PRISM: Prompt Representation Internalization via Selective Compression into LoRA Weights

Eshita Jha, Mingxian Wang, Andrei Barbu, Chetan Aggarwal
Amazon.com Inc, USA
{eshjha, mingxiw, barbu, caggar}@amazon.com

Abstract

Deploying large language models for specialized tasks requires lengthy prompts combining schema, demonstrations, and instructions creating a fundamental tension between prompt richness and inference feasibility. We identify and characterize two distinct prompt regimes: *pattern-dominant* tasks, where demonstrations drive performance through implicit formatting and extraction conventions, and *procedure-dominant* tasks, where explicit instructions carry the critical signal. To surface this distinction, we design a two-inference diagnostic that compares model performance with instructions alone versus examples alone, revealing which component to target for compression. Guided by this diagnostic, we introduce PRISM, a self-distillation framework that generates silver training data from the model's own full-prompt outputs and trains lightweight LoRA adapters to internalize the dominant component's behavior. PRISM recovers 131% of example benefit on CUAD and 151% on MultiWOZ (pattern-dominant), and 39% of instruction benefit on GSM8K (procedure-dominant), eliminating 72–79% of prompt tokens at inference time. Silver data captures 80–94% of gold-labeled performance without human annotations. Beyond compression, PRISM frees context capacity: replacing a long prompt with LoRA weights enables 89.5% compliance with new runtime instructions versus 75.4% with recency prompting. The diagnostic suggests a practical pre-training heuristic to indicate when distillation is likely to succeed, helping practitioners avoid costly training on non-beneficial components.

1 Introduction

Large language models (LLMs) achieve strong few-shot performance through in-context learning (ICL), where task demonstrations are prepended to the input prompt (Brown et al., 2020). However, this paradigm carries significant practical costs: each demonstration consumes context length, increases inference latency proportional to the number of tokens processed, and limits the input budget available for the actual task. For specialized tasks such as legal document analysis or multi-turn dialogue, these costs are particularly acute a prompt with 50 demonstrations can exceed 20,000 tokens, leaving little room for the input itself. Furthermore, in-context examples are constrained by the selection available at prompt design time, and prior work has shown that performance is sensitive to example choice and ordering (Lu et al., 2022; Liu et al., 2022), introducing variance that is difficult to control in production settings.

This raises a natural question: *can the knowledge conveyed by prompt components be compressed into model weights, enabling equivalent performance at reduced inference cost?*

We investigate this through **component distillation** a two-stage approach where a prompted model first generates silver training data from its own example-augmented outputs, then lightweight LoRA adapters (Hu et al., 2022) are fine-tuned to internalize the demonstrated behavior. At inference time, the adapter replaces the distilled component entirely: the model receives a shorter prompt yet produces outputs comparable to or exceeding the fully prompted model.

A central question, however, is *when* will this approach work. Not all in-context examples serve the same function. To investigate this, we first construct structured long prompts (7K–23K tokens) composed of three modular components: domain schema, task instructions, and input-output demonstrations. By systematically ablating these components evaluating all combinations of inclusion and exclusion we isolate the contribution of each component to task performance (§2.2).

This ablation reveals a striking asymmetry. On some tasks, demonstrations are the dominant driver of performance, contributing 2-5× more

than instructions. On others, instructions alone nearly match the full prompt. We formalize this observation as a **distillation diagnostic**: the performance gap between instructions(I) and examples(E) predicts which component can be compressed into weights. When E $\gg$ I, examples encode *implicit pattern knowledge* output formatting, extraction boundaries, domain conventions amenable to weight-based compression. When I $\gg$ E, instructions encode *procedural knowledge* that can similarly be distilled.

We validate this framework across three tasks that span the pattern–procedure spectrum:

- **CUAD** (legal contract clause extraction): Examples improve F1 by 17.8 points over instructions alone, demonstrating extraction boundaries and formatting conventions that resist verbalization. LoRA recovers **131%** of example benefit with 72% fewer tokens.
- **MultiWOZ** (dialogue state tracking): Examples improve JGA by 13 points, providing slot-value formatting patterns across 7 domains. LoRA achieves **151%** recovery substantially *exceeding* in-context example performance with 79% fewer tokens.
- **GSM8K** (mathematical reasoning): Instructions alone achieve 88% accuracy, *outperforming* examples (80%). Example LoRA matches examples at 80% but cannot improve over instructions. However, instruction LoRA recovers 39% of instruction benefit (71%→77.6%), confirming that the dominant component whichever it is can be distilled.

Our contributions are: (1) a self-distillation method for compressing prompt components (examples or instructions) into LoRA adapters, where the model generates its own silver training data; (2) a **distillation diagnostic** the I vs E performance gap that predicts *which* component to distill before any training investment; (3) empirical validation across five distillation settings (three tasks, two component types, two models), showing 39-151% recovery with 72–79% fewer tokens, with layer ablation revealing distilled knowledge concentrates in late transformer layers; and (4) empirical demonstration that LoRA compression frees context capacity: replacing a long prompt with LoRA weights enables 89.5% compliance with new runtime instructions versus 75.4% with recency prompting and 1.7% when the instruction is buried mid-prompt; (5) a comparison with context distillation showing PRISM substantially outperforms KL-based compression (37.8% vs 10.4% F1), which fails to converge effectively under matched hyperparameters.

2 Method

Our approach consists of three stages: structured prompt design with component ablation, silver data generation, and LoRA fine-tuning (Figure 1).

2.1 Problem Formulation

Given a prompt $P = C_1 \oplus C_2 \oplus \ldots \oplus C_k \oplus x$, where C_i are prompt components (schema, examples, instructions) and x is the task input, we seek a lightweight adapter θ such that $\text{LLM}_\theta(P \setminus C_j, x) \approx \text{LLM}(P, x)$, where C_j is the dominant component identified by the diagnostic. The adapter internalizes C_j's contribution into weights, eliminating its tokens at inference time.

