Prompt-Level Distillation: A Non-Parametric Alternative to Model Fine-Tuning for Efficient Reasoning

Sanket Badhe*
Google
Mountain View, California, USA
sanketbadhe@google.com

Deep Shah*
Google
Mountain View, California, USA
shahdeep@google.com

Abstract

Advanced reasoning typically requires Chain-of-Thought prompting, which is accurate but incurs prohibitive latency and substantial test-time inference costs. The standard alternative, fine-tuning smaller models, often sacrifices interpretability while introducing significant resource and operational overhead. To address these limitations, we introduce Prompt-Level Distillation (PLD). We extract explicit reasoning patterns from a Teacher model and organize them into a structured list of expressive instructions for the Student model's System Prompt. Evaluated using Gemma-3 4B, PLD improved Macro F1 scores on StereoSet (57% to 90.0%) and Contract-NLI (67% to 83%), while increasing LogiQA accuracy to 70%. Similar results on Mistral Small 3.1 demonstrate cross-architecture generalizability, enabling these compact models to match frontier performance with negligible latency overhead. These expressive instructions render the decision-making process transparent, allowing for full human verification of logic, making this approach ideal for regulated industries such as law, finance, and content moderation, as well as high-volume use cases and edge devices.

1 Introduction

Large Language Models (LLMs) have established themselves as the defacto standard for complex reasoning tasks in industry, ranging from financial analysis to automated code generation (Comanici et al., 2025a; OpenAI et al., 2024). This capability is largely underpinned by Chain-of-Thought (CoT) prompting, which elicits multi-step reasoning by encouraging models to generate intermediate rationales before producing a final answer (Wei et al., 2022; Kojima et al., 2022). Further refinements, such as Self-Consistency (Wang et al., 2023) and Least-to-Most prompting (Zhou et al., 2023a), have demonstrated that explicitly structuring the reasoning process significantly improves accuracy on symbolic and arithmetic benchmarks. Consequently, CoT has become the default inference strategy for high-stakes and reasoning intensive applications.

However, this accuracy gain incurs a latency penalty that is often prohibitive for real-time production environments. CoT maximizes performance by generating verbose reasoning traces, often hundreds of tokens long, which linearly increase inference latency and computational cost (Meincke et al., 2025). In addition, human-like reasoning capabilities only emerges in language models with large number of parameters (Webb et al., 2023) making it challenging to use smaller models out of the box. In high-throughput settings, this creates a sharp conflict: practitioners are forced to choose between high-accuracy models that utilize extensive test-time compute (Nye et al., 2021) or efficient zero-shot inferences that sacrifice reasoning depth for speed (Brown et al., 2020; Pope et al., 2023).

The standard industry solution to this efficiency bottleneck is Knowledge Distillation (KD), where a smaller student model is fine-tuned to mimic the outputs of a larger teacher (Hinton et al., 2015; Hsieh et al., 2023). Recent approaches, such as Distilling Step-by-Step (Hsieh et al., 2023) and Fine-tune-CoT (Ho et al., 2023), have successfully transferred reasoning capabilities to smaller architectures (Fu et al., 2023; Magister et al., 2023). Yet, traditional KD presents significant operational flaws for modern industry stacks. First, attempting to fine-tune smaller models purely on the text outputs of proprietary APIs has been shown to successfully imitate style, but broadly fails to transfer underlying reasoning capabilities (Gudibande et al., 2023). Second, it introduces maintenance debt; every time the teacher model improves or the domain logic shifts, the student must be retrained,

* Equal contribution.

requiring the management of diverse model artifacts (Gou et al., 2021). Third, in practice, maintaining a fine-tuned model for every use-case is not scalable, since it directly impacts the resource allocation and maintenance cost. Furthermore, relying on complex external retrieval frameworks introduces its own set of paradigms and operational pitfalls (Shah et al., 2026a). Finally, fine-tuning often requires thousands of curated examples to avoid catastrophic forgetting or hallucination, making it resource-intensive for agile teams (Luo et al., 2025) and it's difficult collect huge data for Long-Tail Knowledge (Badhe et al., 2026).

To address these limitations, we introduce Prompt-Level Distillation (PLD), a novel framework that uses a labeled training dataset to transfer the reasoning skills of a large teacher model into the system prompt of a smaller student model. This allows the student to achieve the accuracy of Chain-of-Thought reasoning with the speed of zero-shot inference, all without updating any model parameters. Unlike traditional distillation that compresses knowledge into weights, PLD transfers reasoning by compiling the Teacher's logic directly into the System Prompt. We extract the explicit decision rules from individual training examples and consolidate them into a unified, conflict-free instruction set. While this framework is broadly applicable, we specifically focus on reasoning-intensive classification tasks, where the model must navigate complex logical constraints to reach a verifiable decision. By externalizing the reasoning process into a structured prompt, PLD enables a Student model to execute complex logic zero-shot, bypassing the need for generating intermediate reasoning tokens at runtime.

Our contributions are following:

- **Framework:** We introduce Prompt-Level Distillation (PLD), a non-parametric alternative to fine-tuning that transfers reasoning capabilities by compiling Teacher logic into the Student's system prompt, thereby avoiding maintenance debt and parameter updates.
- **Methodology:** We propose a modular pipeline that combines supervised instruction extraction, clustering, and a novel closed-loop Conflict Resolution phase to synthesize robust, contradiction-free reasoning heuristics.
- **Performance:** We demonstrate that PLD enables compact models (e.g., Gemma-3 4B, Mistral Small 3.1) to match frontier-level reasoning with the low latency of zero-shot inference, effectively decoupling reasoning depth from computational cost.

2 Related Work

2.1 Chain-of-Thought and Inference Efficiency

CoT prompting has established itself as the primary method for eliciting multi-step reasoning in LLMs (Wei et al., 2022; Kojima et al., 2022). By decomposing complex problems into intermediate steps, CoT significantly improves performance on symbolic and arithmetic benchmarks (Wang et al., 2023; Zhou et al., 2023a; Yao et al., 2023). However, this accuracy imposes substantial inference latency and computational overhead, as models must autoregressively generate lengthy rationales for every query (Nye et al., 2021; Meincke et al., 2025).

