# GreenTEA: Gradient Descent with Topic-modeling and Evolutionary Auto-prompting

Zheng Dong
Amazon
Seattle, USA
zhengo@amazon.com

Luming Shang
Amazon
Seattle, USA
lmshang@amazon.com

Gabriela Olinto
Amazon
Seattle, USA
olintogo@amazon.com

## Abstract

High-quality prompts are crucial for Large Language Models (LLMs) to achieve exceptional performance. However, manually crafting effective prompts is labor-intensive and demands significant domain expertise, limiting its scalability. Existing automatic prompt optimization methods either extensively explore new prompt candidates, incurring high computational costs due to inefficient searches within a large solution space, or overly exploit feedback on existing prompts, risking suboptimal optimization because of the complex prompt landscape. To address these challenges, we introduce GreenTEA, an agentic LLM workflow for automatic prompt optimization that balances candidate exploration and knowledge exploitation. It leverages a collaborative team of agents to iteratively refine prompts based on feedback from error samples. An analyzing agent identifies common error patterns resulting from the current prompt via topic modeling, and a generation agent revises the prompt to directly address these key deficiencies. This refinement process is guided by a genetic algorithm framework, which simulates natural selection by evolving candidate prompts through operations such as crossover and mutation to progressively optimize model performance. Extensive numerical experiments conducted on public benchmark datasets suggest the superior performance of GreenTEA against human-engineered prompts and existing state-of-the-arts for automatic prompt optimization, covering logical and quantitative reasoning, commonsense, and ethical decision-making.

## CCS Concepts

• **Computing methodologies** → **Artificial intelligence**.

## Keywords

Prompt optimization, Feedback guidance, Topic modeling, Evolutionary Algorithm

**ACM Reference Format:**
Zheng Dong, Luming Shang, and Gabriela Olinto. 2025. GreenTEA: Gradient Descent with Topic-modeling and Evolutionary Auto-prompting. In *First International KDD Workshop on Prompt Optimization, August 4, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
*KDD '25, Toronto, ON, Canada.*
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-1245-6/25/08
https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 Introduction

The performance of LLMs is highly dependent on the quality of the prompts. Manual prompt engineering [3, 19], though widely adopted, requires substantial human effort and domain expertise, posing challenges when adapting to diverse tasks. To mitigate this dependency, automatic prompt optimization has emerged as a promising direction [26]. Early methods primarily adopt discrete search strategies [16, 36] by enumerating a wide range of prompt candidates, enabling extensive exploration but often incurring high computational costs due to inefficient search dynamics. An alternative line of research focuses on feedback-driven refinement [6, 24, 29, 34] by learning from error samples and improve existing prompts in an iterative way. While effective in exploiting learned knowledge, these approaches are susceptible to suboptimal convergence in the presence of complex and non-convex landscapes in prompt solution space. These limitations highlights the necessity of a well-designed optimization algorithm for navigating the complex prompt space in a cost-efficient way.

In this paper, we introduce GreenTEA, an agentic workflow for automatic prompt optimization that emphasizes both effectiveness and efficiency. GreenTEA adopts a feedback-driven, iterative framework in which one LLM agent analyzes the deficiencies of current prompts and another generates improved candidates based on the identified deficiencies. To guide this process, we introduce a topic-modeling mechanism that clusters error samples by the current prompts. This facilitates the identification of major error patterns and ensures that the collected samples sent to the analyzing agent are semantically coherent. New prompts are generated via a novel gradient-guided genetic algorithm (GA), which produces a population of candidates through evolutionary operations on existing prompts, guided by the feedback. This design draws on the success of evolutionary algorithms in prompt optimization [1, 14, 32], striking a balance between exploration and exploitation within the prompt space. The main contributions of this paper include:

- We present GreenTEA, an agentic workflow for automatic prompt optimization that emphasizes both effectiveness and efficiency.
- We introduce an error topic modeling mechanism to improve the optimization efficiency by extracting the major error and collecting feedback based on semantically coherent samples.
- We develop a gradient-guided evolutionary algorithm that enhances the robustness of prompt optimization by balancing exploration and exploitation in the prompt search space.
- Comprehensive experiments show that GreenTEA consistently achieves expert-level prompts across different scenarios.

## 2 Related works

**LLMs for automatic prompt optimization.** Early work regards the prompt as a learnable artifact and leverages LLMs to serve as black–box optimizer [35] to solve the prompt optimization problem. Various approaches adopt self-refinement via LLMs to critique answers and regenerate the prompts iteratively [25, 33]. Recent work of DSPy [17] generalizes this idea into a compiler that represents an unified reasoning and optimization pipeline. These studies establish the premise that prompt optimization can be automated by exploiting the existing knowledge, yet they adopt generic random or greedy edits that may potentially lead to local optima [14].

