

RESEARCH ARTICLE

Toward Fully Automated Prompting: An LLM-Powered Genetic Algorithm for Prompt Generation, Optimization, and Self-Evaluation

Leandro A. Loss^{1,2,*} and Pratikkumar Dhuvad¹

¹AI R&D, AML RightSource, USA

²MBA Department, ESSCA School of Management, France

Abstract: Large language models (LLMs) have gained recognition as valuable assets across virtually all industries, yet they rely heavily on manually crafted input prompts. In real-world applications, the dependence on specialized staff, skilled prompt engineering, and domain-specific knowledge often leads to suboptimal performance and increased costs. In this study, we investigate the use of genetic algorithms (GAs) to generate, evolve, and judge LLM prompts in a completely autonomous fashion. The main novelty presented here is the integration of general-purpose LLM-guided genetic operators with LLM-based fitness evaluation, enabling prompt optimization without human intervention. Full prompting automation is possible via the customization of a standard GA implementation to handle textual individuals, which are manipulated by LLM-guided genetic operators that iteratively create and enhance candidate prompts. Additionally, LLMs are employed to assess the correctness of outputs, forming the basis of our GA's fitness function. Our experimental results indicate that our approach produces solutions that are, on average, 22% more correct than those generated by humans. This conclusion is supported by extensive testing utilizing 10 public datasets and 6 modern LLMs by OpenAI, Meta, and MistralAI. Ablation studies and sensitivity analysis further substantiate our approach's robustness under probabilistic scenarios. Our findings suggest that this GA-driven prompt engineering approach can produce superior solutions compared with those written by prompt engineers who possess technical skills but lack domain-specific knowledge and a full understanding of vendor-specific prompting idiosyncrasies. Ultimately, this study highlights the viability and potential of fully automated optimization for minimizing human effort in writing performant prompts.

Keywords: genetic algorithms, LLM prompt optimization, automated prompt engineering, LLM-guided genetic operators

1. Introduction

It is unquestionable that large language models (LLMs) have transformed the field of artificial intelligence (AI). By utilizing transformer-based neural networks, massive datasets, and fine-tuned reinforcement learning strategies, these models excel at understanding and generating human-like text, bringing substantial benefits to numerous applications [1, 2]. However, their accuracy still relies heavily on users providing clear, precise input instructions, in an activity known as prompt engineering [3]. This requirement poses challenges, especially for teams lacking resources, working with domain-specific content, or migrating across different model providers. Therefore, automating prompt engineering has the potential to significantly enhance LLM performance across various tasks and industries by reducing reliance on human input.

To address these challenges, novel methods for prompt generation and optimization have been proposed, aiming at automating this process [4–6]. The task is, however, complex due to vendor-specific requirements and the nuanced characteristics of natural language itself. Combined, these characteristics cause minor variations in wording and input structure to produce distinct responses from an LLM [7]. For example, it has been shown that in sentiment analysis, a prompt that includes context and some specific linguistic features can yield more accurate results than a generic, less structured prompt [8]. In categorization tasks, input–output examples provided as context can greatly improve outcomes [2]. Currently, prompt optimization involves domain-specific instructions, manual adjustment, and frequent trial and error. These practices are time-consuming and may not yield optimal results, particularly across wide sets of tasks. Therefore, there is certainly a need for efficient approaches that can systematically explore a large space of candidate prompts and enhance LLM performance [9].

*Corresponding author: Leandro A. Loss, AI R&D, AML RightSource, USA and MBA Department, ESSCA School of Management, France. Email: leandro.loss@essca.eu

In this study, we build on our initial exploration of genetic algorithms (GAs) for prompt optimization. Here, we introduce enhancements to the methods presented in our preliminary works and greatly extend their experimental setups [10, 11]. Among the various optimization algorithms available today, we chose GAs for this work because they have been shown to be highly effective across a wide range of optimization problems (Section 2). Moreover, by mimicking natural selection and evolution, GAs are able to balance exploration (discovery of new solutions) with exploitation (refinement of existing solutions) [12], providing an ideal optimization framework for prompt engineering.

GAs start with a population of random individuals (i.e., candidate solutions) that “evolve” through genetic operators, such as selection, crossover, and mutation. Individuals are continually evaluated by a fitness function that judges their outcome quality. This process is repeated over several generations, gradually converging toward optimal or near-optimal solutions [13]. That also enables GAs to efficiently search large spaces and find innovative solutions, making them ideal for prompt optimization.

The primary goal of this study is to assess, both quantitatively and qualitatively, the effectiveness of GAs in automating prompt engineering for modern LLMs. By treating prompts as individuals in a GA, we can autonomously explore various configurations, potentially discovering strategies that surpass manual methods. More specifically, we perform adaptations to standard GA operators to work with textual individuals, data samples, and *meta prompts* (i.e., generic LLM instructions). The proposed customizations direct the GA to perform generation and gradual refinement of prompts and drive optimization. This automated method also overcomes challenges such as vendor-specific guidelines and language barriers by using the same LLM for both prompt engineering and problem-solving. In this study, we investigate (i) whether GAs can consistently find high-performing prompts without human intervention or explicit domain knowledge and (ii) how these prompts compare to those created by humans. To this end, we compare our GA-optimized prompts against manually crafted prompts using modern LLMs, including OpenAI’s GPT-4o, Meta’s Llama-3.3-70B, and MistralAI’s Mistral Large, across various domain-specific datasets.

We believe GA-optimized prompts can facilitate LLM adoption in specialized fields, offering seamless adaptability to niche problems with minimal human effort. This study contributes to the fields of computational optimization, generative AI, and evolutionary computing by proposing an LLM-powered GA method to fully automate prompt engineering.

To investigate the feasibility of fully automated prompt optimization for LLMs, this study makes the following contributions:

Method: We propose an LLM-powered GA in which prompts evolve using meta-prompt-based initialization, mutation, and crossover operators.

Automation: We introduce a fully automated optimization loop where an LLM generates candidate prompts and a second LLM evaluates outputs through the fitness function.

Evaluation: We validate our approach across 10 datasets and 6 modern LLMs, comparing GA-optimized prompts against no-prompt baselines and prompts written by experienced engineers.

Analysis: We examine the reliability of LLMs serving as evolutionary operators and fitness evaluators (i.e., LLM-as-a-judge), including, among other factors, the impact of evaluation misjudgments on the overall evolutionary process.