2.2 Structured Prompt Design

A key challenge in understanding in-context learning is that prompts are typically treated as monolithic text blocks, making it difficult to attribute performance to specific components. We address this by constructing long prompts (7K–23K tokens) from three modular, independently toggleable components. Each component is manually authored per task: schema defines the domain ontology, examples are drawn from training data (30–80 per task), and instructions specify output format and extraction rules. Representative prompt snippets are provided in Appendix C, and our prompt-engineering methodology (component authoring and coverage-based example selection) is detailed in Appendix B.

- **S (Schema)**: Task-specific domain ontology slot definitions for dialogue, clause type taxonomy for legal extraction, problem categories for math.
- **E (Examples)**: Input-output demonstrations drawn from training data (30–80 examples, comprising 7K–19K tokens). These show the model *what good outputs look like* through concrete demonstrations.
- **I (Instructions)**: Explicit procedural guidance output format specifications, extrac-

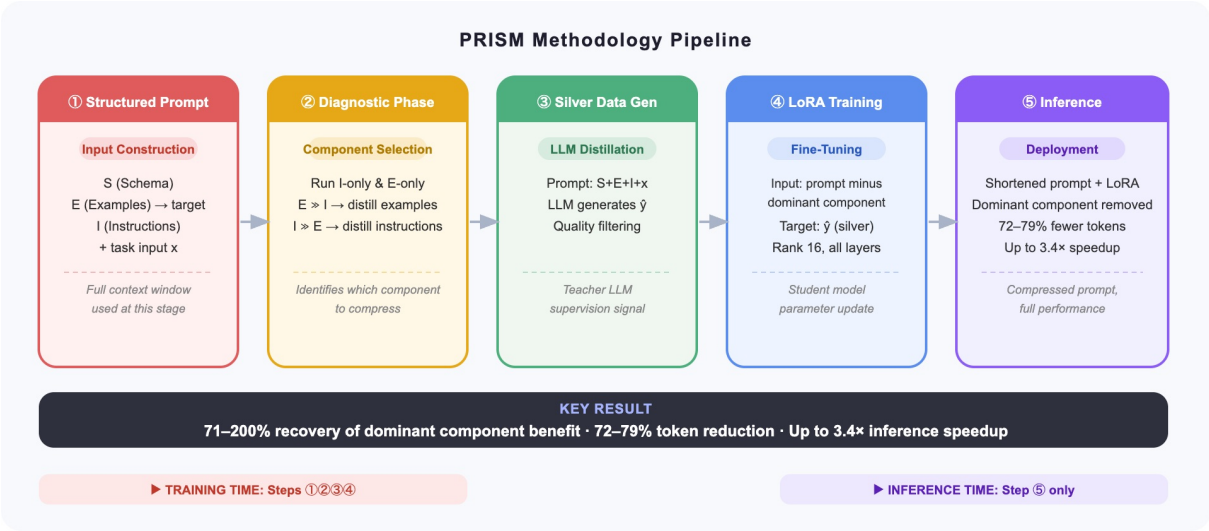

Figure 1: PRISM methodology pipeline. The diagnostic phase identifies which prompt component (examples or instructions) to distill. Silver data is generated from the full prompt, then LoRA is trained to internalize the dominant component's behavior. At inference, the dominant component is removed, reducing tokens by 72–79%.

tion rules, step-by-step reasoning strategies. These tell the model *how to produce outputs* through verbal description.

This decomposition reflects primary prompt engineering strategies in production LLM systems: schema provides domain structure, examples demonstrate desired behavior through input-output pairs, and instructions specify procedural rules. While other strategies such as chain-of-thought(COT) prompting and retrieval-augmented generation(RAG) exist, they can be viewed as specialized forms of these components: COT as a type of instruction, and RAG as dynamically selected examples.The full prompt is $P_{\text{full}} = \text{S} \oplus \text{E} \oplus \text{I} \oplus x$, where x is the task input. By including or excluding each component, we construct 7 configurations ($\varnothing$, S, I, E, S+E, S+I, S+E+I) that isolate each component's contribution. This ablation identifies whether a task is *pattern-dominant* (E drives performance) or *procedure-dominant* (I drives performance), yielding the distillation diagnostic (§5.1).

2.3 Silver Data Generation

We generate silver training data as follows:

1. Sample N inputs $\{x_1, \ldots, x_N\}$ from the task's training split, disjoint from the in-context examples used in E to avoid data leakage.

2. For each input x_i, construct the full prompt $P_{\text{full}}(x_i) = \text{S} \oplus \text{E} \oplus \text{I} \oplus x_i$ and generate $\hat{y}_i = \text{LLM}(P_{\text{full}}(x_i))$ using greedy decoding.

3. Apply task-specific quality filtering to obtain the silver set $\mathcal{D}_{\text{silver}} = \{(x_i, \hat{y}_i)\}$.

The model's outputs under example-augmented prompts reflect behavioral patterns those examples demonstrate (formatting conventions, extraction boundaries, domain-specific output structure). By training a LoRA adapter that receives no examples to reproduce these outputs, we force it to internalize those patterns into its weights. By training on these outputs, we transfer pattern knowledge from the prompt context into model weights (representative silver instances are shown in Appendix D).

Why Silver Data, Not Gold Labels? A natural alternative is to train LoRA directly on gold-labeled data. While this would likely produce a competent model, it conflates multiple learning signals task knowledge, format conventions, domain patterns making it impossible to isolate what in-context examples specifically contribute. Our silver data approach, inspired by knowledge distillation (Hinton et al., 2015) and its application to LLMs (Hsieh et al., 2023), uses the model's own example-augmented outputs as training targets. The training signal is therefore precisely the *behavioral difference* between the model with and without examples. This approach requires no labeled data the model is its own teacher. We val-

idate this empirically: silver LoRA achieves 80–94% of gold LoRA performance across CUAD and MultiWOZ (§5.3).

Quality Filtering. We apply task-specific filters remove low-quality silver examples: CUAD requires ≥80% word overlap with the input passage (38% retention; selection bias and generalization analysis in Appendix E.1); MultiWOZ requires proper slot-value formatting (99% retention); GSM8K retains only correct numerical answers (84% retention). Silver data statistics for all datasets are provided in Appendix E.