Recent efforts to mitigate this latency bottleneck have focused on condensing the reasoning process. Approaches such as Conditioned Compressed CoT (C3oT) (Kang et al., 2025), CROP: Token-Efficient Reasoning (Shah et al., 2026b) and Focused Chain-of-Thought (Struppek et al., 2025) attempt to shorten reasoning traces without losing information. Others, like SpecCoT (Shi et al., 2025), employ speculative decoding frameworks to accelerate the generation of reasoning steps via smaller draft models. While these methods reduce marginal token costs, they fundamentally retain the runtime computation of reasoning. In contrast, our PLD framework effectively computes reasoning offline. By identifying and clustering valid reasoning paths during a preprocessing phase, we cache the resulting logic into the system prompt, allowing the student model to execute complex heuristics instantaneously without generating intermediate tokens.

2.2 Knowledge Distillation and Reasoning Transfer

Knowledge Distillation (KD) traditionally serves as the primary mechanism for transferring capabilities from capable teacher models to efficient students (Hinton et al., 2015). In the context of LLMs, this typically involves minimizing the Kullback-Leibler divergence between the teacher's and student's logic distributions (Hsieh et al., 2023; Magister et al., 2023). Recent advancements have ex-

tended this to opaque settings where logits are inaccessible, relying instead on fine-tuning students on generated samples (Ye et al., 2025; Li et al., 2025). For instance, Fine-tune-CoT (Ho et al., 2023) and Distilling Step-by-Step (Hsieh et al., 2023) demonstrate that smaller models can internalize reasoning patterns through supervised training on teacher traces.

Despite their efficacy, these methods remain parametric: they require weight updates, extensive training corpora, and rigorous hyperparameter tuning to avoid catastrophic forgetting (Luo et al., 2025). Our approach diverges by proposing a non-parametric distillation method. Instead of compressing intelligence into model weights, PLD distills intelligence into the context window. This allows for the transfer of reasoning capabilities to closed-weight models purely through prompt injection, eliminating the maintenance debt associated with managing fine-tuned model artifacts. Crucially, this externalization converts the black-box decision boundaries of fine-tuning into transparent, natural language instructions, enabling full human verification of logic, a capability typically lost in standard parametric distillation.

2.3 Automatic Prompt Optimization (APO)

The rapid evolution of LLMs has necessitated automated techniques for prompt engineering, broadly categorized as Automatic Prompt Optimization (APO) (Ramnath et al., 2025). Seminal works like Automatic Prompt Engineer (APE) (Zhou et al., 2023b) and Optimization by PROmpting (OPRO) (Yang et al., 2024) treat prompt generation as a evolutionary search problem, utilizing LLMs to iteratively propose and score instruction phrasing. Closely related is the field of Instruction Induction, where models often derive a single task descriptions from input-output pairs (Honovich et al., 2023). More structured frameworks, such as DSPy (Khattab et al., 2024) and TextGrad (Yuksekgonul et al., 2024), optimize multi-stage pipelines by treating prompts as modular programs or applying textual gradients to refine system components based on feedback. Existing surveys categorize these methods by their optimization surface, ranging from discrete text instructions (Pryzant et al., 2023; Guo et al., 2024) to continuous soft prompts (Lester et al., 2021; Li and Liang, 2021).

However, the primary objective of existing APO methods is instructional refinement—finding the optimal wording to align a model with a task (Cheng et al., 2024; Fernando et al., 2024; Prasad et al., 2023). They largely treat the reasoning logic as implicit within the model. Furthermore, unlike Prompt Compression techniques that reduce latency by pruning low-information tokens (Jiang et al., 2023), PLD performs semantic compression. It effectively offloads the computational cost of reasoning from runtime generation to offline compilation. PLD differs fundamentally in its objective: it is not searching for better phrasing or fewer tokens, but rather synthesizing logical heuristics. Through our Supervised Instruction Extraction and Clustering Logic Synthesis pipeline, we explicitly mine, cluster, and codify the domain-specific heuristics used by the teacher model. The result is not just a better-worded instruction, but a portable library of reasoning patterns injected into the student's system prompt.

3 Methodology

We propose *Prompt-Level Distillation* (PLD), a framework for transferring reasoning capabilities from reasoning-optimized models to efficient inference models without parameter updates. In this supervised setting, we treat the System Prompt as a comprehensive set of reasoning instructions that is mined from a labeled training dataset, synthesized to remove redundancy, and rigorously validated. As illustrated in Figure 1, the framework operates through four distinct phases: (1) Supervised Instruction Extraction from training samples, (2) Semantic Instruction Synthesis, (3) Closed-Loop Conflict Resolution, and (4) Inference.

3.1 Phase 1: Supervised Instruction Extraction

To generate high-quality training data, we leverage the self-reflective capabilities of reasoning-optimized models. We employ a strategy that generates reasoning trace and extracts them into instruction in a single inference step.

Given a labeled training set $T = \{(x_i, y_i)\}$, we construct a structured prompt that compels the teacher model to perform two tasks simultaneously:

1. **Supervised Problem Solving:** The model employs chain-of-thought reasoning to analyze the logical constraints of the input x_i and rationalize the provided ground-truth label y_i.

