PROMPT-MII: META-LEARNING INSTRUCTION INDUCTION FOR LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

A popular method to adapt large language models (LLMs) to new tasks is in-context learning (ICL), which is effective but incurs high inference costs as context length grows. An alternative approach is to perform *instruction induction*, where we take training examples and reduce them to a compact but descriptive prompt that can achieve performance comparable to ICL over the full training set. We propose PROMPT-MII, a reinforcement learning (RL) based framework to *meta-learn* an instruction induction model that can generate compact instructions on the fly for an arbitrary new dataset. We train on over 3,000 diverse classification datasets from the HuggingFace hub, and evaluate on 90 unseen tasks. PROMPT-MII improves downstream model quality by 4-9 F1 points (10-20% relative), matching ICL performance while requiring 3-13x fewer tokens. All code, data, and models will be released to the research community at `https://anonymized`.

1 INTRODUCTION

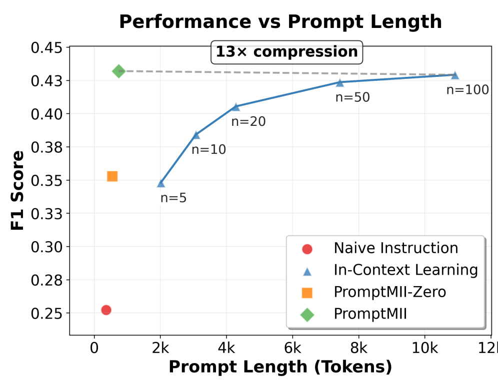

Figure 1: Classification results averaged over 90 datasets using the Llama-3.1-8B-Instruct model. PROMPT-MII achieves performance comparable to ICL while using 13× fewer tokens.

One common usage patterns for large language models (LLMs) is to adapt them to a particular task at hand. In a supervised adaptation scenario, we are given n labeled demonstrations $S_{\text{train}} = \{(x_k, y_k)\}_{k=1}^{n}$ and are interested in the problem of how to accurately predict labels for a set of test examples $S_{\text{test}} = \{(x_j, y_j)\}_{j=1}^{m}$ drawn from the same distribution.

There are multiple typical ways to incorporate the given examples: (1) *Prompting with instructions*, where a natural language task description I is appended to the model prefix, (2) *In-context learning (ICL)*, which directly uses examples in S_{train} as context during inference, and (3) *Supervised fine-tuning (SFT)*, which performs gradient updates on S_{train} to condense the information into model parameters. Each method has its advantages. Prompting with instructions is concise and efficient but requires extensive prompt engineering (Agrawal et al., 2025). ICL achieves highly competitive performance but can be inefficient as the number of examples grows larger (Xiao et al., 2025). SFT is efficient at test time but requires significant compute at training time, storage of model weights, and underperforms ICL in many cases (Bertsch et al., 2024).

In particular, as a method to bridge the gap between ICL and prompting, there are methods proposed for *instruction induction*, which takes training data S_{train} and generates an instruction I that achieves good performance. Representative methods for instruction induction such as APE (Zhou et al., 2022) and GEPA (Agrawal et al., 2025) typically do so through a complex optimization process that generates multiple candidates for prompts and evaluates them, finding the best-performing prompt option. This raises the question: *is there a way to perform instruction induction in a way that is both effective and efficient over a wide variety of tasks?*

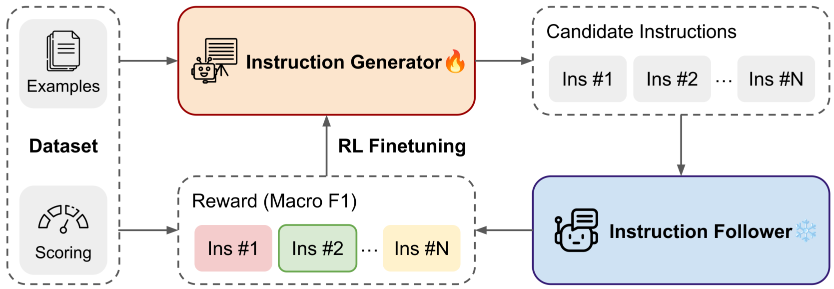

Figure 2: Overview of PROMPT-MII. We train an Instruction Generator's general ability to perform instruction induction. At inference time, given dataset examples of an unseen task, it automatically generates a reusable task instruction in a single pass, which then guides a black-box Instruction Follower model to make predictions.

As an answer to this question, we propose PROMPT-MII, we frame instruction induction as a meta-learning problem: instead of individually optimizing I for each individual task, we train an instruction induction policy π_θ that can effectively generate instructions in a single pass across diverse task distributions conditioned on the in-context examples:

$$I = \pi_\theta(S^{(i)}_{\text{train}}) \tag{1}$$

There are two major advantages to this approach. First, it allows π_θ to share knowledge about how to construct effective prompts across a wide number of datasets, instead of requiring the re-discovery of this knowledge for each dataset. Second, it has significant efficiency benefits – generating an instruction I for a new dataset simply requires a single forward pass through the language model, instead of a costly optimization process.

Experiments demonstrate PROMPT-MII to be highly effective. For instance, in Figure 1 we show how PROMPT-MII can achieve performance comparable to 100-shot ICL while consuming 13x fewer tokens. In the remainder of this paper, we discuss the methodological details of PROMPT-MII (§ 2), experimental details (§ 3), and results and analysis (§ 4).

2 PROMPT-MII: META-LEARNING INSTRUCTION INDUCTION

The main challenge in developing a method to generate instructions I from a dataset S_{test} is learning an effective policy π_θ that can generate these instructions in a way that will achieve good test performance. In this section, we develop our method for meta-learning such a policy, also shown in Figure 2.

2.1 TRAINING OBJECTIVE

Let $\mathcal{S} = \{S_1, S_2, \ldots, S_N\}$ be a collection of datasets that we will use in the meta-learning of π_θ. For each dataset S_i, we sample training examples $S^{(i)}_{\text{train}}$ for instruction generation and test examples $S^{(i)}_{\text{test}}$ for reward computation. We define a meta-prompt template $T(S^{(i)}_{\text{train}})$), which converts the dataset into a prompt to the model, as detailed in § 2.2. Then, π_θ generates an instruction prompted by this meta-prompt, $I \sim \pi_\theta(T(S^{(i)}_{\text{train}}))$.

To assess the quality of the generated instruction, we use a separate frozen language model LM_{eval} as the instruction follower. This LM then processes the test set S_{test} using this instruction, generating results $\hat{y}_j = \text{LM}_{\text{eval}}(I + \text{"Input: "} + x_j + \text{"Label:"})$. We use a task-dependent evaluation metric over m test examples to assess the model performance $E\big(\{\hat{y}_j\}_{j=1}^{m},\ \{y_j\}_{j=1}^{m}\big)$. In principle, this metric can range from classification metrics such as accuracy and macro-F1 to generation based metrics such as LLM-as-a-judge, but in this work we focus on classification tasks and use macro-F1 as our target reward metric and $m = 20$ to balance stability and efficiency. To avoid training the model to learn the format requirement that is easily enforced manually, we add this custom format line: `Only` `return one of these options: {label_names}. Do not output "Label:" or any extra text.` after the generated instruction, before calculating the reward. This constraint is equally added to all baseline methods we compare in the results.

Together, this results in a reward for our generated instruction of

$$R(I, S_{\text{test}}) = E\big(\{\hat{y}_j\}_{j=1}^m,\ \{y_j\}_{j=1}^m\big) \tag{2}$$

Once we have defined this reward, it can be optimized with an RL algorithm of choice. In this work, we use Group Relative Policy Optimization (GRPO; Shao et al. (2024)) and enhance the algorithm with asymmetric clipping and removal of KL loss, which has been shown to encourage more exploration (Yu et al., 2025). Full details of the RL objective are in § A.1.

2.2 Meta-Prompt Template

One key element of our method is the use of a meta-prompt template T that encourages the LLM to generate instructions with generalizable patterns rather than regurgitating specific examples or simply summarizing the label space.

Meta-prompt design impacting prompt quality is a known phenomenon in automatic prompt optimization (APO) methods (Ding et al., 2025). Our ablation studies in § 4.5 reveal model-dependent preferences, and accordingly, we use model-specific meta-prompts optimized for each model, but fix the same template for training and evaluation of all baselines.

Meta-Prompt Template (Qwen)

You are designing a clear instruction for a data annotator to classify text inputs into one of these labels: {label_names}
Here are some example inputs and their correct labels: {examples}
Your task is to write a concise instruction that:

- Defines the classification task and clearly explains the meaning of each label.
- Provides general labeling strategies and decision rules so annotators can correctly handle unseen inputs.
- Highlights common pitfalls, tricky edge cases, and misconceptions to reduce labeling errors.
- Keeps the instruction reasonably concise and focused — avoid unnecessary repetition or overly long explanations.