This paper is organized as follows: After this Introduction, we perform a brief yet targeted literature review in Section 2, offering an overview of the current technical landscape for contextualization of our study. In Section 3, we outline the methodology we utilized for integrating GAs with LLMs for prompt optimization, along with a full description of our experiments. In Section 4, we present the results of our research, their implications, and a comprehensive rationale for the findings. Finally, Section 5 discusses our findings, their significance within the broader field of generative AI and prompt engineering automation, and potential directions for future research.

2. Related Literature

The release of transformer-based language models in the late 2010s, such as those introduced by Vaswani et al. [14] and Devlin et al. [15], marked a turning point in AI and natural language processing (NLP). Subsequent work demonstrated that performance gains could be significantly improved through more efficient scaling and training strategies [16]. This trajectory resulted in the release of OpenAI’s GPT-4 in 2023 [17], which catalyzed the widespread adoption of LLMs across various sectors, including ZenDesk’s AI-powered customer support solution, JasperAI’s marketing automation tool, Thomson Reuters’s Legal AI assistant, and BloombergGPT’s financial analyzer [18], to cite a few. As LLMs have become ubiquitous, their ability to perform tasks with natural language guidance, understand complex queries, and generate human-like text has made them appealing across many industries. Moreover, their emergent capabilities, such as decision-making and reasoning, combined with other advancements in other technologies, such as agentic workflows and retrieval-augmented generation, have further enhanced their utility, making LLMs faster, more cost-effective, and more reliable [19].

Despite their powerful capabilities, the adoption of LLMs has posed challenges, particularly in industries where technical expertise is limited. Manual prompt engineering remains a critical step for optimizing LLM performance [20, 21], yet it can be a barrier, for example, for non-experts or easy migration between competing providers. To address these challenges, researchers have developed semi-automated methods to facilitate LLM utilization. Notable innovations include AutoPrompt by Shin et al. [4], which uses gradient-guided search for prompt generation, and FLUENT PROMPT by Shi et al. [5], which applies linguistic constraints for effective prompt creation. Reinforcement learning-based approaches, like RLPrompt [6] and TEMPERA [22], have also emerged, offering dynamic prompt adaptation. Methods such as automatic prompt optimization [23] and learning-to-plan [24] further reduce the reliance on technical expertise, streamlining the use of LLMs across various problems and providers.

In the field of optimization, GAs have long been effective in solving complex optimization problems. They have been successfully utilized across numerous tasks, from engineering design [13] to financial modeling [25], demonstrating their versatility and robustness in exploring complex, large search spaces. The application of GAs in NLP, though relatively less explored, is gaining traction. Studies like those by Lehman et al. [26] and Meyerson et al. [27] demonstrate the use of LLMs to enhance genetic programming and prompt evolution. Tanaka et al. [28] introduced a GA-based prompt engineering solution for the Japanese language. Other works, such as EvoPrompt [29], further integrate LLMs with GAs and other evolutionary algorithms under human guidance, while research by Guo et al. [30] and Hsieh et al. [31]

investigates GA-based prompt optimization across different tasks. These studies show promise but often require some type of human intervention, either by constraining initial criteria or artificially driving genetic operations based on domain-specific instructions. Despite this, the potential for LLMs as evolutionary operators in GAs continues to grow, with active research striving to overcome these limitations.

For a side-by-side comparison, Table 1 provides a compact guide to the differences between our proposed LLM-powered GA and five of the previously cited LLM-driven prompt optimization solutions most similar to ours.

3. Method and Experiments

In this section, we describe our method and the experiments conducted to explore whether LLMs can autonomously generate effective and optimal prompts without human intervention or the need for domain-specific knowledge. Our goal is to determine if LLMs can act as their own prompt engineers in a fully automated manner. To this end, we developed a GA with custom genetic operators designed to iteratively propose new input prompts. These prompts are generated based on domain-agnostic meta prompts and a set of problem-specific input-output

Table 1
Compact guide to the differences between our proposed method and similar solutions cited in Section 2

Method	Need for labels/ground truth	Manual templates or seed prompts	Human-in-the-loop	Optimization target	Degree of automation
AutoPrompt [4]	Yes: requires labeled training examples to compute gradients	Yes: fixed prompt template with trigger tokens	No during optimization, but template design required	Discrete token triggers inserted into a template to maximize downstream task accuracy	Low-moderate: automated token search but constrained by a human-designed template
RLPrompt [6]	Yes: reward computed from labeled task performance	Often requires an initial prompt template or structure	No during training but task/reward design required	Reinforcement learning policy that generates discrete prompt tokens maximizing reward	Moderate: automated token generation but depends on labeled reward signal
TEMPERA [22]	Yes: uses task outputs or evaluation signals for RL reward	Yes: edits an existing prompt rather than generating from scratch	Indirect: initial prompt supplied by user	Test-time prompt editing to improve task performance	Moderate: automated editing but relies on a starting prompt and reward signal
EvoPrompting [29]	Often yes: task-based evaluation	Yes: typically initialized from human prompts or templates	Yes: evolutionary search often guided or seeded by humans	Evolutionary optimization of prompt structures or programs	Moderate: evolutionary search but human-guided initialization
LLMs as optimizers [30]	Not strictly required: but evaluation signals usually provided	Often yes: optimization target and prompt structure defined manually	Typically yes: humans define objective and evaluation pipeline	LLM iteratively proposes improved solutions for a defined objective	Moderate-high: automated proposal generation but human-defined optimization loop
Proposed LLM-powered GA (this work)	Yes: uses input-output examples for automated fitness evaluation	No domain-specific templates: only generic meta prompts	No human intervention after initialization	Evolutionary search over full prompt text using LLM-driven initialization, mutation, crossover, and evaluation	High-fully automated: prompt generation, evolution, and self-evaluation performed autonomously

examples. Our experiments explore 6 state-of-the-art LLMs and 10 problems, each represented by a unique dataset. The following sections provide a comprehensive overview of our method and experiments, including the reasoning behind each methodological choice.

3.1. LLM-powered genetic algorithm

In this study, we customized a standard GA to work with textual individuals, which are manipulated using operators defined by LLMs. Each individual in our GA is represented as an LLM prompt, serving as a candidate solution to the given task. We replaced traditional genetic operators like initialization, crossover, and mutation with meta prompts: generic, domain-agnostic prompts crafted to drive prompt generation and the evolution of individuals. Tables B1–B3 (Appendix B) illustrate generated prompts for initialization, mutation, and crossover. As discussed in Section 2, similar adaptations have been proposed in previous studies [26, 27, 32], with more recent examples proposed in References [33, 34]. The concept of domain-agnostic meta prompts has also been explored by Yang et al. [35], though their effectiveness has been debated by Ye et al. [36]. Our method is unique in that it minimizes user influence and reliance on domain-specific instructions, relying instead on insights that can be directly inferred from data samples. To ensure diversity and prevent premature convergence of the fitness function, a common problem with GAs, we used a tournament selection strategy with a size of 3¹.