2.4 LoRA Fine-Tuning

We fine-tune LoRA adapters (Hu et al., 2022) on the silver data. The input template excludes the dominant component:for **example distillation**, input is S $\oplus$ I $\oplus$ x_i (no demonstrations), target is $\hat{y}_i$ from S+E+I; for **instruction distillation**, input is S $\oplus$ E $\oplus$ x_i (no instructions).For MultiWOZ, where ablation showed instructions degrade performance when combined with examples (§4.1), we use S$\oplus$x (schema only, no instructions) as the LoRA input template instead of the standard S$\oplus$I$\oplus$x.[1]

Training Configuration. We use LoRA rank $r = 16$, $\alpha = 32$, targeting all attention and MLP projections following Hayou et al. (2024), with learning rate 1×10^{-4} (2×10^{-4} for MultiWOZ). Full training details are provided in Appendix A.

Inference. At deployment, the LoRA adapter is merged with the base model. The input prompt excludes the distilled component S+I+x for example distillation or S+E+x for instruction distillation reducing prompt length by 72–79% compared to the full S+E+I prompt.

3 Experimental Setup

3.1 Tasks and Datasets

We select three tasks spanning the pattern–procedure spectrum. Table 1 provides published reference points for context. Our goal is not to match SOTA performance but to measure how much of the prompted performance gap LoRA can recover relative to the best prompted configuration on each task.

Task	Published Baseline	Score
GSM8K	Qwen2.5-7B, 4-shot[1]	85.4% Acc
	Llama-3-8B, 8-shot CoT[2]	79% Acc
CUAD	DeBERTa-xl, fine-tuned[3]	47% AUPR
MultiWOZ	Fine-tuned DST SOTA[4]	∼60% JGA
	ChatGPT, few-shot[5]	27% JGA

Table 1: Published reference points. [1]Yang et al. (2024) (4-shot), [2]Meta AI (2024), [3]Hendrycks et al. (2021), [4]Feng et al. (2023), [5]Hudeček and Dusek (2023). Our experiments use 30–80 shot prompts with 7B models; these numbers contextualize task difficulty.

CUAD (Hendrycks et al., 2021) requires extracting specific clause types from legal contracts. We evaluate on 500 test examples using token-level F1. **MultiWOZ 2.2** (Zang et al., 2020) is a 7-domain dialogue state tracking benchmark. We evaluate on 500 test dialogues using Joint Goal Accuracy (JGA)[2] and Slot F1. **GSM8K** (Cobbe et al., 2021) consists of grade school math problems requiring multi-step reasoning. We evaluate on 500 test problems using exact-match accuracy.

3.2 Models

We evaluate **Qwen2.5-7B-Instruct** (Yang et al., 2024) (128K context, 4-bit QLoRA) on all tasks. To assess cross-model generalization, we further evaluate two additional 7B instruction-tuned families trained on the *same* Qwen-generated silver data: **Mistral-7B-Instruct-v0.3** (Jiang et al., 2023a) (32K context) on CUAD, and **InternLM2.5-7B-chat** (32K context) on CUAD and MultiWOZ.

3.3 Evaluation Protocol

We define the **recovery rate** to quantify what fraction of the best prompted performance LoRA captures:

$$\text{Recovery} = \frac{f(M_\theta, P \setminus C_j) - f(M, P \setminus C_j)}{f^*(M) - f(M, P \setminus C_j)} \tag{1}$$

where $f^*(M) = \max_{C \subseteq \{S,E,I\}} f(M, C \oplus x)$ is the best performance achievable with any prompted configuration, and $P \setminus C_j$ is the prompt without the distilled component.

[1]Using the standard S$\oplus$I$\oplus$x template for MultiWOZ LoRA yields 55.7% Slot F1 vs 77.3% with the minimal S$\oplus$x template, confirming that instruction interference persists during LoRA training. This means MultiWOZ LoRA benefits from both internalizing example patterns *and* avoiding instruction interference; the reported recovery rate reflects both effects jointly.

[2]JGA measures the fraction of dialogue turns where all predicted slot-value pairs exactly match the gold state.

Config	CUAD-Q (F1)	CUAD-M (F1)	MW JGA (%)	GSM8K (Acc)
∅ (none)	13.3	5.4	0.0	21.0
S+I only (instructions)	14.4	16.4	0.0	**88.0**
S+E only (examples)	**31.5**	**53.8**	**18.8**	80.4
S+I (base, ex. distil.)	10.7	19.3	5.8	76.2
S+E (base, in. distil.)	27.6	53.5	8.6	71.0
S+E+I (full prompt)	31.4	53.2	12.8	86.8
LoRA (example)	**37.8**	**54.2**	**25.4**	79.6
LoRA (instruction)	26.3	—	—	**77.6**

Table 2: Prompt ablation (n=500). CUAD-Q=Qwen, CUAD-M=Mistral. Bold per column = best prompted config. LoRA rows (green) show distillation results: example LoRA exceeds the best prompted config on CUAD and MultiWOZ; instruction LoRA recovers instruction benefit on GSM8K.

Recovery $> 100\%$ indicates LoRA *exceeds* the best prompted configuration (Section 5.1).

Task	Full	Base	LoRA	Saved (%)	Rec. (%)
Example Distil.					
CUAD-Q (F1)	31.4	10.7	37.8	72	131
CUAD-M (F1)	53.2	19.3	54.2	72	103
MW (JGA)	18.8	5.8	25.4	79	151
MW (SF1)	66.8	27.5	74.3	79	119
GSM8K (Acc)	88.0	76.2	79.6	91	29
Instruction Distil.					
GSM8K (Acc)	88.0	71.0	77.6	91	39
CUAD-Q (F1)	31.4	27.6	26.3	72	−34

Table 3: Distillation results (n=500). Q=Qwen, M=Mistral (cross-model transfer, trained on Qwen silver;). **Full**: best prompted config (f^*). **Base**: prompt without distilled component. **Rec.**: recovery rate. GSM8K example LoRA confirms example distillation fails on procedure-dominant tasks; instruction LoRA (39%) confirms the dominant component can be distilled.