**Evolutionary prompt search.** Evolutionary algorithms (EAs) have emerged as a powerful tool for automatic prompt optimization by iteratively refining a *population* of candidate solutions through biologically inspired operators. Prior studies [1, 21] demonstrate the effectiveness of LLM-based operators, while EVOPROMPT [14] presents a generic framework that couples LLMs with EAs, enabling a broad family of EA variants within a unified pipeline. Recent work of GAAPO [28] compares different evolution strategies in a modularized framework via comprehensive evaluations. In contrast to these approaches, our `GreenTEA` integrates a gradient-guided exploration mechanism that directs evolution explicitly towards the major failure modes.

**Gradient-guided genetic algorithm.** Genetic algorithm (GA) [15] is a well-established class of evolutionary search methods and have been widely adopted for prompt optimization [31, 32]. They balance exploration with exploitation via fitness-based selection, achieved through crossovers and mutations that traverse the vast prompt space. Direction-guided variants further accelerate convergence by biasing mutations with gradient rather than random perturbations, consistently achieving enhanced performance across domains such as protein–ligand docking [13], system-fairness testing [12], and molecular generation [38]. While they remain largely unexplored for prompt evolution, our `GreenTEA` closes that gap by embedding a topic-guided gradient into the GA loop, yielding faster convergence and more interpretable evolutionary trajectories.

## 3 Methodology: GreenTEA

Given a LLM $\mathcal{M}$, the goal of prompt optimization is to find the best-performing prompt $\mathcal{P}^*$ on a training set $\mathcal{D}_{\text{tr}} = \{(q_1, a_1), \ldots, (q_N, a_N)\}$ under a performance metric $s$, *i.e.*, $\mathcal{P}^* = \text{argmax}_{\mathcal{P} \in \Omega}\, s(\mathcal{P}, \mathcal{D}_{\text{tr}}, \mathcal{M})$. Here $(q_i, a_i)$ represents the $i$-th question-answer pair. This optimization problem is highly non-convex and intractable. In the following, we introduce our `GreenTEA` framework that use LLM agents to find the approximate solution in an iterative way.

### 3.1 Overview: Iterative prompt optimization

The framework contains a feedback-collection stage and a prompt generation stage that performed in an iterative loop to progressively enhance the prompt, as illustrated in Figure 1. The algorithm starts with a initial population of prompts $\mathcal{P}^{(0)}$ of size $K$. These prompts are simple and generic without being fine-crafted for the task.

In the first stage of $t$-th iteration, each prompt $\mathcal{P}_k^{(t)}$ in the existing population $\mathcal{P}^{(t)}$ will be evaluated on $\mathcal{D}_{\text{tr}}$ using the LLM predictor $\mathcal{M}$. We then apply the topic modeling to all the cases

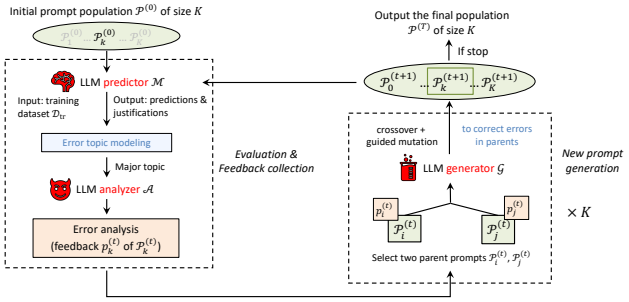



**Figure 1: GreenTEA framework. The first stage (left) evaluates the prompts on the predictor $\mathcal{M}$ and collect feedback on them via the analyzer $\mathcal{A}$. The second stage (right) generates new prompts via the generator $\mathcal{G}$. The two stages are performed iteratively until termination criteria are met.**

that $\mathcal{M}$ have made the wrong predictions on to group them into different topic clusters. Therefore, cases in the same cluster are semantically closer with each other than those in different clusters. We collect those in the cluster with the largest size and input them to the second agent, named the LLM analyzer $\mathcal{A}$. The analyzer summarizes the mistakes and outputs an error analysis $p_k^{(t)}$ as the feedback of $\mathcal{P}_k^{(t)}$ that pinpoints the key deficiencies of the prompt.

After the feedback collection, the second stage contains the third agent, named the LLM generator $\mathcal{G}$, that produce new prompts to address the deficiencies of the existing prompt. We use a gradient-guided GA to evolute the prompts. Instead of optimizing a single prompt, the generator $\mathcal{G}$ takes two existing prompts (the parents) as the input, combine them into one, and revise the prompt based on the feedback of the parents to generate a new prompt (the child). The generator $\mathcal{G}$ will perform the generation by $K$ times to have the new population $\mathcal{P}_k^{(t+1)}$ of size $K$, which will be evaluated and further refined in the next iteration. This cycle continues until predefined termination criteria are met. This evolutionary procedure enhances the exploratory ability of the framework and avoids the optimization trajectory being trapped in the local minima.

