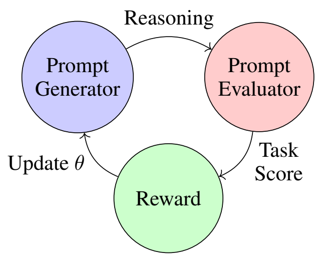
PRL: Prompts from Reinforcement Learning

Paweł Batorski[*] **Adrian Kosmala**[*] **Paul Swoboda**
{pawel.batorski,adrian.kosmala,paul.swoboda}@hhu.de
Heinrich Heine Universität Düsseldorf

Abstract

Effective prompt engineering remains a central challenge in fully harnessing the capabilities of LLMs. While well-designed prompts can dramatically enhance performance, crafting them typically demands expert intuition and a nuanced understanding of the task. Moreover, the most impactful prompts often hinge on subtle semantic cues, ones that may elude human perception but are crucial for guiding LLM behavior. In this paper, we introduce PRL (Prompts from Reinforcement Learning), a novel RL-based approach for automatic prompt generation. Unlike previous methods, PRL can produce novel few-shot examples that were not seen during training. Our approach achieves state-of-the-art performance across a range of benchmarks, including text classification, simplification, and summarization. On the classification task, it surpasses prior methods by 2.58% over APE and 1.00% over EvoPrompt. Additionally, it improves the average ROUGE scores on the summarization task by 4.32 over APE and by 2.12 over EvoPrompt and the SARI score on simplification by 6.93 over APE and by 6.01 over EvoPrompt. Our code is available at https://github.com/Batorskq/prl.



Method	Gen.	Ref.	Few-Shot
Manual Instr.	✗	✗	✗
APE	✓	✗	✗
EvoPrompt	✓	✓	✗
APO	✓	✓	∼
PRL	✓	✓	✓

Figure 1: Left: Illustration of our RL-based prompt optimization cycle, showing the iterative process of prompt generation, evaluation, and refinement. Right: Comparison of prompt engineering methods, highlighting that PRL not only automates prompt generation and refinement but also uniquely incorporates novel task-specific few-shot examples, resulting in superior overall performance. A yellow tilde (∼) is added for APO to indicate that, although it produces few-shot examples, they are sourced from the training data which significantly bounds the performance, whereas PRL generates entirely new instances unseen during training.

1 Introduction

Prompt engineering has emerged as a key technique for enhancing the performance of large language models (LLMs) [Sahoo et al., 2024, Chen et al., 2023]. By crafting precise input prompts, LLMs can be guided to perform complex tasks without requiring additional fine-tuning. However, the effectiveness of a prompt often hinges on subtle phrasing. As shown by Razavi et al. [2025], even

[*]Equal contribution

minor rewordings can significantly alter model predictions, underscoring the fragility of prompt-based control. Moreover, the DeepSeek-R1 paper Guo et al. [2025] states that even a model as large as DeepSeek-R1, with 671 billion parameters, is sensitive to prompts.

Few-shot prompting, in which a prompt includes a small set of input-output examples, is another widely used approach to guide LLMs. While often beneficial, Reynolds and McDonell [2021] demonstrate that zero-shot prompting can sometimes outperform few-shot approaches, suggesting that the usefulness of examples may depend on task familiarity or pretraining exposure. These findings collectively highlight the challenge of designing effective prompts and motivate the need for automated, task-specific prompt optimization.

Recent work has explored automatic prompt generation [Zhang et al., 2022] and refinement [Guo et al., 2023, Pryzant et al., 2023]. Existing methods, with the partial exception of [Pryzant et al., 2023], fail to integrate tailored few-shot examples. We propose PRL, a RL-based prompt optimization algorithm based on reinforcement learning. PRL is capable of automatically determining whether few-shot examples should be included and if so, to create them to maximize task performance. Interestingly, the incorporation of few-shot examples is spontaneously emerging during the prompt generation training and is not explicitly encouraged. Additionally, PRL incorporates a reasoning phase prior to prompt generation, where the model first produces a rationale to guide its final output. We additionally mitigate training instability and noisy feedback with a prompt selection strategy that improves robustness in the limited data setting.

Contributions. This paper makes the following contributions:

- We propose PRL, to our knowledge the first RL-based prompt optimization method capable of generating and selecting novel task-specific few-shot examples.
- We demonstrate the effectiveness of PRL across text classification, summarization, and simplification tasks.
- We show that integrating explicit reasoning before answer generation significantly boosts performance, echoing findings by Guo et al. [2025].
- We provide detailed ablation studies to evaluate the impact of each component.
- Our results suggest that RL-based optimization naturally leads to the emergence of few-shot prompting behavior.

2 Related Work

Prompt Engineering enhances model performance without retraining, offering a cost-effective solution. Chain-of-Thought (CoT) prompting [Wei et al., 2022] improves reasoning by including intermediate steps. Tree-of-Thought (ToT) [Yao et al., 2023] extends this by exploring multiple reasoning paths, while Program-of-Thoughts [Chen et al., 2022] and Graph-of-Thoughts [Besta et al., 2024] further enrich prompts using programmatic and graph structures.

Few-shot prompting [Brown et al., 2020] improves performance by embedding task examples in prompts, proving effective in areas like puzzle solving and evidence extraction [Xu et al., 2023, Greenblatt, 2024, Sivarajkumar et al., 2024]. However, such examples can sometimes hurt performance [Reynolds and McDonell, 2021], making their use highly task-dependent. Our method automatically learns whether and how to include few-shot examples based on task performance.

Automated Prompt Engineering improves task performance by replacing manual prompt design with automated methods. The Automatic Prompt Engineer (APE) Zhou et al. [2022] generates prompt candidates from input-output examples and filters them based on performance. As no gains were observed from in-sample refinement, APE remains a pure generation method.