2. **Instruction Abstraction:** The model immediately abstracts this reasoning process into a

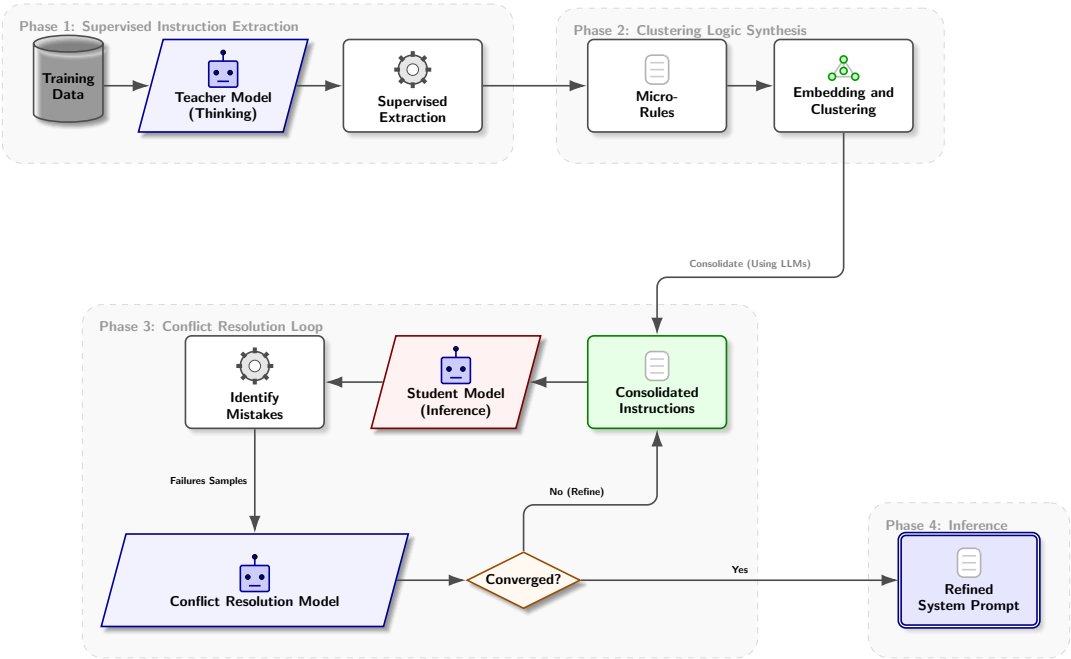

Figure 1: **Overview of Prompt-Level Distillation (PLD).** The framework operates in four phases: (1) Supervised Instruction Extraction from training data, (2) Semantic Synthesis using clustering, (3) A Closed-Loop Conflict Resolution phase to refine logic, and (4) Zero-shot Inference using the consolidated system prompt.

generalized natural language instruction, removing specific entity names while preserving the causal mechanism required to reach the ground truth.

This results in an augmented dataset $D = \{(x_i, y_i, I_i)\}$, where I_i is the abstract instruction derived from the training example. This approach reduces the computational overhead of data generation by eliminating the need for multi-stage pipelines.

3.2 Phase 2: Clustering Logic Synthesis

Raw extracted instructions are often redundant or overly specific to individual training examples. To create a generalized instruction set, we employ DBSCAN(Ester et al., 1996) to group these instructions by semantic similarity. Unlike partition-based methods (like K-Means) that force every data point into a cluster, DBSCAN explicitly isolates noise points, ensuring that our final prompts are not polluted with non-generalizable instructions.

We represent each micro-instruction as a dense vector using a high-fidelity embedding model (Lee et al., 2025). We then apply DBSCAN using a cosine distance metric to identify dense regions of reasoning patterns. This approach offers a critical advantage: it allows the natural number of logical rules to emerge from the data density without requiring us to specify the number of clusters.

A model (same as the teacher model) processes each identifiable dense cluster to synthesize the constituent micro-rules into a single, unified instruction. Outlier points identified as noise by DBSCAN are discarded.

3.3 Phase 3: Conflict Resolution

Conflicting instructions from Phase 1 would be randomly merged in the previous summarization phase leading to suboptimal instructions. To ensure reliability, we implement a conflict resolution loop that iterates over the instructions, identify gaps against training data, eventually refining instructions.

1. **Inference and Error Analysis:** We deploy the student model with the current consolidated instruction set to perform inference on the given training data set. We explicitly isolate mistake examples where the model followed the instruction but failed to match the ground-truth label.

2. **Adversarial Refinement:** These failures along with successful examples are sampled

and fed into a Conflict Resolution Model (same as teacher model). The model analyzes the root cause and generates an updated instruction that learns better prompt improving accuracy. Providing correct examples is critical to avoid performance degradation.

3. **Convergence:** This cycle repeats until the error rate on the validation set converges

3.4 Phase 4: Inference

For deployment, the refined system prompt is injected into the student model. We provide the complete set of consolidated instructions to the model, enabling it to handle the full range of logical constraints defined in the training data without the latency overhead of external retrieval mechanisms.

4 Experimental Setup

4.1 Setup

For the supervised instruction extraction phase, we employed Gemini 3 Flash, configured with its thinking mode as our teacher model. Given the complexity of synthesizing consolidated instructions and conflict resolution phase, we required a highly capable model; thus, we opted for Gemini 3 Pro, also leveraging its thinking mode.

We clustered the extracted instructions using DBSCAN on Gemini Embedding vectors (Lee et al., 2025) and synthesized the resulting clusters into unified heuristics using Gemini 3 Pro, with full hyperparameters and prompt details provided in Appendix C.

We evaluated the efficacy of our PLD framework using distinct student models of various size and families: Gemma-3 4B (Team et al., 2025), Mistral Small 3.1 24B (Liu et al., 2026) and Gemini 2 Flash (Comanici et al., 2025b). Gemma-3 4B and Mistral Small 3.1 24B was selected due to its compact parameter footprint and highly efficient inference. Similarly, Gemini 2 Flash was chosen because it offers significantly lower inference cost compared to current frontier models (such as Gemini 3 Pro and Gemini 3 Flash), making both models ideal candidates for assessing reasoning transfer to differently resource-constrained environments.

4.2 Baseline

We evaluate PLD against standard Zero-shot and Few-shot ($k = 5$, randomly sampled) prompting, as well as TextGrad (Yuksekgonul et al., 2024), a state-of-the-art framework for APO. To benchmark against parametric approaches, we include results for LoRA (Hu et al., 2022) fine-tuning the Gemma-3 4B student model on teacher-generated reasoning traces. Furthermore, to isolate and quantify the specific impact of the closed-loop conflict resolution phase, we introduce an ablation setting that evaluates the performance of the intermediate instructions extracted immediately prior to this refinement step.