The output of `GreenTEA` is the prompt population $\mathcal{P}^{(T)}$ being optimized over $T$ iterations. See Appendix B for algorithm details. In the following, we elaborate on the topic modeling for collecting wrong predictions and the guided GA for prompt generation.

### 3.2 Topic modeling for feedback collection

We observe diverse reasons behind the wrong predictions by the predictor $\mathcal{M}$. To make it easier for the analyzer $\mathcal{A}$ to summarize the error and identify the prompt deficiency, we adopt the topic modeling to guide the selection of wrong prediction samples. Assuming we have the output of $\mathcal{M}$ on each question $q_i$, denoted as $\widehat{a_i} \sim g_{\mathcal{M}}(\cdot|\mathcal{P}_k^{(t)}, q_i)$. Here $g_{\mathcal{M}}$ represents the text generation mechanism of $\mathcal{M}$ that returns the most likely textual output given the input of the prompt and the question. We collect wrong prediction samples, denoted by $\mathcal{W} = \{(q_i, a_i, \widehat{a_i}) : a_i \neq \widehat{a_i}\}$, and perform clustering on $\mathcal{W}$ based on the textual embedding of each $a_i$. Both the choices of the text embedding model and the clustering algorithm are flexible. For instance, we apply a pretrained sentence BERT [27] to create the embedding and the K-nearest neighbors for clustering.

Samples in the largest cluster are retrieved as the final input to the analyzer $\mathcal{A}$ for error analysis, each including $q_i$, $a_i$, and $\widehat{a}_i$. By doing so, the picked samples are semantically close, allowing for the analyzer to focus on the major type of error during its analysis.

The output of the analyzer $\mathcal{A}$ is an analysis of the error that $\mathcal{M}$ has made by comparing the model prediction $\widehat{a}_i$ with the true answer $a_i$. The error analysis contains a guidance on how the existing prompt can be improved to address its deficiencies and output the correct predictions. Note that we make the improvement guidance not question-specific but as general as possible to be able to applied to similar questions that have not been encountered.

## 3.3 Prompt generation via guided GA

Given the current population $\mathcal{P}^{(t)}$, the guided GA generates a new prompt via the following steps:

- **Parent selection**. We use roulette wheel selection method [20] to select two parent prompts $\mathcal{P}_i^{(t)}$ and $\mathcal{P}_j^{(t)}$ based on their fitness scores. The fitness score $f_k$ for each prompt $\mathcal{P}_k^{(t)}$ is measured by the scoring function on the training data, *e.g.,* $f_k = s(\mathcal{P}_k^{(t)}, \mathcal{D}_{\text{tr}}, \mathcal{M})$. The probability of selecting the $k$-th prompt as a parent is $e_k = f_k / \sum_{k'=1}^{K} f_{k'}$.
- **Child generation**. The LLM generator $\mathcal{G}$ takes the selected parent prompts together with their feedback as the input, *e.g.,* $\{\mathcal{P}_i^{(t)}, \mathcal{P}_j^{(t)}, p_i^{(t)}, p_j^{(t)}\}$, and generates a child prompt $\mathcal{P}_k^{(t+1)}$. The generator performs this prompt evolution by i) a crossover to combine the parent prompts into an child prompt that inherits their traits; ii) a mutation to introduce modification to the child prompt. We use corresponding instructions to guide the mutation to incorporate feedback into the child prompt.

We repeat the above steps by $K$ times to produce $K$ child prompts. These child prompts will be evaluated on $\mathcal{D}_{\text{tr}}$ to get their fitness score. We then merge them with the current population and retain the top $K$ prompts with the highest fitness scores to form the updated population $\mathcal{P}^{(t+1)}$. This process ensures the improvement of the population's overall quality, concluding with the best prompt in the updated population being designated as the optimal prompt.

# 4 Experiments

We present the numerical results of prompt optimization that demonstrate the superior performance of `GreenTEA` across different tasks.

## 4.1 Experiment setup

We conduct experiments on the following datasets, including:

- **GSM8K** [5]: A dataset of 8.5K grade-school math word problems designed to evaluate quantitative reasoning and step-by-step problem solving. Each instance includes a problem description and a detailed, free-form answer explanation.
- **ETHOS** [23]: An English hate speech detection dataset containing 997 online comments labeled for attributes such as hate speech, discrimination, and offensive language. It evaluates a model's capability in ethical and socially sensitive reasoning.
- **PIQA** [2]: A physical commonsense reasoning benchmark with 16,000 multiple-choice questions. Each question describes a real-world scenario, and the model must select the more plausible of two possible solutions.