Pryzant et al. [2023] introduced Automatic Prompt Optimization (APO), which iteratively improves prompts using natural language critiques, simulating gradient descent. APO includes few-shot examples in its prompt, but is restricted to examples seen during training. It enhances efficiency via minibatching, beam search, and bandit selection. Guo et al. [2023] proposed EvoPrompt, which evolves a population of prompts with LLMs and evolutionary operators, achieving strong results without needing model gradients. PRLis, to our knowledge, the only method that can create novel few-shot examples not seen during training and we argue that the choice of which and whether to incorporate them is more flexible and directly related to task performance.

Reasoning. Incorporating intermediate reasoning steps significantly boosts LLM performance on complex tasks [Wei et al., 2022, Wang et al., 2022, Kojima et al., 2022]. Guo et al. [2025] propose a two-stage RL-based method where the model generates reasoning traces before final answers, improving interpretability and generalization. Muennighoff et al. [2025] show that supervised fine-tuning on curated datasets with reasoning traces also teaches reasoning effectively. We adopt the RL-based approach from Guo et al. [2025] for its scalability and independence from manual annotations.

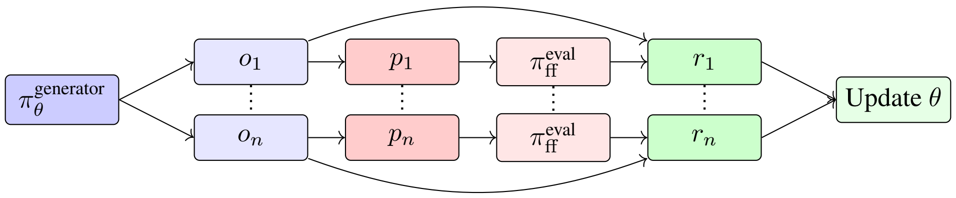

$\pi_\theta^{\text{generator}}$
o_1
o_n
p_1
p_n
$\pi_{\text{ff}}^{\text{eval}}$
$\pi_{\text{ff}}^{\text{eval}}$
r_1
r_n
Update θ

Figure 2: Training scheme of PRL. First, the Prompt Generator $\pi_\theta^{\text{generator}}$ generates a set of outputs $o_1, \dots, o_n$ (reasoning + generated prompt) from which the corresponding prompts $p_1, \dots, p_n$ are extracted. Each prompt is then evaluated by the Evaluation Model π^{eval} (a language model with frozen parameters), which produces corresponding answers. These answers, along with the outputs from the Prompt Generator, are used to compute rewards $r_1, \dots, r_n$. Finally, the rewards are used to update the parameters of the Prompt Generator through RL.

3 Method

Our method comprises the following components:

- **Prompt Generator:** A trainable language model that generates prompts with help of a reasoning process, see Figure 3.
- **Evaluation Model:** A frozen LLM that produces an answer based on the generated prompt.
- **Prompt Selection:** We learn the prompt generator through RL with a reward incorporating both formatting and task performance. We choose the best overall prompt by regularly querying the prompt generator model for prompt candidates and evaluate those.

We now provide a detailed description of each component of our method.

Reward Function Our reward signal is composed of two parts: formatting rewards for the Prompt Generator and task performance rewards for the evaluation model.

Prompt Generator Reward: We assess the format of the Prompt Generator's output using the following scheme:

- A reward of $\frac{r_{\text{token}}}{4}$ is assigned for the correct usage of each of the four key tokens: `<think>`, `</think>`, `<answer>`, and `</answer>`, provided that each token appears exactly once. If all tokens are used correctly, the model receives the full token reward of r_{token}. This encourages the model to output both a reasoning trace and a final answer, each clearly marked and occurring only once.
- An additional reward of $r_{\text{structure}}$ is granted if the generated response exactly matches the required structure: `<think>` reasoning trace `</think>` `<answer>` final answer `</answer>`. This structural reward ensures that the model produces a well-formed response consisting of a reasoning trace followed by a final answer and nothing beyond this expected format.

Evaluation Model Reward: To assess the utility of the prompts, we assign:

- A reward of r_{format} is assigned if the Evaluation Model's response follows the required format. This reward is applicable only when the output must adhere to a specific structure, such as selecting a predefined class; otherwise, r_{format} is set to 0.
- A reward of $r_{\text{alignment}}$ is granted if the model's response is factually correct or aligned with the intended task objective. This reward is typically based on task-specific metrics such as accuracy or any other evaluation criterion that reflects successful performance.

These rewards are task-specific and will be discussed in detail in the experimental section for each respective task.

The overall reward function R is the sum of all elementary applicable rewards r_{token}, $r_{\text{structure}}$, r_{format} and $r_{\text{alignment}}$.

Prompt Generator is a language model designed to refine a given base prompt. First, it generates a reasoning trace about the refinement process, followed by the production of the final prompt (an example is shown in Figure 3). During training, the Prompt Generator learns task semantics and produces suitable prompts, potentially incorporating examples for few-shot prompting.

At each training step, the Prompt Generator produces a set of outputs $o_1, \dots, o_n$ from which candidate prompts $p_1, \dots, p_n$ are extracted. These prompts are then passed to the Evaluation Model, which generates answers conditioned on each prompt. We denote the Prompt Generator as $\pi_\theta^{\text{generator}}$.

Evaluation Model. The *Evaluation Model* assesses the quality of each prompt generated by the Prompt Generator by evaluating its performance on a randomly sampled subset of the training data. For each prompt, a reward is computed for every observation based on its effectiveness, and these rewards are averaged to obtain a final score for the prompt. The Evaluation Model is implemented as a frozen language model, used exclusively for inference. We denote the Evaluation Model as π^{eval}.

