore. Unless otherwise noted, the main relevance-paraphrasing experiments are run with standard non-zero generation temperature to reflect common practical usage, while an additional sensitivity analysis considers temperature-zero decoding.

4.7. Results: Study I: Position Bias in Zero-Shot Summarization

We begin by examining the structural robustness of zero-shot summarizers through the lens of position bias. The central question in this study is whether models draw information from the same regions of the source document that humans rely on when writing summaries.

4.7.1. Gold Summaries Exhibit Distinct Positional Structures Across Datasets. The estimated positional distributions reveal that the datasets are not homogeneous. *CNN/DM* and *News* exhibit lead-oriented gold-summary distributions, whereas *Reddit* and *XSum* are substantially more positionally uniform. This difference is important because positional preference should only be treated as undesirable when it differs from the empirical source-allocation behavior of human-written summaries.

Figure 4.3 visualizes these positional distributions for gold and model-generated summaries. The more these distributions diverge for a given model and dataset, the more positionally biased the model is for that setting.

4.7.2. LLMs Generally Exhibit Lower Position Bias Than Encoder-Decoder Baselines. Figure 4.4 compares summarization performance and position bias across models. Overall, the LLMs tend to achieve both stronger ROUGE performance and lower position bias than the pretrained encoder-decoder models. GPT-3.5_{Turbo} consistently combines strong ROUGE scores with low Wasserstein distance, while Llama-2 also performs strongly among open-weight models. In contrast, BART exhibits comparatively lower zero-shot performance and stronger positional mismatch across datasets.

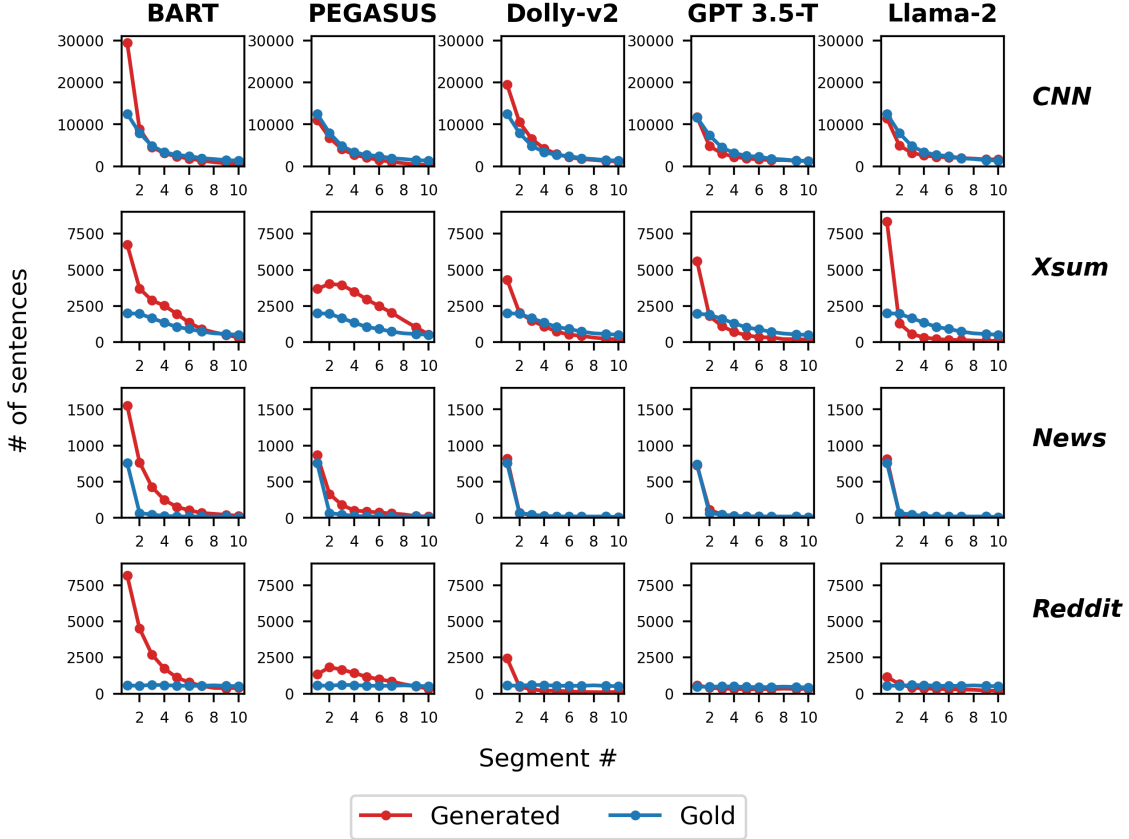


FIGURE 4.3. Visualizing positional distributions of gold and model-generated summaries for multiple datasets. The more these distributions differ for a given dataset and model, the greater the model’s position bias.

These results suggest that modern instruction-tuned LLMs are, on average, better aligned with human source-allocation behavior than zero-shot encoder-decoder baselines.

4.7.3. Extreme Summarization Reveals Stronger Positional Brittleness. A notable exception arises on *XSum*, which constitutes an extreme summarization setting: source articles are relatively long, while gold summaries are typically only one or two sentences. In this regime, both LLMs and encoder-decoder models exhibit much stronger lead bias, despite the gold summaries being comparatively positionally uniform. This suggests that extreme compression places particular pressure on source selection and can amplify positional brittleness even in otherwise strong models.

Comparing *XSum* with *Reddit* is especially instructive. Although both datasets can involve short summaries, *Reddit* does not induce the same degree of lead-oriented model behavior. This indicates

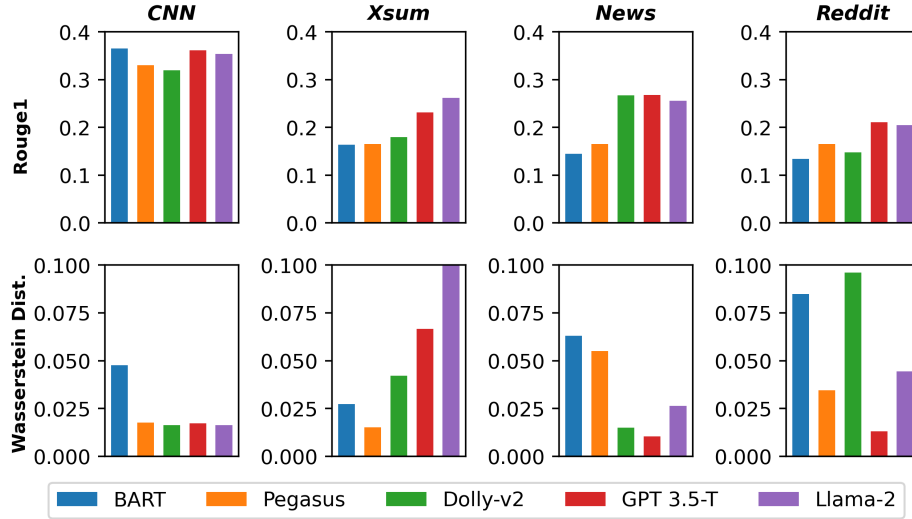


FIGURE 4.4. Measuring summarization performance (R^1 score) and position bias (Wasserstein distance between gold and generated summaries’ positional distributions). Lower Wasserstein distance values correspond to lower position bias.

that the summarization regime itself, not only summary length in isolation, matters for positional robustness.

4.7.4. Position Bias and ROUGE Capture Different Aspects of Summarization Quality. An important finding of Study I is that strong ROUGE performance does not reliably imply low position bias. Some models attain competitive overlap-based quality while still exhibiting substantial positional mismatch, especially on *XSum*. Position bias therefore captures a dimension of summarization behavior that is not directly visible from standard summary-quality metrics alone.

This point is reinforced by the additional ROUGE analyses in Figure 4.5, where the same broad trends hold for ROUGE-2 and ROUGE-L. The existence of these consistent trends across metrics strengthens the interpretation that position bias is a meaningful and stable diagnostic rather than an artifact of one particular overlap score.

4.7.5. Implication of Study I. Taken together, the results of the first study show that zero-shot summarization quality cannot be fully understood from output overlap alone. Models differ not only in how well their summaries match references, but also in how they allocate attention across the source document. This motivates the second study: even if a model appears to draw from

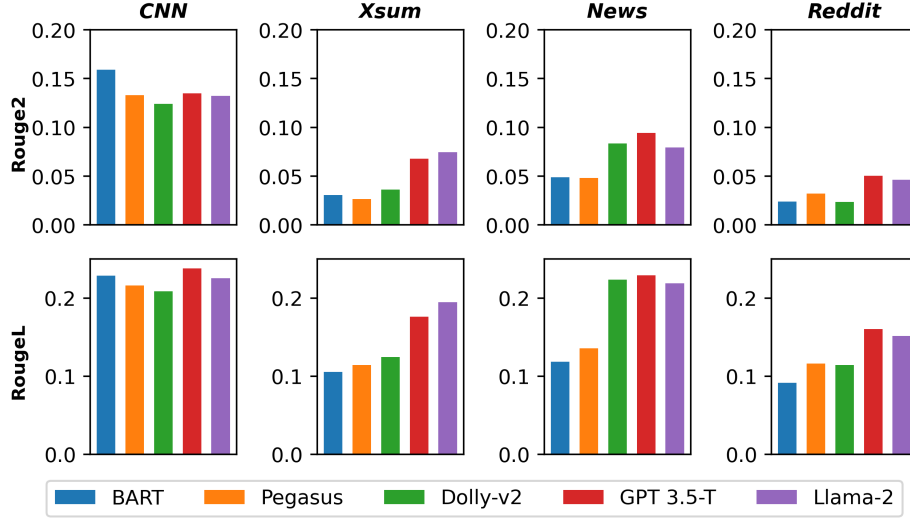


FIGURE 4.5. Additional results for R^2 and R^L metrics.

the right general regions of the document, it may still fail to behave stably when the most salient source content is re-expressed in a meaning-preserving way.

4.8. Results: Study II: Robustness Through Relevance Paraphrasing

We now turn from *where* models draw information from to *how stably* they use that information when salient source content is paraphrased. The second study therefore probes behavioral robustness through controlled, semantics-preserving perturbations of the input articles.

4.8.1. Relevance Paraphrasing Consistently Degrades Summarization Performance.

We present the relative performance change¹ for the original summaries and the summaries obtained after relevance paraphrasing in Table 4.1. We evaluate with four summarization metrics: ROUGE-1, ROUGE-2, ROUGE-L, and BERTScore.

The dominant pattern is clear: summarization performance usually declines after paraphrasing the sentences most relevant to the gold summary. The largest degradations occur on *CNN/DM* and *News*, with particularly large ROUGE-2 drops. The most affected model is Dolly-v2_{7B}, while Mistral_{7B} appears comparatively more robust. GPT-3.5_{Turbo} and Mistral_{7B} show a few isolated improvements, most notably on *XSum*, but these do not overturn the broader trend.

¹Computed as $(\text{new} - \text{old}) / \text{old} \times 100\%$.

TABLE 4.1. Performance change (%) observed after relevance paraphrasing across datasets/LLMs.

Datasets	Metrics	Llama-2 _{13B}	GPT-3.5 _{Turbo}	Dolly-v2 _{7B}	Mistral _{7B}
Performance Change (%)					
CNN	ROUGE-1	(-)7.354	(-)8.750	(-)13.77	(-)6.814
	ROUGE-2	(-)21.20	(-)23.73	(-)31.66	(-)27.72
	ROUGE-L	(-)9.431	(-)13.54	(-)15.70	(-)11.99
	BertScore	(-)0.311	(-)0.689	(-)5.754	(-)0.522
XSum	ROUGE-1	(-)2.837	(+)16.19	(+)0.680	(-)3.680
	ROUGE-2	(-)8.077	(+)12.99	(-)3.607	(-)13.91
	ROUGE-L	(-)3.764	(+)11.41	(+)1.465	(-)3.649
	BertScore	(-)0.092	(+)0.321	(-)0.524	(+)0.047
News	ROUGE-1	(-)10.90	(-)15.41	(-)39.60	(-)7.457
	ROUGE-2	(-)28.43	(-)36.96	(-)50.30	(-)19.43
	ROUGE-L	(-)13.15	(-)17.00	(-)41.79	(-)10.65
	BertScore	(-)0.080	(-)0.707	(-)7.083	(+)0.528
Reddit	ROUGE-1	(-)3.158	(-)6.600	(-)21.85	(-)2.974
	ROUGE-2	(-)13.10	(-)24.13	(-)13.20	(-)13.89
	ROUGE-L	(-)3.529	(-)7.646	(-)27.64	(-)1.700
	BertScore	(-)0.070	(-)0.750	(-)18.84	(+)2.104

4.8.2. LLMs Are Not Fully Consistent Summarizers. Figure 4.6 examines how source-sentence usage changes after relevance paraphrasing. Using the proxy mapping ψ , we estimate which input article sentences contribute to the generated summaries before and after perturbation. The figure shows that models often begin drawing on different source sentences once the relevant sentences are paraphrased. This means that a minor, semantics-preserving perturbation can induce a substantial shift in source selection and, in turn, summary quality.

This behavior is somewhat less pronounced for Mistral_{7B}, but is clearly present across the other LLMs. In this sense, the relevance-paraphrasing results provide behavioral evidence complementing the metric-based results of Table 4.1: the models do not merely score lower after perturbation, but often change the underlying evidence they rely on.

4.8.3. Human Evaluation Supports the Same Trend. To further validate the automatic findings, we conduct human evaluation of article-summary triplets following the methodology of [179, 180]. We recruit 22 student annotators, each of whom evaluates up to 30 article-summary triplets consisting of the gold summary, the original model summary, and the summary produced after relevance paraphrasing. We evaluate all four LLMs on the *XSum* and *Reddit* datasets, with each dataset-model pair annotated by three annotators.

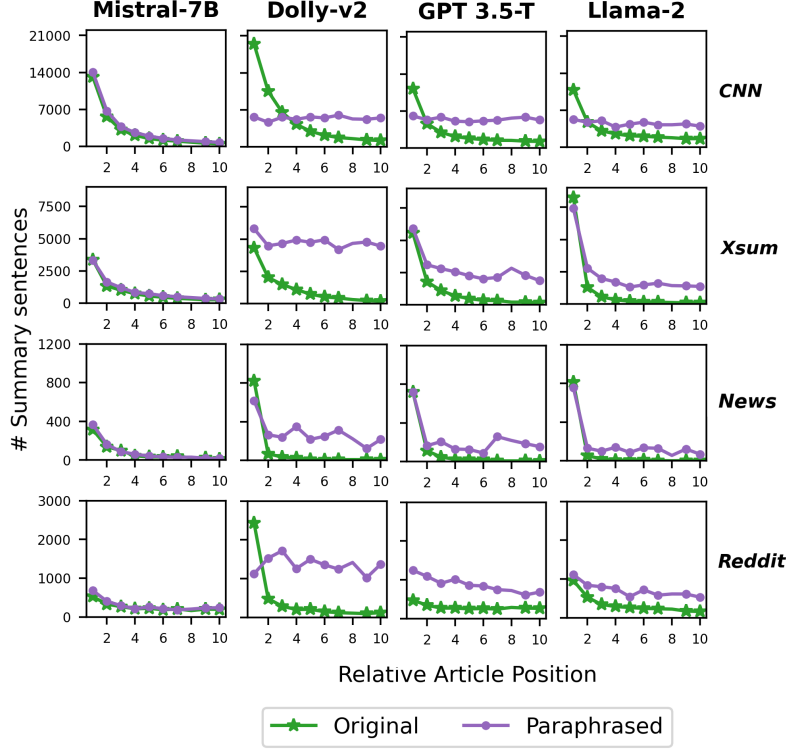


FIGURE 4.6. Paraphrasing results in different summaries.