4 Results

4.1 Prompt Ablation

Table 2 presents the prompt ablation results. Examples dominate on CUAD and MultiWOZ, providing 2–3× improvement over instructions. On GSM8K, instructions alone (88%) nearly match the full prompt (87%). Notably, component interactions can be destructive: on MultiWOZ, S+E+I (12.8%) *degrades* performance compared to E alone (18.8%), probably due to long-context interference, motivating our use of a minimal template for MultiWOZ LoRA training. Recovery rates exceeding 100% (CUAD: 131%, MultiWOZ: 151%) are consistent with the adapter internalizing example patterns while avoiding the cost of processing a long prompt; we note that reduced long-context interference is likely one of several contributing factors rather than the sole cause (see §5.6). The pattern is consistent across architectures: Mistral and InternLM2.5 show the same E-dominant profile on CUAD(InternLM CUAD LoRA reaches 23.7% F1). InternLM2.5 exhibits similar behavior on MultiWOZ , with its LoRA adapter trained on Qwen-generated silver data reaches 18.2% JGA / 66.8% Slot-F1.

4.2 LoRA Distillation

Table 3 presents LoRA adapters trained on silver data compared against prompted baselines. On **CUAD**, Qwen example LoRA achieves 37.8% F1 (131% recovery), exceeding the full prompt (31.4%), while Mistral trained on Qwen-generated silver data achieves 54.2% F1 (103% recovery), demonstrating that behavioral patterns captured in silver data are not model-specific artifacts but reflect genuine task structure that generalizes across architectures; instruction LoRA provides no benefit (26.3% F1, −34% recovery), confirming that distilling the non-dominant component is ineffective. On **MultiWOZ**, LoRA achieves 25.4% JGA and 74.3% Slot F1, substantially exceeding E (18.8% JGA, 66.75% Slot F1) a 151% JGA recovery rate. On **GSM8K**, example LoRA (79.6%) matches E (80.4%) but cannot improve over I (88%), confirming example distillation is unnecessary when instructions dominate, while instruction LoRA recovers 39% of instruction benefit (71%→77.6%), demonstrating that the dominant component can be distilled (§5.1). Both conditions eliminate 91% of prompt tokens (12.9K→1.2K). Instruction LoRA on CUAD achieves 26.3% F1 (−34% recovery), confirming that distilling the non-dominant component is ineffective.

5 Analysis

5.1 The Distillation Diagnostic

The distillation diagnostic predicts which prompt component to distill based on the I vs E performance gap: example distillation succeeds when E $\gg$ I (CUAD: 131%, MultiWOZ: 151% recovery), instruction distillation when I $\gg$ E (GSM8K: 39% recovery), and distilling the non-dominant component provides no benefit (Figure 2). The diagnostic requires only two inference runs and correctly predicts outcomes across all three tasks evalu-

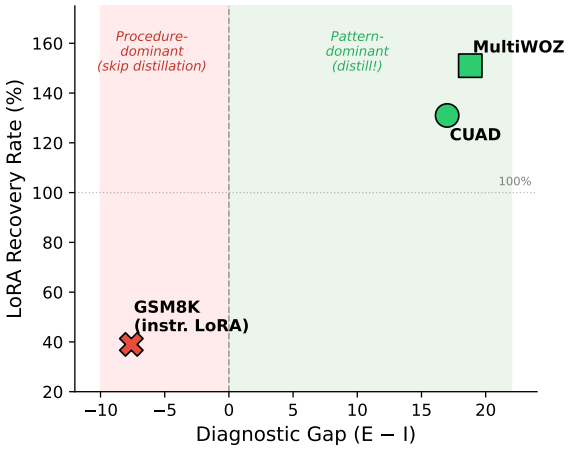

Figure 2: Distillation diagnostic (n=500). The E−I gap predicts LoRA recovery rate. Pattern-dominant tasks (positive gap) succeed with example distillation; GSM8K (procedure-dominant, negative gap) uses instruction LoRA.

ated here; broader validation across additional task families remains future work.

Bidirectional validation. Distilling the non-dominant component provides no benefit: instruction LoRA on CUAD achieves only 26.3% F1 (−34% recovery), and example LoRA on GSM8K matches E (80%) but cannot exceed I (88%). The diagnostic thus operates bidirectionally: it identifies not just *whether* to distill, but *which component* to distill (Table 3).Output-level analysis confirms that LoRA outputs are 2–4× more likely to match E outputs than S+I outputs across all tasks (Appendix E.2).

5.2 Inference Latency

The latency benefit scales with the ratio of example tokens to total input length (Table 4). On MultiWOZ, where examples constitute approximately 75% of the prompt, LoRA achieves a 4.4× speedup by eliminating the example component entirely. On CUAD, the speedup is negligible because the contract passage dominates input length regardless of prompt configuration. On GSM8K, LoRA shows no meaningful latency difference from the full prompt (26.2s vs 27.1s per query), consistent with instruction tokens being a small fraction of total generation time on reasoning tasks. In all cases, LoRA adds no adapter overhead as weights are merged at load time.

Task	Config	Tokens	Latency	Speedup
CUAD	Full Prompt	15.9K	12.7s	1.0×
	LoRA	11.3K	11.9s	1.1×
GSM8K	Full Prompt	12.9K	27.1s	1.0×
	LoRA	1.2K	26.2s	1.0×
MW	Full Prompt	25.9K	21.2s	1.0×
	LoRA	5.7K	4.8s	**4.4**×

Table 4: Inference latency (Qwen2.5-7B, 4-bit, $n = 500$). Speedup relative to full prompt.

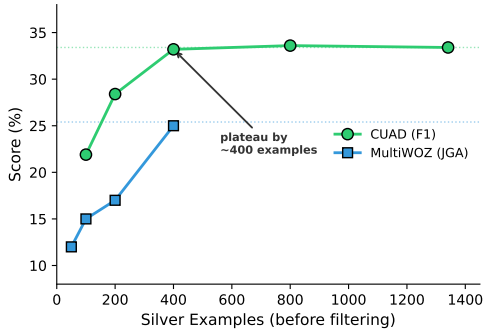

Figure 3: Data efficiency on CUAD (F1) and MultiWOZ (JGA) (Qwen2.5-7B). Both plateau near their full-data results (dotted lines) by ∼400 silver examples; on CUAD, 200 examples already recover 85% of the maximum.