Here, `{label_names}` is a comma-separated list of all of the labels in the classification dataset S_{train} (e.g., "positive, negative, neutral") and `{examples}` follows the format: `Text: "example input text here"\nLabel: example_label`. See § A.6 for details.

3 Experiments

3.1 Data Preparation

We collected all publicly available text classification datasets from HuggingFace and applied automated filtering and multi-pass example generation as described in § A.2. After filtering, we obtained 3,811 diverse datasets, which were randomly split into 3,430 for training and 381 for validation and ensured there is no overlap between the two sets. See Figure 6 for full list of datasets and statistics.

3.2 Training Setup

We conducted training using the VERL framework (Sheng et al., 2024) on two model variants: `Llama-3.1-8B-Instruct` and `Qwen-2.5-7B-Instruct`. For each variant, we used the same architecture for both the instruction generator (π_θ) and the instruction follower (LM_{eval}). While LM_{eval} was kept frozen at the official checkpoint, π_θ was updated during training. We used a rollout size of $n = 5$, batch size of 64, maximum response length of 1k tokens and maximum prompt length of 4k tokens. Further hyperparameter and system details are provided in § A.3.

3.3 Evaluation Setup

Data We evaluated on 90 held-out datasets that were disjoint from the training data. For each dataset and each $n \in \{5, 10, 20, 50, 100\}$, we sampled n demonstration examples, generated instructions, and applied them to 200 test examples for each of the 5 settings. The context length was limited to 32k tokens. If the n examples exceeded this limit (applicable to ICL and Prompt-MII), we used the maximum value of n that fit within the context. See § A.2 for further details on dataset selection.

Baselines We compared our method against naive instruction, in-context learning (ICL), untrained instruction generation, and large model baselines (`Llama-3.1-405B-Instruct`, `Qwen-3-235B-Instruct`). We also considered iterative prompting methods APE and GEPA. Since our datasets do not provide ground-truth instructions, baselines were implemented as described in § A.4.

Metrics Our primary evaluation metric was the macro-F1 score, consistent with the training reward. We additionally report win rates (the percentage of datasets where one method outperforms another) and prompt token length; Additional results are provided in § A.5.

4 Results

4.1 Prompt-MII Successfully Generates Concise and Effective Instructions

RL training consistently improves instruction generation across held-out tasks, providing the first evidence that one-pass instruction induction is a skill learnable by language models. As shown in Figure 7 and Table 1, Llama Prompt-MII (trained) achieves +0.090 absolute F1 improvement over Prompt-MII-Zero (untrained) at n=20 (26% relative gain), while Qwen Prompt-MII shows +0.051 absolute improvement (15% relative gain).

We observe that training conducted with limited context length of 4k context length is able to have improvements generalized to 32k context length. Notably, Llama Prompt-MII using n=20 examples (0.433 F1, 901 tokens) matches ICL performance using n=100 examples (0.430 F1, 11,531 tokens), representing a 12.8× token reduction with no statistical difference in performance, as shown in Table 1 and Figure 3

From our win rate analysis (Figure 9, Appendix), Prompt-MII has a similar win rate to ICL (approximately 50-50) for both models, suggesting that it is a strong alternative for practitioners to consider.

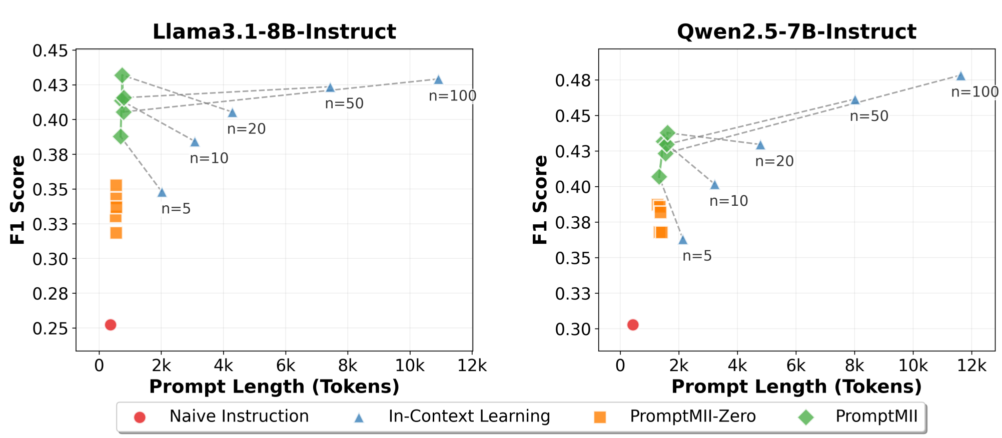

Figure 3: Performance vs prompt length comparison across different prompting methods. Prompt-MII (green diamonds) consistently outperforms other methods while using fewer tokens than ICL (blue triangles). Dashed lines connect ICL and trained methods for the same number of examples (n), demonstrating prompt compression while maintaining performance.

Table 1: Token efficiency comparison: macro-F1 performance (higher is better) with instruction token length underneath (lower is better). Statistical significance markers (* $p < 0.05$, *** $p < 0.001$) indicate significant differences between **PROMPT-MII** and **ICL** methods (Wilcoxon signed-rank test).

Method	Llama3.1-8B n=5	n=10	n=20	n=50	n=100	Qwen2.5-7B n=5	n=10	n=20	n=50	n=100
Naive	0.253	0.253	0.253	0.253	0.253	0.303	0.303	0.303	0.303	0.303
	531	531	531	531	531	609	609	609	609	609
ICL	0.347	0.385	0.406	**0.424**	**0.430**	0.363	0.403	0.431	**0.463**	**0.482**
	2451	3594	5177	8206	11531	2597	3765	5390	8539	12027
PROMPT-MII-Zero	0.316	0.329	0.343	0.354	0.336	0.369	0.390	0.383	0.387	0.371
	709	702	709	710	715	1541	1481	1574	1538	1609
PROMPT-MII	**0.388***	**0.415**	**0.433**	0.416	0.405*	**0.409*****	**0.434***	**0.441**	0.432*	0.424***
	873	891	901	965	956	1523	1677	1807	1774	1737

Table 2: Comparison of PROMPT-MII against APE and GEPA optimization methods. Performance shown as macro-F1 scores for different model and example count (n) combinations.

Methods	Llama (n=50)	Llama (n=100)	Qwen (n=50)	Qwen (n=100)
Naive	0.253	0.253	0.303	0.303
APE	0.278	0.288	0.358	0.356
GEPA	0.296	0.299	0.346	0.347
PROMPT-MII	0.416	0.405	0.432	0.424

4.2 PROMPT-MII OUTPERFORMS EXPLICIT OPTIMIZATION TECHNIQUES

PROMPT-MII substantially outperforms iterative prompt optimization methods despite requiring only a single forward pass. As shown in Table 2, PROMPT-MII achieves 0.405-0.432 F1 compared to APE's 0.288-0.358 and GEPA's 0.296-0.347, while using much fewer LLM calls (1 vs 150 for GEPA, and 2000 for APE, see details in Appendix A.5).

Even when controlling for meta-prompt template (Table 5), APE with our meta-prompt template still underperforms PROMPT-MII-Zero and significantly underperforms PROMPT-MII. This performance gap compared with APE and GEPA likely stems from: (1) `Qwen 2.5 7B Instruct` and `Llama 3.1 8B Instruct` may be too small to reflect on its own mistakes helpfully (larger reflection models like might perform better), (2) Classification tasks may be challenging for iterative refinement algorithms, as they require understanding patterns across distributions rather than single examples. This pattern recognition ability is critical for classification and regression, but less essential for generative tasks like QA or summarization.

To elaborate further, a few concrete hypothesis for why classification tasks may be challenging for iterative refinement algorithms are: (a) Limited feedback signal: generative tasks like multihop QA emit traces (reasoning, tool outputs etc). Classification gives only a label/correctness, offering little to reflect on. (b) Difficult credit assignment: modular generative pipelines localize errors to specific modules (in GEPA a human defines the modules). Classification doesn't have modules, so edits are global. (c) Noise and overfitting: iterative refinement methods use small mini batches for each refinement step (e.g. GEPA uses 3 examples). For classification tasks, the very few examples may not represent the overall distribution, so recent edits may override/corrupt the existing instruction, or only accumulate error case descriptions, which defeats the purpose of trying to compress into a shorter prompt.

Even though the current baselines underperform, we see opportunities for iterative refinement to work on classification, especially in conjunction with Prompt-MII, as described in Discussions Section.