Algorithm 1: LLM-Powered GA for Prompt Optimization

```

0: input→population size, number of generations, mutation
probability, LLM name, and data samples
1: Initialize population using system and initialization meta
prompts, and data samples
2: Evaluate fitness of individual prompts using fitness and system
fitness meta prompts
3: While number of generations is not reached:
4: Perform selection using trio tournament selection
5: Perform pair crossover using crossover meta prompt
6: Perform conditional mutation using mutation meta prompt
7: Evaluate fitness of resulting prompts using fitness and system
fitness meta prompts
8: Increment generation count
9: End While loop
10: output←individual prompt with highest fitness

```

To maintain consistency across experiments and manage computational costs, we fixed the population size at 20 individuals and limited the number of generations to 20. These values proved sufficient across most problems in our experiments. Fitness was evaluated using 500 randomly preselected data samples, even when larger datasets were available. This methodology allowed us to assess our method’s efficiency under controlled and reproducible conditions. The detailed solution is presented in

¹The literature reveals mixed experiences with selection strategies when applied to prompt optimization. For example, Tanaka et al.’s findings [28] also used tournament selection, but Guo et al. [30] recommended roulette wheel selection. Our experience is that maintaining population diversity proved challenging with every traditional strategy applied individually, and we plan to explore the use of novel and concurrent use of these strategies in the future.

Algorithm 1, with the associated code and datasets made publicly available via Reference [37]. Ablation studies in the Appendix of this work analyze the influence of distinct setups.

3.2. Meta prompts

Our customization of the GA involves replacing traditional random initialization and bitwise operators with textual instructions. These instructions, referred to here as meta prompts, are predefined and domain-agnostic prompts that guide the language model in creating, modifying, and assessing individuals, which are also prompts, to simulate processes like initialization, crossover, mutation, and fitness evaluation. The meta prompts are designed to use only generic language, sample data (for initialization), previously generated GA prompts (for crossover and mutation), and output answers (for fitness evaluation). This design enables our method to be applied across a diverse range of supervised learning problems typically represented by input–output pairs. More specifically, during LLM-based initialization, a meta prompt instructs the GA to generate a population of candidate prompts, each addressing the problem defined by the sample data in its own way. Mutation and crossover meta prompts further instruct the GA to create new prompts based on one or two existing prompts, respectively. Finally, a meta prompt instructs the GA’s fitness function to compare the ground truth and the resulting output of candidate prompts. Appendix B illustrates prompts generated during initialization, mutation, and crossover by our GA.

To enhance the consistency and reliability of the LLM’s responses, we designed system prompts to establish basic safeguards and guide the overall model behavior. Our system prompts set overarching parameters and guidelines and work alongside each specific meta prompt. Table 2 outlines our meta prompts and their roles in the LLM-powered GA. The sample data used by initialization prompts comprise a pre-determined number of randomly selected input–output pairs. Similar to other machine learning algorithms, these data allow our method to derive context and infer domain knowledge without specialized instructions. The exact number of samples varies across problems and models and is calculated based on the length of each input–output pair and the available context length in each LLM. For example, LLMs with 128k-token context windows can accommodate up to 200 samples when categorizing short sentences, but only 10–20 samples when interpreting lengthy financial reports. As with traditional machine learning tasks, the performance of generated solutions is influenced by the quality and quantity of data samples. However, in this context, it is more difficult to establish a direct relationship between input samples and output quality because LLMs have the unique advantage of being pre-trained on extensive datasets, which can help mitigate the effects of insufficient or inadequate context. Nonetheless, the sample data are fixed across all models (subject to LLM context-length constraints), so if a suboptimal sample selection occurs, it should affect all models equally on that specific problem.

3.3. Fitness function

The formulation of fitness functions is crucial for the effectiveness and efficiency of GAs, as extensively shown by independent literature [38–40]. To maintain the automated nature of the optimization process, we developed meta prompts that instruct a separate LLM to evaluate the generated output against ground truth and judge their semantic equivalence (refer to Table 2). This operation yields a binary “yes” for matching answers or “no”

Table 2
Meta prompts: generic LLM instructions that drive GA initialization, mutation, crossover, and fitness evaluation

Role	Prompt
System	You are an AI that helps people solve problems. Avoid comments outside the proposed prompt as the user will use your answer to integrate with another downstream system
Initial	Create a [LLM_MODEL] prompt that solves the problem exemplified by the following examples: [SAMPLE_QUESTION_1] [SAMPLE_ANSWER_1] [SAMPLE_QUESTION_2] [SAMPLE_ANSWER_2] ...
Crossover	Given the following drafts for two prompts that aim to solve the same particular problem, create a better prompt using only ideas from them. [PROMPT_A] [PROMPT_B]
Mutation	Given the following draft for a prompt that aims to solve a particular problem, create a better prompt using ideas from it. [PROMPT_A]
System fitness	You are an AI that validates automated answers against ground truth. You only answer “yes” or “no” with no extra comments, notes, or explanations as the user will use your answer to integrate with another downstream system.
Fitness	My ground truth is [TRUTH]. Does the automated output [ANSWER] linguistically, symbolically, conceptually, or fundamentally match my ground truth?

otherwise. Because the judging model solves a purely discriminative task without any knowledge of the underlying generative process, it is unlikely to favor responses by any specific model under evaluation. An individual’s fitness score in any generation is determined by the average rate of matching answers across an evaluation dataset, which is separate from the data samples used during initialization. Refer to Tables C1 and C2 (Appendix C) for examples of positive and negative matches between LLM outputs and the expected golden answer.

In addition to quality metrics, the fitness function can potentially incorporate other elements. Factors such as the length of prompts and responses, their costs, and other considerations can significantly impact the GA’s selection of individuals and better tune it to users’ requirements. These factors, along with their desired scaling coefficients, can be easily added to the formulation:

$$\text{Fitness} = \frac{\sum_{(i=1)}^N (\alpha \times \text{matching}_i \pm \beta \times \text{length}(\text{prompt}_i) \pm \gamma \times \text{length}(\text{response}_i) \pm \delta \times \text{cost}(\text{request}_i) \pm \dots)}{N},$$

where N = GA population size,

matching $\in \{0, 1\}$,

length(x) = number of tokens in x,

cost(x) = cost of tokens in x, and

$\alpha + \beta + \gamma + \dots = 1.0$.