5.3 Data Efficiency

A practical question for deployment is how much silver data is needed. We train LoRA adapters on nested subsets of silver data for both CUAD and MultiWOZ and evaluate on held-out test sets (Figure 3).

On **CUAD**, performance plateaus sharply at 400 examples (33.2% F1), matching the full dataset (33.4%); even 200 examples achieve 28.4% (85% of the maximum), suggesting that extraction patterns are learnable from a modest number of demonstrations. Silver data generation for 400 examples requires only ∼1.4 GPU-hours, making the total upfront cost practical. **MultiWOZ** shows a similar trend, with JGA plateauing near its full-data result (∼25%) by ∼400 examples.

Silver vs Gold Training. To validate the self-distillation approach, we train *separate* LoRA adapters on gold-labeled data using the same inputs, swapping silver outputs for gold annotations. Silver LoRA achieves 94% of gold performance on CUAD (33.4% vs 35.6% F1) and 80–93% on MultiWOZ (36.0% vs 45.0% JGA; 77.3%

Active LoRA Layers	F1	Δ vs S+I
None (S+I baseline)	13.8	
Early only (0-13)	26.0	+12.2
Late only (14-27)	**35.2**	+21.4
Full (0-27)	33.4	+19.6

Table 5: Layer ablation on CUAD (Qwen2.5-7B, $n = 100$). Extraction pattern knowledge is concentrated in later layers. Baseline S+I F1 = 13.8% at $n = 100$.

Method	CUAD F1
No prompt (S+I baseline)	10.7
Full prompt (S+E+I)	31.4
Context distillation (KL)	10.4
PRISM (silver cross-entropy)	**37.8**

Table 6: PRISM vs. context distillation on CUAD (n=500). KL distillation barely exceeds the no-prompt baseline, while PRISM achieves 131% recovery.

vs 82.8% SF1), [3] confirming that the model's own example-augmented outputs provide a sufficient training signal without requiring human annotations.

5.4 Layer Ablation: Where Does Pattern Knowledge Live?

To understand *where* distilled pattern knowledge is encoded, we perform a causal intervention: for each layer group, we set the low-rank updates $\Delta W = BA$ to zero, reverting those layers to pre-trained weights while keeping the adapter active on remaining layers. We selectively disable LoRA in either the early (layers 0–13) or late (layers 14–27) half of Qwen2.5-7B (Table 5). Late-layer LoRA alone (35.2% F1) recovers the full benefit of distillation slightly exceeding the full adapter (33.4%), while early-layer LoRA (26.0%) captures less than half the total gain, indicating that extraction patterns are primarily encoded in later layers where task-specific representations are formed. The slight improvement of late-only over full LoRA further suggests that early-layer modifications introduce noise into general language features; restricting LoRA to later layers thus serves as a more parameter-efficient configuration.

5.5 Catastrophic Forgetting

We test whether task-specific LoRA adapters cause catastrophic forgetting (Biderman et al., 2024) by evaluating the CUAD LoRA adapter on GSM8K and MultiWOZ using the full S+E+I prompt. On MultiWOZ, CUAD LoRA preserves performance (18% vs 16% JGA, 63.3% vs 60.0% Slot F1), showing no degradation on a fellow extraction-oriented task. On GSM8K, we observe a modest drop (83% vs 89% accuracy), consistent with extraction-focused weight modifications slightly interfering with chain-of-thought reasoning. In practice, task-specific adapters should be loaded selectively; adapter routing (Hu et al., 2022) can address this in deployment.

[3] These silver-vs-gold comparisons are computed on a matched n=100 subset

5.6 Comparison with Context Distillation

A natural baseline for prompt compression is context distillation (Snell et al., 2022), which minimizes $\mathrm{KL}(p_{\text{teacher}} \| p_{\text{student}})$ where the teacher is the full-prompt model and the student is the LoRA model without the prompt. We implement this on CUAD under matched conditions: same model, LoRA config, 400 training samples, and inference template. Concretely, for each training sample we run two forward passes one with the full S+E+I prompt (teacher) and one with S+I only (student) and minimize the KL divergence between their output distributions at each token position (Table 6).

PRISM (37.8%) substantially outperforms KL distillation (10.4%), which barely exceeds the no-prompt baseline (10.7%). The KL objective fails to converge effectively with training loss oscillating between 0.49–1.02 across epochs, compared to PRISM's clean convergence from 0.498 to 0.045. We attribute this to the difficulty of optimizing the KL objective under matched hyperparameters: it requires the student to match the teacher's full output distribution at every token position, whereas PRISM only requires reproducing sampled text outputs, following the standard knowledge-distillation practice of training a student on teacher outputs (Hinton et al., 2015; Hsieh et al., 2023). We note that a dedicated hyperparameter sweep for the KL baseline might improve results; our comparison establishes that PRISM is easier to tune under matched configurations.

5.7 Context Capacity Liberation

Compressing a long prompt into LoRA weights does more than reduce tokens it frees context capacity for new runtime instructions. When a 15K-token prompt occupies the context window, a new instruction added at inference time is effectively

ignored: we show compliance drops to 1.7% when the instruction is buried mid-prompt (Table 7). PRISM eliminates this bottleneck.

Setup. We use IFEval (Zhou et al., 2023), a benchmark of 541 instruction-following tasks with 25 programmatically verifiable constraint types, as the base evaluation. We construct a 15K-token consulting firm editorial guide as the long prompt and inject a new instruction absent from the prompt: *"Your response must include the word 'prism' at least once."* This word never appears naturally in IFEval outputs (0/541 verified) and conflicts with no IFEval constraint, making compliance a clean signal of whether the new instruction was followed.