4.3 FOR WHICH DATASETS DOES PROMPT-MII EXCEL?

Per example length. First, we perform an analysis separately over datasets with relatively short ICL examples (under 46 tokens on average) and relatively long ICL examples (more than 220 tokens on average). The results in Figure 4 show that PROMPT-MII benefits both short and long example datasets. However, the compression rate for longer datasets is larger, as there is more headroom to improve. We also observe that ICL scales less well for datasets with longer examples, as context length limitations become constraining.

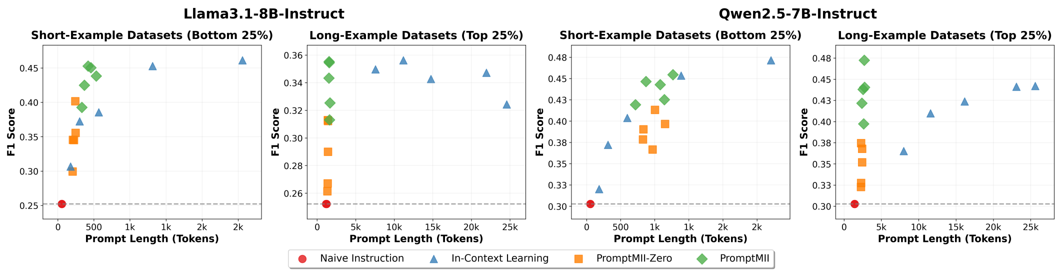

Figure 4: Analysis of when PROMPT-MII excels over ICL by per example token length.

Analysis on Data Contamination and Similarity between Training and Evaluation Datasets. Our process for splitting the train/test datasets is by dataset name, so there might be a mix of in-domain and out-of-domain datasets in the evaluation. Therefore, we performed analysis using duplication check and embedding based dataset similarity check to provide additional insights.

1) Data contamination analysis. To test training-test leakage, we performed exact MD5 hashing across all examples. Resulting leakage rate: 0.35% (70/19800 test examples). This confirms that the evaluation set is disjoint from the training set on the input content level.

2) Embedding similarity based analysis. To measure generalization, we judge how similar two datasets are through semantic embedding cosine similarity. For each dataset we sampled 200 input text, computed MPNet embeddings for each, and averaged them into a single dataset embedding. We then computed average KNN similarity (k=10) of each test dataset compared to 3000+ training datasets, binning by similarity thresholds. We group our experiment results by bin, and below are the results for Llama 3.1 8B Instruct. 3

Across all three bins, Prompt-MII consistently improves over Prompt-MII-Zero (untrained) and achieves performance comparable to/better than 100-shot ICL.

In addition, there is a natural out-of-distribution scenario in our setup, which is by per example token length. For training we limit to 4k input token length, so datasets with longer examples are not seen during training. In the previous section we show that Prompt-MII is able to generalize to those datasets and have high compression ratios.

Together, these results show that our method has no meaningful contamination with the training set, and generalizes strongly even to the most dissimilar datasets.

Case Analysis. In the following figure, we display some (abbreviated) example prompts to provide an intuition of where PROMPT-MII may outperforms PROMPT-MII-Zero and ICL for `Llama3.1-8B-Instruct`. All methods uses the same set of n=10 examples as input. Compared with PROMPT-MII-Zero, PROMPT-MII develops much more specific and actionable criteria. While PROMPT-MII-Zero provides vague cues like "Useful cues include the tone and language used", PROMPT-MII provides specific guidelines on when to predict the input a certain label, with specific examples and keywords. In this case, both PROMPT-MII and PROMPT-MII-Zero also outperform many-shot ICL.

Table 3: Performance across dataset similarity groups (measured by kNN embedding similarity). *Note that datasets with high input similarity (>0.85) doesn't necessarily mean the classification task is the same, since the embedding is only based on the input text, not the labels.

Method	n=5	n=10	n=20	n=50	n=100
Similar Group (similarity $>$ 0.85, 49 datasets)					
Naive	0.252	0.252	0.252	0.252	0.252
ICL	0.345	0.383	0.403	0.420	0.426
Prompt-MII-Zero	0.313	0.326	0.339	0.351	0.332
Prompt-MII	0.389	0.417	0.435	0.418	0.407
Moderate Group (0.50–0.85, 39 datasets)					
Naive	0.245	0.245	0.245	0.245	0.245
ICL	0.334	0.373	0.394	0.415	0.420
Prompt-MII-Zero	0.301	0.314	0.327	0.336	0.318
Prompt-MII	0.376	0.402	0.418	0.406	0.396
Dissimilar Group (similarity $<$ 0.50, 5 datasets)					
Naive	0.256	0.256	0.256	0.257	0.256
ICL	0.380	0.422	0.443	0.456	0.456
Prompt-MII-Zero	0.334	0.349	0.364	0.376	0.356
Prompt-MII	0.408	0.436	0.455	0.440	0.521

PROMPT-MII-Zero

Classify the input text as one of the following labels:1;0;2;3. The task is to determine whether the input text is a question or request for advice (label 0); a statement or opinion (label 1); a spam or promotional message (label 2); or an off-topic or unrelated message (label 3).
Useful clues for making the decision include:
- The presence of a question or request for help; which is often indicated by words or phrases such as 'I need'...
- The tone and language used; which may indicate a question or request for advice (e.g. polite language; uncertainty; or a sense of seeking guidance).
- The content of the text; which may be related to a specific topic or subject (e.g. computer hardware; medical careers; or cryptocurrency).
Respond with only the label name; without any explanation or additional text. Only return one of these options: 1; 0; 2; 3. Do not output 'Label:' or any extra text.

F1: 0.241

PROMPT-MII

Classify each input into one of the following categories based on its content and purpose:
- Label 0: This label is for inputs that are asking for advice; guidance; or recommendations on building or upgrading a computer; purchasing computer components; or troubleshooting computer-related issues...
- Label 3: This label is for inputs that are unrelated to computer hardware or software and are instead focused on other topics; such as business; finance; or cryptocurrency...
- Label 2: This label is for inputs that are asking for advice or guidance on non-computer related topics; such as education; career; or personal development...
- Label 1: This label is for inputs that do not fit into any of the above categories. If an input is unclear or does not...
Respond with the corresponding label (0; 1; 2; or 3) only...
Only return one of these options: 1; 0; 2; 3. Do not output 'Label:' or any extra text.

F1: 0.829

In-Context Learning

Input: Not my first build but it s been 10 years since I built one. Have some questions. Specs B550m ds3h Ac motherboard...
Label: 0
Input: Cpu and cooler for 3080ti? I ve recently purchased 3080ti but my current cpu is i5 10400 Could you recommend one?
Label: 0
Input: need serious explaining and help I use to just play on my PS4; then It broke and I could get it fixed but I've always wanted a gaming pc. Before I ask to build one I need to understand the parts and what they do; which I don't know anything about so this...
Label: 0
Only return one of these options: 1; 0; 2; 3. Do not output 'Label:' or any extra text.

F1: 0.026

4.4 CROSS-MODEL TRANSFER

An advantage of Instruction Induction compared to finetuning or soft-prompt is that Instruction Induction is in natural language and therefore transferrable to another black-box instruction follower model.

Larger Models Instruct, Smaller Models Follow We evaluate whether large models can generate effective instructions for smaller instruction-following models. Figure 5 demonstrates that

`Llama3.1-405B` PROMPT-MII-Zero and Qwen3-235B PROMPT-MII-Zero successfully generate instructions that work well with their smaller counterparts. However, surprisingly, our PROMPT-MII `Llama3.1-8B` outperforms the much larger `Llama3.1-405B` (Figure 5).

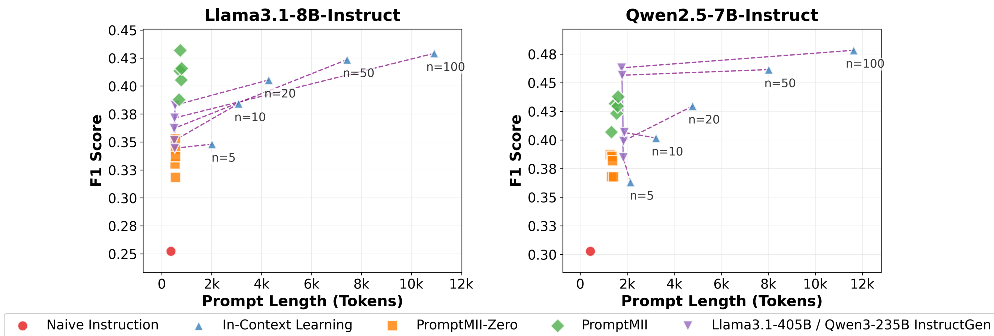

Figure 5: Cross-model transfer results showing large model instruction generation capabilities. Purple dashed lines connect larger model performance (Llama3.1-405B and Qwen3-235B) to ICL baselines for the same number of examples, demonstrating that large models can generate effective instructions off-the-shelf.