**Table 1: Testing accuracy of `GreenTEA` and baselines**

| Method | GSM8K | ETHOS | PIQA | BBH |
|---|---|---|---|---|
| CoT prompt | $82.43_{(1.19)}$ | $78.33_{(1.25)}$ | $83.97_{(0.72)}$ | $77.48_{(2.05)}$ |
| `OPRO` [33] | $88.10_{(0.93)}$ | $83.33_{(2.05)}$ | $86.05_{(0.61)}$ | $79.58_{(1.26)}$ |
| `ProTeGi` [25] | $84.63_{(1.02)}$ | $\mathbf{85.67}_{(1.49)}$ | $85.72_{(0.84)}$ | $78.86_{(1.77)}$ |
| `EVOPROMPT` [14] | $86.27_{(0.58)}$ | $82.00_{(1.41)}$ | $86.43_{(0.36)}$ | $80.12_{(1.43)}$ |
| `GreenTEA` | $\mathbf{91.37}_{(0.45)}$ | $84.67_{(1.89)}$ | $\mathbf{89.86}_{(0.28)}$ | $\mathbf{82.23}_{(1.13)}$ |

*Numbers in parentheses are standard errors for three independent runs.

- **BBH (Big-Bench Hard)** [30]: A challenging subset of the BIG-Bench benchmark consisting of 23 diverse tasks requiring multi-step logical reasoning, such as date manipulation, causal judgment, and word sorting. Each task includes hundreds of examples with high variance in difficulty.

The choice of the agents in `GreenTEA` is flexible and can be either open- or closed-source. While the predictor $\mathcal{M}$ is picked to satisfy the deployment constraints of a given application (latency, cost, privacy), we recommend using a more capable LLM for both the analyzer $\mathcal{A}$ and the generator $\mathcal{G}$ [17, 33], as larger models possess richer world knowledge and stronger reasoning abilities to yield gains when optimizing the prompts. In the following experiments, we use closed-source models from the Anthropic Claude family, choosing the predictor to be *Claude 3 Sonnet*, and the analyzer and the generator to be *Claude 3.5 Sonnet*. See more details of experimental setup and prompt templates in Appendix A.

## 4.2 Comparison with baselines

We compare our `GreenTEA` against three state-of-the-arts in automatic prompt optimization that have methodological relations to `GreenTEA`: (i) `OPRO` [33] leverages LLMs as optimizer to find instructions that maximize the model performance; (ii) `ProTeGi` [25] exploits feedback on the existing prompt and use textual gradient descent to optimize the prompt; (iii) `EVOPROMPT` [14] adopts an evolutionary framework for prompt optimization without using the prompt feedback. We also evaluate the performance of chain-of-thoughts (CoT) prompts [19] and compare it with `GreenTEA`.

Table 1 reports the accuracy of the predictor $\mathcal{M}$ equipped with different prompts on the testing set on different tasks. The prompts optimized by `GreenTEA` achieve a consistently better or comparable performance against those from other prompt optimization baselines. The `GreenTEA` also significantly improve the prompt performance from the manually designed CoT prompts across the evaluated scenarios.

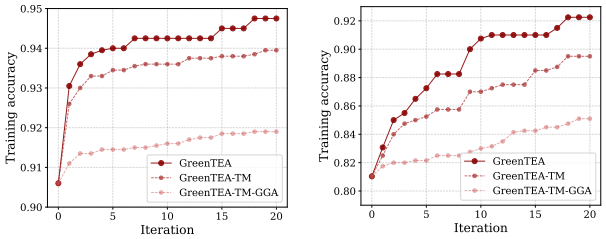
## 4.3 Ablation study

We perform an ablation study to demonstrate the effectiveness of the topic-modeling and gradient-guided GA in prompt optimization. Specifically, we compared two variants of `GreenTEA` that have these key components removed:

- `GreenTEA-TM-GGA`: this variant uses a classic GA instead of the gradient-guided GA to generate child prompts. The LLM generator $\mathcal{G}$ performs random mutation by simply rephrasing the intermediate prompt from crossover of two parent prompts, without deliberately addressing specific errors. Note that we do not need to perform feedback collection, thus topic modeling is also excluded in this variant.