Remark 1. *We have chosen to freeze the evaluation model, since then our method can in principle be used with closed-source LLMs. It also conforms to the setting of existing work that did not finetune the LLM executing the prompt. We also speculate that it might help in training stability. However, jointly training the prompt generator and the task evaluator might yield additional benefits.*

Optimization After obtaining a reward for each prompt, we optimize the Prompt Generator using the Group Relative Policy Optimization (GRPO) update rule [Shao et al., 2024]. A key advantage of GRPO is that it eliminates the need for a separate critic network, significantly reducing memory consumption during the reinforcement learning process. The illustration of this process can be found in Figure 2.

Prompt Selection As the prompt generator evolves during training and each version generates multiple prompts, we obtain a large selection of candidate prompts. We sample in a regular interval a number of prompts and test them on the validation set and keep the overall best one according to the task metric used. The full algorithm is showcased in the supplementary materials A.

System Prompt

A conversation between User and Assistant. The user asks a question, and the Assistant solves it. The assistant first thinks about the reasoning process in the mind and then provides the user with the answer. The reasoning process and answer are enclosed within <think> </think> and <answer> </answer> tags, respectively, i.e., <think> reasoning process here </think><answer> answer here </answer>

User Prompt

Your task is to refine a base prompt for another model that performs a sentiment classification task. Improve the instructions to enhance the model's performance. The base prompt: In this task, you are given sentences from movie reviews. The task is to classify a sentence as 'positive' if the sentiment of the sentence is positive or as 'negative' if the sentiment of the sentence is negative. Return label 'positive' or 'negative' only without any other text.

Figure 3: Prompt used by our model. In the system prompt, we instruct the model to generate a reasoning trace enclosed within <think> and </think> tokens, followed by the final answer encapsulated within <answer> and </answer> tokens. The user message provides the base prompt that it should refine. The model's objective is to produce the prompt that is better than the best prompt.

4 Experiments

Experimental Setup We evaluate the performance of our method across three task types: **classification**, **summarization**, and **simplification**. As Prompt Generator and Evaluation Model we choose Qwen2.5-7B-Instruct (Yang et al. [2024]). Each model is trained separately for each task and dataset. Unless otherwise specified, all experiments are conducted over 48 hours using two NVIDIA A100 GPUs (40 GB each). We fine-tune our models using GRPO (Zhao et al. [2024]) with parameters $\epsilon = 0.2$, $\beta = 0.04$ and weight decay equal to 0.1. We also use Low-Rank Adaptation (LoRA) (Hu et al. [2022]) with a learning rate of 1×10^{-6}, setting $\alpha = 32$ and rank $r = 8$. During training, we sample $n = 4$ prompts per iteration and perform Prompt Selection every 100 iterations.

To ensure fair comparison during Prompt Selection, we adopt the same scoring function as EvoPrompt and we use the identical validation dataset during prompt selection process. Across all tasks, we use reward parameters $r_{\text{token}} = r_{\text{structure}} = 0.75$, unless otherwise stated. The rewards r_{format} and $r_{\text{alignment}}$ are task-specific and will be defined below.

We begin by introducing the baselines used for comparison, followed by detailed descriptions of the classification, summarization, and simplification tasks. Finally, we present ablation studies to analyze key design choices.

Benchmarks We benchmark PRL against both human-written task-specific prompts and a range of general-purpose prompt engineering algorithms.

- `MI (Manual Instruction)` [Zhang et al., 2022]: This benchmark utilizes manually crafted instructions to fine-tune large language models, aiming to enhance their performance on specific tasks through human-written prompts.
- `NI (Natural Instruction)` [Mishra et al., 2021]: NI comprises a diverse set of 61 NLP tasks, each accompanied by human-authored instructions. It is designed to evaluate models' abilities to generalize across tasks by understanding and following natural language instructions.
- `APE (Automatic Prompt Engineer)` [Zhou et al., 2022]: APE introduces a framework for automatically generating and selecting prompts. It leverages large language models to create candidate instructions and selects the most effective ones based on performance evaluations.
- `APO (Automatic Prompt Optimization)` [Pryzant et al., 2023]: APO presents a method for optimizing prompts by iteratively refining them using feedback mechanisms. It treats prompt optimization as a gradient-free problem, employing techniques like beam search to enhance prompt effectiveness.
- `EvoPrompt` [Guo et al., 2023]: EvoPrompt applies evolutionary algorithms to optimize discrete prompts for large language models. It evolves a population of prompts through selection, mutation, and crossover operations to discover high-performing prompts without requiring gradient information.
  - `DE (Differential Evolution)`: This variant employs differential evolution strategies to explore the prompt space.
  - `GA (Genetic Algorithm)`: This approach utilizes genetic algorithms to evolve prompts by simulating natural selection processes, including selection, crossover and mutation to optimize prompt quality over successive generations.

We present a comparison of various methods in Figure 1 (right). Prior work has introduced techniques that automatically generate task-specific prompts and refine them throughout training. To ensure a fair comparison, we utilize the Qwen2.5-7B-Instruct model across all methods, serving both as the prompt generator and the Evaluation Model.

Classification For this task, we evaluate our method on a variety of datasets, including:

- **Binary sentiment classification**: SST-2 (Socher et al. [2013]), MR (Pang and Lee [2005]), CR (Hu and Liu [2004]). These datasets involve classifying whether the semantic meaning of a sentence is `positive` or `negative`.
- **Multiclass sentiment classification**: SST-5 (Socher et al. [2013]) requires classifying a sentence into one of five sentiment categories: `terrible`, `bad`, `okay`, `good`, or `great`. This is more challenging than binary sentiment classification as it involves recognizing more nuanced emotional intensities.
- **Question classification**: TREC Voorhees and Tice [2000]) requires to categorize a question into one of six classes: `Description`, `Entity`, `Expression`, `Human`, `Location`, or `Number`. This task evaluates the model's understanding of the semantic type of the question.
- **News classification**: AG's News (Zhang et al. [2015]) requires to clasify news articles into one of four topics: `World`, `Sports`, `Business`, or `Tech`.
- **Subjectivity classification**: SUBJ (Pang and Lee [2004]) asks to determine whether a sentence is `subjective` or `objective`.