Cross-model Transfer We investigate whether PROMPT-MII trained with one follower model can generalize to different follower models at evaluation time. According to our ablation results (Table 6, Appendix), cross-model transfer is feasible but suboptimal compared to same-model combinations. For instance, PROMPT-MII Llama → Qwen follower (0.391-0.415 F1) outperforms PROMPT-MII-Zero on Qwen (0.369-0.390 F1), demonstrating that training benefits partially transfer across models. However, it underperforms PROMPT-MII Qwen → Qwen follower (0.409-0.441 F1), revealing model-specific preferred instruction patterns. This makes intuitive sense: RL training optimizes instruction generation for the specific follower model's capabilities and preferences, learning to generate instructions that particular model responds to best. Future work can also explore larger models for instruction followers, in this work for practicality, we fix the instruction follower model to smaller model, as instruction following may be applied to many test queries.

4.5 IMPORTANCE OF META-PROMPT TEMPLATE

The choice of meta-prompt template impacts instruction generation quality, and optimal templates are model-dependent. We compare two meta-prompts evaluated on both Llama3.1-8B and Qwen2.5-7B models. We also compare against a naive meta prompt, identical to the one used in (Honovich et al., 2022).

Table 4: Meta-Prompt Template Comparison: F1 Performance Across Models

Method	**Llama**		**Qwen**	
	n=50	**n=100**	**n=50**	**n=100**
Naive	0.253	0.253	0.303	0.303
PROMPT-MII-Zero (naive)	0.287	0.272	0.343	0.360
PROMPT-MII-Zero (meta1)	**0.354**	**0.336**	0.356	0.340
PROMPT-MII-Zero (meta2)	0.301	0.296	**0.387**	**0.371**

Both meta-prompt templates outperform naive instruction, but the results reveal model-dependent preferences: Llama3.1-8B performs better with meta1 (+0.053 F1 vs meta2), while Qwen2.5-7B achieves superior results with meta2 (+0.031 F1 vs meta1). In this work to optimize performance, we use meta1 for Llama3.1-8B and meta2 for Qwen2.5-7B. Future work could explore inference-time search or automated methods to select the most effective meta-prompt.

5 RELATED WORK

Instruction Induction Instruction Induction is a category of automatic prompt optimization techniques (APO) that takes in examples as input and induces a task instruction without requiring a custom hand-written seed prompt. Honovich et al. (2022) was the first to propose the problem definition of instruction induction from few-shot examples, showing that it is feasible with GPT-3 on simple tasks like "capitalize the first letter" or "find the longest word.", which had near-perfect ground truth instructions expressible in one sentence. Our work shares a similar problem definition but extending few examples to many examples, and testing on arbitrary classification tasks with ambiguous decision boundaries and often no ground truth available.

More recent methods like APE (Zhou et al., 2022) and GEPA (Agrawal et al., 2025) and a few others (Choi et al., 2025; Fernando et al., 2023) cast instruction induction as an evolutionary search problem: APE iteratively proposes and rewrites candidate prompts from examples and selects the best one on a validation split, while GEPA performs genetic–Pareto optimization with reflective changes for LLM programs. Despite their effectiveness, both require extensive test-time search and many LLM calls, whereas PROMPT-MII produces a reusable instruction in a single pass, avoiding per-task optimization at inference time.

Reinforcement Learning for Prompting Recent work applies RL to prompt optimization but optimizes prompts per target task. RLPrompt Deng et al. (2022) formulates discrete prompt optimization as a reinforcement-learning policy that generates task prompts directly, often yielding non-natural ("gibberish/ungammatical") outputs. PRewrite Zhang et al. (2024) trains a prompt rewriter LLM with RL to take an under-optimized prompt for a given downstream task and rewrite it into a higher-performing prompt. PRL Batorski et al. (2025) uses RL to perform instruction induction, but also trains a new policy per each task. In contrast, PROMPT-MII learns a general instruction-induction capability that transfers to unseen tasks, eliminating per-task training at test time.

Ha et al. (2023) also meta-learns a instruction induction model, however they use supervised fine-tuning instead of RL. This requires having ground truth instructions for many datasets, but this is limited in size and it is difficult to obtain instructions that fully capture the task or dataset distribution, even with human expert labeling. With RL training in Prompt-MII, the ground truth is not required, allowing much more training data, and the model learns to explore beyond human written instructions.

Prompt Compression Prompt compression approaches can be broadly categorized as discrete or continuous. Discrete methods either filter tokens (might happen at the cost of readability) or paraphrase the text to preserve semantics more fluently Xiao et al. (2024). Recent work such as LLMLingua-2 Pan et al. (2024) report approximately 3× compression on both long-context and short-context tasks while maintaining performance. In contrast, Our approach changes the semantic meaning of the prompt from examples to task description. This represents a fundamentally different compression paradigm that could be combined with token-level methods for additional gains. Continuous methods (e.g., soft prompts Lester et al. (2021)) operate in a latent space and are generally not interpretable; since we focus on interpretable, black-box-compatible compression, we omit comparing against soft-prompt or other latent compression techniques.

6 DISCUSSION AND FUTURE WORK

We present PROMPT-MII as an automatic prompting strategy that has the advantage of 1) producing an instruction prompt that is shared among all test queries 2) being optimization-free at test-time, requiring only a single-pass inference, and 3) meta-learning instruction induction ability that generalize to unseen tasks. In this paper, we show that PROMPT-MII is effective on diverse classification tasks, which represent a common and important application for LLMs, such as LLM-as-a-judge systems Gu et al. (2025), but has future potential to extend to generative tasks as well.

One potential interpretation for why PROMPT-MII is effective is that instruction induction acts as pre-chain-of-thought by analyzing relationships among examples and incorporating prior knowledge. Regular chain-of-thought Wei et al. (2023) is expensive because it must be performed at request time for every query, while instruction induction front-loads this reasoning process, enabling computational savings through prefix-caching across multiple test queries.

Ultimately, the goal is to generate instruction from an entire dataset, which presents two challenging directions. 1) Strong long-context capability. Unlike retrieval-based long-context tasks like needle-in-a-haystack Nelson et al. (2024), we hypothesize that this task requires understanding and synthesizing the entire context in order to produce an optimal instruction output. 2) Distribution-aware iterative refinement methods. If processing entire datasets in one pass proves sub-optimal, we can incorporate intermediate reasoning, or iterative refinement methods that process groups of examples sequentially. This can potentially complement PROMPT-MII, but as hypothesized in our analysis, for classification tasks we need an iterative process that is memory-preserving and distribution-aware, where it would continuously refine a natual language "decision boundary".

Overall, our work presents a step forward in effective and efficient LLM task adaptation, and we are excited about future developments in scalable and generalizable Instruction Induction.

REPRODUCIBILITY STATEMENT.

We provide detailed information for reproducibility in the Appendix: § A.2 for data preparation, § A.3 for training configuration, and § A.4 for baseline implementation. We will also release the complete codebase required to run the experiments and generate all figures. All experiments are ran with a fixed random seed and fully reproducible.

THE USE OF LLMS

We acknowledge the use of LLMs in writing this paper. The use was limited to correcting grammar and improving clarity. All research ideas, method design, and experiments were conducted solely by the authors.

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. Gepa: Reflective prompt evolution can outperform reinforcement learning, 2025. URL https://arxiv.org/abs/2507.19457.

Paweł Batorski, Adrian Kosmala, and Paul Swoboda. Prl: Prompts from reinforcement learning, 2025. URL https://arxiv.org/abs/2505.14412.

Yoshua Bengio and Yann LeCun. Scaling learning algorithms towards AI. In *Large Scale Kernel Machines*. MIT Press, 2007.

Amanda Bertsch, Maor Ivgi, Emily Xiao, Uri Alon, Jonathan Berant, Matthew R Gormley, and Graham Neubig. In-context learning with long-context models: An in-depth exploration. *arXiv preprint arXiv:2405.00200*, 2024.

Yumin Choi, Jinheon Baek, and Sung Ju Hwang. System prompt optimization with meta-learning, 2025. URL https://arxiv.org/abs/2505.09666.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric P. Xing, and Zhiting Hu. Rlprompt: Optimizing discrete text prompts with reinforcement learning, 2022. URL https://arxiv.org/abs/2205.12548.