The following instructions were provided to the annotators² (similar protocol as to [179]: *Annotate the summary quality of the following Article-summary pairs using the definitions. Your scores will be presented in a research article. Definitions:*

- **Faithfulness (Binary Score 0 or 1):** *Whether the summary is faithful to the article or not (on topic).*
- **Coherence (Integer Score between 1 to 5):** *Whether the summary is well-structured and organized. Should not just be a heap of related information*
- **Relevance (Integer Score between 1 to 5):** *Whether the summary selects important content from the source. Summary should only include the important info.*

The results obtained are as follows. For each dataset-model pair the best summary (in bold) is the one with overall high performance on the 3 sub-metrics (to break ties we use the overall highest scores):

²The annotators were grad students of UC Davis residing in Davis, California.

Gold			
Dataset-Model	Faithfulness	Coherence	Relevance
Xsum GPT	0.520(0.242)	3.390(1.257)	2.280(0.440)
Xsum Llama	0.550(0.201)	2.870(0.640)	2.380(0.779)
Xsum Dolly	0.630(0.163)	3.580(0.984)	2.820(0.505)
Xsum Mistral	0.630(0.123)	3.470(0.456)	3.340(0.560)
Reddit GPT	0.720(0.077)	2.750(0.622)	2.820(0.645)
Reddit Llama	0.670(0.133)	2.950(0.572)	2.950(0.230)
Reddit Dolly	0.830(0.095)	3.100(0.272)	3.620(0.308)
Reddit Mistral	0.790(0.160)	3.090(1.057)	3.270(0.500)

Original			
Dataset-Model	Faithfulness	Coherence	Relevance
Xsum GPT	0.950(0.077)	4.000(0.514)	4.380(0.375)
Xsum Llama	0.990(0.030)	3.520(0.600)	3.680(0.380)
Xsum Dolly	0.670(0.116)	3.370(0.552)	2.670(0.289)
Xsum Mistral	0.770(0.177)	3.270(0.668)	2.800(0.608)
Reddit GPT	1.000(0.000)	4.220(0.816)	4.400(0.492)
Reddit Llama	0.920(0.101)	3.850(0.358)	3.900(0.232)
Reddit Dolly	0.520(0.248)	2.340(0.534)	2.070(0.315)
Reddit Mistral	0.400(0.077)	3.100(0.439)	2.220(0.477)

Paraphrased			
Dataset-Model	Faithfulness	Coherence	Relevance
Xsum GPT	0.920(0.073)	3.760(0.545)	3.720(0.146)
Xsum Llama	0.990(0.030)	3.550(0.303)	3.590(0.305)
Xsum Dolly	0.820(0.186)	3.370(0.622)	2.720(0.305)
Xsum Mistral	0.820(0.085)	3.250(0.648)	2.870(0.590)
Reddit GPT	0.890(0.030)	4.130(0.497)	3.920(0.581)
Reddit Llama	0.950(0.071)	3.880(0.363)	4.120(0.093)
Reddit Dolly	0.500(0.179)	2.230(0.737)	2.040(0.506)
Reddit Mistral	0.370(0.173)	2.930(0.509)	1.950(0.469)

Across the eight dataset-model settings, paraphrased summaries are rated best only once. Original summaries are preferred in three cases, while gold summaries are preferred in the remaining four cases. Thus, the human evaluations are broadly consistent with the automatic metric results: relevance paraphrasing tends to reduce summary quality in a way that is perceptible to human readers.

4.9. Discussion

4.9.1. The Two Studies Reveal Complementary Failure Modes. Taken together, the two studies show that robustness in zero-shot summarization has at least two distinct dimensions. The position-bias study identifies *structural* failures in source allocation: a model can achieve strong

	CNN		Reddit	
Metrics	Relevant	Random	Relevant	Random
ROUGE-1	(-)7.135	(-)1.625	(-)3.880	(-)0.665
ROUGE-2	(-)19.84	(-)3.636	(-)16.17	(-)2.212
ROUGE-L	(-)8.498	(-)1.394	(-)4.178	(-)0.179
BertScore	(-)0.238	(-)0.023	(-)0.078	(+)0.060

TABLE 4.2. Effect of paraphrasing summary-relevant sentences versus randomly selected non-relevant sentences.

overlap-based performance while still emphasizing source regions that differ from those emphasized by human-written summaries. The relevance-paraphrasing study identifies *behavioral* failures in selection stability: a model can change its output substantially when salient evidence is re-expressed in a meaning-preserving way.

These are related but non-identical failure modes. A model may draw from roughly the right source regions but still be unstable when those regions are reworded. Conversely, a model may be relatively stable under paraphrasing while still being systematically misaligned in its positional preferences. Together, the studies reinforce the broader claim of this chapter: aggregate summary quality alone is an incomplete proxy for reliability.

4.9.2. Why the Relevance-Paraphrasing Effect Is Meaningful. A central validity question for Study II is whether the observed degradation is truly due to perturbing *salient* content, rather than merely changing the wording of the article in some arbitrary way. Several additional analyses support the stronger interpretation.

First, paraphrasing *non-relevant* sentences has a much smaller effect on summary quality than paraphrasing the source sentences most relevant to the gold summary. This shows that the observed degradation is not simply due to making the article stylistically different; it is specifically tied to changing the wording of the content most central to the summarization task.

Second, paraphrasing a larger proportion of relevant sentences produces somewhat larger performance degradation, especially on *XSum* and *News*. This is consistent with the view that increasing the amount of perturbed salient content further stresses the model’s content-selection stability.

	XSum		NEWS	
Metrics	Top 1	Top 3	Top 1	Top 3
ROUGE-1	(-)2.483	(-)3.091	(-)10.58	(-)11.21
ROUGE-2	(-)5.926	(-)8.302	(-)28.32	(-)33.22
ROUGE-L	(-)2.644	(-)3.831	(-)12.91	(-)13.64
BertScore	(-)0.050	(-)0.076	(-)0.015	(-)0.022

TABLE 4.3. Effect of paraphrasing a larger proportion of relevant source sentences.

TABLE 4.4. Performance change (%) observed after prompting the LLM with the same article-summary pair twice.

Datasets	Metrics	Llama-2 _{13B}	GPT-3.5 _{Turbo}	Dolly-v2 _{7B}	Mistral _{7B}
Performance Change (%)					
CNN	ROUGE-1	(-)0.039	(+)0.328	(-)0.119	(+)0.014
	ROUGE-2	(-)0.088	(+)0.909	(+)0.016	(+)0.134
	ROUGE-L	(-)0.013	(+)0.294	(-)0.040	(-)0.029
	BertScore	(+)4.66E-08	(+)0.009	(+)2.60E-08	(+)7.75E-08
XSum	ROUGE-1	(-)0.014	(+)0.347	(+)0.069	(-)0.033
	ROUGE-2	(-)0.033	(-)1.622	(+)0.049	(-)0.329
	ROUGE-L	(+)0.096	(+)0.128	(-)0.052	(-)0.046
	BertScore	(+)9.91E-08	(+)0.005	(+)1.33E-07	(+)7.43E-08
News	ROUGE-1	(+)0.146	(+)1.202	(+)0.552	(-)0.141
	ROUGE-2	(+)0.530	(-)1.919	(-)0.137	(-)0.343
	ROUGE-L	(+)0.218	(-)0.268	(+)0.431	(+)0.185
	BertScore	(+)0.000	(+)0.059	(+)5.86E-08	(+)1.17E-07
Reddit	ROUGE-1	(-)0.047	(-)2.392	(-)0.160	(+)0.032
	ROUGE-2	(-)0.375	(-)3.429	(-)0.654	(-)0.757
	ROUGE-L	(+)0.035	(-)2.060	(-)0.393	(-)0.004
	BertScore	(-)2.60E-07	(+)0.043	(+)2.60E-07	(-)9.00E-08

Third, successive prompting on the same original articles yields negligible performance change, indicating that the larger drops under relevance paraphrasing are not simply a result of generation variance.

4.9.3. Sensitivity Analyses Strengthen the Main Interpretation. We also re-run the relevance-paraphrasing experiment with temperature set to zero on a 10% sample of the datasets. The magnitude of the effect is reduced in some settings, but the overall pattern remains: performance frequently worsens after paraphrasing salient source content. Table 4.5 and Figure 4.7 summarize these results.

NLI-based evaluation using MENLI [181] likewise shows consistent degradation in summary quality after relevance paraphrasing, supporting the conclusion that the observed effect is not an artifact of ROUGE or BERTScore alone.

TABLE 4.5. Performance change (%) observed after relevance paraphrasing across datasets/LLMs with temperature set to 0.

Datasets	Metrics	Llama-2 _{13B}	GPT-3.5 _{Turbo}	Dolly-v2 _{7B}	Mistral _{7B}
Performance Change (%)					
CNN	ROUGE-1	(-)7.543	(-)5.740	(-)10.45	(-)10.37
	ROUGE-2	(-)20.728	(-)14.74	(-)28.12	(-)28.32
	ROUGE-L	(-)9.842	(-)7.034	(-)13.90	(-)14.33
	BertScore	(-)0.388	(-)0.231	(-)0.353	(-)0.373
XSum	ROUGE-1	(-)5.381	(-)3.448	(-)0.637	(+)1.703
	ROUGE-2	(-)12.405	(-)8.361	(-)4.547	(+)1.414
	ROUGE-L	(-)4.377	(-)3.298	(-)1.420	(+)1.324
	BertScore	(-)0.249	(-)0.125	(-)1.751	(+)0.218
News	ROUGE-1	(+)7.095	(-)2.876	(-)19.37	(-)11.38
	ROUGE-2	(-)3.272	(-)17.79	(-)40.93	(-)34.03
	ROUGE-L	(+)4.312	(-)4.335	(-)22.17	(-)15.59
	BertScore	(+)0.436	(+)0.459	(-)0.331	(-)0.103
Reddit	ROUGE-1	(+)2.174	(-)1.992	(-)23.87	(-)6.461
	ROUGE-2	(-)1.813	(-)9.880	(-)41.31	(-)25.44
	ROUGE-L	(-)0.704	(-)4.075	(-)25.82	(-)6.445
	BertScore	(-)0.020	(-)0.102	(-)22.33	(-)0.053

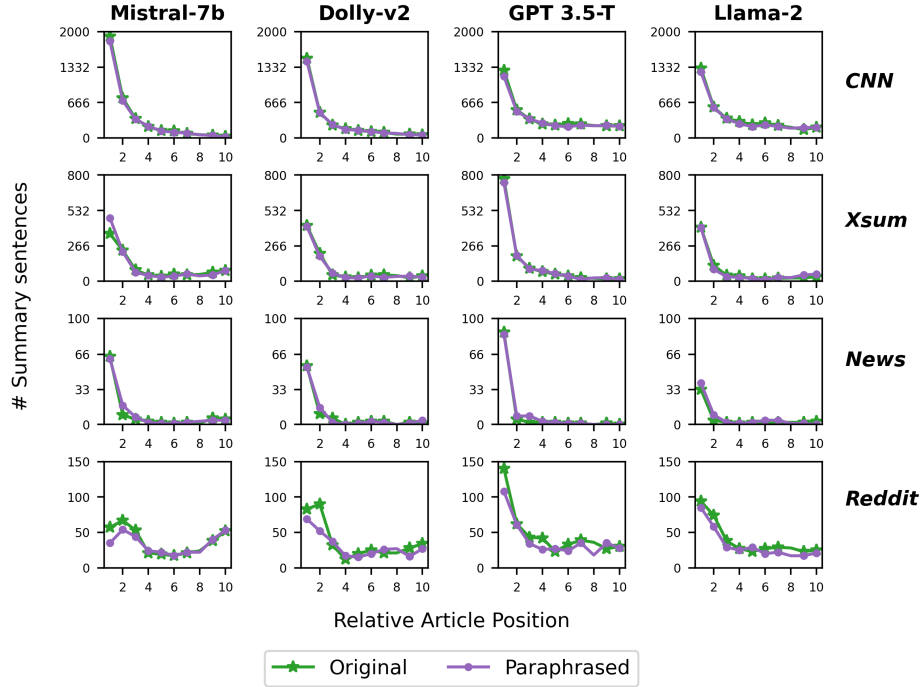


FIGURE 4.7. Paraphrasing results in different summaries for temperature set to 0.

4.9.4. Comparison with Traditional Summarization Models. We also evaluate fine-tuned BART and Pegasus under relevance paraphrasing. Both models show consistent degradation across all datasets and metrics, as summarized in Table 4.6. These results indicate that robustness problems

Dataset	Metric	BART	Pegasus
CNN	Rouge1	(-)13.04	(-)14.1
	Rouge2	(-)33.71	(-)34.61
	RougeL	(-)18.38	(-)19.73
	BertScore	(-)0.676	(-)0.828
Xsum	Rouge1	(-)3.14	(-)2.978
	Rouge2	(-)5.601	(-)4.836
	RougeL	(-)3.53	(-)3.139
	BertScore	(-)0.22	(-)0.218
News	Rouge1	(-)10.71	(-)12.3
	Rouge2	(-)21.9	(-)22.71
	RougeL	(-)11.6	(-)12.64
	BertScore	(-)0.587	(-)0.763
Reddit	Rouge1	(-)5.4	(-)8.81
	Rouge2	(-)15.41	(-)22.43
	RougeL	(-)6.263	(-)10.15
	BertScore	(-)0.137	(-)0.38

TABLE 4.6. Robustness degradation after relevance paraphrasing for BART and Pegasus.

under meaning-preserving perturbations are not unique to LLMs, although the zero-shot LLMs still generally offer stronger overall summarization quality.

4.10. Additional Validating Analyses

4.10.1. Robustness of the Position-Bias Framework to Alternative Design Choices.

The position-bias findings are stable under multiple reasonable implementation choices. Using ROUGE-1 instead of TF-IDF as the proxy mapping function ϕ yields highly similar positional distributions and Wasserstein distances. Likewise, mapping summary sentences to the top- N most similar source sentences rather than only the single best match does not change the main qualitative trends. These analyses suggest that the framework is not overly sensitive to one specific operationalization of source-sentence mapping.

4.10.2. ROUGE and Position Bias Are Not Redundant. We also measure the relationship between ROUGE and position bias more directly. The correlation between Wasserstein distance and ROUGE is highly dataset dependent: it is strongly positive on *XSum*, weak on *CNN/DM*, and strongly negative on *News* and *Reddit*. This confirms that overlap-based metrics do not subsume

positional robustness and that independent source-allocation measurement is needed for a more complete evaluation.