Cond.	Description	IFEval	Compliance
A	Long prompt + task	74.9	—
G	LoRA + task (no new instr.)	75.1	—
D	No prompt + task (no new instr.)	76.3	—
B	Long prompt + instr. (mid)	76.0	1.7
C	Long prompt + instr. (end)	75.1	75.4
E	No prompt + instr. (upper bound)	73.9	94.1
F	**LoRA + new instruction**	**73.0**	**89.5**

Table 7: Context capacity experiment on IFEval (n=541). Compliance = % of outputs containing 'prism'. D = no prompt, no new instruction (baseline). G≈A confirms LoRA preserves task quality; F≫C shows LoRA frees capacity for new instructions.

Results. G (75.1%) ≈ A (74.9%) confirms LoRA successfully internalizes the long prompt without degrading IFEval performance. LoRA (F: 89.5%) dramatically outperforms both mid-prompt placement (B: 1.7%) and recency (C: 75.4%), approaching the no-prompt upper bound (E: 94.1%). The mild IFEval drop in F vs G (73.0% vs 75.1%) is not a LoRA artifact the same drop occurs without any prompt (E: 73.9% vs D: 76.3%) confirming it is caused by the new instruction itself, not by compression. PRISM thus frees context capacity: a model with compressed prompts follows new runtime instructions at 89.5% compliance versus 75.4% with recency prompting and 1.7% when the instruction is buried.

6 Related Work

Context and Rationale Distillation. Snell et al. (2022) propose context distillation, aligning model outputs with and without context through a KL-divergence objective that requires access to the model's output distribution. Distilling Step-by-Step (Hsieh et al., 2023) extracts chain-of-thought reasoning from teacher LLMs to train smaller models. PRISM differs in three ways: (1) it uses only sampled text outputs for silver data generation, decoupling data creation from adapter training; (2) it targets both implicit pattern knowledge and procedural knowledge, not just reasoning chains; and (3) it provides a pre-hoc diagnostic that identifies which component to distill before any training investment.

Prompt Compression. GIST (Mu et al., 2023) compresses prompts into learned soft tokens, while LLMLingua (Jiang et al., 2023b) removes tokens at inference time via perplexity-based heuristics. These approaches either produce non-standard representations or add per-inference overhead. Our method uses standard LoRA weights with zero inference-time cost and provides explicit analysis of compressibility by component type.

In-Context Learning Theory. Brown et al. (2020) demonstrated few-shot ICL; subsequent work investigated example selection (Liu et al., 2022), ordering sensitivity (Lu et al., 2022), and theoretical foundations (Xie et al., 2022; Dai et al., 2023). Min et al. (2022) showed that examples primarily convey format and distribution rather than factual knowledge consistent with our finding that pattern knowledge is distillable.

Parameter-Efficient Fine-Tuning. LoRA (Hu et al., 2022) introduces low-rank adapters for parameter-efficient adaptation;Hayou et al. (2024) further improve it by assigning distinct learning rates to the two low-rank matrices;Prefix tuning (Li and Liang, 2021) and prompt tuning (Lester et al., 2021) also replace explicit tokens with learned parameters, but prepend vectors at every inference call rather than merging into weights, maintaining nonzero context overhead. Crucially, none of these approaches provide a pre-hoc diagnostic for identifying which prompt components are compressible.

7 Conclusion

We presented PRISM, a framework that identifies the dominant prompt components (examples or instructions) via a two-inference diagnostic, then distills it into a lightweight LoRA adapter. Example LoRA recovers 131% on CUAD and 151%

on MultiWOZ; instruction LoRA recovers 39% on GSM8K eliminating 72–79% of prompt tokens in each case. Instruction distillation proves highly effective for format- and guideline-following instructions (IFEval) but only partial for multi-step reasoning (GSM8K). Distilling the non-dominant component yields no benefit, confirming the diagnostic's bidirectionality. Beyond this, PRISM establishes a broader principle: the knowledge encoded in carefully engineered prompts need not be re-supplied at every inference call. It can be internalized into model weights once, amortizing the cost of prompt engineering across all future inferences, and a two-inference diagnostic determines whether that internalization is worth attempting before any training investment is made. In deployment, this directly reduces context window pressure and inference failures caused by lengthy prompts. PRISM also frees context capacity: replacing a long prompt with LoRA weights enables 89.5% compliance with new runtime instructions versus 75.4% with recency prompting. Future work includes extending PRISM to reasoning-heavy transfer, chain-of-thought and retrieval-augmented prompts, larger model families, multilingual settings, and multi-task adapter composition.

Limitations

We evaluate on three 7B-parameter models (Qwen2.5, Mistral, InternLM2.5) and three tasks; generalization to larger models (70B+) and additional task families such as summarization and classification remains untested. We do not compare against GIST (Mu et al., 2023) or LLMLingua (Jiang et al., 2023b); evaluating the trade-offs between static, weight-internalized compression (PRISM) and dynamic, inference-time token reduction (LLMLingua) across different deployment constraints is a critical area for future work. Finally, we do not conduct exhaustive hyperparameter sweeps across LoRA rank or learning rate, and the optimal configuration may vary by task.

Disclosure of AI Assistance

We acknowledge the use of AI assistance for LaTeX formatting and writing polish.These tools were utilized solely to enhance the readability of the manuscript. The final content and scientific integrity of this work remain the sole responsibility of the authors.

References

Dan Biderman, Jacob P. Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, Cody Blakeney, and John Patrick Cunningham. 2024. Lora learns less and forgets less. *Transactions on Machine Learning Research*, 2024.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

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. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. 2023. Why can GPT learn in-context? language models secretly perform gradient descent as meta-optimizers. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4005–4019, Toronto, Canada. Association for Computational Linguistics.

Yujie Feng, Zexin Lu, Bo Liu, Liming Zhan, and Xiao-Ming Wu. 2023. Towards LLM-driven dialogue state tracking. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 739–755, Singapore. Association for Computational Linguistics.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. 2024. LoRA+: Efficient low rank adaptation of large models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pages 17783–17806. PMLR.