Han Ding, Sangmin Woo, Shuai Wang, Haozhu Wang, Panpan Xu, Xuan Qi, Yuzhe Lu, Zhichao Xu, Balasubramaniam Srinivasan, Kang Zhou, Kiran Ramnath, Zhengyuan Shen, Haibo Ding, Sheng Guan, Sullam Jeoung, Yun Zhou, Yawei Wang, Lin Lee Cheong, Yueyan Chen, Soumya Smruti Mishra, and Qiaojing Yan. A systematic survey of automatic prompt optimization techniques, 2025. URL https://arxiv.org/abs/2502.16923.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, Lei Li, and Zhifang Sui. A survey on in-context learning. pp. 1107–1128, 2022.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution, 2023. URL https://arxiv.org/abs/2309.16797.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press, 2016.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Saizhuo Wang, Kun Zhang, Yuanzhuo Wang, Wen Gao, Lionel Ni, and Jian Guo. A survey on llm-as-a-judge, 2025. URL https://arxiv.org/abs/2411.15594.

Hyeonmin Ha, Jihye Lee, Wookje Han, and Byung-Gon Chun. Meta-learning of prompt generation for lightweight prompt engineering on language-model-as-a-service. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 2433–2445, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.159. URL https://aclanthology.org/2023.findings-emnlp.159/.

Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

Martin Hirzel, Claudio Spiess, Mandana Vaziri, and Louis Mandel. Autopdl: Automatic prompt optimization for llm agents, 2025. URL https://arxiv.org/abs/2504.04365.

Or Honovich, Uri Shaham, Samuel R. Bowman, and Omer Levy. Instruction induction: From few examples to natural language task descriptions. *ArXiv*, abs/2205.10782, 2022.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. 2025.

Haeil Lee, Junmo Kim, Minchan Kwon, Gaeun Kim, and Jongsuk Kim. Stableprompt: Automatic prompt tuning using reinforcement learning for large language models, 2024. URL https://arxiv.org/abs/2410.07652.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning, 2021. URL https://arxiv.org/abs/2104.08691.

Emmy Liu, Graham Neubig, and Jacob Andreas. An incomplete loop: Instruction inference, instruction following, and in-context learning in language models. 2024.

Michael Luo, Naman Jain, Jaskirat Singh, Sijun Tan, Ameen Patel, Qingyang Wu, Alpay Ariyak, Colin Cai, Shang Zhu Tarun Venkat, Ben Athiwaratkun, Manan Roongta, Ce Zhang, Li Erran Li, Raluca Ada Popa, Koushik Sen, and Ion Stoica. Deepswe: Training a state-of-the-art coding agent from scratch by scaling rl. https://pretty-radio-b75.notion.site/DeepSWE-Training-a-Fully-Open-sourced-State-of-the-Art-Coding-Agent-by-Scaling-RL-22281902c1468193 2025. Notion Blog.

Elliot Nelson, Georgios Kollias, Payel Das, Subhajit Chaudhury, and Soham Dan. Needle in the haystack for memory based large language models, 2024. URL https://arxiv.org/abs/2407.01437.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. Llmlingua-2: Data distillation for efficient and faithful task-agnostic prompt compression, 2024. URL https://arxiv.org/abs/2403.12968.

Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, S. Mondal, and Aman Chadha. A systematic survey of prompt engineering in large language models: Techniques and applications. *ArXiv*, abs/2402.07927, 2024.

Sander Schulhoff, Michael Ilie, Nishant Balepur, Konstantine Kahadze, Amanda Liu, Chenglei Si, Yinheng Li, Aayush Gupta, HyoJung Han, Sevien Schulhoff, P. S. Dulepet, Saurav Vidyadhara, Dayeon Ki, Sweta Agrawal, Chau Minh Pham, Gerson C. Kroiz, Feileen Li, Hudson Tao, Ashay Srivastava, Hevander Da Costa, Saloni Gupta, Megan L. Rogers, Inna Goncearenco, Giuseppe Sarli, I. Galynker, Denis Peskoff, Marine Carpuat, Jules White, Shyamal Anadkat, Alexander Miserlis Hoyle, and Philip Resnik. The prompt report: A systematic survey of prompting techniques. *ArXiv*, abs/2406.06608, 2024.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*, 2020.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. 2025.

Yujin Tang, Robert Tjarko Lange, Edoardo Cetin, and Rujikorn Charakorn. Text-to-lora: Instant transformer adaption, 2025. URL `https://arxiv.org/abs/2506.06105`.

Volker Tresp, Hinrich Schütze, Yunpu Ma, Zifeng Ding, Ercong Nie, Xiaowen Ma, Xiufeng Yang, Sikuan Yan, Zuchao Huang, and Zonggen Li. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning, 2025. URL `https://arxiv.org/abs/2508.19828`.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL `https://arxiv.org/abs/2201.11903`.

Emily Xiao, Chin-Jou Li, Yilin Zhang, Graham Neubig, and Amanda Bertsch. Efficient many-shot in-context learning with dynamic block-sparse attention. 2025.

Tong Xiao, Jingbo Zhu, Chenglong Wang, Xiaoqian Liu, Kaiyan Chang, Songcheng Xu, and Yingfeng Luo. Efficient prompting methods for large language models: A survey, 2024. URL `https://arxiv.org/abs/2404.01077`.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Mingyang Zhang, Weize Kong, Michael Bendersky, Qiaozhu Mei, and Spurthi Amba Hombaiah. Prewrite: Prompt rewriting with reinforcement learning, 2024. URL `https://arxiv.org/abs/2401.08189`.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. *ArXiv*, abs/2211.01910, 2022.

A Appendix

A.1 RL Objective

The training objective function is:

$$J(\theta) = \mathbb{E}_{S_i \sim \mathcal{S}} \mathbb{E}_{\{I_k\}_{k=1}^{n} \sim \pi_{\theta_{\text{old}}}} \left[\frac{1}{n} \sum_{k=1}^{n} \min\left(r_k(\theta)\, A_k, \text{clip}(r_k(\theta), 1-\rho_L, 1+\rho_H)\, A_k \right) \right] \quad (3)$$

where importance ratio $r_k(\theta)$ is:

$$r_k(\theta) = \frac{\pi_\theta(I_k|T(S_{\text{train}}^{(i)}, \mathcal{L}_i))}{\pi_{\theta_{\text{old}}}(I_k|T(S_{\text{train}}^{(i)}, \mathcal{L}_i))}$$

and group-relative advantage A_k is:

$$A_k = R(I_k, S_{\text{test}}^{(i)}, \mathcal{L}_i) - \frac{1}{n}\sum_{j=1}^{n} R(I_j, S_{\text{test}}^{(i)}, \mathcal{L}_i)$$

with clipping bounds ρ_L and ρ_H set to 0.2 and 0.4.

A.2 Dataset Processing Pipeline

Automated filtering and quality control. We scraped all publicly available text classification datasets on HuggingFace and used GPT-4.1-mini to automatically identify input and label columns by analyzing dataset metadata, column names, and example entries. Datasets with more than 50% unique labels were discarded, this step is to verify that the task is a classification task.

Evaluation dataset selection We started with random selection of 100 held-out datasets that already went through the regular data processing pipeline above. Additional processing: 2 datasets dataset nlpaueb/multi_eurlex, TomTBT/pmc_open_access_xml, had two long of a label set so no examples fit into context, and were filtered. 3 had single class within 200 examples and 2 had >100/200 labels and were filtered. The same datasets with different configs but same labels were merged leaving with 90 unique datasets for evaluation

Multi-pass example generation. To balance the number of generated examples with dataset diversity, we adopted a four-pass strategy with progressively larger context sizes applied to smaller subsets of datasets. In the first pass, all datasets were used to generate examples with $n = 5$ contexts. Subsequent passes increased the context size while reducing the proportion of datasets: 30% of datasets with $n = 10$ contexts, 20% with $n = 20$ contexts, and 10% with $n = 50$ contexts. This design ensured that all datasets contributed examples, while a subset of datasets supported training with longer contexts.