4.10.3. Prompting and Architecture Variants. To further probe the generality of the position-bias framework, we examine prompt and architecture variants. Role-playing prompts modestly reduce position bias on a subset of *XSum*, but do not eliminate the strong lead-oriented tendency. Flan-T5, a more generalized encoder-decoder model, exhibits substantially lower position bias than BART and Pegasus on several datasets, including *XSum*. This suggests that positional behavior is shaped not only by model architecture, but also by training regime and instruction tuning.

4.11. Limitations

This chapter has several limitations. First, both studies rely on proxy mappings from summary sentences back to source sentences. Although sensitivity analyses suggest that the results are robust to reasonable choices of these mappings, they remain approximations. Second, relevance paraphrasing depends on the quality of the paraphrasing model. While the chosen paraphraser preserves semantic content well in both automatic and qualitative inspection, no paraphrasing procedure is perfectly meaning invariant. Third, the analyses focus on zero-shot summarization and therefore do not exhaust the broader space of summarization robustness under fine-tuning, retrieval augmentation, or multi-document settings.

4.12. Conclusion

This chapter introduced two complementary diagnostics for summarization robustness. The first, position bias, measures whether a model’s source allocation is aligned with that of human summaries. The second, relevance paraphrasing, measures whether summary generation remains stable when salient source content is re-expressed in a meaning-preserving way. Together, the findings show that even strong zero-shot summarizers can exhibit meaningful robustness failures that are not visible from aggregate summary metrics alone. Robust summarization evaluation should therefore consider not only final summary quality, but also how a model uses the source document and how stable that behavior remains under benign variation.

Reliability: Avoidance Learning of Failure Modes for Robust Tool Planning

5.1. Introduction

The previous chapters studied large language models as predictors: systems that map an input to an answer, a label, or a summary. A rapidly growing class of deployments instead treats the model as an *agent* that accomplishes a goal by orchestrating external tools, such as search engines, calculators, databases, and application programming interfaces. In this agentic setting the model does not produce a single answer; it decomposes a request into sub-goals, selects an appropriate tool for each sub-goal, binds the arguments of every call with valid types, routes the output of one call into the input of the next, and sequences the whole procedure so that every dependency is satisfied before it is used. The artifact the model produces is therefore a *tool plan*: a structured, multi-step program over an external toolset. As LLMs are increasingly deployed as autonomous agents, the reliability of these plans, rather than the fluency of any individual response, becomes the property that determines whether the system is usable in practice.

This shift from prediction to planning introduces a distinct failure surface. In multi-step tool planning, an LLM must simultaneously select correct tools, bind arguments with valid types, route intermediate outputs as inputs to downstream calls, and order operations correctly. A single local mistake, whether a near-miss tool, a mis-typed argument, or a mis-threaded reference, silently propagates through dependent steps and breaks the entire trajectory. Even frontier models exhibit this brittleness on multi-step tool-calling datasets: GPT-4o reaches only 49% accuracy on ToolHop [182] and 28% full-sequence match on NESTFUL [183]. In multi-step plans no constraint is independent; a single error cascades forward, producing trajectories that appear syntactically well-formed but are deductively broken.

This chapter approaches that problem through the dissertation’s data-model synergy lens, but in a constructive mode. The adaptation chapters used influence functions to measure how existing training data shapes a model’s layers and its choice of demonstrations; the robustness chapter used distributional and perturbation diagnostics to reveal where a model’s behavior is brittle. Tool planning lets us join those two stances. We first characterize how planners fail, in the same spirit that the robustness chapter characterizes summarization failures, and we then turn that characterization into synthesized training data that reshapes the model, in the same spirit that the adaptation chapters reshape it. Concretely, we encode planner failures in a nine-error taxonomy, generate contrastive negative plans that instantiate each error, and use them to train a judge and, through that judge, a planner that learns to avoid the failures directly. The instrument here is not influence estimation but data synthesis grounded in a structural model of failure; the unifying principle, that downstream behavior can be improved by acting deliberately on the relationship between data and model, is the same.

The community has responded to planner brittleness along two axes. The first is *test-time search*: ReAct [184], Reflexion [185], LATS [186], and ToolTree [187] explore reasoning trajectories more carefully through reactive loops or tree search. The second is *structural scaffolding*: GTool [188], GNN4Plan [189], and ToolPlanner [190] inject dependency-graph priors at inference time. RL-based methods that fine-tune tool policies on coarse execution outcomes [191, 192, 193] provide a third alternative. All share an assumption that improved inference-time guidance or scalar success signals suffice; none directly trains models to recognize and avoid the specific structural failure modes that derail multi-step plans. Recent diagnostic work has begun characterizing these failure modes systematically: CriticTool [194] defines a five-type error taxonomy, while ToolScan [195] and FAIL-TALMS [196] catalog recurring failure patterns. These taxonomies, however, remain post-hoc diagnostic instruments rather than training targets, leaving a clear opportunity: train planners and judges to be error-aware by construction.

We introduce FORTE (Figure 5.1), a framework that trains tool planners through failure avoidance. FORTE starts from a 9-error taxonomy for multi-step plans (Table 5.1) and uses it to drive a contrastive data augmentation pipeline: for each gold trajectory we generate nine minimally edited negative plans, each instantiating one taxonomy error while preserving task context. This produces a dense,

FORTE *Failure-Oriented Robust Tool-planning with Error-aware training*

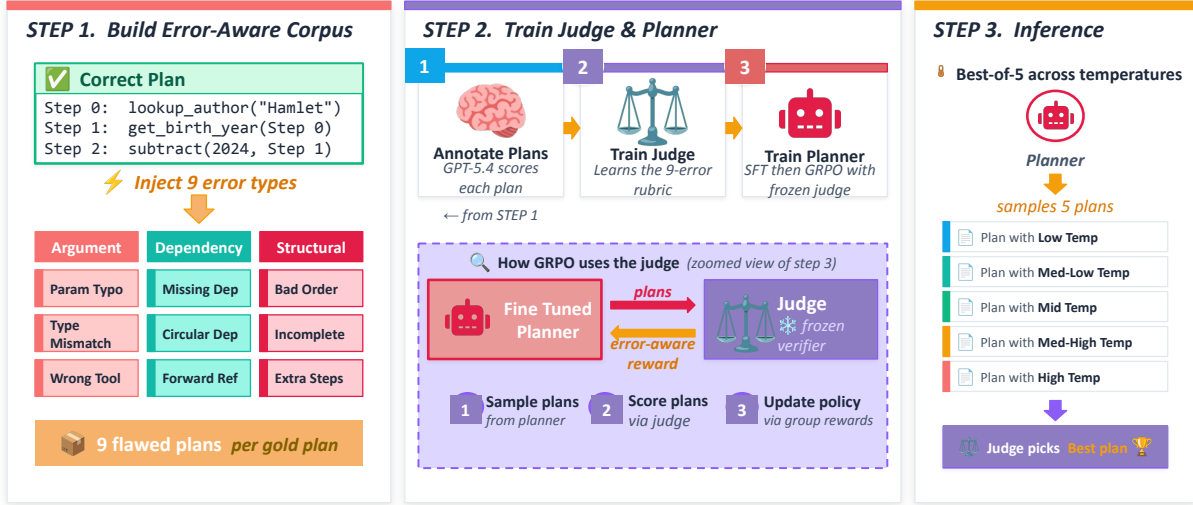


FIGURE 5.1. FORTE trains tool-planning models in three steps. We first build an error-aware corpus by injecting nine taxonomy-grounded error types into correct plans. We then annotate plans with GPT-5.4, fine-tune a judge model on the nine-error rubric, and train the planner with GRPO using the frozen judge as a dense reward signal. At inference, we sample five candidates with Best-of- N selection.

taxonomy-grounded supervision signal at scale without manual annotation. A lightweight judge is fine-tuned on a deductive rubric over these plan pairs, learning to score plan structure and localize specific error types. The planner is then trained with GRPO using this frozen judge as a structured reward signal, and at test time the same judge selects from a temperature-diverse Best-of- N candidate pool [197, 198].

FORTE differs from prior judge-planner [190, 191] and RL methods in three ways. First, the judge is grounded in a human-interpretable error taxonomy rather than in scalar end-task scores, making its judgments more precise and more interpretable. Second, contrastive negative plans densely cover the error space, rather than relying on sparse environment failures or oracle trajectory pairs. Third, the architecture is decoupled in a specific sense that prior methods are not: prior training-based approaches conflate plan-level correctness with tool execution outcomes, whereas FORTE’s judge scores plan structure alone, so the planner is reusable across benchmarks and requires no ensemble components, graph-search modules, or multi-agent decompositions at inference time.

Across ToolHop and NESTFUL, FORTE achieves the highest Judge Success Rate (JSR) on all six backbone and dataset configurations using only 3B to 7B open-source models, beating established baselines (ReAct, α -UMi, GNN4TaskPlan, ToolPlanner) by 2.5 to 13.2 JSR points and exceeding frontier closed-source models: even our smallest backbone outperforms GPT-5.5 by 33.9 ToolHop JSR points. Ablations confirm that the contrastive negatives are the critical ingredient; removing them drops JSR by 14.3 points and collapses Best-of- N selection below the single-sample baseline. Our main contributions are:

- A 9-error taxonomy for multi-step tool plans organized into argument-level, multi-step dependency, and plan-structure failure modes, with three categories that are absent from prior diagnostic frameworks (Table 5.1).
- Two augmented contrastive datasets (ToolHop and NESTFUL) constructed via a scalable annotation pipeline that, for each gold trajectory, instantiates all nine taxonomy errors as minimally edited negatives and annotates the resulting plan pairs against a deductive rubric, supporting the training of error-aware planners and judges.
- A decoupled judge-planner architecture in which a lightweight judge supplies dense GRPO reward and also serves as a verifier for temperature-diversity Best-of- N inference, without requiring tool execution at any stage of training or selection.
- Comprehensive experiments showing FORTE achieves the highest JSR across all six backbone and dataset configurations, with FORTE-Qwen2.5-7B reaching 96.6 / 98.6 JSR on ToolHop / NESTFUL and outperforming GPT-5.5 by 33.9 ToolHop JSR points using only open-source 3B to 7B backbones.

5.2. Related Work

Multi-step tool planning and benchmarks. Tool-augmented LLMs have advanced from single-step function calling [199] to multi-step orchestration at scale [200]. Two recent benchmarks specifically stress multi-step structure: ToolHop [182] requires multi-hop chaining across 995 queries and 3,912 tools (GPT-4o: 49% accuracy), and NESTFUL [183] targets nested API sequences where one call’s output seeds the next (full-sequence match: 28%). These results establish that multi-step

TABLE 5.1. FORTE’s 9-error taxonomy versus prior error-characterization and training frameworks [194, 195, 196, 191, 190, 193]. ✓/✗ indicate whether the error type is explicitly characterized or the methodology property is present. Rows are grouped into argument-level errors (rows 1 to 3), multi-step dependency errors (rows 4 to 6), and multi-step plan structure errors (rows 7 to 9); all three dependency and ordering error types are absent from prior frameworks.

Error Type	<i>Diagnostic Benchmarks</i>			<i>Training Methods</i>			
	Critic Tool	Tool Scan	FAIL-TALMS	Step Tool	Tool Planner	Ava TaR	FORTE (Ours)
Type Mismatch	✗	✓	✗	✗	✗	✗	✓
Parameter Typo	✓	✓	✗	✗	✗	✗	✓
Wrong Tool	✓	✓	✗	✗	✗	✗	✓
Missing Dependency	✗	✗	✗	✗	✗	✗	✓
Circular Dependency	✗	✗	✗	✗	✗	✗	✓
Forward Reference	✗	✗	✗	✗	✗	✗	✓
Inefficient Order	✗	✗	✗	✗	✗	✗	✓
Incomplete Plan	✗	✓	✓	✗	✗	✗	✓
Unnecessary Steps	✗	✓	✗	✗	✗	✗	✓
Errors Covered	2/9	5/9	1/9	0/9	0/9	0/9	9/9
Taxonomy Defined	✓	✓	✗	✗	✗	✗	✓
Used as Training Signal	✗	✗	✗	✓	✓	✓	✓
Contrastive Training Data	✗	✗	✗	✗	✗	✓	✓
Judge Trained on Taxonomy	✗	✗	✗	✗	✗	✗	✓

dependency failures are qualitatively different from single-call errors and persist even in frontier models, motivating methods that act on multi-step structure at training time.

Test-time search and structural scaffolding. Test-time methods improve robustness either by searching harder or by imposing structure. Search-based approaches range from the canonical Thought, Action, Observation loop of ReAct [184] to the recent ToolTree [187], which combines pre- and post-execution LLM evaluation inside MCTS with bidirectional pruning. Structural scaffolding methods instead inject dependency priors via GNNs or graph navigation [189]. We compare directly against representatives of both families (ReAct and GNN4TaskPlan) in our experiments. All of these methods constrain or guide the planner at inference time but leave its underlying error tendencies unchanged, treating brittleness as an inference problem rather than a training deficiency.

RL and contrastive training for tool use. Several works apply RL or contrastive optimization to tool-use policies. StepTool [191] introduces step-grained reward shaping that assigns credit to each tool call based on invocation success and task contribution. ToolPlanner [190] constructs pairwise SFT and RL preference data from task-completion and instruction-following scores. On

the contrastive side, AvaTaR [193] learns a comparator module that performs contrastive reasoning over positive and negative trajectories to generate holistic steering prompts. The supervision in all of these frameworks reduces to coarse execution outcomes, such as binary success, task recall, or completion scores, rather than structured error labels. None trains against a taxonomized error space, so the agent receives no signal distinguishing a dependency-threading failure from a parameter-key error. FORTE addresses this directly, training its judge on a deductive rubric over a 9-error taxonomy and making the contrastive signal causally grounded in specific failure types.

Failure taxonomies and diagnostic benchmarks. Substantial recent work characterizes tool-use failures empirically. CriticTool [194] is the closest prior work, defining a five-type taxonomy covering tool selection, tool hallucination, parameter-key, parameter-value, and environment errors, and introducing the first benchmark for evaluating LLM self-critique in tool-calling; it finds that GPT-4o achieves only 22% error recovery on NESTFUL. ToolScan and FAIL-TALMS each catalog seven or more recurring error patterns and provide diagnostic protocols [195, 196]. The consistent pattern across this line of work is that error taxonomies serve as diagnostic instruments rather than training targets. FORTE closes this loop: the taxonomy is designed to be labeled over individual plan steps, the contrastive pipeline populates that label space at scale, and the judge is trained to reason within it, transforming error characterization from a post-hoc diagnostic into the primary driver of training.

5.3. Method

Here we present the methodology of FORTE.

5.3.1. Problem Formulation. Let q denote a natural-language query and $\mathcal{T} = \{t_1, \dots, t_K\}$ a set of available tools, each described by its name, parameter schema, and return type. A *tool plan* is a sequence $\pi = \langle s_0, \dots, s_L \rangle$ where each $s_i = (v_i, t_i, \mathbf{p}_i)$ specifies an output variable, a tool invocation, and parameter bindings that may reference earlier outputs via $\{\{j\}\}$ syntax; π^* denotes the gold reference plan. The planner f_θ maps $(q, \mathcal{T})$ to a distribution over plans, and the judge g_ϕ maps $(q, \mathcal{T}, \pi)$ to a quality score $\rho \in [0, 100]$ and a predicted error label e drawn from the 9-type taxonomy defined below.