We apply a unified reward function across all classification tasks, with reward parameters set as $r_{\text{format}} = r_{\text{alignment}} = 1$. The component r_{format} is specifically awarded when the Evaluation Model's output is a valid label, i.e. one that belongs to the task's set of permissible labels.

For example, in binary sentiment classification, a reward of $+1$ is given if the output is either `positive` or `negative`. This encourages the Prompt Generator to produce prompts that guide the Evaluation Model toward correct, task-appropriate responses.

The scoring function f used in all classification tasks is accuracy. We set the number of test prompts to $n_{\text{test}} = 10$. For most tasks we sample a subset of 100 samples of our training set. For CR and AG's

Table 1: Accuracy on classification tasks, averaged over three runs. Colours mark the best (red), second-best (orange) and third-best (yellow) numbers in each column; minor differences (≤ 0.05) are treated as ties. The right-most column shows the mean accuracy of each method across the seven datasets.

Method / Dataset	**SST-2**	**CR**	**MR**	**SST-5**	**AG's News**	**TREC**	**Subj**	**Avg**
MI	92.70	87.25	87.40	52.31	82.29	69.20	57.95	75.59
Natural Prompt	95.77	91.50	90.85	51.90	83.43	66.60	68.10	78.31
APO	$93.71_{\pm0.25}$	$93.48_{\pm0.24}$	$89.97_{\pm1.37}$	$53.94_{\pm0.29}$	$83.73_{\pm0.31}$	$71.30_{\pm1.90}$	$69.80_{\pm5.96}$	79.42
APE	$91.23_{\pm0.66}$	$92.87_{\pm0.02}$	$89.90_{\pm0.94}$	$49.37_{\pm5.66}$	$82.58_{\pm1.20}$	$77.07_{\pm1.61}$	$73.92_{\pm1.39}$	79.56
EvoPrompt (GA)	$94.65_{\pm1.04}$	$92.75_{\pm0.40}$	$90.45_{\pm0.72}$	$53.76_{\pm1.13}$	$82.24_{\pm1.00}$	$79.20_{\pm2.83}$	$74.93_{\pm3.12}$	81.14
EvoPrompt (DE)	$93.29_{\pm0.34}$	$93.38_{\pm0.19}$	$89.98_{\pm0.24}$	$55.25_{\pm0.37}$	$82.18_{\pm1.04}$	$76.47_{\pm0.38}$	$73.08_{\pm4.95}$	80.52
PRL (–PS) (ours)	$95.98_{\pm0.19}$	$92.17_{\pm0.02}$	$90.72_{\pm0.05}$	$54.80_{\pm1.10}$	$83.84_{\pm0.33}$	$72.00_{\pm0.86}$	$66.98_{\pm2.86}$	79.50
PRL (ours)	$96.32_{\pm0.04}$	$92.83_{\pm0.24}$	$91.27_{\pm0.05}$	$56.21_{\pm0.15}$	$84.36_{\pm0.08}$	$77.07_{\pm2.36}$	$76.90_{\pm0.95}$	82.14

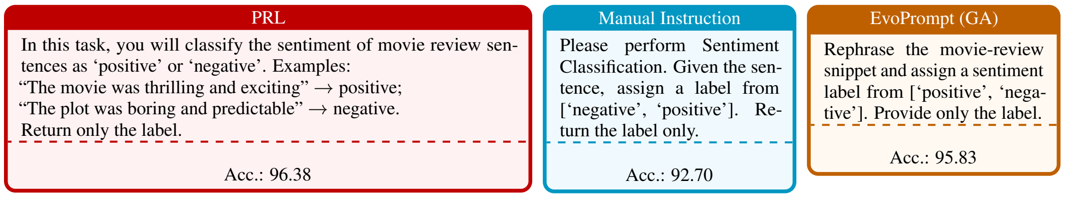
News, due to longer average sentence lengths (which increases training and evaluation time), we reduce this to 30 samples.

We present our results in Table 1, where our method achieves state-of-the-art performance on all classification datasets. Notably, on the subjectivity classification task, our approach improves accuracy by 19% compared to the manual prompt baseline.

Figure 4 presents a comparison of the manual prompt, the PRL-generated prompt, and the EvoPrompt-generated prompt for the SUBJ classification task. As shown, the prompt generated by PRL is more detailed and explicit, providing clearer guidance for the model. Moreover, it is automatically tailored with task-specific few-shot examples, which contributes to its superior performance.

The remaining prompts for other classification tasks are included in Appendix B. As illustrated, all of these prompts incorporate few-shot examples, emphasizing the critical role of few-shot prompting in text classification. Interestingly, the examples generated by PRL do not appear in the training set, indicating that the model is able to synthesize relevant and task-aligned examples autonomously. This is in contrast to APO, which also can incorporate few-shot examples, but which are always selected from the training set. Specifically, few-shot examples in APO are selected from training samples which were incorrectly classified.

Remark 2. *In EvoPrompt [Guo et al., 2023] the task accuracy is computed by extracting the corresponding word for the classification from the full response. Hence, even when additional text is generated and hence does the output does not strictly conform to the desired format, often a classification can still be obtained. In our work we train and evaluate EvoPrompt by only accepting a response that is comprised of a single word denoting the classification. For summarization and simplification tasks we do not modify the EvoPrompt training and evaluation process.*



PRL

In this task, you will classify the sentiment of movie review sentences as 'positive' or 'negative'. Examples:
"The movie was thrilling and exciting" $\longrightarrow$ positive;
"The plot was boring and predictable" $\longrightarrow$ negative.
Return only the label.

Acc.: 96.38

Manual Instruction

Please perform Sentiment Classification. Given the sentence, assign a label from ['negative', 'positive']. Return the label only.