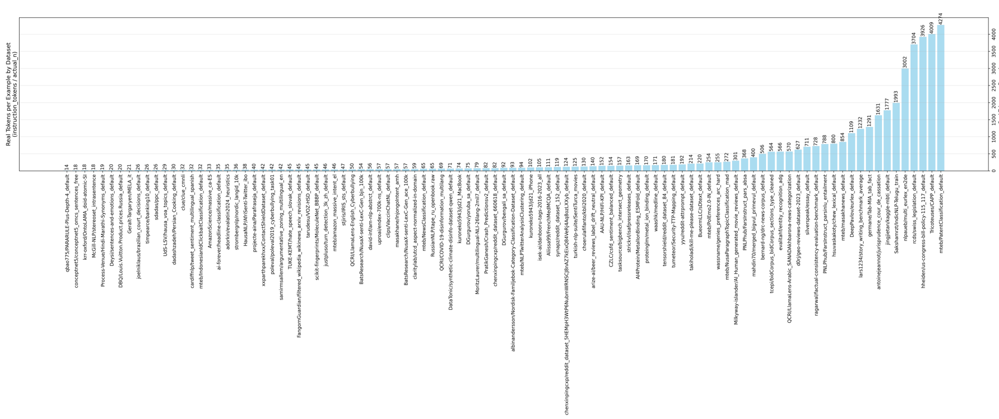

Figure 6: Distribution of example lengths in the evaluation datasets.

A.3 Detailed Training Configuration

Hyperparameters We grouped $n = 5$ instructions per prompt and set the batch size to 64 prompts. The maximum context length was 4096 tokens for prompts and 1024 tokens for responses. The model was trained with a learning rate of 2×10^{-6} with a 3.3% warmup schedule for 15 epochs.

We applied asymmetric clipping (DAPO) with `clip_ratio_low` $= 0.2$, while disabling the KL penalty (`use_kl_loss` $=$ `False`) to encourage exploration and aggregating the loss with the `seq-mean-token-mean` mode. Decoding used a temperature of 1.0 and top-$p = 1.0$.

Computational Resources We used 8 H100 GPUs per training job, with each model trained for approximately 48 hours. Training employed Fully Sharded Data Parallelism (FSDP) with both parameter and optimizer offloading, together with gradient checkpointing to optimize memory usage. To handle high concurrency (128 simultaneous requests) during batch reward computation and prefix caching, we deployed SGLang Serving for reward computation on 4 H100 GPUs, enabling efficient prefill-decode disaggregation.

A.4 Baseline Implementation Details

We append identical format constraints *"Only return one of these options: {label_names}. Do not output 'Label:' or any extra text."* to the instructions for all methods, including APE and GEPA. Without explicit constraints, responses occasionally include redundancy, which hinders reliable scoring and prompt selection.

We used Qwen2.5-7B-Instruct for both baselines (instruction generation and prediction for APE; task and reflection language model for GEPA) to ensure a fair comparison with Prompt-MII-Zero.

Prompt for Naive Baseline

Classify the Input. Only return one of these options: $\{\text{label}_1$, label_2, ... $\text{label}_n\}$. Do not output 'Label:' or any extra text.

Prompt for ICL Baseline

Classify the Input. Only return one of these options: $\{\text{label}_1$, label_2, ... $\text{label}_n\}$. Do not output 'Label:' or any extra text.
Input: {Example 1}
Label: {Label 1}
...
Input: {Test case}
Label:

Automatic Prompt Engineer (APE). We evaluated APE using both its default meta-prompt and a custom meta-prompt derived from Prompt-MII. Our setup followed the instruction induction experiments in Zhou et al. (2022), using the same hyperparameters. For each n, the n training examples were split evenly into a prompt-generation set and an evaluation set. While initial experiments used accuracy as the selection metric, we found that using F1 score yielded higher final F1 scores on the test subset.

GEPA (Genetic-Pareto). We split the n training examples into training and validation sets in a 1:2 ratio, following the procedure in the original paper for most datasets. We implemented a Classification Adapter based on the default GEPA adapter, with only minor modifications to the language model invocation logic. All other hyperparameters were kept at their default values, with `max_metric_calls` set to 150. The seed prompt was initialized with our naive instruction prompt.

Baseline	F1 (n=50)	F1 (n=100)
APE	0.358	0.356
APE_META	0.353	0.384

Table 5: F1 score comparison of APE using different meta-prompt. APE_META uses Prompt-MII's template, while APE uses original template.

A.5 Additional Results

Figures and tables in the appendix provide additional results: Figure 7 shows the RL training curve; Figure 8 illustrates F1 performance trends across different values of n; Table 6 reports F1 scores for different n; and Figure 9 presents win-rate matrices comparing different baselines and PROMPT-MII.

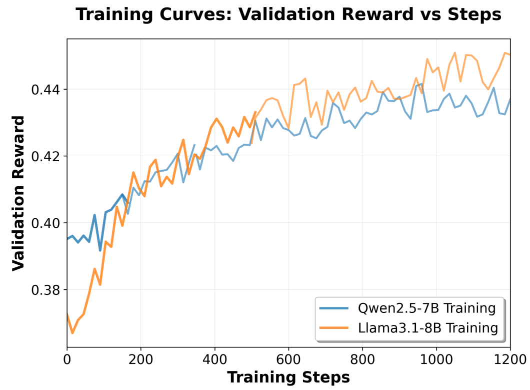

Figure 7: RL training curves of validation reward progression for Qwen2.5-7B and Llama3.1-8B.

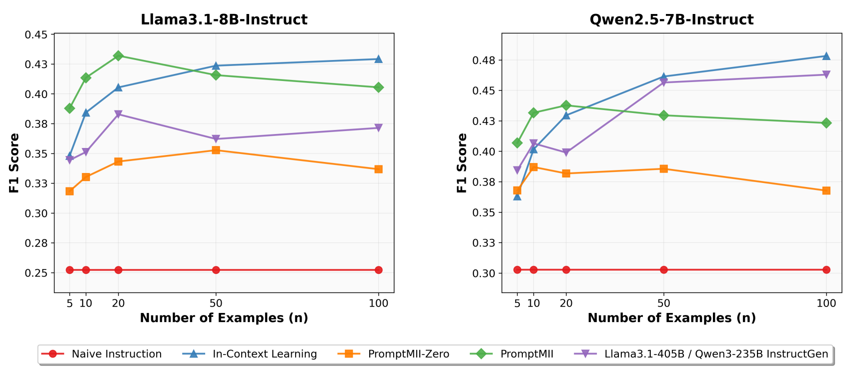

Figure 8: F1 performance trends across different values of n. The plots show how each method's performance changes as the number of training examples increases from 5 to 100. Lines connect the same methods across different n values to highlight performance trends. Notably, Qwen3-235B PROMPT-MII-Zero shows the best scalability as n increase.

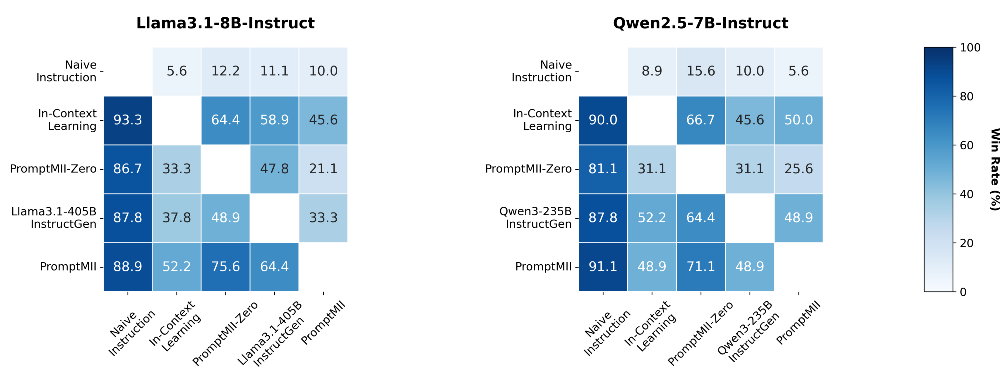

Figure 9: Win rate matrices showing pairwise comparison results between different methods. Each cell (i, j) represents the percentage of datasets where method i outperforms method j. Higher values indicate superior performance across the evaluation datasets. For Llama 3.1 8B, PROMPT-MII shows a hight winrate of 52.2% compared to ICL 45.6%

Table 6: F1 Performance across different values of n. * indicates significance between ICL and PROMPT-MII (Wilcoxon signed-rank test). All models are Instruct models instead of Base models

Llama3.1-8B-Instruct					
Method	**n=5**	**n=10**	**n=20**	**n=50**	**n=100**
Naive	0.253	0.253	0.253	0.253	0.253
ICL	0.347	0.385	0.406	**0.424**	**0.430**
PROMPT-MII-Zero	0.316	0.329	0.343	0.354	0.336
PROMPT-MII (Llama3.1-405B)	0.345	0.352	0.381	0.361	0.370
PROMPT-MII	**0.388***	**0.415**	**0.433**	0.416	0.405*
PROMPT-MII (Qwen2.5-7B)	0.342	0.358	0.353	0.347	0.311
APE	–	–	–	0.278	0.288
GEPA	–	–	–	0.296	0.299
Qwen2.5-7B-Instruct					
Method	**n=5**	**n=10**	**n=20**	**n=50**	**n=100**
Naive	0.303	0.303	0.303	0.303	0.303
ICL	0.363	0.403	0.431	**0.463**	**0.482**
PROMPT-MII-Zero	0.369	0.390	0.383	0.387	0.371
PROMPT-MII-Zero (Qwen3-235B)	0.386	0.408	0.404	0.461	0.465
PROMPT-MII	**0.409*****	**0.434***	**0.441**	0.432*	0.424***
PROMPT-MII (Llama3.1-8B)	0.391	0.412	0.438	0.434	0.415
APE	–	–	–	0.358	0.356
GEPA	–	–	–	0.346	0.347

Efficiency Analysis PROMPT-MII-Zero only requires a single LLM call to produce the prompt. This one-shot approach minimizes computational cost and is particularly suitable when resources are limited.