5.3.2. Error Taxonomy. We define nine concrete failure modes in three categories.

Argument-level errors occur within individual tool invocations. *Parameter Typo* distorts one parameter value while preserving syntactic validity. *Type Mismatch* replaces a parameter with one of an incompatible type, for example an integer where a string is required. *Wrong Tool* substitutes the correct tool with a semantically adjacent but functionally incompatible alternative.

Multi-step dependency errors arise from incorrect data flow. *Missing Dependency* replaces an inter-step reference $\{\{j\}\}$ with a hardcoded literal. *Circular Dependency* introduces a reference cycle. *Forward Reference* lets an earlier step reference a step not yet executed.

Multi-step plan structure errors affect global plan organization. *Inefficient Order* permutes steps so the logical sequence is broken while individual invocations remain correct. *Incomplete Plan* truncates the plan before the final sub-question is resolved. *Unnecessary Steps* inserts a redundant tool call.

These nine types cover the recurring patterns identified in prior diagnostic work [194, 195, 196], extending coverage to the three dependency and ordering errors unique to multi-step plans (Table 5.1).

5.3.3. Contrastive Data Augmentation. Given a gold plan π^* for query q , we construct nine taxonomy-grounded negative plans $\{\pi_e^-\}_{e \in \mathcal{E}}$, one per error type, via deterministic, minimally invasive edits to π^* . Each edit targets a single step and preserves all other steps verbatim, so the negative differs from the gold in exactly the way prescribed by e .

Argument-level negatives apply a character-level perturbation (typo), substitute a type-incompatible constant from the schema (type mismatch), or swap the tool name for the most-similar alternative in $\mathcal{T}$ (wrong tool). Dependency negatives replace $\{\{j\}\}$ with a hardcoded string (missing dependency), introduce a back-edge that closes a cycle (circular dependency), or advance a reference past the current step (forward reference). Structural negatives swap two steps with a data-flow dependency (inefficient order), drop the final step (incomplete plan), or duplicate an intermediate step (unnecessary steps).

This yields plan pairs $(q, \mathcal{T}, \pi^*, \pi_e^-, e)$ without manual annotation. Applied independently to the training splits of ToolHop and NESTFUL, the pipeline produces two dataset-specific augmented corpora that systematically cover the full error space.

5.3.4. Judge Training. We fine-tune a judge model g_ϕ to score plans on a deductive rubric aligned with the taxonomy. Given the query, tool schemas, and proposed plan, the judge outputs a JSON object containing a quality score $\rho \in [0, 100]$, a categorical success prediction, a natural-language reasoning trace, and a list of detected issues with their predicted error types (inference prompt in Section 5.3.7).

Reference annotations. For each pair $(q, \mathcal{T}, \pi^*, \pi_e^-, e)$ we query GPT-5.4 with the inference-time rubric prompt (Section 5.3.7). Starting from a base score of 100, the rubric deducts on a four-tier severity schedule: critical (-30) for circular dependencies, forward references, missing required parameters, and non-existent tools; high (-20) for hardcoded values where $\{\{j\}\}$ references are expected; medium (-10) for ordering and redundancy issues; and low (-5) for stylistic issues. The injected error type e is *not* revealed to GPT-5.4: the annotator identifies issues independently and applies the rubric, so each negative’s score reflects errors GPT-5.4 actually detects rather than a deterministic mapping from the injection label. Empirically, gold plans score 95 to 100 and negatives score 70 to 95.

Syntactic-only scoring. The rubric scores syntactic plan correctness rather than semantic task success, and the judge is instructed never to type-check parameter values containing $\{\{j\}\}$, since these are dependency references whose runtime type is determined upstream. This decoupling enables training the judge on plan structure alone, with no tool execution environment required.

The judge is LoRA fine-tuned [68] from Qwen2.5-7B-Instruct on the reference annotations and used identically across all planner experiments. At inference it is served via batched vLLM as both the GRPO reward model and the Best-of- N scorer.

5.3.5. Planner Training. Supervised fine-tuning. We initialize the planner from a pre-trained instruction-tuned base model and perform SFT on the gold plans of the relevant corpus with the prompt template in Section 5.3.7. The objective is token-level cross-entropy on response tokens only. The SFT checkpoint serves as both the RL initialization and the KL reference policy.

RL with GRPO. We use Group Relative Policy Optimization [201], which removes the value network and computes advantages via group-normalized rewards: for each query we sample G candidate plans, score each with the frozen judge, and normalize the group’s rewards by their mean

and standard deviation. The judge weights are frozen throughout RL, so the planner’s reward signal is determined entirely by plan structure.

5.3.6. Inference: Temperature-Diversity Best-of- N . At test time we generate $N = 5$ candidate plans per query and return the highest-scoring candidate. To maximize candidate diversity without sacrificing per-sample quality, we draw exactly one candidate at each temperature in $\tau \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$. Each candidate is scored by the judge with the same rubric used during RL training (Section 5.3.7), and we return $\hat{\pi} = \arg \max_{i \in [N]} g_{\phi}(q, \mathcal{T}, \pi_i)$.

5.3.7. Prompts and Scoring Rubric. This subsection collects the three prompts used by FORTE verbatim from the released codebase. The *annotation prompt* is used by GPT-5.4 to generate training labels for the judge; the *judge inference prompt* is sent to the trained judge during GRPO reward computation and Best-of- N selection; and the *planner prompt* is used identically for SFT, GRPO, and Best-of- N inference.

GPT-5.4 is prompted with the deductive rubric below, which assigns a base score of 100 and subtracts points per identified syntactic error according to a four-tier severity scale (critical -30 , high -20 , medium -10 , low -5). The rubric explicitly forbids the judge from flagging semantic issues or type-checking parameters containing dependency references $\{\{N\}\}$.

System message : annotation judge

You are an expert at evaluating multi-step tool execution plans.
You only check syntactic correctness, not semantic correctness.
NEVER type-check parameters containing $\{\{N\}\}$ - they are dependency references and are always type-valid.

User message : annotation judge

You are evaluating a multi-step tool execution plan using an
OBJECTIVE ERROR-BASED RUBRIC.

****USER QUERY:**** {query}
****AVAILABLE TOOLS:**** {tools_desc}
****PROPOSED PLAN:**** {plan_desc}
****GROUND TRUTH PLAN (for reference):**** {ground_truth_desc}

=====

OBJECTIVE SCORING RUBRIC

=====

```

SCORING FORMULA:
1. Start with a base score of 100.
2. Identify ALL errors in the plan.
3. For EACH error, deduct points based on severity:
   - CRITICAL error: -30 points
   - HIGH error:      -20 points
   - MEDIUM error:   -10 points
   - LOW error:       -5 points
4. Final score = max(0, 100 - total_deductions)

=====
SYNTAX-ONLY EVALUATION
=====
Check ONLY syntactic correctness:
- Are all required parameters provided?
- Are LITERAL parameter types correct (skip values containing {{N
}})?
- Are dependencies valid (no forward refs, no circular deps)?
- Are all steps present to complete the task?
- Is the tool name spelled correctly?

Do NOT check semantics:
- Whether tool outputs will work together
- Whether the plan produces the correct final answer
- Whether tool choice "makes sense" for the task
- Whether parameter VALUES produce the right result

ABSOLUTE RULE: any parameter value containing {{N}} is a dependency
reference and is ALWAYS type-valid. Skip type checking entirely.
Quotes around {{N}} do NOT make it a literal string.

=====
ERROR SEVERITY DEFINITIONS
=====
CRITICAL (-30): missing required param; circular dependency; non-
existent
    tool; forward reference; literal-value type mismatch; missing
critical step.
HIGH (-20):      hardcoded value where {{N}} expected; incorrect
parameter
    value changing output significantly; missing non-critical step.
MEDIUM (-10):   inefficient ordering (works but suboptimal); typo
in
    parameter value; unnecessary step that doesn't break the plan.
LOW (-5):       minor formatting; non-standard but functional
format;
    trivial inefficiency; style issues.

SUCCESS PREDICTION MAPPING:
90-100 -> "yes"          75-89 -> "likely_yes"
50-74  -> "uncertain"    25-49 -> "likely_no"    0-24 -> "no"

=====
YOUR TASK

```

```

=====
1. IDENTIFY ALL ERRORS step by step.
2. CLASSIFY SEVERITY using the rubric above.
3. CALCULATE SCORE: start at 100, deduct per error.
4. SHOW YOUR WORK in the "reasoning" field.

RESPONSE FORMAT (JSON only, no markdown):
{
  "quality_score": <0-100>,
  "success_prediction": "<yes|likely_yes|uncertain|likely_no|no>",
  "confidence": <0.0-1.0>,
  "reasoning": "Base score: 100\nError 1: [severity] description (-
X)\n...\nFinal score: Z",
  "issues": [
    {
      "type": "type_mismatch|missing_dependency|wrong_tool|
parameter_typos|
      circular_dependency|inefficient_order|
incomplete_plan|
      unnecessary_steps|forward_reference|other",
      "severity": "critical|high|medium|low",
      "step": <step_number or null>,
      "description": "Specific description of the error",
      "suggestion": "How to fix this error",
      "points_deducted": <30|20|10|5>
    }
  ]
}

```

At RL training time, the trained judge receives the prompt below to produce a reward signal for GRPO; at inference time, the same prompt is used to score each of the N candidate plans for Best-of- N selection. The system message and JSON output schema match the annotation prompt above so the judge can be trained end-to-end on the GPT-5.4 annotations.

System message : inference judge

You are an expert judge for evaluating tool execution plans. Your task is to:

1. Analyze the plan's correctness and efficiency
2. Assign a quality score (0-100)
3. Predict success likelihood (yes/likely_yes/uncertain/likely_no/no)
4. Identify specific issues with severity levels
5. Provide detailed reasoning

Scoring guidelines:

- 100: Perfect execution, no errors
- 80-99: Minor issues, likely to succeed
- 60-79: Moderate issues, uncertain outcome

- 40-59: Major issues, likely to fail
- 0-39: Critical errors, will fail

User message : inference judge

Query: {query}

{tools_desc}

Plan to Evaluate:
{plan_str}

Please evaluate this plan and provide:

1. Quality score (0-100)
2. Success prediction (yes/likely_yes/uncertain/likely_no/no)
3. Detailed reasoning
4. List of issues (if any)
5. Confidence (0.0-1.0)

Format your response as JSON:

```
{
  "quality_score":    <int>,
  "success_prediction": "<string>",
  "reasoning":        "<string>",
  "issues":           [...],
  "confidence":       <float>
}
```

The same planner prompt template is used for SFT, GRPO, and inference, ensuring that no distribution shift is introduced between training and deployment. The user message instructs the planner to use `{{N}}` variables for dependency threading and the exact parameter names from the tool signatures.

System message : planner

You are an expert at creating multi-step tool execution plans.
Given a query and available tools, generate a correct sequence of
tool calls to answer the query.

User message : planner

Generate a tool execution plan to answer this query.

Query: {question}

```

{tools_desc}

Generate a step-by-step plan using the available tools. Each step
should:
1. Call exactly one tool
2. Use output variables {{0}}, {{1}}, {{2}}, etc. for results
3. Reference previous step outputs using {{N}} - never substitute a
   hardcoded
   value for an output that comes from a prior step
4. Use the exact parameter names shown in the tool signatures above
5. Provide all required parameters

Generate only the steps the query requires - no redundant steps, no
missing steps.

Format each step as: Step N: {{N}} = tool_name(param1=value1,
param2=value2, ...)

```

5.4. Experiments

Here we expand upon our experimental setup to conduct the experiments and analyze the results.

5.4.1. Datasets and Evaluation. We evaluate FORTE on two complementary multi-step tool-planning benchmarks.

ToolHop [182] contains 995 multi-hop queries each requiring a 2-to-6-step tool plan over a per-query tool set of 3-to-8 tools. The benchmark stresses cross-step reasoning where each query is explicitly decomposed into sub-questions whose answers feed downstream tool calls. We hold out 100 queries for evaluation and apply the augmentation pipeline of Section 5.3.3 independently to each split: for every query we instantiate one error-free reference plan and nine taxonomy-grounded negatives, one per error type, yielding 1000 annotated evaluation plans that cover the full error space.

NESTFUL [183] contains 1861 multi-step instances designed to evaluate nested function calling, where outputs of one tool call are used as arguments of subsequent calls via labeled references such as `$var_1.result$`. The benchmark spans two domains: a mathematical-reasoning subset (1,415 examples drawn from MathQA covering 40 tools with an average of 5.1 tool calls per sample) and a coding subset (446 examples synthesized from StarCoder2-Instruct covering 881 distinct tools with an average of 2.1 tool calls per sample). We apply a similar augmentation pipeline as ToolHop to create our training and test sets.

Metrics. We report four metrics, all expressed as percentages averaged over the test set. *Judge Success Rate* (JSR) is the fraction of generated plans for which the judge assigns a score of at least 80, the threshold calibrated to the rubric’s “likely to succeed” band. The remaining three are deterministic structural metrics computed against gold plans. *Functional Match* (FM) requires three conditions to hold jointly on a single example: step-count agreement with the gold plan, at least 50% Jaccard keyword overlap on tool names, and at least 50% parameter accuracy. *Mean Parameter Accuracy* (PA) is the fraction of ground-truth parameters reproduced correctly. *Mean Dependency Accuracy* (DA) is the fraction of $\{\{N\}\}$ output-variable references correctly threaded through the plan.

5.4.2. Implementation Details. Models. We use three planner backbones: Qwen2.5-3B-Instruct, Qwen2.5-7B-Instruct, and Llama3.2-3B-Instruct. The judge is always Qwen2.5-7B-Instruct fine-tuned on GPT-5.4 reference annotations, regardless of the planner.

Training. SFT runs for 4 epochs with AdamW (peak learning rate 5×10^{-5} , cosine schedule). GRPO uses group size $G = 5$, learning rate 1×10^{-6} , PPO clip $\varepsilon = 0.2$, and KL coefficient $\beta = 0.01$. We train for 4 epochs and report the checkpoint that maximizes joint performance on judge success and structural metrics on a held-out validation split. All training runs use $8 \times$ NVIDIA RTX 6000 Ada GPUs with bfloat16 mixed precision.

Inference. Best-of- N uses $N = 5$ with temperatures $\tau \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$ as described in Section 5.3.6. Top- p is fixed at 0.9 for all candidates. The judge runs as a vLLM server with batch size 64 and is queried in parallel for the five candidates per query.