Dan Hendrycks, Collin Burns, Anya Chen, and Spencer Ball. 2021. CUAD: An expert-annotated NLP dataset for legal contract review. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

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, Toronto, Canada, July 9-14, 2023*, pages 8003–8017. Association for Computational Linguistics.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Vojtěch Hudeček and Ondrej Dusek. 2023. Are large language models all you need for task-oriented dialogue? In *Proceedings of the 24th Meeting of the Special Interest Group on Discourse and Dialogue, SIGDIAL 2023, Prague, Czechia, September 11 - 15, 2023*, pages 216–228. Association for Computational Linguistics.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023a. Mistral 7b. *CoRR*, abs/2310.06825.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023b. LLMLingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 13358–13376. Association for Computational Linguistics.

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, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 3045–3059. 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, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4582–4597. Association for Computational Linguistics.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out: The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures, DeeLIO@ACL 2022, Dublin, Ireland and Online, May 27, 2022*, pages 100–114. Association for Computational Linguistics.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 8086–8098. Association for Computational Linguistics.

Meta AI. 2024. Introducing meta Llama 3. `https://ai.meta.com/blog/meta-llama-3/`. Accessed: 2026-08-30.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 11048–11064. Association for Computational Linguistics.

Jesse Mu, Xiang Li, and Noah D. Goodman. 2023. Learning to compress prompts with gist tokens. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Charlie Snell, Dan Klein, and Ruiqi Zhong. 2022. Learning by distilling context. *CoRR*, abs/2209.15189.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2022. An explanation of in-context learning as implicit Bayesian inference. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 22 others. 2024. Qwen2.5 technical report. *CoRR*, abs/2412.15115.

Xiaoxue Zang, Abhinav Rastogi, Srinivas Sunkara, Raghav Gupta, Jianguo Zhang, and Jindong Chen. 2020. MultiWOZ 2.2: A dialogue dataset with additional annotation corrections and state tracking baselines. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 109–117, Online. Association for Computational Linguistics.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and

Le Hou. 2023. Instruction-following evaluation for large language models. *CoRR*, abs/2311.07911.

A Training Configuration

We use LoRA rank $r = 16$, $\alpha = 32$, dropout 0.05, targeting all attention projections (q, k, v, o) and MLP projections (gate, up, down). Training proceeds for 5 epochs with gradient accumulation over 16 steps, batch size 1, and fp16 precision on a single NVIDIA L4 GPU (24GB). We use 1×10^{-4} as our base learning rate. For MultiWOZ, we additionally evaluated 2×10^{-4}, which yielded substantially better results (77.3% vs 55.7% Slot F1). Within each training run, we select the best checkpoint based on eval loss on a 10% stratified held-out split. Input sequences are truncated to 4,096 tokens with answer-centered truncation for CUAD to ensure the gold span remains within the context window.

B Prompt Engineering Methodology

To reach dataset baseline performance, we applied several prompt-engineering tactics: (i) **instruction refinement**—for example, on CUAD we moved from brief directives ("extract exact spans," "be precise") to more precise extraction rules such as Verbatim Extraction, Minimal Sufficient Span, and Complete Clause Extraction, each with explicit correct/incorrect cases; (ii) **stratified example selection** so demonstrations are representative of the data e.g., on MultiWOZ the examples cover all five active domains (restaurant, hotel, train, attraction, and taxi); (iii) **decomposing** each prompt into schema, examples, and instructions rather than a monolithic block, which is what enables the component ablation to identify the dominant part. We applied these tactics to reach industry-benchmark performance, but we emphasize that our objective is not to find the single best prompt rather, to study which prompt components can be internalized into weights. Prompt engineering is therefore a means to a fair comparison, not the central contribution of this work.

C Prompt Examples

We provide representative snippets of the prompt configurations used in our experiments. Full prompts range from 1K to 23K tokens; we show truncated versions here.

C.1 CUAD: E (Examples Only)

```
## EXAMPLES

Example 1:
Passage: EXHIBIT 10.6
DISTRIBUTOR AGREEMENT
THIS DISTRIBUTOR AGREEMENT (the
"Agreement") is made by and between
Electric City Corp., a Delaware
corporation ("Company") and Electric
City of Illinois LLC ("Distributor")
this 7th day of September, 1999. ...
Question: Highlight the parts related
to "Document Name".
Answer: DISTRIBUTOR AGREEMENT

Example 2:
Passage: WEB SITE HOSTING AGREEMENT ...
Question: Highlight the parts related
to "Parties".
Answer: Centrack International, INC.

[... 28 more examples ...]

## TASK
Passage: {context}
Question: {question}
Answer:
```

C.2 CUAD: S+I (Schema + Instructions)

The S+I prompt contains detailed system context, a full clause type taxonomy (29 types), and comprehensive extraction rules (2,667 tokens total). We show a truncated version:

```
## SYSTEM
You are a legal contract analysis
system specialized in extracting
specific clause information from
commercial agreements.

## APPLICATIONS
Contract Clause Types:
- Document Name: title of the agreement
- Parties: entities entering the
↪  agreement
- Agreement Date: when signed/effective
- Governing Law: jurisdiction
[... 25 more clause types with detailed
 descriptions and valid values ...]

## INSTRUCTIONS
### Core Extraction Principles

Principle 1: Verbatim Extraction
Copy text exactly as it appears. Do not
paraphrase, summarize, fix grammar, or
change capitalization.

Principle 2: Minimal Sufficient Span
Extract the shortest span that fully
answers the question.

Principle 3: Complete Clause Extraction
For complex provisions, include the full
operative language.

[... negatives, cross-references,
 boilerplate vs negotiated terms,
 common extraction errors ...]
```

```
## Output Format
Answer: [extracted text or "Not found"]

## TASK
Passage: {context}
Question: {question}
Answer:
```

C.3 GSM8K: I (Instructions Only)

```
## INSTRUCTIONS
Show your solution step by step,
then end with: Answer: <number>

The final answer must be:
- Just the numerical value
- No units
- An integer if whole number

### Solution Structure
Step 1: Understand the problem
Step 2: Plan the solution
Step 3: Execute calculations
Step 4: State the answer

### Common Mistakes to Avoid
- Misreading the question
- Arithmetic errors
- Order of operations
[... detailed guidance ...]

## TASK
Problem: {question}
Solution:
```