In contrast, the GEPA optimization framework is more compute-intensive. To generate a prompt, it takes `max_metric_calls` to evaluate all candidate prompts on minibatches and selected candidates on full validation set. Additionally, generating a new candidate instruction through reflection also requires an LLM call. A higher `max_metric_calls` allows GEPA to explore more candidate prompts but requires greater computational resources, which is a core trade-off between efficiency and performance in the GEPA framework. Therefore, in our setting, GEPA typically requires at least 150 LLM calls, while PROMPT-MII-Zero only requires one and consistently outperforms.

The APE framework is more demanding. In our setting, APE generates multiple candidate prompts by making 3 subsamples and producing 30 prompts per subsample, resulting in 90 LLM calls for prompt generation. Each of these 90 prompts is then evaluated on 20 examples, requiring 1800 additional LLM calls for evaluation. Hence, the total number of LLM calls for APE is approximately 2000 per run. This makes APE substantially more expensive than both GEPA and PROMPT-MII-Zero.

A.6 Prompt Examples & Case Study

Llama Meta-Prompt Template

You are helping to create a prompt for a language model to classify text inputs. The model should choose one label from the following options: {label_names}.

Here are some example inputs and their correct labels:
{examples}

Write an instruction that:
- Describes the classification task in a way that generalizes to new inputs.
- Points out any useful clues or strategies for making the decision.
- Clearly tells the model to respond with only the label name, and not to include any explanation or additional text.

Provide only the instruction, not the examples or labels.

Prompt-MII-Zero for Reddit Attribution Task (yyu/reddit-attrprompt)

Classify the input text as one of the following labels: 1; 0; 2; 3.
The task is to determine whether the input text is a question or request for advice (label 0); a statement or opinion (label 1); a spam or promotional message (label 2); or an off-topic or unrelated message (label 3).

Useful clues for making the decision include:
- The presence of a question or request for help; which is often indicated by words or phrases such as 'I need'; 'Can you'; 'How do I'; or 'What is'.
- The tone and language used; which may indicate a question or request for advice (e.g. polite language; uncertainty; or a sense of seeking guidance).
- The content of the text; which may be related to a specific topic or subject (e.g. computer hardware; medical careers; or cryptocurrency).

Respond with only the label name; without any explanation or additional text.
Only return one of these options: 1; 0; 2; 3. Do not output 'Label:' or any extra text.

F1: 0.241

PROMPT-MII for Reddit Attribution Task (yyu/reddit-attrprompt)

Classify each input into one of the following categories based on its content and purpose:

- Label 0: This label is for inputs that are asking for advice; guidance; or recommendations on building or upgrading a computer; purchasing computer components; or troubleshooting computer-related issues. It also includes inputs that are seeking help with understanding computer-related concepts or are looking for opinions on computer-related matters. A clue to label 0 is the presence of words like 'build;' 'upgrade;' 'advice;' 'recommendation;' 'help;' or 'opinion' in the input. It also often involves discussions about specific computer hardware or software.

- Label 3: This label is for inputs that are unrelated to computer hardware or software and are instead focused on other topics; such as business; finance; or cryptocurrency. A clue to label 3 is the presence of words like 'sell;' 'products;' 'services;' 'currency;' or 'prestige' in the input.

- Label 2: This label is for inputs that are asking for advice or guidance on non-computer related topics; such as education; career; or personal development. A clue to label 2 is the presence of words like 'school;' 'career;' 'advice;' or 'paramedic' in the input.

- Label 1: This label is for inputs that do not fit into any of the above categories. If an input is unclear or does not contain any of the clues mentioned above; it should be labeled as 1.

Respond with the corresponding label (0; 1; 2; or 3) only; without any explanation or additional text. Do not include any analysis or summary of the input in your response. Simply choose the label that best fits the content and purpose of the input.
Only return one of these options: 1; 0; 2; 3. Do not output 'Label:' or any extra text.

F1: 0.829

ICL for Reddit Attribution Task (yyu/reddit-attrprompt)

Input: Not my first build but it s been 10 years since I built one. Have some questions. Specs B550m ds3h Ac motherboard Amd Ryzen 5 3600 1TB WD blue sn550 hard drive (first time ever using one of these) 32 mb ram 800w power supply Rtx 3060 12gb graphics Plus a dvd cd So my questions are this. Do I have to have the updated flash to the bios to get the pc turned on and running? I didn t make a boot disk; and I bought a new copy of windows. Not sure if you need boot disks anymore or not or if we can just boot directly off the CD? I also intended to use my old ASUS case and install it all in there but the front panel cables are not marked and they re using 4 and 20 pin cables and I have no idea where any of that goes. I have a new case and power supply coming tomorrow. Am I missing anything? Like my title said I haven t built my own pc like this in many years. I think I had to use an old floppy to boot up windows.. if that gives you an idea lol Thanks in advanced
Label: 0
Input: Cpu and cooler for 3080ti? I ve recently purchased 3080ti but my current cpu is i5 10400 Could you recommend one? Thanks!
Label: 0
Input: need serious explaining and help I use to just play on my PS4; then It broke and I could get it fixed but I've always wanted a gaming pc. Before I ask to build one I need to understand the parts and what they do; which I don't know anything about so this is why I'm making this post. Is it really cheaper then buying a prebuilt; what good parts are in my price range which isn't that large?
Label: 0
Input: Need advice on a pc for my baby brother (and myself)) Hello; everyone! Hope you all are safe and well!! I need advice on this build I composed for my baby brother. I was planning to buy him a PS5 but I wasn't able to get it; and naturally I thought it was a good time to get a PC that both of us can use. Ever since I was a little girl; I dreamed of getting a computer exclusively for gaming. I never had the funds for it before (or the time since) so it never happened. I'm hoping I can play all the games I never got to play with this build. My 12 y o brother will be playing games like Minecraft; Genshin Impact; Terraria; Among Us and I plan on playing some CS GO; Hearthstone; Portal 2; Detroit Become Human and a bunch of indie games I bought on Steam. I'm mainly looking for a PC that can handle 1080p gaming comfortably. I live in the UAE so buying online from Newegg; Amazon US is really a no no since I'm forced to pay shipping costs up to 200 300. The parts I'm gonna buy are mostly from local merchants and a few can be ordered online (from local websites). I'm mainly looking for critique or advice. I've checked for the compatibility but I just want to make sure that all the components work well together for the games that will be played. PCPartPicker Part List CPU AMD Ryzen 5 3600X 3.8 GHz 6 Core Processor Motherboard MSI B550 A PRO ATX AM4 Motherboard Memory G.Skill Ripjaws V 16 GB (2 x 8 GB) DDR4 3600 CL16 Memory Storage Crucial P1 500 GB M.2 2280 NVME Solid State Drive Video Card Asus GeForce GTX 1660 SUPER 6 GB STRIX GAMING OC Video Card Case MSI MPG Sekira 100R ATX Mid Tower Case Power Supply Thermaltake Smart 650 W 80 Bronze Certified ATX Power Supply Operating System Microsoft Windows 10 Home OEM 64 bit If you've read this far; thank you so much! Have a good day)
Label: 0
Input: Something that will help Doge If you sell goods; products or services; Make some available exclusively for Doge transactions. This will continue to solidify the Coin as a currency as well as something exclusively and filled with prestige.
Label: 3
Input: Which one should I go with? Idk if I should go with my first choice or my second one. There isn't much difference but I still don't know which one I should go with. Any help is appreciated. Choice 1 Choice 2
Label: 0
Input: I Need Better Storage for my Legion y7000 Yes; it's a gaming laptop; sue me. But I love it and so far it's played most games without issue. But the issue I've had as of

....

F1: 0.026

PROMPT-MII-Zero for Brazilian Court Decisions (joelniklaus/brazilian_court_decisions)

Classify the given text as one of the following: no, partial, yes.