For this study, we set $\alpha = 1.0$ and $\beta, \gamma, \delta, \dots = 0.0$ in all experiments. Failure to normalize measurements or to tune coefficients appropriately (e.g., overweighting prompt length relative to matching rate) may yield poor results, causing the GA to favor short but inaccurate prompts over longer, more accurate ones. Ablation studies in Appendix A of this work analyze the impact of distinct setups.

3.4. Datasets

To validate our method across a range of tasks, including multilingual and domain-specific text classification, comprehension, reasoning, and question answering, we evaluated it on 10 public datasets: (1) Academic Topic Classification, (2) Financial Math Reasoning, (3) Bulgarian Academic Q&A, (4) ScienceQA, (5) TruthfulQA, (6) China Law, (7) MS MARCO, (8) Liar, (9) Banking77, and (10) Criminality Sentiment Analysis. Some of these datasets are established NLP benchmarks, whereas others are less widely used; however, all are documented in the technical literature and available through Hugging Face Datasets (<https://huggingface.co/datasets>). In preliminary experiments, we also considered traditional benchmarks, such as SST-2, SST-5, and AG’s News [4], but these proved insufficiently challenging for modern LLMs: the GA’s initial population often achieved perfect scores, likely reflecting the models’ capabilities and/or prior exposure to these datasets during training or fine-tuning. Our goal is to demonstrate effectiveness in tasks where expert prompt engineering is critical to performance. For that reason, we excluded many popular but comparatively easy datasets from this study.

The sections below describe each dataset. For each dataset, we first randomly separated a fixed evaluation set of 500 samples from the remaining data used for initialization, subject to the LLMs’ context-length constraints. We then reduced each dataset to the input–output fields only, removing auxiliary metadata to enforce a consistent format and to simplify the shareable version of our implementation.

Academic topic classification: Released by S. Anandhu (huggingface.co/datasets/anandhu-sct/topic_classification), this dataset contains over 3000 academic problems labeled across 254 topics spanning mathematics, computer science, and logic. The large label space and fine-grained distinctions make it a stringent test for multi-class classification and prompt design.

Financial math reasoning: Described by Cheng et al. [41], this dataset includes nearly 1500 reading-comprehension items derived from financial news. It requires reasoning about domain-specific numerical and temporal information (amounts, dates,

market indicators, etc.), making it suitable for evaluating prompt engineering under mixed textual–numerical constraints.

Bulgarian academic Q&A: Released by Hardalov et al. [42], this dataset includes over 2600 multiple-choice questions in Bulgarian across several academic domains. Its non-English language setting and specialized content test multilingual prompting and domain adaptation.

ScienceQA: Described by Saikh et al. [43], this dataset contains approximately 21,000 science questions spanning multiple disciplines. It targets domain-specific comprehension and reasoning tasks and serves as a robust prompt-engineering testbed.

TruthfulQA: Introduced by Lin et al. [44], this dataset contains over 800 questions across domains such as health and politics and is designed to assess whether models avoid producing false yet plausible answers. It is therefore useful for evaluating prompts aimed at improving truthfulness.

China law: Studied by Yuan et al. [45], this dataset comprises over 2.6 million Chinese legal cases with annotations for legal judgment prediction. The combination of technical legal language and a non-English setting makes this dataset particularly challenging for LLMs and prompt optimization in general.

MS MARCO: Introduced by Nguyen et al. [46], MS MARCO contains over one million real user queries from the Bing search engine with human-annotated answers. Its open-ended, organic queries provide a demanding setting for prompt design.

Liar dataset: Compiled by Wang [47], this dataset includes 12,800 human-labeled statements from PolitiFact. The task requires distinguishing degrees of truthfulness using nuanced language and contextual cues, making it appropriate for evaluating prompt strategies for fact-checking and classification.

Banking77: Introduced by Casanueva et al. [48], this dataset contains roughly 13,000 banking-related user queries labeled with 77 intents. It provides a focused benchmark for intent classification in a specialized domain.

Criminality sentiment analysis: This newly compiled dataset [37] includes approximately 2000 criminal news articles with manually annotated questions and answers concerning entities' criminal participation and facilitation. It supports evaluation of LLM capabilities in named-entity recognition, coreference resolution, domain-specific sentiment analysis, and causal inference, all of which benefit from careful prompt generation.

3.5. Large language models

In our study, we evaluated six LLMs to assess prompt optimization using our customized GA. Our selected models comprise various versions, families, sizes, and access regimes (proprietary Application Programming Interface (API) models and open-weight models) to provide coverage across different architectures and instruction-tuning conventions. Although this selection is not exhaustive, it encompasses a diverse set of prompting interfaces and constraints, and it subjects prompt-optimization methods to distinct context handling, formatting requirements, and response patterns. The evaluated models are summarized below.

OpenAI's GPT-4o (version: 2024-08-06): GPT-4o was developed by OpenAI and is reported to perform strongly on general and complex tasks such as writing, analysis, and comprehension (<https://openai.com/index/hello-gpt-4o/>). It is available via a proprietary API. Its source code and weights were not publicly released.

OpenAI's GPT-4o-mini (version: 2024-07-18): GPT-4o-mini is a smaller, low-latency, and low-cost variant intended to retain strong performance on many NLP tasks while improving

efficiency. It is also provided via a proprietary API with closed source and closed weights.

MistralAI's Mistral Large 24.11: Mistral Large is a high-capacity model from MistralAI designed for strong general instruction-following performance across domains (<https://mistral.ai/news/>). It is offered as a proprietary model; its source code and weights are not publicly released.

MistralAI's Mistral NeMo 12B: Mistral NeMo is a 12B-parameter model positioned for efficient inference and adaptation to domain-specific tasks (<https://mistral.ai/news/mistral-nemo/>). Unlike Mistral Large, it is released as an open-weight model (Apache 2.0).

Meta's Llama-3.3-70B-Instruct: Llama-3.3-70B-Instruct is a 70B-parameter instruction-tuned model designed for complex instruction-following and text-generation workloads (<https://www.llama.com/>). It is distributed as an open-weight model under Meta's Llama license (i.e., not an OSI-approved open-source license).

Meta's Llama-3.2-11B-Vision-Instruct: Llama-3.2-11B-Vision-Instruct is an 11B-parameter multimodal instruction-tuned model intended for vision-and-language use cases (<https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/>). We included it to represent smaller open-weight models and multimodal prompting; it is also distributed under the Llama license.