Acc.: 92.70

EvoPrompt (GA)

Rephrase the movie-review snippet and assign a sentiment label from ['positive', 'negative']. Provide only the label.

Acc.: 95.83

Figure 4: Comparison of a manual instruction, the best PRL prompt, and the best EvoPrompt prompt along with their accuracies on SST-2 task.

Summarization We evaluate PRL on a summarization task, where the model is required to extract and condense the most important information from a given text. The objective is to generate a concise summary that preserves key content while omitting irrelevant or redundant details.

Our experiments are conducted on the SAMSUM dataset [Gliwa et al., 2019], which comprises English-language chat dialogues resembling real-life messenger conversations. These dialogues were synthetically generated by linguists to reflect informal, everyday exchanges and are accompanied by manually written abstractive summaries. To assess summarization performance, we adopt the widely used ROUGE metrics (Lin [2004]), reporting scores for the following variants:

- **ROUGE-1**: Measures the overlap of individual words (unigrams) between the generated summary and the reference summary, focusing on content selection.
- **ROUGE-2**: Measures the overlap of consecutive word pairs (bigrams), capturing the ability of the model to preserve local coherence and phrasing.
- **ROUGE-L**: Measures the longest common subsequence of words between the generated and reference summaries, evaluating the overall fluency and structure alignment.

For this task, we set $r_{\text{format}} = 0$, as summarization does not involve selecting from a fixed label set. Instead, we use $r_{\text{alignment}}$, which computes the reward based on the average of the three ROUGE metrics.

Table 2: Text summarization results averaged over three runs.

Method	ROUGE-1	ROUGE-2	ROUGE-L
MI	32.76	10.39	28.97
APE	$37.12_{\pm 2.02}$	$12.97_{\pm 0.74}$	$33.32_{\pm 1.68}$
EvoPrompt (GA)	$39.69_{\pm 1.76}$	$14.47_{\pm 1.00}$	$35.84_{\pm 1.63}$
EvoPrompt (DE)	$33.91_{\pm 4.04}$	$12.53_{\pm 1.47}$	$31.05_{\pm 3.79}$
PRL (ours)	$42.47_{\pm 0.83}$	$16.17_{\pm 0.24}$	$37.73_{\pm 0.36}$

The results, shown in Table 2, indicate that PRL significantly outperforms all baseline methods on the summarization task. Interestingly, PRL consistently opts to generate prompts without incorporating few-shot examples.

We include the generated prompts in Figure 5, along with the corresponding average ROUGE scores. Notably, the two prompts produced by EvoPrompt are nearly identical in structure and wording, yet they yield significantly different results. This underscores how seemingly minor variations in prompt phrasing that are semantically equivalent to humans can lead to substantial differences in LLM performance.

PRL

When rephrasing text in a few words, focus on capturing the main idea and key points succinctly. Your summary should be concise, accurate, and free of unnecessary details. Aim for clarity and brevity, and ensure that the essence of the original text is preserved. Aim for a sentence or two that encapsulates the core message.

Av ROUGE: 31.60

Manual Instruction

How would you rephrase that in a few words?

Av ROUGE: 24.04

EvoPrompt (GA)

Concisely summarize the main points using clear and brief language by removing redundancies and unnecessary details.

Av ROUGE: 24.89

EvoPrompt (GA)

Create a concise summary that retains the semantic meaning and omits unnecessary details.

Av ROUGE: 29.85

Figure 5: Comparison of averaged ROUGE metrics based on prompts generated by PRL, EvoPrompt, and Manual Instruction for the summarization task. This figure highlights the importance of precise prompt design: although the two prompts generated by EvoPrompt on two different seeds are superficially similar, they result in significantly different performance. In contrast, the PRL prompt is both more effective and better aligned with the task objective.

PRL

Improve the instructions by specifying the aspects of simplification and providing examples. Here's an enhanced version of the prompt: Simplify the following text by:
1. Shortening sentences.
2. Using simpler vocabulary.
3. Removing unnecessary or redundant words.
Example of simplified text: Original: The quick brown fox jumped over the fence, which was quite a high jump for such a small creature. Simplified: The quick brown fox jumped over the high fence. Your task is to make the given text easier to understand while maintaining its original meaning.

SARI: 57.19

Manual Instruction

Simplify the text

SARI: 43.77

EvoPrompt (GA)

Rephrase the given sentence to make it simpler and easier to understand.

SARI: 46.60

Figure 6: Comparison of SARI metric for prompts generated by PRL, EvoPrompt and Manual Instruction for the simplification task.

Simplification We evaluate PRL on the sentence simplification task using the ASSET dataset [Alva-Manchego et al., 2020]. ASSET is a crowdsourced corpus designed to assess the performance of simplification models across multiple rewriting operations, including lexical paraphrasing, sentence splitting, deletion, and reordering. Each of the original sentences is accompanied by human-written simplifications, providing a rich set of references for evaluating model outputs. This multi-reference setup enables more robust and comprehensive evaluation of simplification systems.

To evaluate simplification quality, we adopt the SARI metric Xu et al. [2016], which compares the system output against both the original sentence and a set of reference simplifications. SARI assesses the quality of words that are added, deleted, and kept by the system, providing a comprehensive measure of simplification performance. It has been shown to correlate well with human judgments of simplicity, making it a reliable metric for this task.

We set $r_{\text{format}} = 0$ for this task, as there is no fixed output format to enforce. For the alignment reward, we use the SARI metric. For the final scoring function, we also use the SARI score.

Table 3: Results on task simplification averaged over three runs.

Method	SARI
MI	43.77
APE	$45.33_{\pm 0.83}$
EvoPrompt(GA)	$46.25_{\pm 0.47}$
EvoPrompt(DE)	$45.79_{\pm 0.35}$
PRL (ours)	$52.26_{\pm 3.51}$

The results are presented in Table 3. Our baselines perform on average comparable to a manually written prompt. Generated prompts from PRL, EvoPrompt, and Manual Instruction are provided in Figure 6. For this task baselines generated comparatively simple prompts which fail to provide sufficient guide the model's ouput. The prompt generated by EvoPrompt, despite being structurally similar to the Manual Instruction prompt, yields significantly lower performance. In contrast, the PRLprompt is precise, comprehensive, includes a well-constructed example and leads to a substantial performance improvement.