The task involves determining the outcome of a legal appeal or review.

Useful clues for making the decision include:

- The presence of words like "conhecido" (known), "provido" (granted), or "denegada" (denied), which often indicate the outcome of the appeal.
- The use of phrases like "em parte" (in part) or "parcialmente procedente" (partially granted), which suggest a partial outcome.
- The overall tone and language used in the text, which may convey a sense of approval, denial, or partial acceptance.

Respond with only the label name, without any additional text or explanation.
Only return one of these options: no, partial, yes. Do not output "Label:" or any extra text.

F1: 0.147

PROMPT-MII for Brazilian Court Decisions (joelniklaus/brazilian_court_decisions)

Classify each input as 'yes,' 'partial,' or 'no' based on the model's analysis of the text. To make this determination, pay close attention to the presence and wording of phrases such as 'CONHECIDO E PROVIDO,' 'CONHECIDO E NÃO PROVIDO,' 'CONHECIDO, EM PARTE, E PARCIALMENTE PROVIDO,' and 'DENEGADA.'

- 'CONHECIDO E PROVIDO' typically indicates a 'yes' label, as it implies the request was fully granted.
- 'CONHECIDO E NÃO PROVIDO' typically indicates a 'no' label, as it implies the request was fully denied.
- 'CONHECIDO, EM PARTE, E PARCIALMENTE PROVIDO' typically indicates a 'partial' label, as it implies the request was granted but not fully.
- 'DENEGADA' typically indicates a 'no' label, as it directly translates to 'denied.'
- If the input does not contain any of these phrases, look for other indicators such as 'revisão criminal parcialmente procedente,' which suggests a 'partial' label.

Respond with only the corresponding label ('yes,' 'partial,' or 'no') for each input, without any explanation or additional text. Do not include any analysis or context; simply choose the most accurate label based on the input.
Only return one of these options: no, partial, yes. Do not output "Label:" or any extra text.

F1: **0.937**

ICL for Stereotype Detection (McGill-NLP/stereoset)

Input: I love my BLANK grandfather.
Label: gender
Input: The Iranian had a BLANK in his personal affects.
Label: race
Input: Most prisoners are BLANK people
Label: profession
Input: People from Afghanistan are BLANK.
Label: race
Input: Jordan is know for its BLANK.
Label: race
Now classify the next input. Only return one of these options: religion, profession, gender, race. Do not output "Label:" or any extra text.

F1: 0.270

PROMPT-MII for Stereotype Detection (McGill-NLP/stereoset)

Classify each input as belonging to one of the following categories: religion, profession, gender, or race. To make this decision, look for contextual clues such as the type of noun that fits into the blank (e.g. 'grandfather' implies gender, 'prisoners' implies profession, 'Iranian' or 'Afghanistan' implies race, and 'BLANK in his personal affects' implies a characteristic associated with a person's cultural or racial background).

Key indicators for each category:
- **Gender**: References to family relationships (grandfather, grandmother, son, daughter), gendered pronouns, or gender-specific roles
- **Race**: References to countries, nationalities, ethnic groups, or cultural backgrounds (Iranian, Afghanistan, Jordan, etc.)
- **Profession**: References to job-related contexts, workplace situations, or occupational groups (prisoners, workers, etc.)
- **Religion**: References to religious practices, beliefs, institutions, or religious groups

When encountering a sentence with a BLANK, consider what type of word would logically complete the sentence and which category that word would most likely belong to. Focus on the subject and context of the sentence to determine the most appropriate classification.

Respond with only the category name (religion, profession, gender, or race), without any explanation or additional text.
Only return one of these options: religion, profession, gender, race. Do not output "Label:" or any extra text.

F1: 0.930

Table 7: Evaluation Datasets: Number of Labels, and Avg Tokens per Example

Dataset	# Labels	# Token Length
Milkyway-islander/AI_Human_generated_movie_reviews	2	300
turkish-nlp-suite/SentiTurca	2	109
scikit-fingerprints/MoleculeNet_BBBP	2	116
hsuvaskakoty/chew_lexical	2	762
AAU-NLP/HiFi-KPI	2	155
poleval/poleval2019_cyberbullying	2	143
kuroneko5943/jd21	2	72
DGurgurov/bengali_sa	2	101
TUKE-KEMT/hate_speech_slovak	2	40
Geralt-Targaryen/MELA	2	21
proteinglm/metal_ion_binding	2	170
Process-Venue/Hindi-Marathi-Synonyms	2	13
xxparthparekhxx/ContactShieldDataset	2	41
mteb/IndonesianIdClickbaitClassification	2	32
justpluso/turn_detection_3k_zh	2	53
projecte-aina/Parafraseja	2	40
germane/Tab-MIA	2	1199
tarudesu/VOZ-HSD	2	45
AI4Protein/MetallonBinding_ESMFold	2	168
qbao775/PARARULE-Plus-Depth-4	2	14
uproai/endex-700k-ns	2	57
FangornGuardian/filtered_wikipedia_wikinews_arxiv_revisions	2	43
krr-oxford/OntoLAMA	2	118
ragarwal/factual-consistency-evaluation-benchmark	2	718
mahdin70/merged_bigvul_primevul	2	397
DGurgurov/yoruba_sa	2	74
QCRI/COVID-19-disinformation	2	68
clue/clue	3	37
BatsResearch/NusaX-senti-LexC-Gen	3	54
stjiris/IRIS_sts	3	47
MoritzLaurer/multilingual-NLI-26lang-2mil7	3	78
CZLC/csfd_sentiment_balanced	3	116
arize-ai/beer_reviews_label_drift_neutral	3	144
waashk/medline	3	167
cardiffnlp/tweet_sentiment_multilingual	3	32
albinandersson/Nordisk-Familjebok-Category-Classification-Dataset	3	87
joelniklaus/brazilian_court_decisions	3	26
david-inf/am-nlp-abstract	3	52
PratikGanesh/Crash_Predictionsv2	3	79
Sakshamrzt/IndicNLP-Telugu	3	2275
BueormLLC/sDtext	3	218
Tricoteuses/CAPP	3	3883
conceptnet5/conceptnet5	3	18
isek-ai/danbooru-tags-2016-2023	3	104
HausaNLP/AfriSenti-Twitter	4	39
adorkin/evalatin2024	4	35
yyu/reddit-attrprompt	4	181
sdadaas/ppc	4	27
mteb/NewsClassification	4	64
McGill-NLP/stereoset	4	20
RussianNLP/tape	4	60
mteb/PolEmo2.0-IN	4	263
tcepi/bidCorpus	5	564
UdS-LSV/hausa_voa_topics	5	31
ai-forever/headline-classification	6	33
QCRI/LlamaLens-English	6	50
strombergnlp/nordic_langid	6	36
samirmsallem/argumentative_zoning_multilingual	7	42
mteb/masakahnews	7	841
silverspeak/essay	7	704
QCRI/LlamaLens-Arabic	7	574
antoinejeannot/jurisprudence	8	1600
mteb/NusaParagraphTopicClassification	8	269
dadashzadeh/Persian_Cooking	8	22
DBQ/Louis.Vuitton.Product.prices.Russia	8	20
DataTonic/synthetic-climate-disinfo-dataset-qwen	8	70
rds/swiss_legislation	8	3385
chenxinpingcxp/reddit_dataset_5HEMpH3WtP6NubmWRNSCJ8nAZ7kEixQ84VeRjA4g8ozLXXyb	9	122
mteb/PatentClassification	9	4028
strickvl/isafpressreleases	9	155
timepearc/banking10	10	26
Deysi/sentences-and-emotions	10	20
chenxinpingcxp/reddit_dataset_660618	13	80
jingjietan/kaggle-mbti	16	1746
choerulaffianto/kblbi2020	21	200
bernard-ng/drc-news-corpus	22	505
lars1234/story_writing_benchmark	26	1235
mteb/NLPTwitterAnalysisClustering	26	58
synapz/reddit_dataset_152	29	130
Aliissa99/FrenchMedMCQA	31	111
hheiden/us-congress-bill-policy-115_117	32	3844
mteb/amazon_massive_intent	39	397
tasksource/bigbench	39	156
masakhane/InjogonIntent	40	78
AmazonScience/massive	60	397
clips/VaccinChatNL	70	57
claritylab/utcd	83	59
takiholadi/kill-me-please-dataset	98	205
DeepPavlov/eurlex	142	1060
PNLPHub/FarsInstruct	156	223
tumeteor/Security-TTP-Mapping	222	133
PNLPHub/FarsInstruct	254	458
evalitahf/entity_recognition	364	567
tensorshield/reddit_dataset_84	493	179
d0rj/geo-reviews-dataset-2023	503	595