3.6. Experimental design

To evaluate the effectiveness of our customized GA in automating prompt engineering across various problems and models, we conducted three sets of experiments:

Experiment I: We tested our GA-based prompt-optimization approach using the three smaller language models: GPT-4o-mini, Mistral NeMo, and Llama-3.2-11B. The models were applied to four datasets: Academic Topic Classification, Financial Math Reasoning, Bulgarian Academic Q&A, and ScienceQA. This experiment aimed to demonstrate the effectiveness of our method with both open-source and proprietary models.

Experiment II: We focused our attention on one of the models used in Experiment I: Mistral NeMo, and applied it to another six datasets: TruthfulQA, China Law, MS MARCO, Liar Dataset, Banking77, and Criminality Sentiment Analysis. The objective of this experiment was to illustrate the applicability of our method across a broader range of problems.

Experiment III: Finally, we compared the performance of our approach with the larger models: GPT-4o, Mistral Large 24.11, and Llama-3.3-70B. In this experiment, we tested LLMs on the Academic Topic Classification dataset. The aim was to evaluate how our GA performs under the guidance of large, top-performing models.

In each experiment described above, we recruited experienced prompt engineers to solve the same problems with the same models. These engineers, proficient in utilizing LLMs for diverse computational tasks, had not worked on the specific problems selected for our study before. They were instructed to use the sample data provided for context and problem understanding, mirroring the GA's initialization process. Although they were proficient in English, they were not familiar with Bulgarian or Chinese, which was relevant to two of the datasets (Bulgarian Academic Q&A and China Law). Each engineer solved five problems using the smaller models. Due to budget constraints, we engaged only one engineer to work with the larger models. Engineers were allotted four hours per problem, including computing time.

Finally, to establish an experimental baseline, the models were also tested using prompts containing only the input data, without any additional instructions. Our objective for this “no-prompt” approach was to determine whether the datasets could have been part of the model’s training or fine-tuning (i.e., data leakage, memorization, etc.), which would inherently embed the solutions into the model itself, and thus isolate the impact of prompt engineering.

Our results are described in Section 4. Additionally, we present ablation studies and sensitivity analysis substantiating our design decisions and parameter choices in Appendix A.

4. Results

In this section, we show the results from Experiments I, II, and III, along with our technical observations and the implications of our findings. We start by examining Figure 1, which provides curves resulting from Experiment I across the smaller tested LLMs: GPT-4o-mini, Mistral NeMo, and Llama-3.2-11B. Sub-figures (a)–(c) display the comparative results of our GA, manual prompt engineering, and the no-prompt baseline for each dataset.

Next, Figure 2 presents curves resulting from Experiment II, which again compares our GA-based optimization with manual

Figure 1

Results for Experiment I: plots for our GA, no-prompt, and manual approaches using GPT-4o-mini, Mistral NeMo, and Llama-3.2-11B applied to four datasets. Curves labeled Best, Average, Manual, and No-prompt represent the maximum fitness per generation, the average fitness per generation, the human performance, and the no-prompt performance achieved on each problem, respectively

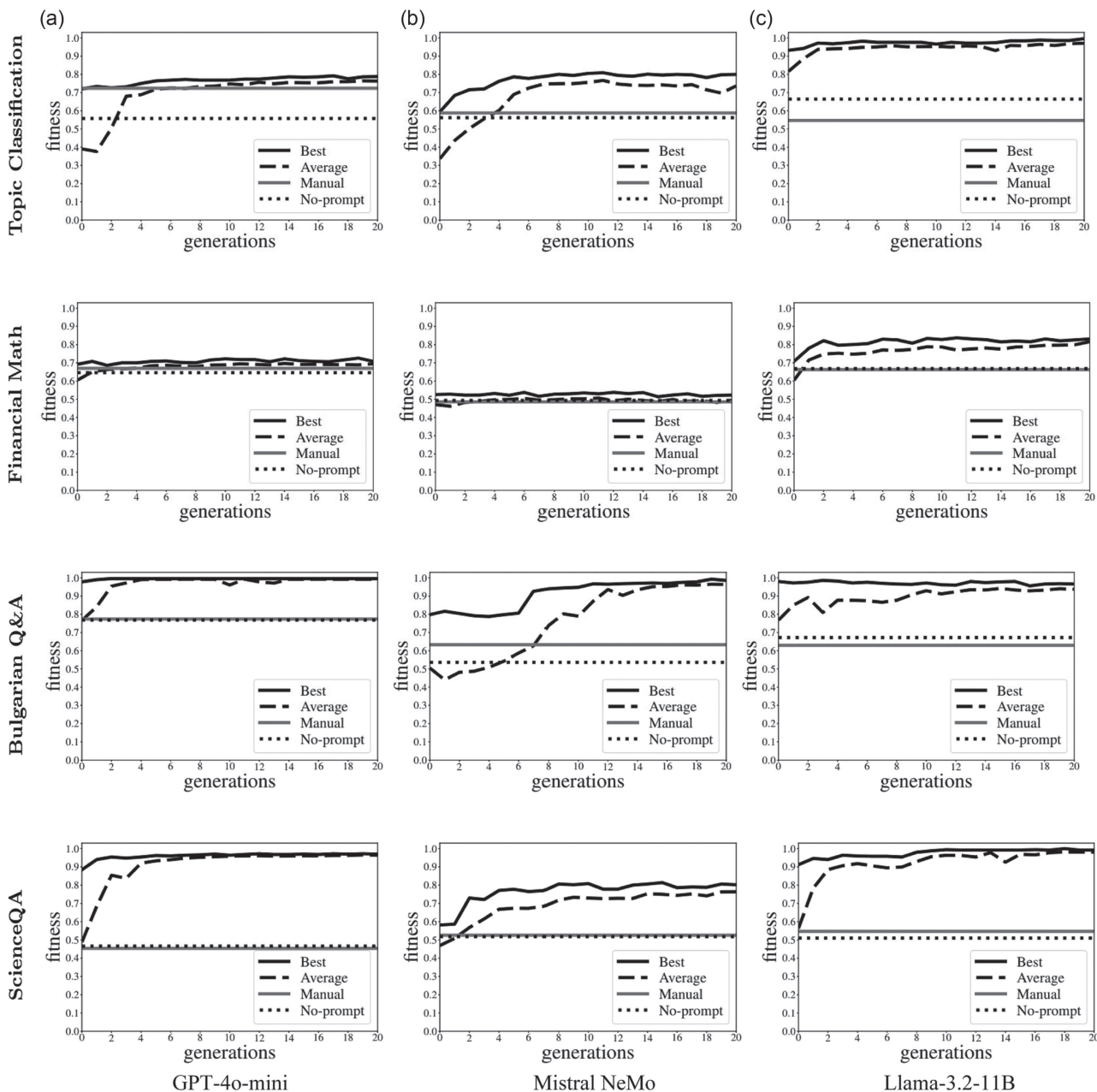
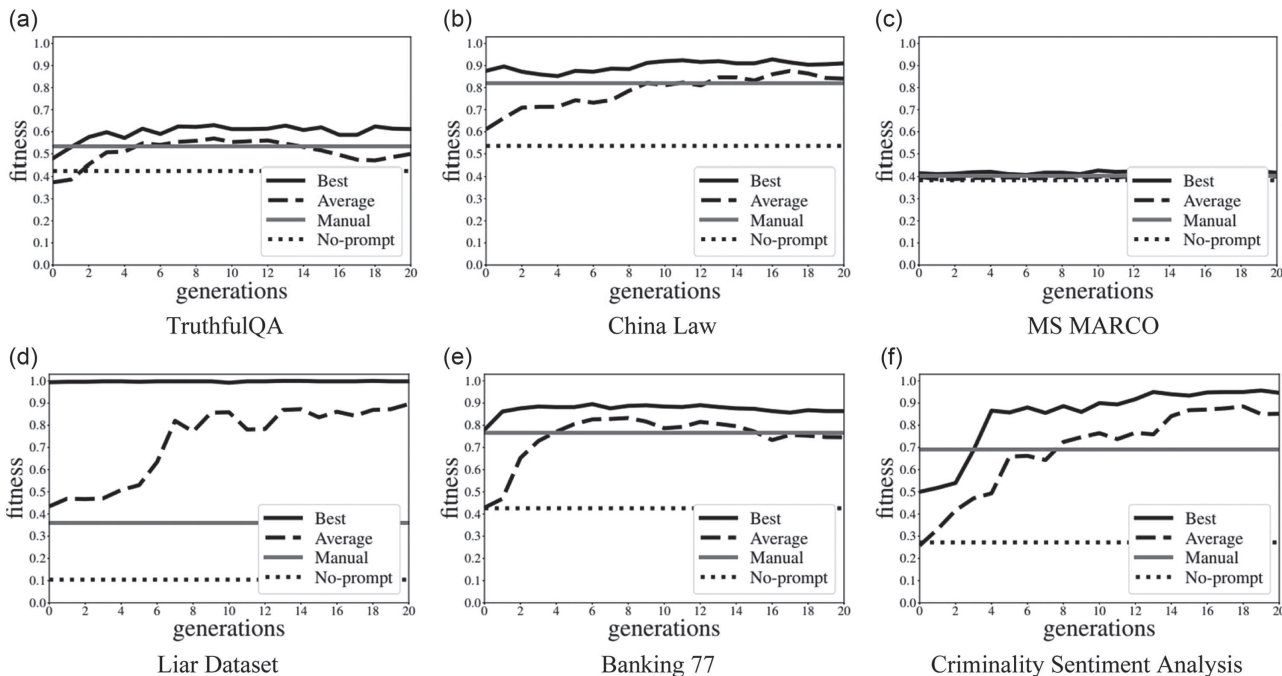