5.4.3. Baselines. We compare FORTE against four established multi-step tool-planning baselines. All baselines use our SFT-trained planner as their underlying generation model unless stated otherwise. **ReAct** [184] runs a 3-shot Thought-Action loop with a single generation per query. **α -UMi** [202] uses the multi-model Planner-Caller decomposition, with both roles filled by separate instances of the non-fine-tuned instruction-tuned backbone. **GNN4TaskPlan** [189] is a three-stage pipeline (sub-action decomposition $\rightarrow$ GraphSAGE retriever $\rightarrow$ parameter filling); we train the retriever on the training-set co-occurrence graph with BPR loss, two hard negatives per positive,

TABLE 5.2. Main results on ToolHop and NESTFUL. **Best** (green shading) and **second best** are marked within each backbone group. JSR: Judge Success Rate (%), score ≥ 80). FM: Functional Match (%). PA: Mean Parameter Accuracy (%). DA: Mean Dependency Accuracy (%).

Backbone LLM	Method	ToolHop				NESTFUL			
		JSR	FM	PA	DA	JSR	FM	PA	DA
Llama3.2-3B	ReAct	75.5	74.4	<u>69.8</u>	88.2	88.9	30.8	53.2	83.8
	α -UMi	0.0	0.0	20.0	11.5	17.3	0.0	26.1	9.2
	GNN4TaskPlan	14.0	12.0	25.1	51.8	12.6	2.8	29.4	58.6
	ToolPlanner	<u>83.2</u>	91.7	74.4	<u>96.5</u>	<u>95.0</u>	<u>31.4</u>	<u>59.1</u>	85.1
	FORTE (ours)	92.9	<u>80.2</u>	63.6	98.9	98.9	37.2	60.2	<u>84.0</u>
Qwen2.5-3B	ReAct	<u>92.7</u>	<u>84.7</u>	70.0	<u>93.1</u>	83.3	27.5	45.6	67.7
	α -UMi	18.0	5.0	26.3	52.6	46.7	3.9	30.3	49.5
	GNN4TaskPlan	2.0	2.0	8.9	13.9	4.7	1.7	4.7	20.4
	ToolPlanner	60.1	82.7	66.7	85.4	<u>84.5</u>	<u>29.1</u>	<u>52.3</u>	79.7
	FORTE (ours)	96.6	89.2	<u>69.5</u>	98.8	97.7	33.6	56.2	87.8
Qwen2.5-7B	ReAct	74.9	80.0	<u>74.6</u>	<u>91.6</u>	89.4	32.6	52.6	80.6
	α -UMi	29.0	21.3	29.8	49.6	67.3	15.7	42.1	51.1
	GNN4TaskPlan	23.0	25.0	30.9	59.9	24.4	5.3	28.7	57.2
	ToolPlanner	87.7	<u>85.4</u>	67.6	86.2	<u>96.1</u>	<u>41.6</u>	<u>59.0</u>	85.9
	FORTE (ours)	96.6	97.3	77.1	98.9	98.6	44.3	61.6	89.0

for 20 epochs. **Tool-Planner** [190] performs reward-guided beam search with a width of 3, using task-completion and instruction-following scores. This protocol grants the strongest baselines (ReAct, Tool-Planner, GNN4TaskPlan) the same SFT initialization as FORTE, so any remaining gap isolates the contribution of contrastive judge training and GRPO from that of supervised fine-tuning alone. We also compare against closed-source models GPT-5.5 and GPT-5-Nano.

5.4.4. Main Results. FORTE achieves the highest judge success in every setting.

Table 5.2 shows that FORTE obtains the best JSR across all six backbone-dataset configurations. On ToolHop, FORTE improves over the next-best method by 9.7, 3.9, and 8.9 points for Llama3.2-3B, Qwen2.5-3B, and Qwen2.5-7B, respectively. On NESTFUL, the corresponding gains are 3.9, 13.2, and 2.5 points. This consistent improvement suggests that failure-avoidance training improves overall plan quality across both benchmarks and model families. Qualitative examples of generated plans are presented in Section 5.6.

FORTE is strongest on the harder NESTFUL benchmark. NESTFUL is more structurally challenging, with substantially lower FM scores than ToolHop across methods. Despite this, FORTE achieves the best NESTFUL FM for all three backbones, reaching 37.2, 33.6, and 44.3. It also obtains the best PA for all three backbones and the best DA for both Qwen models. These results indicate that FORTE is especially effective when tool planning requires stronger structural generalization.

FORTE nearly saturates dependency accuracy on ToolHop. On ToolHop, FORTE achieves the best DA for every backbone, with scores of 98.9, 98.8, and 98.9. This shows that the generated plans correctly preserve dependencies across multi-step tool calls. Since dependency errors can make otherwise plausible plans non-executable, these gains are important for reliable tool-use planning.

Prior planning baselines are less robust. ToolPlanner and ReAct are competitive in isolated cases, but their performance varies across backbones and datasets. For example, ToolPlanner achieves strong FM on Llama3.2-3B ToolHop, but FORTE dominates JSR across all settings and wins most structural metrics on NESTFUL. Meanwhile, α -UMi and GNN4TaskPlan perform poorly in most configurations, suggesting that methods relying on online execution feedback do not transfer well to offline plan generation.

FORTE benefits from stronger backbone models. The strongest structural results are obtained with Qwen2.5-7B, where FORTE wins all four metrics within its backbone group on both ToolHop and NESTFUL. On ToolHop, it reaches 96.6 JSR, 97.3 FM, 77.1 PA, and 98.9 DA; on NESTFUL, it reaches 98.6 JSR, 44.3 FM, 61.6 PA, and 89.0 DA. This indicates that FORTE scales positively with backbone capacity rather than saturating at the 3B scale.

Closed models approach FORTE on NESTFUL but miss the strict format bar. GPT-5.5 reaches NESTFUL FM=43.0, within 1.3 points of FORTE-Qwen2.5-7B (44.3) and ahead of our smaller backbones as seen in Table 5.3. JSR however remains 9.6 points behind: GPT-5.5 produces semantically reasonable plans but consistently misses format, hallucinating parameter values absent from the tool signature and occasionally compressing multi-step plans (Section 5.6). Training on a curated failure taxonomy beats raw capacity where structural compliance matters most.

TABLE 5.3. Closed-source frontier reference. Models evaluated on 100 examples per benchmark with identical prompts, judge, and eval as our trained planners; greedy single-shot. **Best** (green shading) per column.

Method	ToolHop				NESTFUL			
	JSR	FM	PA	DA	JSR	FM	PA	DA
GPT-5-Nano	17.0	44.0	50.3	78.0	56.0	17.0	36.6	59.1
GPT-5.5	59.0	50.0	56.9	82.5	89.0	43.0	55.0	73.0
FORTE-Llama3B	92.9	80.2	63.6	98.9	98.9	37.2	60.2	84.0
FORTE-Qwen3B	96.6	89.2	69.5	98.8	97.7	33.6	56.2	87.8
FORTE-Qwen7B	96.6	97.3	77.1	98.9	98.6	44.3	61.6	89.0

5.5. Discussion

In this section we discuss the components of the FORTE pipeline and elaborate on key design choices and ablation studies conducted as part of our experiments.

Contrastive negatives are critical for judge training. Ablating the contrastive component of FORTE’s judge by retraining on only perfect plans (`quality_score=100`) severely degrades the full system on Qwen2.5-3B / ToolHop: JSR drops 14.3 points, FM drops 20.9, and PA drops 12.8 (Figure 5.2). The BoN gain inverts to -2.6 points, meaning the ablated judge’s top-1 selection is on average 2.6 quality points worse than simply taking the lowest-temperature sample, and the judge labels 100% of its picks as successful. Without exposure to flawed plans, the judge loses discriminative ability and provides neither useful reward during GRPO nor useful selection at inference. Damage is concentrated at the argument level, while DA is essentially unchanged (-0.3 points) since SFT already establishes that capability. The contrastive signal is therefore load-bearing rather than incidental: the same negatives that supply the judge’s reward also supply the discriminative ability required at inference.

Best-of- N saturates quickly, justifying $N = 5$. Averaged across the three backbones on ToolHop, JSR rises from 93.15% at $N = 1$ to 95.36% at $N = 5$ and 95.87% at $N = 10$. The bulk of the achievable gain (+2.21 points) is captured by $N = 5$, with five further candidates contributing only +0.51 points. $N = 5$ thus realises roughly 80% of the $N = 10$ improvement at half the inference cost, keeping FORTE’s per-query budget comparable to non-search baselines without sacrificing the candidate diversity the judge needs to discriminate.

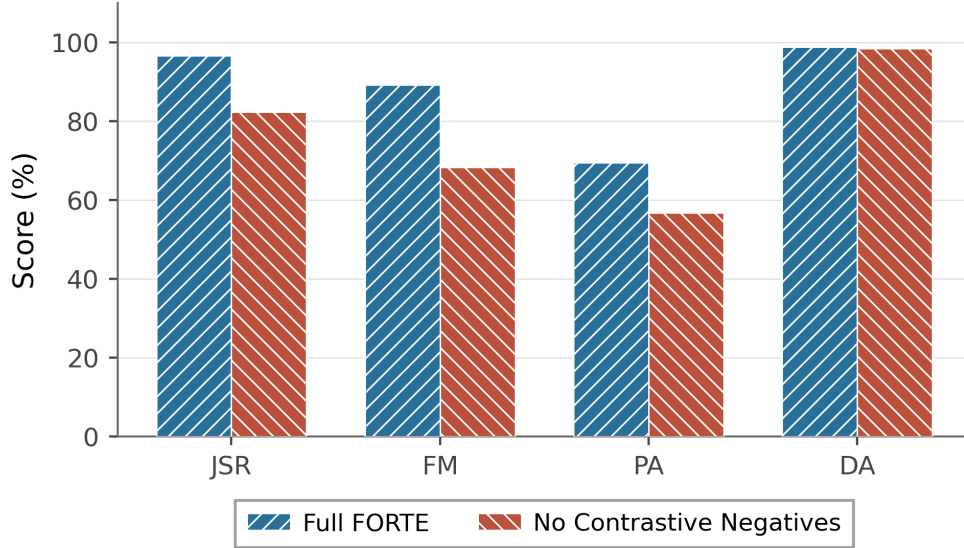


FIGURE 5.2. Comparison between training with FORTE’s contrastive negatives over the error taxonomy versus not.

Residual failures are bounded by FM’s step-count requirement. We measure the fraction of judge-passing plans (judge ≥ 80) that also satisfy FM (FM=1: step-count match, $\geq 50\%$ tool-name Jaccard, and $\geq 50\%$ parameter accuracy jointly). On ToolHop, agreement is high and rises monotonically with backbone capacity: 80.4%, 89.5%, and 97.2% for Llama3.2-3B, Qwen2.5-3B, and Qwen2.5-7B. On NESTFUL the conditional drops to 37.1%, 33.6%, and 44.0%, but 79.2%, 75.4%, and 72.8% of NESTFUL FM failures across the three backbones are pure step-count mismatches with otherwise correct tool selection, parameter binding, and dependency threading (Section 5.6): the planner emits compressed plans that the rubric scores as valid but FM rejects for departing from the gold’s exact step decomposition. The 97.2% ToolHop agreement of our largest backbone, where step-count compression is rarely viable, is consistent with this interpretation. The remaining 21 to 27% of NESTFUL FM failures involve true structural errors, motivating plan-length calibration as the natural next step rather than further work on the structural primitives FORTE already learns reliably.

Inference cost is comparable to baselines and dominated by open-source compute. FORTE’s Best-of- N inference issues five planner forward passes and five parallel judge calls per query, matching ReAct (3 to 5 iterations) and TOOL-PLANNER (beam width 3 plus scoring) and

well below tree-search methods like LATS or ToolTree (20 to 50 LLM calls). Running on local 3B to 7B open-source backbones, per-query cost is dominated by GPU compute on commodity hardware, placing FORTE roughly an order of magnitude below closed-source frontier baselines while matching or beating them on plan quality (Table 5.3). Frontier-grade structural compliance is therefore achievable through targeted failure-mode training rather than scale.

5.6. Qualitative Analysis

We present three qualitative examples drawn from our held-out evaluation: one example of FORTE producing a correct plan despite the reference being seeded with a wrong-tool error, one residual failure that exposes a secondary failure mode, and one closed-source comparison illustrating GPT-5.5’s hallucinated parameters and compressed step counts.

Wrong-tool robustness (NESTFUL). Although the dataset’s reference for this query was seeded with the wrong tool (`non_whitespace_substrings`), FORTE generates the correct `swap_by_index` call directly from the query, demonstrating that exposure to wrong-tool negatives during training does not bias the planner toward reproducing them.

NESTFUL : reference seeded with wrong_tool

```

QUERY:
Find the unique common values in two lists [1, 2, 3, 4, 5] and
[4, 5, 6, 7, 8]. Then, swap the values at index 0 and 1 in the
resulting list.

REFERENCE PLAN (deliberately flawed : wrong_tool):
Step 0: {{0}} = unique_common_values(list1=[1, 2, 3, 4, 5],
list2=[4, 5, 6, 7, 8])
Step 1: {{1}} = non_whitespace_substrings(values={{0}}, i=0, j=1)

FORTE GENERATED PLAN (T=0.2):
Step 0: {{0}} = unique_common_values(list1=[1, 2, 3, 4, 5],
list2=[4, 5, 6, 7, 8])
Step 1: {{1}} = swap_by_index(values={{0}}, i=0, j=1)

JUDGE: score=100, prediction="yes", confidence=0.99.
```

Residual failure: step-count compression (NESTFUL, judge=100, FM fails). This is the dominant NESTFUL FM failure mode: the planner compresses the 4-step gold decomposition into a

3-step solution composed of compound arithmetic calls, which the rubric-based judge correctly scores as syntactically valid but which the structural evaluator marks as a functional-match failure due to the step-count mismatch. The judge’s deductive rubric does not verify that the compressed plan computes the same intermediate values as the gold, exposing the syntactic-only scoring limitation discussed in the Limitations section.

```

NESTFUL : Forte-Qwen2.5-7B,  $T=0.4$ 

QUERY:
If n is a prime number greater than 5, what is the remainder when n
^2
is divided by 12?

GOLD PLAN (4 steps):
Step 0: {{0}} = add(arg_0=5, arg_1=2)
Step 1: {{1}} = power(arg_0={{0}}, arg_1=2)
Step 2: {{2}} = multiply(arg_0=12, arg_1=4)
Step 3: {{3}} = subtract(arg_0={{1}}, arg_1={{2}})

FORTE GENERATED PLAN (3 steps, all 5 candidates scored 100):
Step 0: {{0}} = multiply(arg_0=4, arg_1=3)
Step 1: {{1}} = divide(arg_0={{0}}, arg_1=2)
Step 2: {{2}} = power(arg_0={{1}}, arg_1=2)

JUDGE: score=100, prediction="yes", confidence=0.99.
STRUCTURAL EVAL: step_count_match=False (3 vs 4),
                  exact_tool_accuracy=0.0, functional_match=False.

```

Comparison with GPT-5.5 on the same ToolHop query. GPT-5.5 exhibits two of the failure modes FORTE’s training data systematically penalises: (i) it hallucinates an `additional_filters` parameter that does not appear in the `film_release_year_finder` signature, and an `output_format="sum_unique_prime_factors"` value that does not exist; (ii) it compresses the plan from four steps to three, folding the final `sum_digits` call into a non-existent `prime_factorization` output format. FORTE’s plan for the same query issues four schema-compliant steps and scores 100.

```

ToolHop : GPT-5.5 (closed-source frontier,  $T=0.0$ )

QUERY:
What is the sum of the (unique) prime factors of the year when the
actress that plays Kaylee Frye in the series "Firefly" also stars
with
Alan Dale in a sci-fi tv film released in?