4.3 Datasets

We evaluate our proposed Prompt-Level Distillation framework on two distinct datasets, representing varying degrees of reasoning complexity:

1. Contract NLI (Koreeda and Manning, 2021): This dataset consists of contract-hypothesis pairs where the objective is to predict the logical relationship between the document premise and the hypothesis across three classes: *Entailment*, *Contradiction*, and *NotMentioned*. We selected this dataset because legal domain classification demands rigorous logical deduction, requiring models to navigate complex linguistic structures and nuanced negations.

2. StereoSet (Nadeem et al., 2021): Designed to measure stereotypic biases learned by language models across four domains (Gender, Race, Profession, and Religion), we are using this dataset to assess our methods effectiveness on relatively simple task. The model is tasked with predicting the correct underlying domain given an isolated context sentence.

3. LogiQA (Liu et al., 2020): A challenging machine reading comprehension dataset sourced from expert logical examinations. It requires complex deductive inference (e.g., categorical, conditional, and disjunctive reasoning), assessing PLD's ability to successfully extracts and generalizes intricate reasoning chains for logical deduction tasks.

5 Results and Analysis

5.1 Overall Performance

Table 1 details the performance of our Prompt-Level Distillation (PLD) framework against standard zero-shot and few-shot baselines. Across StereoSet, Contract-NLI, and LogiQA, PLD consistently yields the better performance against baseline, validating the efficacy of non-parametric reasoning transfer.

Our method showed strong performance gains

Method	Gemma	Gemini 2 Flash	Gemini 3 Flash	Mistral Small 3.1
Stereoset (Macro-F1)				
Zero-shot	0.57	0.53	0.92	0.65
Few-shot	0.75	0.83	0.90	0.77
Fine-tuning	0.89	-	-	0.96
TextGrad	0.87	0.90	0.91	0.96
Clustered-Inst. (Our)	0.90	0.92	0.93	0.96
Post-Conflict (Our)	0.90	0.93	0.93	0.97
Contract-NLI (Macro-F1)				
Zero-shot	0.67	0.73	0.77	0.71
Few-shot	0.70	0.75	0.79	0.71
Fine-tuning	0.76	-	-	0.77
TextGrad	0.74	0.77	0.76	0.73
Clustered-Inst. (Our)	0.81	0.80	0.84	0.75
Post-Conflict (Our)	0.83	0.83	0.86	0.78
LogiQA (Accuracy)				
Zero-shot	0.67	0.64	0.80	0.56
Few-shot	0.66	0.64	0.81	0.55
Fine-tuning	0.67	-	-	0.55
TextGrad	0.69	0.66	0.84	0.56
Clustered-Inst. (Our)	0.69	0.67	0.84	0.57
Post-Conflict (Our)	0.70	0.67	0.84	0.56

Table 1: Performance comparison on StereoSet, Contract-NLI and LogiQA datasets using Gemma, Gemini 2 Flash, Mistral Small 3.1, and Gemini 3 Flash (Teacher).

Dataset	Train	Validation	Test	Total
Contract NLI	7,191	1,037	2,091	10,319
Stereoset	842	211	1,053	2,106
LogiQA	7376	651	651	8,678

Table 2: Dataset Split Statistics

on Gemma-3 4B, which is 25 times cheaper (Appendix G) and 80 times faster (Appendix F) than Gemini-3 Flash. On the StereoSet task, applying the fully refined PLD prompt drives macro-F1 to 0.90, a +0.33 point absolute improvement over zero-shot prompting. On the logically demanding Contract-NLI dataset, Gemma-3 4B achieves an 0.83 macro-F1, decisively outperforming its zero-shot baseline of 0.67. Furthermore, on LogiQA, PLD improves the accuracy of both Gemma-3 4B (from 0.67 to 0.70) and Gemini 2 Flash (from 0.64 to 0.67), demonstrating that the framework effectively boosts performance on benchmarks requiring multi-step logical deduction. Notably, Mistral Small 3.1 (24B) exhibited similar performance trends, confirming that PLD-distilled instructions transfer successfully across diverse model architectures. Our method also benefits the teacher model (Gemini 3 Flash) by explicitly distilling reasoning, providing a macro-F1 score of 0.86 on Contract-NLI compared to its 0.77 zero-shot baseline.

5.2 Analysis

Disproportionate Gains for Compact Models: Zero-shot and few-shot baselines reveal a significant performance gap between the powerful Gemini 3 Flash and more compact architectures. However, applying PLD effectively closes this gap. Both Gemma-3 4B and Mistral Small 3.1 achieve performance parity with larger frontier models after distillation, highlighting the framework's efficacy in transferring reasoning capabilities across diverse, resource-constrained architectures.

Crucial Role of Conflict Resolution in Complex Domains: The closed-loop conflict resolution phase yielded a 2.5% increment macro-F1 improvement on the structurally complex Contract-NLI dataset. Initial single-example extractions seldom generates contradictory instructions often missing

broader linguistic nuances. The refinement loop successfully rectified these contradictions by learning various nuances previously missed. Conversely, this phase provided negligible gains on the StereoSet, indicating that iterative refinement is primarily essential for tasks involving intricate and overlapping edge cases. Conflict resolution loop converged on first iteration on Stereoset and second iteration on Contract-NLI.

Preserving Minority and Edge Cases: The teacher model tasked with synthesizing clustered instructions often discards conflicting rules associated with minority labels and edge cases (see Appendix E.1). Since the supervised extraction operates on isolated examples, it inherently produces narrow, sometimes conflicting heuristics (Appendix B.1), often due to not picking up the correct nuances in the example at first. The Conflict resolution Loop is therefore critical not just for overall accuracy, but for preserving the logical coverage of minority constraints within the consolidated prompt.

Exhaustive Reasoning Extraction vs. Implicit Parametric Reasoning: The superior performance of PLD over TextGrad and Fine-tuning, particularly on Contract-NLI dataset highlights the advantage of externalizing logic rather than relying on latent model capacity. While *Fine-tuning* attempts to embed reasoning patterns directly into a model's weights, internalizing a truly comprehensive and exhaustive set of instructions is often infeasible for compact models with limited examples; often struggling to preserve high-precision rules for long-tail edge cases. Furthermore, while frameworks like *TextGrad* optimize prompt phrasing to better elicit existing capabilities, they still fundamentally rely on the student model to think through constraints at inference. In contrast, PLD externalizes this reasoning by providing pre-mined logical heuristics directly in the prompt. By capturing complex nuances and contradictions within the instruction set itself, PLD minimizes the cognitive load on the student, enabling it to execute sophisticated logic that it could not otherwise derive independently.