Figure 2

Results for Experiment II: plots for our GA, no-prompt, and manual approaches using Mistral NeMo applied to an additional six datasets. Similar to the previous plot, curves labeled Best, Average, Manual, and No-prompt represent the maximum fitness per generation, the average fitness per generation, the human performance, and the no-prompt performance achieved on each problem, respectively



prompt engineering and the no-prompt baseline, this time focusing solely on Mistral NeMo as the operating LLM. Subfigures (a)–(f) display the comparative results of the approaches on each new dataset.

Next, Figure 3 displays curves resulting from Experiment III, comparing GA-based prompt optimization, manual prompt engineering, and no-prompt baseline runs using the larger LLMs: GPT-4o, Mistral Large 24.11, and Llama-3.3-70B. Results from the corresponding smaller models are repeated here for side-by-side comparisons between model sizes. Results are shown for the Academic Topics Classification dataset only. Curves illustrate the best fitness achieved across each generation by each approach.

Regardless of each model’s specific strengths and weaknesses, which can indeed have a localized impact on fitness, our experimental results clearly demonstrate that our proposed LLM-powered GA generated and evolved effective prompts across all datasets and models. The curves representing the best individuals reveal that all models improved the fitness of initial solutions after processing them through our genetic operators. Additionally, the average curves indicate that the entire population benefited from this evolutionary process.

Quantitative comparisons between our LLM-powered GA and the no-prompt and manual prompt engineering baselines showed superior results across most datasets, as detailed in Table 3. Overall improvements between approaches averaged 31.2% and 22.0%, respectively. While the former indicates how much our method was able to enhance what the LLM could already achieve on its own, the latter highlights the viability of fully automated prompt engineering and aligns closely with findings from Guo et al. (up to 25% improvement) and Chen et al. (22% improvement on a slightly different application). It is important to note that manual prompting excelled on the Financial