```

```

GPT-5.5 GENERATED PLAN:
Step 0: {{0}} = character_actor_lookup(character_name="Kaylee Frye",

    series_name="Firefly", exact_match=true)
Step 1: {{1}} = film_release_year_finder(actors=[{{0}}, "Alan Dale
"],
    genre="sci-fi", additional_filters={"format": "tv film"})
Step 2: {{2}} = prime_factorization(number={{1}},
    output_format="sum_unique_prime_factors")

JUDGE: score=70, prediction="uncertain", confidence=0.78.
Step count: 3 vs gold's 4 (step_count_match=False).

FORTE-Qwen2.5-7B GENERATED PLAN (T=1.0):
Step 0: {{0}} = character_actor_lookup(character_name='Kaylee Frye',

    series_name='Firefly', exact_match=True, case_sensitive=
False)
Step 1: {{1}} = film_release_year_finder(actors=['{{0}}', 'Alan
Dale'],
    genre='sci-fi')
Step 2: {{2}} = prime_factorization(number={{1}},
    output_format='unique_list', algorithm='general_purpose')
Step 3: {{3}} = sum_digits(number={{2}}, output_format='number')

JUDGE: score=100, prediction="yes", confidence=0.98.

```

5.7. Limitations

We note several limitations. First, FORTE trains a separate planner per benchmark: the augmented corpus, judge training, and RL fine-tuning are all dataset-specific. Second, our error taxonomy is derived from observed failure patterns in ToolHop and NESTFUL; additional error categories may exist in domains we do not evaluate, such as long-horizon GUI agents or browser-based tool use. Third, the judge’s deductive rubric scores syntactic plan correctness rather than semantic task success; this decoupling is what enables training stability without a tool execution environment, but it means FORTE does not directly optimize for end-task reward, and may miss failure modes that are only visible at execution time. Finally, the 9-error taxonomy treats each failure mode as independent; in practice plan errors can co-occur, and the judge is trained to detect but not necessarily to disentangle compound failures.

5.8. Conclusion

We presented FORTE, a framework that trains multi-step tool planners by exposing them to a systematic taxonomy of failure modes during training. By grounding judge training in deductive structural scoring over a 9-error rubric and using contrastive negatives drawn from minimally edited gold plans, FORTE produces planners that achieve the highest Judge Success Rate across all six backbone and dataset configurations on ToolHop and NESTFUL, beating established baselines by 2.5 to 13.2 JSR points and exceeding GPT-5.5 by up to 33.9 points using only 3B to 7B open-source models. Our ablations show that contrastive exposure is the critical ingredient: without it, the judge loses discriminative ability and Best-of- N selection becomes counterproductive. FORTE’s decoupled judge-planner architecture, together with the augmented contrastive datasets we construct for ToolHop and NESTFUL, transforms error taxonomies from post-hoc diagnostic instruments into the primary driver of training. In the terms of this dissertation, it shows that the data-model synergy perspective extends naturally from analysis and diagnosis to construction: a structural account of how a model fails can be synthesized into data that trains the failure out, suggesting that targeted exposure to failure modes during training is a simple and effective alternative to inference-time search and structural scaffolding for tool-augmented agents.

CHAPTER 6

Concluding remarks and future work

This dissertation argued that large language models can be made more effective and more reliable by studying the relationship between training data, model internals, and downstream behavior in a more principled way. Across four chapters, the dissertation developed this perspective through the unifying theme of *data-model synergy*, the idea that pretrained models acquire useful but uneven inductive biases, and that these biases can be identified, measured, and leveraged to improve post-training adaptation and evaluation.

The first part of the dissertation focused on *utility*. Chapter 2 introduced **LayerIF**, a framework for estimating task-specific layer quality in large language models through layer-localized influence analysis. Rather than relying only on model-internal heuristics such as weight statistics or activation patterns, LayerIF used sensitivity to training data as a signal of how strongly different layers contribute to downstream task behavior. The results showed that layer quality is often task dependent, and that these estimates can guide practical post-training decisions such as LoRA-MoE expert allocation and layer-wise sparsity allocation for pruning. More broadly, this chapter demonstrated that post-training resource allocation benefits from being informed by the interaction between training data and model components, rather than by model-centric diagnostics alone.

The second part focused on *generalization*. Chapter 3 introduced **Indirect In-Context Learning**, a generalized framework for demonstration selection when the candidate pool is imperfect, including settings with task mismatch, heterogeneity, noise, or label corruption. This chapter showed that influence-guided selection can improve inference-time adaptation even when standard assumptions of direct task alignment do not hold. In doing so, it reframed demonstration selection not simply as a problem of semantic similarity, but as a problem of *inductive bias alignment*, selecting examples that better activate useful latent behavior for the target task. This extended the data-model synergy perspective from parameter-space adaptation to prompt-time adaptation.

The third part focused on *robustness*. Chapter 4 examined zero-shot abstractive summarization through two complementary studies, **position bias**, which characterizes whether summarization models draw information from the same regions of a source document as human summaries, and **relevance paraphrasing**, which tests whether summarizers remain stable when the most salient source content is rewritten in semantically equivalent ways. Together, these studies showed that strong aggregate summarization performance does not necessarily imply reliable source-grounded behavior. Models may appear strong according to standard metrics while still exhibiting structural misalignment in source allocation or instability under benign perturbations. This chapter therefore argued that robustness should be evaluated more directly at the level of source use and behavioral consistency.

The fourth part focused on *reliability* in agentic tool use. Chapter 5 introduced **FORTE**, a framework for improving the structural reliability of multi-step tool planners without executing tools. Whereas the robustness chapter characterized failure modes to diagnose unreliable behavior, FORTE used an explicit characterization of failure modes to construct targeted supervision for reducing such behavior during planning. Specifically, a nine-error taxonomy of tool-plan failures was used to synthesize contrastive negative plans, which trained a structural judge that supplied both the reward for reinforcement-learning fine-tuning and the verifier for Best-of- N inference. The chapter showed that this failure-mode-aware training produces planners that match or exceed substantially larger closed-source models on structural reliability, using only modest open-source backbones. In doing so, it extended the data-model synergy perspective from analysis and diagnosis to intervention, showing that a structural account of model failure can be operationalized as training data that improves downstream planning behavior.

Taken together, these contributions support a broader view of LLM post-training as a *data-driven methodology*. This dissertation showed that useful structure can be uncovered by analyzing how data, model internals, and downstream predictions interact, and that this structure can be acted upon, whether by reallocating capacity, selecting better evidence, or constructing data that targets known failure modes. This perspective helps make post-training decisions more principled, from where to add or remove capacity, to which demonstrations to provide at inference time, to how to diagnose

whether a model is behaving robustly in deployment, to how to improve the structural reliability of agentic planning systems.

Beyond the specific methods introduced here, this research also has broader implications for the development of more capable and trustworthy AI systems. Improving the utility of LLMs can help reduce unnecessary computational waste by allocating capacity more effectively. Improving generalization in low-resource and imperfect-supervision settings can broaden access to language technologies for domains and communities where clean, task-specific labeled data is scarce. Improving robustness and source-grounding can make deployed systems more reliable in settings where errors are costly and where users must be able to trust not only what a model outputs, but also how that output is produced. Improving the structural reliability of agentic tool plans matters for a related reason: as LLMs are given the ability to act through external tools, structurally sound plans are less likely to fail silently partway through execution, which is precisely the kind of failure that is difficult to detect and costly in deployment. In this sense, the dissertation contributes to a broader effort to make large language models not only more capable, but also more interpretable, adaptable, and dependable.

Several promising directions for future work emerge naturally from this dissertation. First, the LayerIF framework could be extended beyond text-only language models to *vision-language models*, where analogous questions arise about which layers and modal components are most responsible for grounding, generation, and multimodal reasoning. Second, the Indirect ICL framework could be expanded to genuinely low-resource languages and specialized domains, where demonstration scarcity, distribution shift, and weak supervision are not merely experimental settings but central practical challenges. Third, the robustness analyses developed for text summarization could be extended to *video summarization* and other multimodal generation tasks, where questions of source allocation, bias, stability, and fairness become even more complex. Finally, the reliability framework of Chapter 5 suggests several extensions of its own. A natural next step is to train a single planner that generalizes across tool-planning benchmarks rather than one planner per dataset. Another is to ground the structural judge in execution feedback, so that its reward captures semantic task success in addition to syntactic plan correctness. A third is to model compound failures, in which several error types co-occur within a single plan, rather than treating each failure mode as independent.

More broadly, the same failure-mode-driven construction could be applied to longer-horizon settings such as browser-based or graphical-user-interface agents, whose plans span many more steps and whose failure taxonomies are correspondingly richer.

In sum, this dissertation advanced a unified research program for understanding and improving large language models through the lens of data-model synergy. By studying how training data shapes model internals and how those internals govern downstream behavior, and by constructing targeted data that addresses structurally defined failure modes, it presented methods for improving utility, generalization, robustness, and reliability in a common conceptual framework. The central lesson is that better language models will not come only from scaling, but also from better understanding how their learned structure can be measured, aligned, and exploited in principled ways.

Bibliography

- [1] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of EMNLP*, pages 2383–2392. Association for Computational Linguistics, 2016.
- [2] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering, 2018.
- [3] Adina Williams, Nikita Nangia, and Samuel R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of NAACL-HLT*, 2018.
- [4] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In Jill Burstein, Christy Doran, and Tamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.
- [5] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence?, 2019.
- [6] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra,

- Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021.
- [7] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.
 - [8] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.
 - [9] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.
 - [10] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on machine learning research*, 2023.
 - [11] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.
 - [12] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.
 - [13] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.
 - [14] Frank R Hampel. The influence curve and its role in robust estimation. *Journal of the american statistical association*, 69(346):383–393, 1974.
 - [15] R Dennis Cook and Sanford Weisberg. Characterizations of an empirical influence function for detecting influential cases in regression. *Technometrics*, 22(4):495–508, 1980.
 - [16] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR, 2017.

- [17] Anshuman Chhabra, Bo Li, Jian Chen, Prasant Mohapatra, and Hongfu Liu. Outlier gradient analysis: Efficiently improving deep learning model performance via hessian-free influence functions. *International Conference on Machine Learning*, 2025.
- [18] Yongchan Kwon, Eric Wu, Kevin Wu, and James Zou. Datainf: Efficiently estimating data influence in lora-tuned llms and diffusion models. *arXiv preprint arXiv:2310.00902*, 2023.
- [19] Andrea Schioppa, Polina Zablotskaia, David Vilar, and Artem Sokolov. Scaling up influence functions. In *AAAI Conference on Artificial Intelligence*, 2022.
- [20] Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, et al. Studying large language model generalization with influence functions. *arXiv preprint arXiv:2308.03296*, 2023.
- [21] Anshuman Chhabra, Peizhao Li, Prasant Mohapatra, and Hongfu Liu. " what data benefits my classifier?" enhancing model performance and interpretability through influence-based data selection. In *The Twelfth International Conference on Learning Representations*, 2024.
- [22] Zhuoming Liu, Hao Ding, Huaping Zhong, Weijia Li, Jifeng Dai, and Conghui He. Influence selection for active learning. In *IEEE/CVF International Conference on Computer Vision*, 2021.
- [23] Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A Roberts. The unreasonable ineffectiveness of the deeper layers. *arXiv preprint arXiv:2403.17887*, 2024.
- [24] Oscar Skea, Md Rifat Arefin, Dan Zhao, Niket Patel, Jalal Naghiyev, Yann LeCun, and Ravid Shwartz-Ziv. Layer by layer: Uncovering hidden representations in language models. *arXiv preprint arXiv:2502.02013*, 2025.
- [25] Yang Zhang, Yanfei Dong, and Kenji Kawaguchi. Investigating layer importance in large language models. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 469–479, 2024.
- [26] Lu Yin, You Wu, Zhenyu Zhang, Cheng-Yu Hsieh, Yaqing Wang, Yiling Jia, Gen Li, Ajay Jaiswal, Mykola Pechenizkiy, Yi Liang, Michael Bendersky, Zhangyang Wang, and Shiwei Liu. Outlier weighed layerwise sparsity (owl): A missing secret sauce for pruning llms to high sparsity, 2024.

- [27] Haiquan Lu, Yefan Zhou, Shiwei Liu, Zhangyang Wang, Michael W Mahoney, and Yaoqing Yang. Alphapruning: Using heavy-tailed self regularization theory for improved layer-wise pruning of large language models. *Advances in Neural Information Processing Systems*, 37:9117–9152, 2024.
- [28] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- [29] Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. Metaicl: Learning to learn in context. *arXiv preprint arXiv:2110.15943*, 2021.
- [30] Yu Fei, Yifan Hou, Zeming Chen, and Antoine Bosselut. Mitigating label biases for in-context learning. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14014–14031, Toronto, Canada, July 2023. Association for Computational Linguistics.
- [31] Chen Cheng, Xinzhi Yu, Haodong Wen, Jingsong Sun, Guanzhang Yue, Yihao Zhang, and Zeming Wei. Exploring the robustness of in-context learning with noisy labels. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2025.
- [32] Sewon Min, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 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, 2022.
- [33] Dylan Bouchard. An actionable framework for assessing bias and fairness in large language model use cases. *arXiv preprint arXiv:2407.10853*, 2024.
- [34] Isabel O. Gallegos, Ryan A. Rossi, Joe Barrow, Md Mehrab Tanjim, Sungchul Kim, Franck Dernoncourt, Tong Yu, Ruiyi Zhang, and Nesreen K. Ahmed. Bias and fairness in large language models: A survey. *Computational Linguistics*, 50(3):1097–1179, September 2024.
- [35] Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th annual meeting of the*

- association for computational linguistics*, pages 1906–1919, 2020.
- [36] Matt Grenander, Yue Dong, Jackie Chi Kit Cheung, and Annie Louis. Countering the effects of lead bias in news summarization via multi-stage training and auxiliary losses. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6019–6024, Hong Kong, China, November 2019. Association for Computational Linguistics.
 - [37] Chenguang Zhu, Ziyi Yang, Robert Gmyr, Michael Zeng, and Xuedong Huang. Leveraging lead bias for zero-shot abstractive news summarization. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, pages 1462–1471, 2021.
 - [38] Mingyu Jin, Qinkai Yu, Jingyuan Huang, Qingcheng Zeng, Zhenting Wang, Wen Yue Hua, Haiyan Zhao, Kai Mei, Yanda Meng, Kaize Ding, et al. Exploring concept depth: How large language models acquire knowledge and concept at different layers? *arXiv preprint arXiv:2404.07066*, 2024.
 - [39] Siqi Fan, Xin Jiang, Xiang Li, Xuying Meng, Peng Han, Shuo Shang, Aixin Sun, Yequan Wang, and Zhongyuan Wang. Not all layers of llms are necessary during inference. *arXiv preprint arXiv:2403.02181*, 2024.
 - [40] Sofia Serrano and Noah A Smith. Is attention interpretable? *arXiv preprint arXiv:1906.03731*, 2019.
 - [41] Samira Abnar and Willem Zuidema. Quantifying attention flow in transformers, 2020.
 - [42] Sarah Wiegrefe and Yuval Pinter. Attention is not not explanation, 2019.
 - [43] Yaru Hao, Li Dong, Furu Wei, and Ke Xu. Self-attention attribution: Interpreting information interactions inside transformer, 2021.
 - [44] Juhan Bae, Wu Lin, Jonathan Lorraine, and Roger Grosse. Training data attribution via approximate unrolled differentiation. *arXiv preprint arXiv:2405.12186*, 2024.
 - [45] Qidong Liu, Xian Wu, Xiangyu Zhao, Yuanshao Zhu, Derong Xu, Feng Tian, and Yefeng Zheng. When moe meets llms: Parameter efficient fine-tuning for multi-task medical applications. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development*