C.4 MultiWOZ: E (Examples Only)

```
## EXAMPLES
Example 1:
Dialogue:
USER: i need a place to dine in the
  center thats expensive
SYSTEM: I have several options; do you
  prefer African, Asian, or British?
USER: Any sort of food would be fine.
  Could I get the phone number?
Slots: restaurant-area: centre,
  restaurant-pricerange: expensive

Example 2:
Dialogue:
USER: I need a train from cambridge
  to london on tuesday.
Slots: train-departure: cambridge,
  train-destination: london,
  train-day: tuesday

[... 78 more examples ...]

## TASK
Dialogue:
{dialogue}
Slots:
```

D Silver Data Examples

Silver training data is generated by prompting the model with the full S+E+I configuration and collecting outputs. The LoRA adapter is then trained on these (input, output) pairs using the S+I prompt (no examples).

D.1 CUAD Silver Example

```
  Training Input (S+I prompt)
## SYSTEM
You are a legal contract analysis
system ...
## APPLICATIONS
[clause type taxonomy]
## INSTRUCTIONS
[extraction rules]
## TASK
Passage: EXHIBIT 10.6 DISTRIBUTOR
AGREEMENT THIS DISTRIBUTOR
(the "Agreement") is made by and
between Electric City Corp. ...
Question: Highlight the parts related
to "Document Name".
Answer:

  Training Target (silver output)
DISTRIBUTOR AGREEMENT

  Gold Label
DISTRIBUTOR AGREEMENT
```

D.2 GSM8K Silver Example

```
  Training Input (S+I prompt)
## INSTRUCTIONS
[step-by-step guidance]
## TASK
Problem: Natalia sold clips to 48 of
her friends in April, and then she
sold half as many clips in May. How
many clips did Natalia sell altogether?

Solution:

  Training Target (silver output)
Step 1: In April, Natalia sold 48 clips
Step 2: In May, she sold half: 48/2 = 24
Step 3: Total = 48 + 24 = 72
Answer: 72

  Gold Answer: 72 (Correct: True)
```

D.3 MultiWOZ Silver Example

```
  Training Input (S+I prompt)
## SYSTEM
You are a Dialogue State Tracking
system ...
## APPLICATIONS
[domain and slot definitions]
## INSTRUCTIONS
[extraction and formatting rules]
## TASK
Dialogue:
USER: i need a place to dine in the
  center thats expensive
SYSTEM: I have several options ...
USER: Sounds good. Also, could you
  recommend an expensive hotel?
Slots:

  Training Target (silver output)
hotel-bookday: saturday,
hotel-bookpeople: 2,
```

```
hotel-bookstay: 2,
hotel-name: university arms hotel,
hotel-pricerange: expensive,
hotel-type: hotel
```

E Silver Data Statistics

	CUAD	GSM8K	MultiWOZ
Generated	3,000	2,000	1,992
After filtering	1,134	1,673	1,984
Retention rate	38%	84%	99%
Filter criterion	Word overlap	Answer correct	Format valid
Train split	1,021	1,506	1,786
Eval split	113	167	198

Table 8: Silver data statistics. CUAD has the lowest retention due to strict verbatim extraction filtering ($\geq$80% word overlap with passage).

E.1 Silver Data Filtering Analysis

Our quality filter for CUAD retains 38% of silver examples (1,134 of 3,000) by requiring $\geq$80% word overlap between the model's output and the input passage, ensuring the model extracts rather than hallucinates. This raises a natural concern about selection bias: are retained examples systematically easier? We analyze the filtered-out examples and find they fall into two categories: (1) outputs where the model generated "Not found" or empty responses, and (2) outputs where the model hallucinated text not present in the passage (374 examples with average output length of 11.5 words vs 21.2 for retained). The filter thus primarily removes generation failures, not difficult examples. However, retention rates do vary by clause type: factual clauses (document names: 98%, agreement dates: 87%) are retained at higher rates than complex legal provisions (liability caps: 8%, IP ownership: 0%). This is consistent with findings in self-training literature that model confidence correlates with task simplicity (Hsieh et al., 2023), and quality filtering inherently favors examples where the teacher model performs well. Critically, this bias does not prevent generalization. The test set spans 24 clause types including complex provisions (IP ownership, liability caps, anti-assignment) with low or zero training retention. LoRA achieves 33.4% F1 across this full distribution, exceeding the examples-only baseline (31.6%). This suggests that patterns learned from simpler clause types (extraction boundaries, output formatting) transfer to harder ones consistent with Min et al. (2022)'s finding that in-context examples primarily convey format and distribution rather than task-specific factual knowledge.

E.2 Behavioral Validation

To validate that LoRA internalizes example behavior rather than learning independent patterns, we compare outputs across conditions and categorize prediction errors (Table 9). Across all three tasks, LoRA outputs are 2–4$\times$ more likely to match E outputs than S+I outputs (CUAD: 15% vs 5% exact match with E; GSM8K: 43% vs 22%; MultiWOZ: 33% vs 10%), confirming the adapter shifts behavior toward the example-augmented condition.

	Qwen			Mistral		
Category	S+I	E	LoRA	S+I	E	LoRA
Exact match	5	12	**14**	0	14	**13**
Partial	8	**35**	35	30	**43**	42
Hallucination	52	**53**	34	**69**	43	45
Empty	**35**	0	17	1	0	0
F1	.138	.316	**.334**	.249	.384	**.358**

Table 9: Error distribution on CUAD ($n = 100$, percentages). LoRA reduces hallucination and increases exact matches relative to S+I, closely tracking E's distribution.

LoRA achieves the highest F1 (.334 Qwen, .358 Mistral), reducing hallucination (Qwen: 52%$\rightarrow$34%; Mistral: 69%$\rightarrow$45%) while increasing exact matches. Both models achieve strong F1 through LoRA but via qualitatively different mechanisms — Qwen learns abstention (17% empty), Mistral learns aggressive extraction — revealing architecture-dependent distillation effects.