| Iteration | Feedback from last iteration | Prompt | Acc |
|---|---|---|---|
| 0 | / | Decide if the comment is hateful. | 82.30 |
| 2 | Determine if the comment expresses hatred, hostility, or promotes violence specifically targeting people based on their race, ethnicity, religion, gender, or other protected characteristics. The mere presence of negative language or harsh policy views, without targeting protected groups, should not be classified as hate speech. | Evaluate if the comment expresses hatred, hostility, or promotes violence specifically targeting protected characteristics (race, ethnicity, religion, gender, etc.). Note that general negative language or harsh policy views without targeting protected groups should not be classified as hate speech. | 84.00 |
| 6 | Add explicit criteria for what constitutes targeting of protected characteristics. The comment must explicitly connect negative statements to protected characteristics. Mere use of offensive language without clear targeting of protected groups is insufficient. | Follow this process to evaluate hate speech in comments:<br>Does the comment target protected characteristics through either: 1. Explicit targeting: a) Direct mentions of race, ethnicity, religion, gender, etc; b) Clear derogatory language or slurs against protected groups. 2. Implicit targeting: a) Cultural/ethnic stereotypes; b) Coded language commonly used against protected groups c) Context suggesting targeting of specific communities. 3. If the comment: a) Discusses protected groups without promoting hatred; b) Criticizes systemic discrimination; c) Uses general negative language without targeting protected groups → Label as NOT hate speech (0) | 89.00 |
| 11 | Add a new evaluation criterion for "SEVERITY & CREDIBILITY":<br>- Assess if the comment presents a credible expression of hatred or bias<br>- Distinguish between genuine hate speech and:<br>* Nonsensical or grammatically impossible statements<br>* Obviously absurd or illogical claims<br>* Random mentions of protected groups without coherent negative messaging<br>- Require both targeting AND a credible expression of hatred/bias for hate speech classification | Follow this process to evaluate hate speech in comments:<br>1. GROUP TARGET CHECK: a) Identify explicit or implicit targeting of protected characteristics (race, ethnicity, religion, gender, etc.); b) Check for derogatory language, slurs, stereotypes, or coded language against protected groups. 2. SEVERITY & CREDIBILITY ASSESSMENT: a) Evaluate if the content presents a credible expression of hatred/bias; b) Filter out: a) Nonsensical or grammatically impossible statements; b) Obviously absurd or illogical claims; c) Random mentions without coherent negative messaging. 3. CONTEXT & INTENT ANALYSIS: Label as hate speech (1) if ALL conditions are met: a) Targets protected groups; b) Contains credible expression of hatred/bias; c) Promotes discrimination, violence, or hostile stereotyping. Label as NOT hate speech (0) if: a) Lacks credible targeting of protected groups; b) Contains nonsensical or absurd statements; c) Discusses protected groups without promoting hatred | 94.00 |

**Figure 2: Prompt evolution on ETHOS dataset. Key messages in the feedback and the corresponding adjustments in the prompts at each iteration are highlighted in the same color.**




**(a) GSM8K** **(b) ETHOS**

**Figure 3: Training accuracy evolution by GreenTEA and two ablated variants on GSM8K and ETHOS datasets.**

- GreenTEA-TM: this variant uses the gradient-guided GA but replaces the topic modeling with random sampling, collecting incorrect samples by randomly selecting a subset of wrong predictions from the available pool.

To better investigate the performance difference between variants, we visualize the evolution of the average model accuracy over the whole population on the training set of GSM8K and ETHOS datasets after each iteration in Figure 3. We observe that the complete GreenTEA consistently achieves higher accuracy and faster convergence compared to both ablated variants, as it rapidly improves within the first few iterations and continues to exhibit steady gains. The performance drop in GreenTEA-TM highlights the benefits of topic modeling in improving the optimize efficiency as it facilitates more targeted feedback learning. The earlier plateau and lower final accuracies of GreenTEA-TM-GGA further underscore the significance of gradient-guided evolution in prompt optimization.

## 4.4 Model interpretation

To illustrate how GreenTEA systematically enhances prompt quality, Figure 2 presents the feedback and corresponding optimized prompts collected across three iterations for the ETHOS dataset. At each iteration, the LLM analyzer $\mathcal{A}$ identifies key deficiencies in the prompt—such as lack of clarity on group targeting, missing criteria for hate speech, or ambiguity in assessing intent and severity. These critical insights from the LLM analyzer $\mathcal{A}$ (highlighted in color) are explicitly integrated into the prompt by the LLM generator $\mathcal{G}$. Beginning with a basic prompt like "Decide if the comment is hateful", the GreenTEA with the backbone of Claude model family progressively refines instructions into detailed, multi-step guidelines. This structured refinement effectively guides the LLM predictor $\mathcal{M}$ toward more accurate classifications. These improvements of the prompts directly correlate with substantial accuracy gains at iterations 2, 6, and 11 in Figure 3(b), demonstrating GreenTEA's capability in optimizing more interpretable and high-performing prompts.

## 5 Discussion

We propose GreenTEA, an agentic workflow for prompt optimization that emphasizes both effectiveness and efficiency. It iteratively refines prompts by incorporating feedback from previous performance and imitating the natural evolutionary process to generate stronger solutions. Numerical experiments show the superior performance of GreenTEA against current state-of-the-arts. An ablation study further demonstrates the benefits of topic modeling and guided genetic algorithms in enhancing the efficiency and robustness of the optimization process in complex prompt solution space. Extensions of GreenTEA include adopting other clustering methods, *e.g.*, spectral clustering that has demonstrated strong empirical performance [22] to identify informative error patterns; incorporating advanced scoring metrics beyond aggregated accuracy to capture key features that reflect prompt quality [33]; designing efficient workflows [18] to manage the growing complexity of prompts over iterations; and leveraging its non-parametric optimization capabilities for broader solution optimization tasks, *e.g.*, high-dimensional event modeling [4, 7–11, 37], that go beyond prompt refinement.