Interestingly, our experimental results differ from those reported in the original EvoPrompt paper, which claimed that the differential evolution (DE) variant of prompt generation outperforms the genetic algorithm (GA) variant. In contrast, across all three types of tasks we observe that the GA variant consistently yields superior accuracy compared to DE. We attribute this discrepancy to the use of different underlying language models in our reproduction study. These findings suggest that the relative effectiveness of EvoPrompt's evolutionary strategies is sensitive to the choice of the base model.

Ablation Study: Influence of Prompt Selection We analyze the impact of the Prompt Selection process on overall performance. In the ablation setting, instead of selecting the best prompt iteratively during training, we simply report the final prompt at the end of training. To do so after training we sample n_{test} prompts to choose the best one according to the validation set. This comparison is performed across all classification tasks.

The results, shown in Table 1, demonstrate that Prompt Sampling not only improves final performance but also enhances training efficiency. By selecting strong prompts throughout the training process, Prompt Selection leads to both better and faster results. We believe this phenomenon arises from two main factors. First, the use of reinforcement learning to train our LLMs, which is inherently unstable. Second, our method is highly vulnerable to overfitting due to the limited number of samples in the training dataset.

Table 4: Model accuracy vs. number of test samples

n_{test}	1	5	10	15
Accuracy (%)	$90.92_{\pm 0.17}$	$91.25_{\pm 0.15}$	$91.27_{\pm 0.11}$	$91.35_{\pm 0.11}$

Ablation Study: Impact of Prompt Sampling Size in Prompt Selection We investigate how the number of prompt samples used during inference in the Prompt Selection technique affects final performance. Specifically, we evaluate PRL on the MR classification dataset while varying the number of sampled prompts: $n_{\text{test}} = 1$, 5, 10, and 15. Each configuration is run three times, and we report the average accuracy. The results are presented in Table 4.

Performance remains stable when using more than five prompt samples. Using only a single prompt leads to a noticeable drop in performance. This degradation might be due to the nature of the task and the limited size of the training dataset, which make the method more sensitive to prompt variability.

Although the results suggest that five prompts are sufficient for stable performance, we recommend using ten prompts to provide an additional buffer against potential sensitivity in other tasks or datasets.

Ablation Study: Influence of Reasoning To investigate the role of explicit reasoning in our method, we conduct an ablation study based on the prompt design illustrated in Figure 3. In the standard setup, the model is instructed to perform a reasoning process before producing the final answer. To evaluate the effect of removing this step, we modify the prompt to omit the reasoning phase and instead directly request the model to generate the answer within `<answer>` `</answer>` tokens.

We perform this experiment on the SUBJ dataset, training two identical models: one using the reasoning-augmented prompt and the other using the direct-answer prompt. Each experiment is repeated three times. The results, shown in Table 5, indicate that resoning consistently improves upon the version without.

Table 5: Comparison of accuracy between models employing reasoning and those without reasoning on the SUBJ dataset. Results are averaged over three runs.

Reasoning	No Reasoning
$75.05_{\pm 1.63}$	$60.12_{\pm 1.62}$

Ablation Study: PRL on Larger Models It is widely held true that larger and better model are less sensitive to prompt variations. To investigate this phenomenon, we use Qwen2-14B-Instruct and Qwen2-32B-Instruct as the Evaluation Model models, while keeping Qwen2-7B-Instruct as the Prompt Generator. For this experiment, we utilize 6 A100 GPUs for the 32B model and 4 A100 GPUs for the 14B model.

We compare the performance using the base prompt against prompts generated by PRL on the MR dataset. The results, presented in Table 6, show that all model sizes benefit from PRL, demonstrating two key findings: (i) Even larger LLMs remain vulnerable to prompt variation. (ii) PRL is capable of effectively tailoring prompts for both smaller and larger models, significantly improving their performance.

5 Conclusions & Limitations

Table 6: Comparison of accuracy across different model sizes of the Evaluation Model on the MR dataset. Results are averaged over three runs.

Number of Parameters	**7B**	**14B**	**32B**
MI	87.40	89.20	90.15
PRL (ours)	$91.27_{\pm 0.05}$	$92.03_{\pm 0.13}$	$92.52_{\pm 0.02}$

We have introduced an RL-based algorithm for prompt generation that consistently outperforms other approaches across classification, summarization and simplification tasks. Even though we use recent LLMs, better prompts can still significantly increase task performance, indicating that LLMs are still sensitive to differences in semantically equivalent prompts. Interestingly, this holds true even for the largest LLM we have tested on, the Qwen2-32B-instruct model. Additionally, our results underscore that there is no single recipe to generate good prompts across different tasks, as some tasks benefit from few-shot examples or other subtle semantic cues, while others do not. Our approach effectively navigates such delicate prompt crafting issues. In line with current work our prompt generator profits from increased inference time compute by allowing it to reason about effective prompts.

Currently, improved performance is obtained via a significantly greater computational expense than used by the comparatively simpler related work. Another limitation is that we retrain the prompt generator for each new task. An universal prompt generator is a desideratum.

6 Broader & Societal Impact

We do not foresee any negative societal impacts resulting from this work. Our approach can boost LLM performance on promptable tasks to a certain degree, but within the overall capability horizon of current LLMs.

References

Fernando Alva-Manchego, Louis Martin, Antoine Bordes, Carolina Scarton, Benoît Sagot, and Lucia Specia. Asset: A dataset for tuning and evaluation of sentence simplification models with multiple rewriting transformations. *arXiv preprint arXiv:2005.00481*, 2020.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17682–17690, 2024.