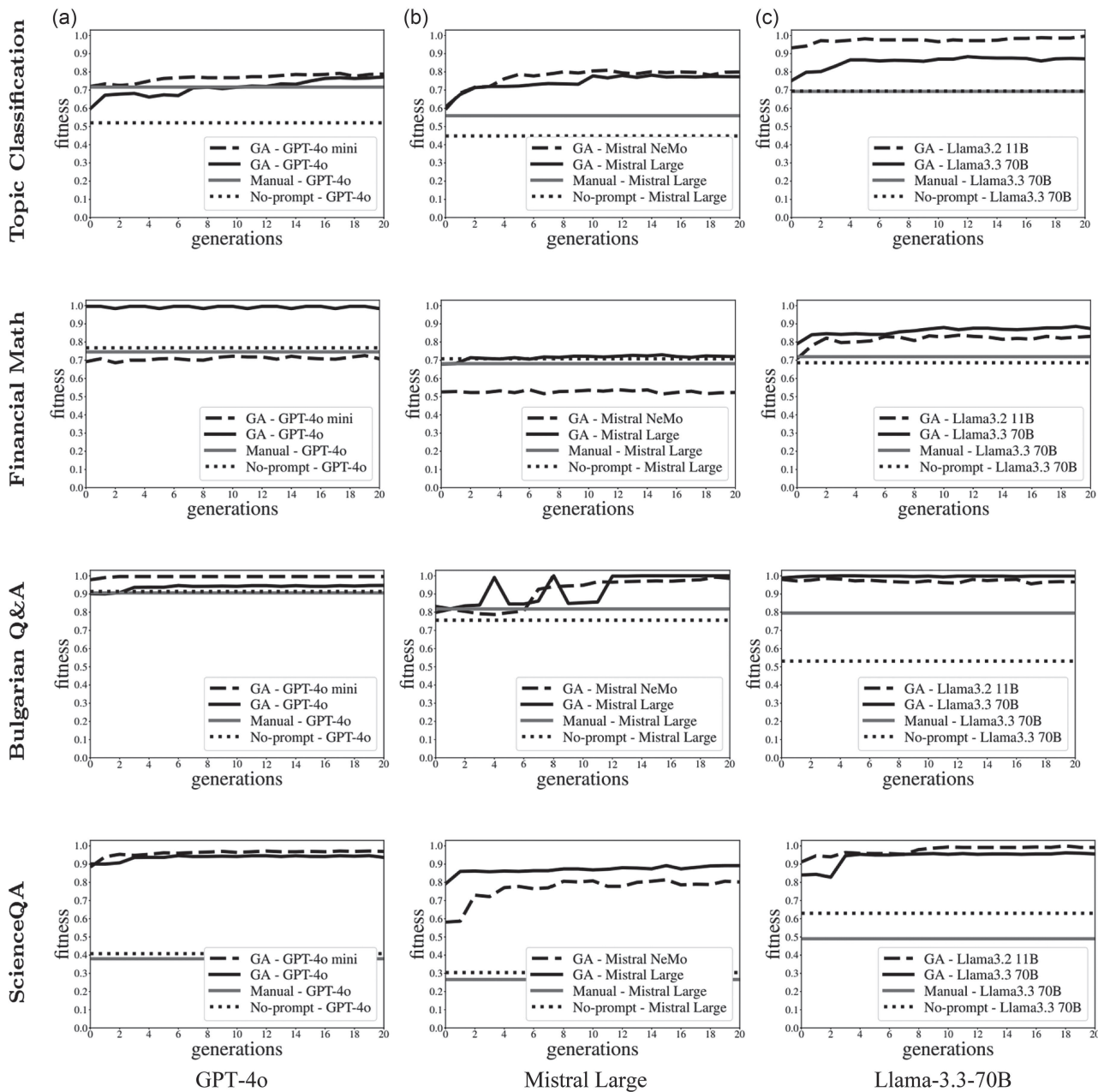
Math Reasoning dataset using one of the strongest models (i.e., Mistral Large). That is likely because the engineer intentionally employed Chain-of-Thought (CoT) [49], a powerful prompting strategy that our automated method did not consistently evolve for this task in particular. In a separate study, we address that specific problem with reasoning-capable models [11].

Qualitatively, we identified a few emergent features. For instance, GA-guided LLMs were capable of generating a variety of prompts using known prompt engineering techniques, such as employing examples and templates, contextual information, role-playing, and step-by-step guidance. While some strategies led to fitness improvements, others did not and were pruned out of the population. Another interesting feature is that our GA allowed LLMs to explore different approaches concurrently. More specifically, in the Academic Topic Classification dataset, we noted that Mistral NeMo addressed the problem from at least three distinct perspectives: topic inference (open categories), topic categorization (predefined categories), and interactive learning experience design (comprehensive descriptions of core topics, takeaways, and learning objectives). Finally, Llama and Mistral models were able to generate prompts entirely in Bulgarian for the Academic Q&A and in Chinese for China Law. These observations showcase behaviors not typically seen among human prompt engineers.

Overall, we observed that larger models, while generating prompts that were essentially similar to those produced by smaller models, generally exhibited slightly higher accuracy and finer detail in their responses. Nevertheless, the smaller models (GPT-4o-mini, Mistral NeMo, and Llama-3.2-11B) achieved superior fitness scores. We attribute this counterintuitive result to the higher misjudgment rates by these models when serving as fitness evaluators. In our experiments, we observed that the LLMs’

Figure 3

Results for Experiment III: plots for our GA, no-prompt, and manual approaches using GPT-4o, Mistral Large, and Llama-3.3-70B applied to the Academic Topic Classification dataset. Curves represent the best fitness per generation, as well as results by manual and no-prompt baselines



performance as judges of output quality varied significantly. While GPT-4o, GPT-4o-mini, Mistral NeMo, and Mistral Large performed well and within a consistent and acceptable margin of error (between 1% and 5% misjudgment rates), Meta’s Llama pair, Llama-3.2-11B and Llama-3.3-70B, exhibited rather inconsistent performance with a higher margin of error. Sample analysis revealed instances where these models produced misjudgment rates anywhere between 1% and 35%, more often overestimating the equality of different answers than vice versa. These results varied across datasets, with categorization tasks generally performing best and free-text Q&A tasks performing worst. Appendix A describes our ablation studies and sensitivity analysis, touching on this topic in a little more depth. Nonetheless, due to the

subjective nature of such judgments in various tasks, users of this method may disagree with the standard matching produced by our fitness meta prompts and may tailor them, with little effort, to perform more aggressively or conservatively according to their needs.

It is important to note, though, that one should not infer that future, more advanced models will not possess better innate prompt engineering and self-judgment capabilities. On the contrary, we anticipate that our evolutionary optimization methodology will greatly benefit from more capable models with inherent strengths in prompt engineering.

Ultimately, our results reveal that automated methods outperformed manual prompt engineering across most models and

Table 3

Summary of results from Experiments I, II, and III, including performance differences and average fitness scores across approaches. Each row shows the highest fitness achieved by each model on each dataset, while the final two columns report the difference between automated and baseline results (NP: No-Prompt and MP: Manual Prompting).