## References


[1] Jill Baumann and Oliver Kramer. 2024. Evolutionary multi-objective optimization of large language model prompts for balancing sentiments. In *International conference on the applications of evolutionary computation*. Springer, 212–224.

[2] Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 7432–7439.

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

[4] Xiuyuan Cheng, Zheng Dong, and Yao Xie. 2025. Deep spatio-temporal point processes: Advances and new directions. *arXiv preprint arXiv:2504.06364* (2025).

[5] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168* (2021).

[6] Yihong Dong, Kangcheng Luo, Xue Jiang, Zhi Jin, and Ge Li. 2024. PACE: Improving Prompt with Actor-Critic Editing for Large Language Model. In *Findings of the Association for Computational Linguistics ACL 2024*. 7304–7323.

[7] Zheng Dong, Xiuyuan Cheng, and Yao Xie. 2023. Spatio-temporal point processes with deep non-stationary kernels. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=PsIk0kO3hKd

[8] Zheng Dong, Zekai Fan, and Shixiang Zhu. 2025. Conditional Generative Modeling for High-dimensional Marked Temporal Point Processes. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*. 248–259.

[9] Zheng Dong, Jorge Mateu, and Yao Xie. 2024. Spatio-temporal-network point processes for modeling crime events with landmarks. *arXiv preprint arXiv:2409.10882* (2024).

[10] Zheng Dong, Matthew Repasky, Xiuyuan Cheng, and Yao Xie. 2023. Deep graph kernel point processes. In *Temporal Graph Learning Workshop@ NeurIPS 2023*.

[11] Zheng Dong, Shixiang Zhu, Yao Xie, Jorge Mateu, and Francisco J Rodríguez-Cortés. 2023. Non-stationary spatio-temporal point process modeling for high-resolution COVID-19 data. *Journal of the Royal Statistical Society Series C: Applied Statistics* 72, 2 (2023), 368–386.

[12] Ming Fan, Wenying Wei, Wuxia Jin, Zijiang Yang, and Ting Liu. 2022. Explanation-guided fairness testing through genetic algorithm. In *Proceedings of the 44th International Conference on Software Engineering*. 871–882.

[13] Boxin Guan, Changsheng Zhang, and Jiaxu Ning. 2016. EDGA: a population evolution direction-guided genetic algorithm for protein–ligand docking. *Journal of Computational Biology* 23, 7 (2016), 585–596.

[14] Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. 2024. Connecting Large Language Models with Evolutionary Algorithms Yields Powerful Prompt Optimizers. In *The Twelfth International Conference on Learning Representations*.

[15] John H Holland. 1992. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press.

[16] Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics* 8 (2020), 423–438.

[17] Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T Joshi, Hanna Moazam, et al. 2023. Dspy: Compiling declarative language model calls into self-improving pipelines. *arXiv preprint arXiv:2310.03714* (2023).

[18] Sehoon Kim, Suhong Moon, Ryan Tabrizi, Nicholas Lee, Michael W Mahoney, Kurt Keutzer, and Amir Gholami. 2024. An llm compiler for parallel function calling. In *Forty-first International Conference on Machine Learning*.

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

[20] Adam Lipowski and Dorota Lipowska. 2012. Roulette-wheel selection via stochastic acceptance. *Physica A: Statistical Mechanics and its Applications* 391, 6 (2012), 2193–2196.

[21] Fei Liu, Xialiang Tong, Mingxuan Yuan, and Qingfu Zhang. 2023. Algorithm evolution using large language model. *arXiv preprint arXiv:2311.15249* (2023).

[22] Xiaoyue Liu, Praveen Muthukrishnan, and Benoit Montreuil. 2025. Network Design and Capacity Management in Hyperconnected Urban Logistic Networks. In *Proceedings of 11th International Physical Internet Conference*.

[23] Ioannis Mollas, Zoe Chrysopoulou, Stamatis Karlos, and Grigorios Tsoumakas. 2022. ETHOS: a multi-label hate speech detection dataset. *Complex & Intelligent Systems* 8, 6 (2022), 4663–4678.

[24] Archiki Prasad, Peter Hase, Xiang Zhou, and Mohit Bansal. 2023. GrIPS: Gradient-free, Edit-based Instruction Search for Prompting Large Language Models. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*. 3845–3864.

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

[26] Kiran Ramnath, Kang Zhou, Sheng Guan, Soumya Smruti Mishra, Xuan Qi, Zhengyuan Shen, Shuai Wang, Sangmin Woo, Sullam Jeoung, Yawei Wang, et al. 2025. A Systematic Survey of Automatic Prompt Optimization Techniques. *arXiv preprint arXiv:2502.16923* (2025).