- in *Information Retrieval*, pages 1104–1114, 2024.
- [46] Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Jun Zhao, Wei Shen, Yuhao Zhou, Zhiheng Xi, Xiao Wang, Xiaoran Fan, et al. Loramoe: Alleviate world knowledge forgetting in large language models via moe-style plugin. *arXiv preprint arXiv:2312.09979*, 2023.
 - [47] Ted Zadouri, Ahmet Üstün, Arash Ahmadian, Beyza Ermiş, Acyr Locatelli, and Sara Hooker. Pushing mixture of experts to the limit: Extremely parameter efficient moe for instruction tuning. *arXiv preprint arXiv:2309.05444*, 2023.
 - [48] Chongyang Gao, Kezhen Chen, Jinmeng Rao, Baochen Sun, Ruibo Liu, Daiyi Peng, Yawen Zhang, Xiaoyuan Guo, Jie Yang, and VS Subrahmanian. Higher layers need more lora experts, 2024.
 - [49] Peijun Qing, Chongyang Gao, Yefan Zhou, Xingjian Diao, Yaoqing Yang, and Soroush Vosoughi. Alphalora: Assigning lora experts based on layer training quality. *arXiv preprint arXiv:2410.10054*, 2024.
 - [50] Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot, 2023.
 - [51] Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023.
 - [52] Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. Linguistic knowledge and transferability of contextual representations. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1073–1094, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.
 - [53] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What does BERT look at? an analysis of BERT’s attention. In Tal Linzen, Grzegorz Chrupała, Yonatan Belinkov, and Dieuwke Hupkes, editors, *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy, August 2019. Association for Computational Linguistics.

- [54] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information, 2017.
- [55] Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. What do you learn from context? probing for sentence structure in contextualized word representations, 2019.
- [56] Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes, 2018.
- [57] Charles H. Martin and Michael W. Mahoney. Implicit self-regularization in deep neural networks: Evidence from random matrix theory and implications for learning, 2018.
- [58] Lloyd S Shapley. A value for n-person games. In Harold W. Kuhn and Albert W. Tucker, editors, *Contributions to the Theory of Games II*, pages 307–317. Princeton University Press, Princeton, 1953.
- [59] Yang Zhang, Yanfei Dong, and Kenji Kawaguchi. Investigating layer importance in large language models, 2024.
- [60] Oscar Skea, Md Rifat Arefin, Dan Zhao, Niket Patel, Jalal Naghiyev, Yann LeCun, and Ravid Shwartz-Ziv. Layer by layer: Uncovering hidden representations in language models, 2025.
- [61] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1989.
- [62] Babak Hassibi and David Stork. Second order derivatives for network pruning: Optimal brain surgeon. In S. Hanson, J. Cowan, and C. Giles, editors, *Advances in Neural Information Processing Systems*, volume 5. Morgan-Kaufmann, 1992.
- [63] Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Gloriosi, and Daniel A. Roberts. The unreasonable ineffectiveness of the deeper layers, 2025.
- [64] Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. Shortgpt: Layers in large language models are more redundant than you expect, 2024.
- [65] Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models, 2023.

- [66] Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks, 2019.
- [67] Sehoon Kim, Coleman Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W. Mahoney, and Kurt Keutzer. Squeezellm: Dense-and-sparse quantization, 2024.
- [68] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.
- [69] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarczyk, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.
- [70] Dengchun Li, Yingzi Ma, Naizheng Wang, Zhengmao Ye, Zhiyuan Cheng, Yinghao Tang, Yan Zhang, Lei Duan, Jie Zuo, Cal Yang, et al. Mixlora: Enhancing large language models fine-tuning with lora-based mixture of experts. *arXiv preprint arXiv:2404.15159*, 2024.
- [71] Shu Yang, Muhammad Asif Ali, Cheng-Long Wang, Lijie Hu, and Di Wang. Moral: Moe augmented lora for llms’ lifelong learning. *arXiv preprint arXiv:2402.11260*, 2024.
- [72] Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Wei Shen, Limao Xiong, Yuhao Zhou, Xiao Wang, Zhiheng Xi, Xiaoran Fan, et al. Loramoe: Alleviating world knowledge forgetting in large language models via moe-style plugin. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1932–1945, 2024.
- [73] Tongxu Luo, Jiahe Lei, Fangyu Lei, Weihao Liu, Shizhu He, Jun Zhao, and Kang Liu. Moelora: Contrastive learning guided mixture of experts on parameter-efficient fine-tuning for large language models. *arXiv preprint arXiv:2402.12851*, 2024.
- [74] Wenfeng Feng, Chuzhan Hao, Yuewei Zhang, Yu Han, and Hao Wang. Mixture-of-loras: An efficient multitask tuning method for large language models. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 11371–11380, 2024.
- [75] Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. Lorahub: Efficient cross-task generalization via dynamic lora composition, 2024.

- [76] Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *International Conference on Machine Learning (ICML)*. PMLR, 2019.
- [77] Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nick Hynes, Nezihe Merve Gürel, Bo Li, Ce Zhang, Dawn Song, and Costas J Spanos. Towards efficient data valuation based on the shapley value. In *International Conference on Artificial Intelligence and Statistics*, 2019.
- [78] Yongchan Kwon and James Zou. Beta Shapley: a unified and noise-reduced data valuation framework for machine learning. In *International Conference on Artificial Intelligence and Statistics*, 2022.
- [79] Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nezihe Merve Gurel, Bo Li, Ce Zhang, Costas Spanos, and Dawn Song. Efficient task specific data valuation for nearest neighbor algorithms. *Proceedings of the VLDB Endowment*, 2018.
- [80] Chih-Kuan Yeh, Joon Kim, Ian En-Hsu Yen, and Pradeep K Ravikumar. Representer point selection for explaining deep neural networks. *Advances in Neural Information Processing Systems*, 2018.
- [81] Yuanyuan Chen, Boyang Li, Han Yu, Pengcheng Wu, and Chunyan Miao. Hydra: Hypergradient data relevance analysis for interpreting deep neural networks. In *AAAI Conference on Artificial Intelligence*, 2021.
- [82] Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. Datamodels: Predicting predictions from training data. *arXiv preprint arXiv:2202.00622*, 2022.
- [83] Andrea Schioppa, Katja Filippova, Ivan Titov, and Polina Zablotskaia. Theoretical and practical perspectives on what influence functions do. *Advances in Neural Information Processing Systems*, 2024.
- [84] Anshuman Chhabra, Bo Li, Jian Chen, Prasant Mohapatra, and Hongfu Liu. Outlier Gradient Analysis: Efficiently Identifying Detrimental Training Samples for Deep Learning Models. In *International Conference on Machine Learning (ICML)*. PMLR, 2025.
- [85] Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. Trak: Attributing model behavior at scale. *arXiv preprint arXiv:2303.14186*, 2023.
- [86] Irina Bejan, Artem Sokolov, and Katja Filippova. Make every example count: On the stability and utility of self-influence for learning from noisy nlp datasets. In *Conference on Empirical*

Methods in Natural Language Processing, 2023.

- [87] Megh Thakkar, Tolga Bolukbasi, Sriram Ganapathy, Shikhar Vashishth, Sarath Chandar, and Partha Talukdar. Self-influence guided data reweighting for language model pre-training. In *Conference on Empirical Methods in Natural Language Processing*, 2023.
- [88] SungYub Kim, Kyungsu Kim, and Eunho Yang. Gex: A flexible method for approximating influence via geometric ensemble. *Advances in Neural Information Processing Systems*, 36, 2024.
- [89] Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 33:19920–19930, 2020.
- [90] Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. Less: Selecting influential data for targeted instruction tuning. *arXiv preprint arXiv:2402.04333*, 2024.
- [91] Guillaume Charpiat, Nicolas Girard, Loris Felardos, and Yuliya Tarabalka. Input similarity from the neural network perspective. *Advances in Neural Information Processing Systems*, 2019.
- [92] R Douglas Martin and Victor J Yohai. Influence functionals for time series. *The annals of Statistics*, pages 781–818, 1986.
- [93] Bill Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Third international workshop on paraphrasing (IWP2005)*, 2005.
- [94] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.
- [95] Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in Neural Information Processing Systems*, 35:2507–2521, 2022.
- [96] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*,

- 2018.
- [97] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018.
 - [98] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.
 - [99] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.
 - [100] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.
 - [101] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.
 - [102] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, L  lio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timoth  e Lacroix, and William El Sayed. Mistral 7b, 2023.
 - [103] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivi  re, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, L  onard Hussenot, Pier Giuseppe Sessa, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Am  lie H  liou, Andrea Tacchetti, Anna Bulanov  , Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Cl  ment Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Christian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, Justin Mao-Jones, Katherine Lee, Kathy Yu, Katie Millican, Lars Lowe Sjoesund,

- Lisa Lee, Lucas Dixon, Machel Reid, Maciej Mikula, Mateo Wirth, Michael Sharman, Nikolai Chinaev, Nithum Thain, Olivier Bachem, Oscar Chang, Oscar Wahltinez, Paige Bailey, Paul Michel, Petko Yotov, Rahma Chaabouni, Ramona Comanescu, Reena Jana, Rohan Anil, Ross McIlroy, Ruibo Liu, Ryan Mullins, Samuel L Smith, Sebastian Borgeaud, Sertan Girgin, Sholto Douglas, Shree Pandya, Siamak Shakeri, Soham De, Ted Klimenko, Tom Hennigan, Vlad Feinberg, Wojciech Stokowiec, Yu hui Chen, Zafarali Ahmed, Zhitao Gong, Tris Warkentin, Ludovic Peran, Minh Giang, Clément Farabet, Oriol Vinyals, Jeff Dean, Koray Kavukcuoglu, Demis Hassabis, Zoubin Ghahramani, Douglas Eck, Joelle Barral, Fernando Pereira, Eli Collins, Armand Joulin, Noah Fiedel, Evan Senter, Alek Andreev, and Kathleen Kenealy. Gemma: Open models based on gemini research and technology, 2024.
- [104] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.
- [105] Sewon Min, Xinxu Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? In *EMNLP*, 2022.
- [106] Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. What makes good in-context examples for gpt-3? *arXiv preprint arXiv:2101.06804*, 2021.
- [107] Cem Anil, Yuhuai Wu, Anders Andreassen, Aitor Lewkowycz, Vedant Misra, Vinay Ramasesh, Ambrose Slone, Guy Gur-Ari, Ethan Dyer, and Behnam Neyshabur. Exploring length generalization in large language models. *Advances in Neural Information Processing Systems*, 35:38546–38556, 2022.
- [108] Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12697–12706. PMLR, 18–24 Jul 2021.
- [109] Kishaloy Halder, Alan Akbik, Josip Krapac, and Roland Vollgraf. Task-aware representation of sentences for generic text classification. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3202–3213, 2020.

- [110] Ohad Rubin, Jonathan Herzig, and Jonathan Berant. Learning to retrieve prompts for in-context learning. *arXiv preprint arXiv:2112.08633*, 2021.
- [111] Shivanshu Gupta, Matt Gardner, and Sameer Singh. Coverage-based example selection for in-context learning. *arXiv preprint arXiv:2305.14907*, 2023.
- [112] Wenpeng Yin, Muhao Chen, Ben Zhou, Qiang Ning, Kai-Wei Chang, and Dan Roth. Indirectly supervised natural language processing. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 6: Tutorial Abstracts)*, 2023.
- [113] Bangzheng Li, Ben Zhou, Xingyu Fu, Fei Wang, Dan Roth, and Muhao Chen. Famicom: Further demystifying prompts for language models with task-agnostic performance estimation. *arXiv preprint arXiv:2406.11243*, 2024.
- [114] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.
- [115] Shengnan An, Zeqi Lin, Qiang Fu, Bei Chen, Nanning Zheng, Jian-Guang Lou, and Dongmei Zhang. How do in-context examples affect compositional generalization? *arXiv preprint arXiv:2305.04835*, 2023.
- [116] Yiming Zhang, Shi Feng, and Chenhao Tan. Active example selection for in-context learning. *arXiv preprint arXiv:2211.04486*, 2022.
- [117] Hila Gonen, Srini Iyer, Terra Blevins, Noah A Smith, and Luke Zettlemoyer. Demystifying prompts in language models via perplexity estimation. *arXiv preprint arXiv:2212.04037*, 2022.
- [118] Taylor Sorensen, Joshua Robinson, Christopher Michael Rytting, Alexander Glenn Shaw, Kyle Jeffrey Rogers, Alexia Pauline Delorey, Mahmoud Khalil, Nancy Fulda, and David Wingate. An information-theoretic approach to prompt engineering without ground truth labels. *arXiv preprint arXiv:2203.11364*, 2022.
- [119] Shivanshu Gupta, Clemens Rosenbaum, and Ethan R Elenberg. Gistscore: Learning better representations for in-context example selection with gist bottlenecks. *arXiv preprint arXiv:2311.09606*, 2023.