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

Banghao Chen, Zhaofeng Zhang, Nicolas Langrené, and Shengxin Zhu. Unleashing the potential of prompt engineering in large language models: a comprehensive review. *arXiv preprint arXiv:2310.14735*, 2023.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*, 2022.

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. Samsum corpus: A human-annotated dialogue dataset for abstractive summarization. *arXiv preprint arXiv:1911.12237*, 2019.

Ryan Greenblatt. Getting 50% (sota) on ARC-AGI with GPT-4o, 2024. URL `https://redwoodresearch.substack.com/p/getting-50-sota-on-arc-agi-with-gpt`.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. *arXiv preprint arXiv:2310.08510*, 2023.

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

Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 168–177. Association for Computing Machinery, 2004.

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

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, 2004.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. Cross-task generalization via natural language crowdsourcing instructions. *arXiv preprint arXiv:2104.08773*, 2021.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.

Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 271–278. Association for Computational Linguistics, 2004.

Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 115–124. Association for Computational Linguistics, 2005.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with" gradient descent" and beam search. *arXiv preprint arXiv:2305.03495*, 2023.

Amirhossein Razavi, Mina Soltangheis, Negar Arabzadeh, Sara Salamat, Morteza Zihayat, and Ebrahim Bagheri. Benchmarking prompt sensitivity in large language models. In *European Conference on Information Retrieval*, pages 303–313. Springer, 2025.

Laria Reynolds and Kyle McDonell. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended abstracts of the 2021 CHI conference on human factors in computing systems*, pages 1–7, 2021.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, and Junxiao Song. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Sonish Sivarajkumar, Mark Kelley, Alyssa Samolyk-Mazzanti, Shyam Visweswaran, and Yanshan Wang. An empirical evaluation of prompting strategies for large language models in zero-shot clinical natural language processing: algorithm development and validation study. *JMIR Medical Informatics*, 12:e55318, 2024.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642. Association for Computational Linguistics, 2013.

Ellen M Voorhees and Dawn M Tice. Building a question answering test collection. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 200–207, 2000.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

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

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415, 2016.

Yudong Xu, Wenhao Li, Pashootan Vaezipoor, Scott Sanner, and Elias B Khalil. Llms and the abstraction and reasoning corpus: Successes, failures, and the importance of object-based representations. *arXiv preprint arXiv:2305.18354*, 2023.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

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

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

Yuze Zhao, Jintao Huang, Jinghan Hu, Xingjun Wang, Yunlin Mao, Daoze Zhang, Zeyinzi Jiang, Zhikai Wu, Baole Ai, Ang Wang, Wenmeng Zhou, and Yingda Chen. Swift:a scalable lightweight infrastructure for fine-tuning, 2024. URL `https://arxiv.org/abs/2408.05517`.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *The Eleventh International Conference on Learning Representations*, 2022.

A Appendix - PRL Pseudo Code

In Algorithm 1 we provide the PRL algorithm.

Algorithm 1 PRL

```
Require: $\pi_\theta^{\text{generator}}$: prompt generator
    $\pi^{\text{eval}}$: frozen Evaluation Model
    $T, V$: training and validation datasets
    $n, n_{\text{test}}$: number of prompts during training/Prompt Selection
    $k$: number of samples per iteration
    $I$: total number of iterations
    $t$: Prompt Selection frequency
    $R$: reward operator
    $f$: scoring function
 1: $best\_score \leftarrow 0$, $best\_prompt \leftarrow$ “”
 2: for $i = 1$ to $I$ do
 3:     Sample $k$ training samples $D \sim T$
 4:     Generate answers $o_1, \ldots, o_n \sim \pi_\theta^{\text{generator}}$
 5:     Extract answers $p_1, \ldots, p_n$ from $o_1, \ldots, o_n$
 6:     Compute rewards $r_j = R(\pi^{\text{eval}}, D, p_j, o_j)$ for each $j = 1, ..., n$
 7:     Update $\theta$ using GRPO with rewards $\{r_j\}$
 8:     if $i \bmod t = 0$ then
 9:         Generate test prompts $p_1, \ldots, p_{n_{\text{test}}} \sim \pi_\theta^{\text{generator}}$
10:         with torch.no_grad():
11:         Compute scores $s_j = f(\pi^{\text{eval}}, V, p_j)$
12:         Let $j^* = \arg\max_j s_j$
13:         if $s_{j^*} > best\_score$ then
14:             $best\_score \leftarrow s_{j^*}$, $best\_prompt \leftarrow p_{j^*}$
15:         end if
16:     end if
17: end for
18: return $best\_prompt$
```

B Appendix – Prompts for Classification

This subsection provides the most effective prompts used for the classification task in our method.

PRL

In this task, you are to classify the opinion in a given sentence from a review as either subjective or objective.
- A subjective sentence expresses personal feelings, opinions, or attitudes.
- An objective sentence presents facts that can be verified and are not influenced by personal feelings.
Examples:
- Subjective: "This movie is the best I’ve ever seen." (Opinion expressed)
- Objective: "The movie won five awards this year." (Fact stated)
When classifying, focus only on the opinion, not the facts. Return the label ’subjective’ or ’objective’ only, without any additional text.
Example:
Input: "The food was delicious." Output: subjective

Acc.: 77.95

Figure 7: Best prompt generated by PRL for SUBJ classification task along with accuracy.

PRL

In this task, you will classify the sentiment of movie review sentences as ’positive’ or ’negative’. Examples:
"The movie was thrilling and exciting" -> positive;
"The plot was boring and predictable" -> negative.
Return only the label: ’positive’ or ’negative’.

Acc.: 96.38

Figure 8: Best prompt generated by PRL for SST2 classification task along with accuracy.