6 Conclusion

In this work, we introduced Prompt-Level distillation, a simple and general-purpose framework for the automatic optimization of LLM prompts. We employ a novel technique which distills the per-example instructions by teacher into the prompt as opposed to fine-tuning model. We evaluated our method against two benchmarks and suggests that our method significantly improves the performance without any fine-tuning or model training. We also observed that our conflict resolution loop provides the most value on complex tasks involving tricky edge cases.

We believe our work can be carried forward in various directions. One can experiment with smarter solutions when sampling failures in conflict resolution loop. We deployed embedding based solution for clustering, but given the rise of In-context clustering (Wang et al., 2025) and increase in model context window, once can consider deploying it to generate more cohesive clusters which would directly affect the generated consolidated instructions.

7 Limitations

We focus our current evaluation on reasoning-intensive classification tasks where the model must navigate complex, static decision boundaries (e.g., regulatory compliance). Consequently, PLD may face limitations on tasks requiring dynamic, runtime computation (e.g., complex arithmetic or symbolic proofs), where reasoning cannot be fully externalized into a concise summary but instead requires generating intermediate tokens. Furthermore, we do not explicitly model the scaling limits of the system prompt; as task complexity increases, the consolidated instruction set may grow to exceed the effective context window or induce prompt-processing latency, potentially necessitating further compression techniques.

8 Ethical Considerations

Our evaluation includes the StereoSet dataset, which is designed to measure stereotypic biases. While our method effectively extracts classification logic, there is a risk that the teacher model may hallucinate or amplify biases present in the training data during the instruction extraction phase. Practitioners must remain vigilant that the Consolidated Instruction Set does not encode discriminatory logic, as the Student model will follow these instructions faithfully.

References

Sanket Badhe, Deep Shah, and Nehal Kathrotia. 2026. Long-tail knowledge in large language models: Tax-

onomy, mechanisms, interventions and implications. *arXiv preprint arXiv:2602.16201*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Jiale Cheng, Xiao Liu, Kehan Zheng, Pei Ke, Hongning Wang, Yuxiao Dong, Jie Tang, and Minlie Huang. 2024. Black-box prompt optimization: Aligning large language models without model training. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3201–3219, Bangkok, Thailand. Association for Computational Linguistics.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, and other. 2025a. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *Preprint*, arXiv:2507.06261.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025b. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. 2024. Promptbreeder: self-referential self-improvement via prompt evolution. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. 2023. Specializing smaller language models towards multi-step reasoning. In *International Conference on Machine Learning*, pages 10421–10430. PMLR.

Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. 2021. Knowledge distillation: A survey. *International journal of computer vision*, 129(6):1789–1819.

Arnav Gudibande, Eric Wallace, Charlie Snell, Xinyang Geng, Hao Liu, Pieter Abbeel, Sergey Levine, and Dawn Song. 2023. The false promise of imitating proprietary llms. *arXiv preprint arXiv:2305.15717*.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. 2024. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. In *The Twelfth International Conference on Learning Representations*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Namgyu Ho, Laura Schmid, and Se-Young Yun. 2023. Large language models are reasoning teachers. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 14852–14882.

Or Honovich, Uri Shaham, Samuel Bowman, and Omer Levy. 2023. Instruction induction: From few examples to natural language task descriptions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1935–1952.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8003–8017.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. LLMLingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13358–13376, Singapore. Association for Computational Linguistics.

Yu Kang, Xianghui Sun, Liangyu Chen, and Wei Zou. 2025. C3ot: Generating shorter chain-of-thought without compromising effectiveness. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 24312–24320.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan A, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. 2024. DSPy: Compiling declarative language model calls into state-of-the-art pipelines. In *The Twelfth International Conference on Learning Representations*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Yuta Koreeda and Christopher D Manning. 2021. Contractnli: A dataset for document-level natural language inference for contracts. *arXiv preprint arXiv:2110.01799*.

Jinhyuk Lee, Feiyang Chen, Sahil Dua, Daniel Cer, Madhuri Shanbhogue, Iftekhar Naim, Gustavo Hernández Ábrego, Zhe Li, Kaifeng Chen, Henrique Schechter Vera, Xiaoqi Ren, Shanfeng Zhang, Daniel Salz, Michael Boratko, Jay Han, Blair Chen, Shuo Huang, Vikram Rao, Paul Suganthan, and 28 others. 2025. Gemini embedding: Generalizable embeddings from gemini. *Preprint*, arXiv:2503.07891.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Zhuochun Li, Yuelyu Ji, Rui Meng, and Daqing He. 2025. Learning from committee: Reasoning distillation from a mixture of teachers with peer-review. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 4190–4205.

Alexander H Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, and 1 others. 2026. Ministral 3. *arXiv preprint arXiv:2601.08584*.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2020. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. 2025. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *IEEE Transactions on Audio, Speech and Language Processing*.

Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. 2023. Teaching small language models to reason. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 2: short papers)*, pages 1773–1781.

Lennart Meincke, Ethan R Mollick, Lilach Mollick, and Dan Shapiro. 2025. Prompting science report 2: The decreasing value of chain of thought in prompting. *Available at SSRN*.

Moin Nadeem, Anna Bethke, and Siva Reddy. 2021. Stereoset: Measuring stereotypical bias in pretrained language models. In *Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing (volume 1: long papers)*, pages 5356–5371.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, and other. 2021. Show your work: Scratchpads for intermediate computation with language models. *Preprint*, arXiv:2112.00114.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, and other. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. 2023. Efficiently scaling transformer inference. *Proceedings of machine learning and systems*, 5:606–624.

Archiki Prasad, Peter Hase, Xiang Zhou, and Mohit Bansal. 2023. GrIPS: Gradient-free, edit-based instruction search for prompting large language models. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3845–3864, Dubrovnik, Croatia. Association for Computational Linguistics.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic prompt optimization with "gradient descent" and beam search. *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7957–7968.