[27] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 3982–3992.

[28] Xavier Sécheresse, Antoine Villedieu de Torcy, et al. 2025. GAAPO: Genetic Algorithmic Applied to Prompt Optimization. *arXiv preprint arXiv:2504.07157* (2025).

[29] Hong Sun, Xue Li, Yinchuan Xu, Youkow Homma, Qi Cao, Min Wu, Jian Jiao, and Denis Charles. 2023. Autohint: Automatic prompt optimization with hint generation. *arXiv preprint arXiv:2307.07415* (2023).

[30] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, et al. 2023. Challenging BIG-Bench Tasks and Whether Chain-of-Thought Can Solve Them. In *Findings of the Association for Computational Linguistics: ACL 2023*. 13003–13051.

[31] Hiroto Tanaka, Naoki Mori, and Makoto Okada. 2023. Genetic Algorithm for Prompt Engineering with Novel Genetic Operators. In *2023 15th International Congress on Advanced Applied Informatics Winter*. IEEE, 209–214.

[32] Hanwei Xu, Yujun Chen, Yulun Du, Nan Shao, Wang Yanggang, Haiyu Li, and Zhilin Yang. 2022. GPS: Genetic Prompt Search for Efficient Few-Shot Learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. 8162–8171.

[33] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. 2024. Large Language Models as Optimizers. In *The Twelfth International Conference on Learning Representations*.

[34] Tianjun Zhang, Xuezhi Wang, Denny Zhou, Dale Schuurmans, and Joseph E. Gonzalez. 2023. TEMPERA: Test-Time Prompt Editing via Reinforcement Learning. In *The Eleventh International Conference on Learning Representations*.

[35] Mingkai Zheng, Xiu Su, Shan You, Fei Wang, Chen Qian, Chang Xu, and Samuel Albanie. 2023. Can gpt-4 perform neural architecture search? *arXiv preprint arXiv:2304.10970* (2023).

[36] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2022. Large Language Models are Human-Level Prompt Engineers. In *The Eleventh International Conference on Learning Representations*.

[37] Shixiang Zhu, Haoyun Wang, Zheng Dong, Xiuyuan Cheng, and Yao Xie. 2022. Neural Spectral Marked Point Processes. In *International Conference on Learning Representations*. https://openreview.net/forum?id=0rcbOaoBXbg

[38] Chris Zhuang, Debadyuti Mukherjee, Yingzhou Lu, Tianfan Fu, and Ruqi Zhang. 2025. Gradient GA: Gradient Genetic Algorithm for Drug Molecular Design. *arXiv preprint arXiv:2502.09860* (2025).

**Prompt template for LLM predictor $\mathcal{M}$**

**Human:** Here is the problem you are asked to solve, provided in the XML tag <question></question>
<question>
{input_text}
</question>

Please follow the instruction in the XML tag <Instruction></Instruction>.
<Instruction>
{adjustable_prompt}
</Instruction>

Present your response in the following format:

---
<Annotation>
Anything you need to write before your final answer
</Annotation>

<Answer>
Your final answer
</Answer>
---

**Assistant:**

**Figure 4: Prompt template for predictor $\mathcal{M}$**

**Prompt template for LLM analyzer $\mathcal{A}$**

**Human:** You are assisting in improving the instruction used by an LLM to solve given problems. The model uses the instruction to reason through the problem and provide its answer. However, the current instruction has led to some incorrect answers.

Your task is to analyze the failure cases and identify how the current instruction contribute to these mistakes. Then, propose a revision that makes the instruction more effective and generalizable.

The current instruction is:
{cur_prompt}

The instruction led to incorrect answers on the following examples:
{example_string}

Analyze why the model failed and what about the instruction might have contributed to those errors. Wrap your analysis in <erroranalysis></erroranalysis>.

Then, provide a revised version or improvement suggestion. Your suggestion should be:
- Concise and actionable
- Focused on general improvement (not tied to any specific example)
- Effective across a wide range of similar cases

Wrap your suggestion in <suggestion></suggestion>.

Use the following format in your response:
---
The current instruction has the following limitations:
<erroranalysis>
</erroranalysis>

To improve it, I suggest the following revision:
<suggestion>
</suggestion>
---

**Assistant:**

**Figure 5: Prompt template for analyzer $\mathcal{A}$**

**Prompt template for LLM generator $\mathcal{G}$**

**Human:** You are assisting in improving the instruction used by an LLM to solve given problems. The instruction guides the LLM's behavior during prediction.

I have two instructions with each of them have some deficiencies. Your task is to help generate a new instruction (i.e., child instruction) based on these two instructions (i.e., parent instructions) and improve it by solve the deficiencies of the parent instructions.