PRL

In this task, you are given sentences from movie reviews. Your goal is to classify each sentence as ’positive’ or ’negative’ based on its sentiment. Pay close attention to the context and nuances in the text, as the sentiment might not be explicitly stated. Examples:
- "The acting was superb, and the plot was engaging." -> positive
- "The movie was so slow and boring that I almost fell asleep." -> negative
Return only the label ’positive’ or ’negative’ without any additional text.

Acc.: 93.00

Figure 9: Best prompt generated by PRL for CR classification task along with accuracy.

PRL

In this task, you are given a sentence from a movie review. Classify the sentence as ’positive’ if the sentiment is positive, or as ’negative’ if the sentiment is negative. Provide only the label ’positive’ or ’negative’ without any additional text. Examples:
- "The acting was superb and the plot was engaging." -> positive
- "The movie was boring and the storyline was predictable." -> negative

Acc.: 91.30

Figure 10: Best prompt generated by PRL for MR classification task along with accuracy.

PRL

In this task, you are given sentences from movie reviews. Your goal is to classify the sentiment of each sentence as ’terrible’, ’bad’, ’okay’, ’good’, or ’great’. Be as accurate as possible. Here are the guidelines for each category:
- ’terrible’: The sentence expresses extreme dissatisfaction or negative feelings.
- ’bad’: The sentence conveys negative feelings but not as strongly as ’terrible’.
- ’okay’: The sentence is neutral or has mixed feelings with no strong positive or negative sentiment.
- ’good’: The sentence conveys positive feelings but not as strongly as ’great’.
- ’great’: The sentence expresses strong positive feelings or high satisfaction.
Consider the overall tone and specific positive or negative words in the sentence to determine the closest sentiment.
If you are not sure, choose the closest option.
Return the label ’terrible’, ’bad’, ’okay’, ’good’, or ’great’ only without any additional text.

Acc.: 56.38

Figure 11: Best prompt generated by PRL for SST-5 classification task along with accuracy.

PRL

In this task, you will be given a news article and asked to classify it into one of the four predefined categories: 'World', 'Sports', 'Business', or 'Tech'.
Follow these detailed instructions to ensure accurate classification
1. Read the article thoroughly to understand its main subject matter.
2. Determine which of the following categories the article's main topic most closely aligns with:
- 'World': articles covering global news, politics, international affairs, etc.
- 'Sports': articles discussing various sports, competitions, athletes, etc.
- 'Business': articles focusing on financial news, corporate activities, markets, etc.
- 'Tech': articles about technology, innovations, companies, gadgets, etc.
3. If the article's content is not clearly related to any of these categories, choose the closest option based on the predominant subject matter.
4. Return the label of the chosen category as a single word without any additional text or explanations, e.g., 'World', 'Sports', 'Business', or 'Tech'.
Example:
Article: "Apple Launches New iPhone Model with Improved Camera Features" Label: Tech
Article: "China and the US Reach a New Trade Agreement" Label: World
Article: "Local Soccer Team Qualifies for the World Cup" Label: Sports
Article: "Oil Prices Drop as OPEC Decides to Cut Production" Label: Business

Acc.: 84.42

Figure 12: Best prompt generated by PRL for AG's News classification task along with accuracy.

PRL

Please perform a Question Classification task. Given a question, classify it into one of the following categories:
- **Description**: Questions asking for descriptions or explanations.
- **Entity**: Questions asking about specific things, objects, or entities.
- **Expression**: Questions asking about how something is expressed or phrased.
- **Human**: Questions asking about people, their characteristics, or roles.
- **Location**: Questions asking about places or geographical locations.
- **Number**: Questions asking for numerical information or quantities.
Return the label 'Description', 'Entity', 'Expression', 'Human', 'Location', or 'Number' only without any additional text. Example:
Question: "What is the capital of France?" Label: Location
Question: "How do you say 'hello' in Spanish?" Label: Expression
Question: "Who is the CEO of Apple?" Label: Human

Acc.: 78.60

Figure 13: Best prompt generated by PRL for TREC classification task along with accuracy.

C Appendix – Benchmark Results Reproduction

To provide a fair comparison with existing benchmarks (APE, APO, and EvoPrompt), we reproduced their results using Qwen2.5-7B-Instruct as both the Prompt Generator and the Evaluation Model, terms named differently in the original papers but are functionally equivalent. The evaluation procedure is identical for all the benchmarks and for PRL, as follows:

- For classification tasks, only a response consisting of a single word denoting the correct class is considered a valid answer.
- For summarization and simplification tasks, the entire response generated by the Evaluation Model is used to compute ROUGE/SARI metrics.

We apply zero-shot evaluation for each method and task.

To ensure that only the label is output by the Evaluation Model, we appended the following instruction to the end of each prompt in the initial population:

```
Return only label {list of labels} without any other text.
```

For example, in the case of the SST-5 dataset, the appended sentence was:

```
Return only label 'terrible', 'bad', 'okay', 'good' or 'great'
without any other text.
```

Depending on the benchmark, the initial population consisted of manual prompts (see MI in Benchmarks) and/or automatic prompts generated by Qwen, following the prompt generation method described in [Zhou et al., 2022].

APO Following the EvoPrompt [Guo et al., 2023] experimental protocol, we ran the APO algorithm using the manual prompt with the best performance as the initial population. However, unlike the EvoPrompt authors, we applied the method to all classification tasks, not only binary ones. APO was not evaluated on text generation tasks (i.e., summarization and simplification), as its optimization algorithm fundamentally relies on binary feedback (correct vs incorrect), which is incompatible with continuous scores such as ROUGE or SARI.

The default parameter setup provided by [Pryzant et al., 2023] was used for each run.

APE and EvoPrompt Following [Guo et al., 2023], the development set size was set to 200 for classification tasks and 100 for simplification and summarization tasks. Each run included 10 iterations. The 10 best prompts for each task served as the initial population (selected from automatic prompts for APE and from both automatic and manual prompts for EvoPrompt).