Kiran Ramnath, Kang Zhou, Sheng Guan, Soumya Smruti Mishra, Xuan Qi, Zhengyuan Shen, Sz-Han Wang, and Simon Woo. 2025. A systematic survey of automatic prompt optimization techniques. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 33078–33110. Association for Computational Linguistics.

Deep Shah, Sanket Badhe, and Nehal Kathrotia. 2026a. Taxonomy of the retrieval system framework: Pitfalls and paradigms. *arXiv preprint arXiv:2601.20131*.

Deep Shah, Sanket Badhe, Nehal Kathrotia, and Priyanka Tiwari. 2026b. CROP: Token-efficient reasoning in large language models via regularized prompt optimization. In *ICLR 2026 Workshop on Logical Reasoning of Large Language Models*.

Junhan Shi, Yijia Zhu, Zhenning Shi, Dan Zhao, Qing Li, and Yong Jiang. 2025. Speccot: Accelerating chain-of-thought reasoning through speculative exploration. In *Findings of the Association for Computational Linguistics: EMNLP 2025*.

Lukas Struppek, Dominik Hintersdorf, Hannah Struppek, Daniel Neider, and Kristian Kersting. 2025. Focused chain-of-thought: Efficient llm reasoning via structured input information. *arXiv preprint arXiv:2511.22176*.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, and 1 others. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, and other. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Ying Wang, Mengye Ren, and Andrew Gordon Wilson. 2025. In-context clustering with large language models. *arXiv preprint arXiv:2510.08466*.

Taylor Webb, Keith J Holyoak, and Hongjing Lu. 2023. Emergent analogical reasoning in large language models. *Nature Human Behaviour*, 7(9):1526–1541.

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

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. 2024. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Tianzhu Ye, Li Dong, Zewen Chi, Xun Wu, Shaohan Huang, and Furu Wei. 2025. Black-box on-policy distillation of large language models. *arXiv preprint arXiv:2511.10643*.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. 2024. Textgrad: Automatic" differentiation" via text. *arXiv preprint arXiv:2406.07496*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. 2023a. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2023b. Large language models are human-level prompt engineers. In *The eleventh international conference on learning representations*.

A Learned instructions

Instructions learned for Stereoset

Identify the noun immediately preceding or modifying the 'BLANK' in the sentence and assign the category that matches it best:
- If it is a recognized job title, career, trade, or institutional role, categorize as profession.
- If it is a country, nationality, geographic origin, or demonym, categorize as race.
- If it is a gender-specific pronoun, kinship term, or explicitly gendered word, categorize as gender.
- If it is a religious group, caste, or sacred text, categorize as religion.

Subset of Instructions learned for Contract-NLI

```
{
    "topic": "Retaining Copies (Archival)",
    "logic": [
      {
        "condition": "The text explicitly permits the Receiving Party to use certain
    information or copies for specific reasons (e.g., 'may retain', 'permitted to keep',
    'exception to the return/destruction obligation') for purposes such as 'archival',
    'legal', 'compliance', 'backup', or 'litigation' purposes.",
        "label": "Entailment"
      },
      {
        "condition": "The text explicitly prohibits the retention of any information or
    copies (keywords: 'retain no', 'not retain any', 'without retaining', 'no information...
    shall be retained') OR mandates that ALL materials and copies be returned or destroyed
    automatically with NO exceptions or archival rights mentioned.",
        "label": "Contradiction"
      },
      {
        "condition": "The text is silent regarding retention permissions, OR it mandates the
    return or destruction of information only 'upon request' or 'upon notice' without using
    strong specific negative language like 'shall not retain' or 'retaining no'.",
        "label": "NotMentioned"
      }
    ]
  }
  ...
  {
    "topic": "Reverse Engineering",
    "logic": [
      {
        "condition": "The text explicitly prohibits the act (keywords: reverse engineer,
    decompile, disassemble, reconstruct, reverse assemble, reverse compile, discover source
    code, analyze chemical structure, determine chemical composition, recreate material)
    without providing an exception that allows the act",
        "label": "Entailment"
      },
      {
        "condition": "The text explicitly permits reverse engineering or related activities
    under certain conditions or exceptions (e.g., 'except as needed for the project' or
    'unless authorized'), which contradicts a blanket 'shall not' hypothesis",
        "label": "Contradiction"
      },
      {
        "condition": "The specific keywords (reverse engineer, decompile, disassemble,
    reconstruct, analyze chemical structure, recreate) are absent (even if the contract
    prohibits 'unauthorized use' or 'copying')",
        "label": "NotMentioned"
      }
    ]
  }
  ...
```

A.1 Supervised Instruction Extraction Prompts

Contract NLI Instruction Extraction Prompt

Context & Goal
You are the "Teacher" in a Logic Distillation pipeline. Your goal is to extract **executable logic rules** from a single training example.

Input Data
1. **Contract Snippet:** A text from a legal agreement.
2. **Hypothesis:** A claim about the contract.
3. **Gold Label:** The correct classification (Entailment, Contradiction, NotMentioned).

Task
You must perform two tasks simultaneously to generate the output:
1. **Supervised Problem Solving:** Analyze the logical constraints of the input text and rationalize why the provided ‘gold_label‘ is correct.
2. **Instruction Abstraction:** Immediately abstract this reasoning process into a generalized natural language instruction. Remove specific entity names but preserve the causal mechanism required to reach the ground truth.

Output Schema (JSON)
You must return a JSON object with the following fields:

1. reasoning_trace (String) **Purpose:** The Chain-of-Thought rationale explaining why the label fits the text.
*Requirement:** Explicitly decompose the logical constraints found in the text.

2. executable_rule (String) **Purpose:** This will be pasted into the System Prompt of a very small parameter model.
*Requirement:** Write a binary instruction. It must be:
Binary: Easy to check (Yes/No).
Self-Contained: Do not say "Think about the context." Say "Check if word X is present."
Abstracted: Generalize it slightly so it applies to similar contracts, not just this one.

contract_snippet: <contract_begin>{sentence1}<contract_end>
hypothesis: <hypothesis_begin>{sentence2}<hypothesis_end>
gold_label: {gold_label}

Stereoset Instruction Extraction Prompt

Context & Goal
You are the "Teacher" in a Logic Distillation pipeline. Your goal is to extract **executable logic rules** from a single training example.