- [120] Hongjin Su, Jungo Kasai, Chen Henry Wu, Weijia Shi, Tianlu Wang, Jiayi Xin, Rui Zhang, Mari Ostendorf, Luke Zettlemoyer, Noah A Smith, et al. Selective annotation makes language models better few-shot learners. *arXiv preprint arXiv:2209.01975*, 2022.
- [121] Jiacheng Ye, Zhiyong Wu, Jiangtao Feng, Tao Yu, and Lingpeng Kong. Compositional exemplars for in-context learning. In *International Conference on Machine Learning*, pages 39818–39833. PMLR, 2023.
- [122] Costas Mavromatis, Balasubramaniam Srinivasan, Zhengyuan Shen, Jiani Zhang, Huzefa Rangwala, Christos Faloutsos, and George Karypis. Which examples to annotate for in-context learning? towards effective and efficient selection. *arXiv preprint arXiv:2310.20046*, 2023.
- [123] Anwoy Chatterjee, Eshaan Tanwar, Subhabrata Dutta, and Tanmoy Chakraborty. Language models can exploit cross-task in-context learning for data-scarce novel tasks. *arXiv preprint arXiv:2405.10548*, 2024.
- [124] R Dennis Cook and Sanford Weisberg. *Residuals and influence in regression*. New York: Chapman and Hall, 1982.
- [125] Shaokun Zhang, Xiaobo Xia, Zhaoqing Wang, Ling-Hao Chen, Jiale Liu, Qingyun Wu, and Tongliang Liu. Ideal: Influence-driven selective annotations empower in-context learners in large language models, 2025.
- [126] Tai Nguyen and Eric Wong. In-context example selection with influences. *arXiv preprint arXiv:2302.11042*, 2023.
- [127] Minh-Hao Van, Xintao Wu, et al. In-context learning demonstration selection via influence analysis. *arXiv preprint arXiv:2402.11750*, 2024.
- [128] Zijian Zhou, Xiaoqiang Lin, Xinyi Xu, Alok Prakash, Daniela Rus, and Bryan Kian Hsiang Low. Detail: Task demonstration attribution for interpretable in-context learning. *Advances in Neural Information Processing Systems*, 37:25284–25310, 2024.
- [129] Andrea Schioppa, Katja Filippova, Ivan Titov, and Polina Zablotskaia. Theoretical and practical perspectives on what influence functions do. *Advances in Neural Information Processing Systems*, 36:27560–27581, 2023.
- [130] Sang Keun Choe, Hwijeen Ahn, Juhan Bae, Kewen Zhao, Minsoo Kang, Youngseog Chung, Adithya Pratapa, Willie Neiswanger, Emma Strubell, Teruko Mitamura, et al. What is

- your data worth to gpt? llm-scale data valuation with influence functions. *arXiv preprint arXiv:2405.13954*, 2024.
- [131] Yinhan Liu. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.
- [132] Naman Agarwal, Brian Bullins, and Elad Hazan. Second-order stochastic optimization for machine learning in linear time. *Journal of Machine Learning Research*, 18(116):1–40, 2017.
- [133] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.
- [134] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022.
- [135] Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361, 2021.
- [136] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
- [137] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.
- [138] Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourier, Nathan Habib, Nathan Sarrazin, Omar Sanseviero, Alexander M. Rush, and Thomas Wolf. Zephyr: Direct distillation of lm alignment, 2023.
- [139] Qwen Team. Qwen2.5: A party of foundation models, September 2024.
- [140] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The

- llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.
- [141] Keqin Peng, Liang Ding, Yancheng Yuan, Xuebo Liu, Min Zhang, Yuanxin Ouyang, and Dacheng Tao. Revisiting demonstration selection strategies in in-context learning. *arXiv preprint arXiv:2401.12087*, 2024.
- [142] Giwon Hong, Emile Van Krieken, Edoardo Ponti, Nikolay Malkin, and Pasquale Minervini. Mixtures of in-context learners. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 26332–26351, 2025.
- [143] Jiazhu Dai, Chuanshuai Chen, and Yufeng Li. A backdoor attack against lstm-based text classification systems. *IEEE Access*, 7:138872–138878, 2019.
- [144] Fanchao Qi, Yangyi Chen, Xurui Zhang, Mukai Li, Zhiyuan Liu, and Maosong Sun. Mind the style of text! adversarial and backdoor attacks based on text style transfer. *arXiv preprint arXiv:2110.07139*, 2021.
- [145] Shizhe Diao, Pengcheng Wang, Yong Lin, Rui Pan, Xiang Liu, and Tong Zhang. Active prompting with chain-of-thought for large language models. *arXiv preprint arXiv:2302.12246*, 2023.
- [146] Tianyi Zhang, Faisal Ladhak, Esin Durmus, Percy Liang, Kathleen McKeown, and Tatsunori B Hashimoto. Benchmarking large language models for news summarization. *arXiv preprint arXiv:2301.13848*, 2023.
- [147] Fei Wang, Wenjie Mo, Yiwei Wang, Wenxuan Zhou, and Muhao Chen. A causal view of entity bias in (large) language models. *arXiv preprint arXiv:2305.14695*, 2023.
- [148] Yiwei Wang, Muhao Chen, Wenxuan Zhou, Yujun Cai, Yuxuan Liang, Dayiheng Liu, Baosong Yang, Juncheng Liu, and Bryan Hooi. Should we rely on entity mentions for relation extraction? debiasing relation extraction with counterfactual analysis. *arXiv preprint arXiv:2205.03784*, 2022.
- [149] Fei Wang, James Y Huang, Tianyi Yan, Wenxuan Zhou, and Muhao Chen. Robust natural language understanding with residual attention debiasing. *arXiv preprint arXiv:2305.17627*, 2023.
- [150] Wenxuan Zhou, Sheng Zhang, Hoifung Poon, and Muhao Chen. Context-faithful prompting for large language models. *arXiv preprint arXiv:2303.11315*, 2023.

- [151] Wentao Ye, Mingfeng Ou, Tianyi Li, Xuetao Ma, Yifan Yanggong, Sai Wu, Jie Fu, Gang Chen, Junbo Zhao, et al. Assessing Hidden Risks of LLMs: An Empirical Study on Robustness, Consistency, and Credibility. *arXiv preprint arXiv:2305.10235*, 2023.
- [152] Ching-Yun Ko, Pin-Yu Chen, Payel Das, Yung-Sung Chuang, and Luca Daniel. On Robustness-Accuracy Characterization of Large Language Models using Synthetic Datasets. In *Workshop on Efficient Systems for Foundation Models@ ICML2023*, 2023.
- [153] Kaijie Zhu, Jindong Wang, Jiaheng Zhou, Zichen Wang, Hao Chen, Yidong Wang, Linyi Yang, Wei Ye, Neil Zhenqiang Gong, Yue Zhang, et al. PromptBench: Towards Evaluating the Robustness of Large Language Models on Adversarial Prompts. *arXiv preprint arXiv:2306.04528*, 2023.
- [154] Boxin Wang, Chejian Xu, Shuohang Wang, Zhe Gan, Yu Cheng, Jianfeng Gao, Ahmed Hassan Awadallah, and Bo Li. Adversarial GLUE: A Multi-Task Benchmark for Robustness Evaluation of Language Models. In *35th Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.
- [155] Andy Zou, Zifan Wang, J Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023.
- [156] Jiong Xiao Wang, Zichen Liu, Keun Hee Park, Muhao Chen, and Chaowei Xiao. Adversarial Demonstration Attacks on Large Language Models. *arXiv preprint arXiv:2305.14950*, 2023.
- [157] Sicheng Zhu, Ruiyi Zhang, Bang An, Gang Wu, Joe Barrow, Zichao Wang, Furong Huang, Ani Nenkova, and Tong Sun. AutoDAN: Automatic and Interpretable Adversarial Attacks on Large Language Models. *arXiv preprint arXiv:2310.15140*, 2023.
- [158] Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How Does LLM Safety Training Fail? In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.
- [159] Fábio Perez and Ian Ribeiro. Ignore Previous Prompt: Attack Techniques For Language Models. In *NeurIPS ML Safety Workshop*, 2022.
- [160] Abhinav Rao, Sachin Vashistha, Atharva Naik, Somak Aditya, and Monojit Choudhury. Tricking LLMs into Disobedience: Understanding, Analyzing, and Preventing Jailbreaks. *arXiv preprint arXiv:2305.14965*, 2023.

- [161] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, 2004.
- [162] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. BERTScore: Evaluating Text Generation with BERT. In *International Conference on Learning Representations*, 2019.
- [163] Kaiqiang Song. Towards improving the robustness of neural abstractive summarization. *Doctoral Dissertation, University of Central Florida*, 2021.
- [164] Lisa Fan, Dong Yu, and Lu Wang. Robust neural abstractive summarization systems and evaluation against adversarial information. *arXiv preprint arXiv:1810.06065*, 2018.
- [165] Xiuying Chen, Mingzhe Li, Xin Gao, and Xiangliang Zhang. Towards improving faithfulness in abstractive summarization. *Advances in Neural Information Processing Systems*, 35:24516–24528, 2022.
- [166] Yiran Chen, Pengfei Liu, Ming Zhong, Zi-Yi Dou, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. Cdevalsumm: An empirical study of cross-dataset evaluation for neural summarization systems. *arXiv preprint arXiv:2010.05139*, 2020.
- [167] Klaus-Michael Lux. On the factual correctness and robustness of deep abstractive text summarization. 2020.
- [168] Kundan Krishna, Yao Zhao, Jie Ren, Balaji Lakshminarayanan, Jiaming Luo, Mohammad Saleh, and Peter J Liu. Improving the robustness of summarization models by detecting and removing input noise. *arXiv preprint arXiv:2212.09928*, 2022.
- [169] Yang Liu and Mirella Lapata. Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, page 3721. Association for Computational Linguistics, 2019.
- [170] Matt Grenander, Yue Dong, Jackie Chi Kit Cheung, and Annie Louis. Countering the effects of lead bias in news summarization via multi-stage training and auxiliary losses. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6019–6024, 2019.

- [171] Linzi Xing, Wen Xiao, and Giuseppe Carenini. Demoting the lead bias in news summarization via alternating adversarial learning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 948–954, 2021.
- [172] Diana McCarthy and Roberto Navigli. Semeval-2007 task 10: English lexical substitution task. In *Proceedings of the fourth international workshop on semantic evaluations (SemEval-2007)*, pages 48–53, 2007.
- [173] Leonid Nisonovich Vaserstein. Markov processes over denumerable products of spaces, describing large systems of automata. *Problemy Peredachi Informatsii*, 5(3):64–72, 1969.
- [174] Byeongchang Kim, Hyunwoo Kim, and Gunhee Kim. Abstractive Summarization of Reddit Posts with Multi-level Memory Networks. In *Proceedings of NAACL-HLT*, pages 2519–2531, 2019.
- [175] Chao Zhao, Faeze Brahman, Kaiqiang Song, Wenlin Yao, Dian Yu, and Snigdha Chaturvedi. NarraSum: A Large-Scale Dataset for Abstractive Narrative Summarization. *arXiv preprint arXiv:2212.01476*, 2022.
- [176] Abigail See, Peter J Liu, and Christopher D Manning. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*, 2017.
- [177] Shashi Narayan, Shay B Cohen, and Mirella Lapata. Don’t give me the details, just the summary! Topic-aware convolutional neural networks for extreme summarization. *arXiv preprint arXiv:1808.08745*, 2018.
- [178] Hadeer Ahmed, Issa Traore, and Sherif Saad. Detecting opinion spams and fake news using text classification. *Security and Privacy*, 1(1):e9, 2018.
- [179] Tianyi Zhang, Faisal Ladhak, Esin Durmus, Percy Liang, Kathleen McKeown, and Tatsunori B Hashimoto. Benchmarking large language models for news summarization. *Transactions of the Association for Computational Linguistics*, 12:39–57, 2024.
- [180] Alexander R Fabbri, Wojciech Kryściński, Bryan McCann, Caiming Xiong, Richard Socher, and Dragomir Radev. Summeval: Re-evaluating summarization evaluation. *Transactions of the Association for Computational Linguistics*, 9:391–409, 2021.

- [181] Yanran Chen and Steffen Eger. Menli: Robust evaluation metrics from natural language inference. *Transactions of the Association for Computational Linguistics*, 11:804–825, 2023.
- [182] Junjie Ye, Zhengyin Du, Xuesong Yao, Weijian Lin, Yufei Xu, Zehui Chen, Zaiyuan Wang, Sining Zhu, Zhiheng Xi, Siyu Yuan, et al. Toolhop: A query-driven benchmark for evaluating large language models in multi-hop tool use. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2995–3021, 2025.
- [183] Kinjal Basu, Ibrahim Abdelaziz, Kiran Kate, Mayank Agarwal, Maxwell Crouse, Yara Rizk, Kelsey Bradford, Asim Munawar, Sadhana Kumaravel, Saurabh Goyal, et al. Nestful: A benchmark for evaluating llms on nested sequences of api calls. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 33526–33535, 2025.
- [184] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022.
- [185] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.
- [186] Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning acting and planning in language models. *arXiv preprint arXiv:2310.04406*, 2023.
- [187] Shuo Yang, Soyeon Caren Han, Yihao Ding, Shuhe Wang, and Eduard Hoy. Tooltree: Efficient llm agent tool planning via dual-feedback monte carlo tree search and bidirectional pruning, 2026.
- [188] Wenjie Chen, Wenbin Li, Di Yao, Xuying Meng, Chang Gong, and Jingping Bi. Gtool: Graph enhanced tool planning with large language model. *arXiv preprint arXiv:2508.12725*, 2025.
- [189] Xixi Wu, Yifei Shen, Caihua Shan, Kaitao Song, Siwei Wang, Bohang Zhang, Jiarui Feng, Hong Cheng, Wei Chen, Yun Xiong, et al. Can graph learning improve planning in llm-based agents? *Advances in Neural Information Processing Systems*, 37:5338–5383, 2024.
- [190] Qinzhuo Wu, Wei Liu, Jian Luan, and Bin Wang. Toolplanner: A tool augmented llm for multi granularity instructions with path planning and feedback. In *Proceedings of the 2024*

- Conference on Empirical Methods in Natural Language Processing*, pages 18315–18339, 2024.
- [191] Yuanqing Yu, Zhefan Wang, Weizhi Ma, Shuai Wang, Chuhan Wu, Zhiqiang Guo, and Min Zhang. Steptool: Enhancing multi-step tool usage in llms via step-grained reinforcement learning. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pages 3952–3962, 2025.
 - [192] Zhiwei Li, Yong Hu, and Wenqing Wang. Encouraging good processes without the need for good answers: Reinforcement learning for llm agent planning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1654–1666, 2025.
 - [193] Shirley Wu, Shiyu Zhao, Qian Huang, Kexin Huang, Michihiro Yasunaga, Kaidi Cao, Vassilis N Ioannidis, Karthik Subbian, Jure Leskovec, and James Zou. Avatar: Optimizing llm agents for tool usage via contrastive reasoning. *Advances in Neural Information Processing Systems*, 37:25981–26010, 2024.
 - [194] Shiting Huang, Zhen Fang, Zehui Chen, Siyu Yuan, Junjie Ye, Yu Zeng, Lin Chen, Qi Mao, and Feng Zhao. Critictool: Evaluating self-critique capabilities of large language models in tool-calling error scenarios. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 26683–26692, 2025.
 - [195] Shirley Kokane, Ming Zhu, Tulika Awalganekar, Jianguo Zhang, Thai Hoang, Akshara Prabhakar, Zuxin Liu, Tian Lan, Liangwei Yang, Juntao Tan, et al. Toolscan: A benchmark for characterizing errors in tool-use llms. *arXiv preprint arXiv:2411.13547*, 2024.
 - [196] Eduardo Treviño, Hugo Contant, James Ngai, Graham Neubig, and Zora Zhiruo Wang. Benchmarking failures in tool-augmented language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2916–2934, 2025.
 - [197] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021.
 - [198] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let’s verify step by step. In *The*

Twelfth International Conference on Learning Representations, 2023.

- [199] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.
- [200] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*, 2023.
- [201] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.
- [202] Weizhou Shen, Chenliang Li, Hongzhan Chen, Ming Yan, Xiaojun Quan, Hehong Chen, Ji Zhang, and Fei Huang. Small LLMs are weak tool learners: A multi-LLM agent. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 16658–16680, Miami, Florida, USA, November 2024. Association for Computational Linguistics.