The first parent instruction is:
{prompt1}

It gets the following examples wrong:
{wrong_examples1}

The problems with this parent instruction and the potential improvements are:
{gradient1}

The second parent instruction is:
{prompt2}

It gets the following examples wrong:
{wrong_examples2}

The problems with this parent instruction and the potential improvements are:
{gradient2}

Please begin by creating a child instruction. The child instruction is obtained by crossover the two parent instructions. Make necessary but modest adjustments to ensure a grammatically correct text. Wrap the child instruction within tags <ChildPrompt></ChildPrompt>.

Then, follow the potential improvements on parent instructions and optimize your child instruction so that:
1. The optimized instruction should solve the parents' problems by following the improvements.
2. The optimized instruction should serves as a valid instruction for the given task.
3. The optimized instruction is concise, effective, and generally applicable to all cases.

Return the optimized instruction, wrapped with <OptimizedPrompt> and </OptimizedPrompt>

Use the following format to produce your response:

---

<ChildPrompt>
Your child prompt after crossover
</ChildPrompt>.

<OptimizedPrompt>
Your optimized child prompt
</OptimizedPrompt>

---

**Assistant:**

**Figure 6: Prompt template for generator $\mathcal{G}$**

**Meta-prompt template.** See the meta-prompt templates for different agents in `GreenTEA` in Figure 4, 5, and 6.

# A Details of experimental setup

**Dataset.** The testing set contain 1,000 samples for GSM8K and PIQA data, contain 100 samples for ETHOS data, and contain 25 samples for each BBH task.

**Experiment configuration.** We set the temperature of the LLM predictor $\mathcal{M}$ to be 0.0 when evaluating the performance of generated prompts, and set the default temperature to be 1.0 for LLM analyzer $\mathcal{A}$ and generator $\mathcal{G}$ to generate diverse outputs. For the generic algorithm, we set the number of iteration to be 20 and the population size to be 4, *i.e.*, generate 4 new prompts in each iteration and keep the best 4 prompts as the updated population after each iteration. We evaluate the testing accuracy of the prompts in the last population and report their average as the final prompt accuracy by `GreenTEA`. The results of the testing accuracy in Table 1 are averaged over three independent runs.

## B Algorithm

we present the full algorithmic workflow of GreenTEA in Algorithm 1. The method maintains a population of prompts that are iteratively refined through a feedback-guided evolutionary process. In each iteration, error patterns are extracted from model predictions via topic modeling, analyzed by an LLM agent, and used to guide new prompt generation through crossover and mutation.

**Algorithm 1** GreenTEA for automatic prompt optimization

**Input:** Population size $K$, max number of iteration $T$, scoring function $s(\cdot)$, training data $\mathcal{D}_{\text{tr}}$, LLM predictor $\mathcal{M}$, LLM analyzer $\mathcal{A}$, LLM generator $\mathcal{G}$

**Initialization:** Population $\mathcal{P}^{(0)} = \{\mathcal{P}_0^{(0)}, \ldots, \mathcal{P}_{k-1}^{(0)}\}$, $t = 0$

**while** $t \leq T$ **do**

  **for** each prompt $\mathcal{P}_k^{(t)}$ in $\mathcal{P}^{(t)}$ that has not been evaluated **do**

    1. Get score $f_k = s(\mathcal{P}_k^{(t)}, \mathcal{D}_{\text{tr}}, \mathcal{M})$ and collect $\{\widehat{a_n}\}_{n=1}^{N}$;

    2. Get wrong samples $\mathcal{W}_k^{(t)}$ using topic modeling;

    3. Input $\mathcal{W}_k^{(t)}$ to $\mathcal{A}$ and obtain feedback $p_k^{(t)}$ of $\mathcal{P}_k^{(t)}$;

  **end for**

  $\mathcal{P}^{(t)}$ = prompts in $\mathcal{P}^{(t)}$ with the top-$K$ scores;

  $\mathcal{P}^{(t+1)} = \mathcal{P}^{(t)}$;

  **for** $k \in \{0, \ldots, K-1\}$ **do**

    1. Select parent prompts $\mathcal{P}_i^{(t)}, \mathcal{P}_j^{(t)}$ from $\mathcal{P}^{(t)}$ using roulette wheel selection;

    2. Generate a new prompt $\mathcal{P}_k^{(t+1)}$ by inputting $\mathcal{P}_i^{(t)}, \mathcal{P}_j^{(t)}, p_i^{(t)}, p_j^{(t)}$ to the generator $\mathcal{G}$;

    3. $\mathcal{P}^{(t+1)} = \{\mathcal{P}^{(t+1)}, \mathcal{P}_k^{(t+1)}\}$

  **end for**

  $t \leftarrow t + 1$;

**end while**

**return** $\mathcal{P}^{(T)}$ = prompts in $\mathcal{P}^{(T)}$ with the top-$K$ scores;