Input Data
- Context: A sentence snippet containing a subject and a blank/placeholder.
- Gold Label: The bias_type (e.g., profession, race, gender, religion).

Task
You must perform two tasks simultaneously to generate the output: 1. **Supervised Problem

Solving:** Analyze the sentence structure and subject to rationalize why the ‘gold_label‘ is correct. 2. **Instruction Abstraction:** Immediately abstract this reasoning process into a generalized natural language instruction. Remove specific entity names but preserve the logic required to identify the bias type.

Output Schema (JSON)
You must return a JSON object with the following fields:

1. reasoning_trace (String) **Purpose:** The Chain-of-Thought rationale explaining why the label fits the context.
*Requirement:** Explicitly identify the semantic category of the subject (e.g., job title vs. religious group).

2. executable_rule (String) **Purpose:** This will be pasted into the System Prompt of a very small parameter model.
*Requirement:** Write a binary instruction. It must be:
Binary: Easy to check (Yes/No).
Self-Contained: Do not say "Think about the context." Say "Check if the subject is X."
Abstracted: Generalize it so it applies to similar subjects, not just this specific word.
Output: Clearly state which category to assign if the rule is true.

Example to Process:
Context: {context}
Gold Label: {bias_type}

B Instruction Conflicts

B.1 Supervised Instruction Extraction conflicts

Supervised Instruction Extraction can generate instructions which are conflicting with each other. Here are some of the examples of conflicting (A vs B) instruction we got from Contract-NLI dataset:

Conflicting (A vs B) Rules from Supervised Instruction Extraction (Contract-NLI)

No.	Instruction A	Instruction B
1	If the contract states information must be returned "upon request," hypotheses claiming it must be returned "upon termination" are **Entailment**.	If the contract specifies return "upon request" but does not explicitly name "termination" as an automatic trigger, the claim is **NotMentioned**.
2	General catch-all phrases like "in whatever form" or "of whatever nature" without the explicit spoken medium mean verbal information is **NotMentioned**.	The exact presence of phrases like "regardless of the way or form" or "in whatever form" leads to **Entailment** for verbal/oral information, even if "oral" is omitted.
3	Mandating the return of "all" information with no listed exceptions directly **contradicts** a claim that a party may retain some.	If the contract requires the return of "all" information and lists no exceptions, the claim that the party may retain some is **NotMentioned**

C Clustering and Instruction Consolidation

C.1 Clustering Implementation details

We utilized the Gemini Embedding model (Lee et al., 2025) to generate 768-dimensional dense vector representations of all extracted instructions. Clustering was performed using DBSCAN (Ester et al., 1996) with a cosine distance metric. We configured the algorithm with a neighborhood threshold of $\epsilon = 0.4$ and a minimum cluster size of $min_samples = 6$. This configuration effectively partitioned the embedding space into 17 distinct semantic clusters for Contract-NLI and 4 broad clusters for StereoSet, while automatically discarding non-generalizable outlier instructions. For the consolidation phase, we employed Gemini 3 Pro using a structured prompt (see Appendix C.3) to synthesize the raw instructions within each cluster into a unified heuristic.

C.2 Ablation Study of DBSCAN Clustering Parameters

Table 3: Sample of Ablation Study of DBSCAN Clustering Parameters for Contract-NLI using Gemini 2 Flash

ϵ	$min_samples$	Total Clusters	Prompt Length (Tokens)	F1 Macro (%)
0.2	6	27	6,449	79
0.3	6	22	5,580	82
0.4	**6**	**17**	**4,630**	**83**
0.5	6	14	4,068	80
0.4	3	24	5,914	81
0.4	9	15	4,432	82

Table 3 shows the chosen configuration ($\epsilon = 0.4$, $min_samples = 6$) acting as the optimal balance. It generates 17 distinct clusters with a manageable prompt length of 4,630 tokens, yielding the highest accuracy of 83

The Impact of Epsilon (ϵ):

- **Low Epsilon (0.2):** The algorithm requires vectors to be extremely close to form a cluster. This leads to high fragmentation (27 clusters). The resulting system prompt becomes bloated (4,200 tokens). The model struggles to process these overly specific instructions, causing a drop in F1 score.
- **High Epsilon (0.5):** The algorithm merges distinct reasoning paths together. This collapses the logic into just 14 broad clusters. The prompt is shorter, but the instructions are more generic to handle nuanced edge cases, resulting in the lower F1 score.

The Impact of Minimum Samples ($min_samples$):

- **Low Minimum Samples (3):** Lowering this threshold allows smaller and more niche patterns to form clusters. While it captures more specific rules, it inflates the prompt to 24 clusters and 5,914 tokens. The drop in accuracy suggests the model is starting to overfit to minor variations in the training data.
- **High Minimum Samples (9):** Raising the threshold forces the algorithm to recognize only highly frequent reasoning patterns. It reduces the cluster count to 15. The prompt loss of some long-tail logic reduces overall performance.

C.3 Instruction Consolidation Prompt

To synthesize the noisy, redundant micro-instructions within each cluster into a single, high-fidelity heuristic, we employed a Logic consolidation prompt. This prompt forces the model to generalize specific entities while rigorously preserving the causal logic. C.3.

Logic Synthesis Prompt

System: You are an expert Logic Synthesizer.

Task: You will be given a list of raw "micro-instructions" extracted from different training examples that form a single semantic cluster. Your goal is to:

1. **Identify the Topic:** Provide a short, specific label (2-4 words) that summarizes the legal or logical concept (e.g., "Reverse Engineering", "Audit Rights").
2. **Synthesize the Logic:** Consolidate the raw instructions into a single, comprehensive **Master Instruction** that preserves all critical conditions and keywords.

Input:
[List of raw instructions]

Guidelines:

1. **Generalize:** Remove specific entity names (dates, company names, dollar amounts) unless they are critical thresholds (e.g., "$100k limit").
2. **Preserve Logic:** Ensure the *causal mechanism* (If X then Y) is preserved.
3. **Format:** Output a single executable instruction in clear natural language.

Output Schema:

```
{
  "topic": "Short Topic Name",
  "instruction": "The synthesized master instruction text..."
}
```

D Scalability Ablation

We ran an ablation study on the Contract-NLI dataset by iteratively increasing the training data size to observe its impact on the total number of clusters and the resulting prompt length. With just 1,030 examples, the framework successfully identified the main logical topics, establishing a baseline of 16 clusters. As the dataset size increased to over 7,000 examples, the total number of clusters plateaued at 18, meaning the prompt length remained highly stable. Instead of generating new topics, the additional data refined the existing clusters, which steadily improved the F1 macro score from 0.77 to 0.83 without introducing prompt bloat.

Table 4: Ablation Study of Dataset Size, Prompt Length (Tokens) and F1 score for Contract-NLI using Gemini 2 Flash

Index	Dataset Size	Total Clusters	Prompt Length (Tokens)	F1 Macro
1	1,030	16	4,062	0.77
2	2,060	18	4,486	0.79
3	3,090	18	4,410	0.80
4	4,119	18	4,580	0.80
5	5,148	18	4,520	0.82
6	6,178	18	4,665	0.83
7	7,190	18	4,630	0.83

E Conflict resolution model

E.1 Examples of Contract-NLI Conflict resolution instructions

Hypothesis #2: Claims the Receiving Party may 'retain' or 'keep' copies (e.g. for legal archival purposes).

```
**Before Conflict resolution model**
{
  "condition": "The text mandates return/destruction of 'all' copies and lists NO
  exceptions",
  "label": "Contradiction"
},
{
  "condition": "The text contains an exception clause (e.g., 'subject to,' 'except
  for,' 'provided that') allowing retention for 'legal,' 'archival,' 'backup,' or
  'compliance' purposes",
  "label": "Entailment"
}
```

```
**After Conflict resolution model**
{
  "condition": "The text explicitly permits the Receiving Party to 'retain', 'keep',
  'store', or 'preserve' copies of Confidential Information (or states
  return/destruction obligations are 'subject to' such retention) for specific
  purposes such as 'legal', 'archival', 'backup', 'compliance', 'audit', 'regulatory',
  'record-keeping', 'defending claims', or for use by 'legal counsel'.",
  "label": "Entailment"
},
{
  "condition": "The text explicitly mandates the 'return', 'destruction', 'deletion',
  or 'erasure' of 'all' Confidential Information (or 'all copies', 'all records', 'all
  materials') upon termination or request, AND does not contain exceptions allowing
  retention for legal/backup/archival purposes (Note: A clause stating items that
  'cannot be returned' must be 'destroyed' is considered a mandate for elimination,
  not retention).",
  "label": "Contradiction"
},
{
  "condition": "The text is silent regarding the return or destruction of Confidential
  Information upon termination/request, or does not explicitly specify whether copies
  may be retained.",
  "label": "NotMentioned"
}
```

Hypothesis #6: Claims the party is prohibited from 'reverse engineering' 'decompiling' or 'disassembling.'

```
**Before Conflict resolution model**
{
  "condition": "The text contains the keywords: reverse engineer, decompile,
  disassemble, reconstruct",
  "label": "Entailment"
},
{
  "condition": "These specific keywords are absent (even if the contract prohibits
  'unauthorized use' or 'copying')",
  "label": "NotMentioned"
}
```

```
**After Conflict resolution model**
{
  "condition": "The text explicitly prohibits the act (keywords: reverse engineer,
  decompile, disassemble, reconstruct, reverse assemble, reverse compile, discover
  source code, analyze chemical structure, determine chemical composition, recreate
  material) without providing an exception that allows the act",
  "label": "Entailment"
},
{
  "condition": "The text explicitly permits reverse engineering or related activities
  under certain conditions or exceptions (e.g., 'except as needed for the project' or
  'unless authorized'), which contradicts a blanket 'shall not' hypothesis",
  "label": "Contradiction"
},
{
  "condition": "The specific keywords (reverse engineer, decompile, disassemble,
  reconstruct, analyze chemical structure, recreate) are absent (even if the contract
  prohibits 'unauthorized use' or 'copying')",
  "label": "NotMentioned"
}
```

F Model Latency

Latency of various models monitored over 1000 requests.

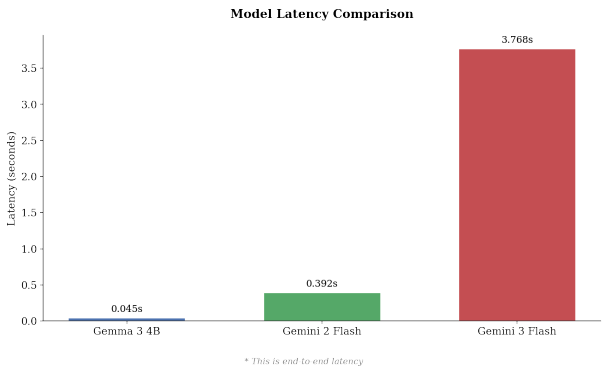

Figure 2: E2E Model Latency between Gemma-3 4B, Gemini 2 Flash, and Gemini 3 Flash

G Model Pricing

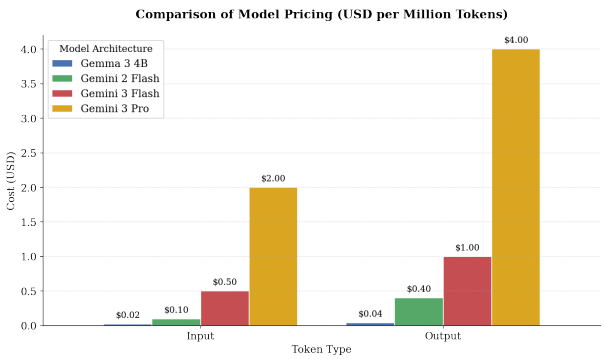

Figure 3: Comparision of Model pricing