The table demonstrates that our LLM-powered GA is not only competitive overall but also outperforms manual prompting across the great majority of datasets and models

LLM	Dataset	NP baseline	LLMed GA	MP baseline	Diff. (GA-NP)	Diff. (GA-MP)
GPT-4o-mini	Topic Classification	0.558	0.788	0.724	0.230	0.064
	Bulgarian Q&A	0.768	0.996	0.774	0.228	0.222
	Financial Math	0.646	0.708	0.670	0.062	0.038
	ScienceQA	0.466	0.970	0.454	0.504	0.516
Llama-3.2-11B	Topic Classification	0.664	0.996	0.547	0.332	0.449
	Bulgarian Q&A	0.672	0.966	0.630	0.294	0.336
	Financial Math	0.668	0.831	0.663	0.163	0.168
	ScienceQA	0.510	0.992	0.547	0.482	0.445
	Topic Classification	0.562	0.800	0.588	0.238	0.212
	Bulgarian Q&A	0.536	0.996	0.633	0.460	0.363
	Financial Math	0.494	0.523	0.486	0.029	0.037
	ScienceQA	0.518	0.802	0.526	0.284	0.276
Mistral NeMo	TruthfulQA	0.426	0.612	0.534	0.186	0.078
	China Law	0.536	0.910	0.820	0.374	0.090
	MS MARCO	0.384	0.418	0.404	0.034	0.014
	Liar Dataset	0.104	0.998	0.360	0.894	0.638
	Banking77	0.426	0.864	0.766	0.438	0.098
	Criminality Sentiment	0.272	0.946	0.691	0.674	0.255
	Topic Classification	0.520	0.772	0.717	0.252	0.055
GPT-4o	Bulgarian Q&A	0.768	0.944	0.910	0.176	0.034
	Financial Math	0.646	0.971	0.751	0.325	0.220
	ScienceQA	0.466	0.938	0.380	0.472	0.558
	Topic Classification	0.694	0.872	0.692	0.178	0.180
Llama-3.3-70B	Bulgarian Q&A	0.672	0.966	0.809	0.294	0.157
	Financial Math	0.668	0.831	0.723	0.163	0.108
	ScienceQA	0.510	0.992	0.488	0.482	0.504
	Topic Classification	0.448	0.773	0.564	0.325	0.209
Mistral Large	Bulgarian Q&A	0.536	0.996	0.823	0.460	0.173
	Financial Math	0.494	0.523	0.682	0.029	-0.159
	ScienceQA	0.518	0.802	0.526	0.284	0.276
Average	All				0.312	0.220

problem domains. It is worth noting, though, that the engineers did not always utilize the full 4-hour period allocated for each problem. We observed that they typically experimented with a few prompting strategies and quickly anchored on what appeared to be the most promising solution. They then concentrated their remaining time on refining that solution, often neglecting to further explore different, potentially superior, prompting strategies. We believe this behavior may have led them to sub-optimal outcomes in some cases. While this may seem to complicate a direct comparison between methodologies, it is, in fact, an enriching observation that highlights some of the drawbacks of manual prompt engineering. Unsurprisingly, it also emphasizes the fact that prompt engineering demands personal investment, discipline, and often feedback from domain experts to drive gradual improvements toward optimal solutions.

Consequently, automated prompting approaches, like the one proposed in this study, are increasingly appealing for real-world applications.

5. Conclusion

The integration of GAs and LLMs for prompt optimization marks a significant advancement in the practical application of generative AI. Our customized GA method proved to address well some challenges associated with manual prompt engineering and limitations of current automated techniques. By leveraging the evolutionary power of GAs, prompt engineers and LLM practitioners can automatically generate effective prompts, discovering strategies that enhance the efficiency and accuracy of LLMs across a wide range of applications and domains.

Our study highlights the potential for GA-optimized prompts to significantly scale the deployment of LLMs in specific industry contexts, allowing generative models to adapt to specialized problems with minimal human intervention. The findings presented here contribute to the broader fields of generative AI, computational optimization, and evolutionary computing in several important ways. First, they demonstrate the ability of GA-guided LLMs to systematically tackle complex problems traditionally addressed by humans only. Second, they provide substantial evidence that LLMs by different vendors and of different sizes can serve as good judges of output correctness and, consequently, drive optimization via their influence on the fitness function. Third, they show numerous instances where GA-guided LLMs exceed the performance of manual prompt engineering in solving domain-specific tasks, aligning with findings from other independent studies. Finally, by making all code and datasets publicly available, our work facilitates replication and encourages further contributions from the research community.

In future research, we plan to explore advanced and possibly concurrent usage of selection strategies aiming at increasing diversity across populations and, consequently, the longer survival of different, albeit undeveloped, prompting strategies. We also plan to continue improving our LLM-based fitness function and to investigate the broader concept of subjective, nondeterministic fitness assessment by LLMs in different optimization problems. To improve performance and reduce costs, we plan to explore early-stopping and memorization strategies to minimize the computational footprint of individual GA executions. Finally, we plan to investigate methods for explicitly eliciting CoT prompting strategies to enrich prompt generation. Ultimately, fully automated prompt engineering represents a major leap forward in leveraging LLMs across various industries, particularly benefiting teams with limited technical expertise. Recent research suggests that combining LLMs with evolutionary algorithms represents a growing direction, with the potential to redefine how complex search and optimization problems are approached [50]. As LLMs continue to evolve, these strategies promise to unlock new capabilities, driving progress in domain-specific applications and transforming how humans interact with LLMs.

Ethical Statement

This study does not contain any studies with human or animal subjects performed by any of the authors.

Conflicts of Interest

The authors declare that they have no conflicts of interest related to this work.

Data Availability Statement

All data that support the findings of this study are in the public domain and openly available on GitHub at <https://www.github.com/leloss/gallm>.

Author Contribution Statement

Leandro A. Loss: Conceptualization, Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization, Supervision, Project administration. **Pratikumar Dhuvad:** Conceptualization, Conceptualization, Methodology, Software, Validation, Formal analysis,

Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization, Supervision, Project administration.

References

- [1] Yin, S., Fu, C., Zhao, S., Li, K., Sun, X., Xu, T., & Chen, E. (2024). A survey on multimodal large language models. *National Science Review*, 11(12), nwae403. <https://doi.org/10.1093/nsr/nwae403>
- [2] Raza, M., Jahangir, Z., Riaz, M. B., Saeed, M. J., & Sattar, M. A. (2025). Industrial applications of large language models. *Scientific Reports*, 15(1), 13755. <https://doi.org/10.1038/s41598-025-98483-1>
- [3] Anagnostidis, S., & Bulian, J. (2024). *How susceptible are LLMs to influence in prompts?* arXiv. <https://doi.org/10.48550/arXiv.2408.11865>
- [4] Shin, T., Razeghi, Y., Iv, R. L. L., Wallace, E., & Singh, S. (2020). AutoPrompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, 4222–4235. <https://doi.org/10.18653/v1/2020.emnlp-main.346>
- [5] Shi, W., Han, X., Gonen, H., Holtzman, A., Tsvetkov, Y., & Zettlemoyer, L. (2023). Toward human readable prompt tuning: Kubrick’s *The Shining* is a good movie, and a good prompt too? In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 10994–11005. <https://doi.org/10.18653/v1/2023.findings-emnlp.733>
- [6] Deng, M., Wang, J., Hsieh, C.-P., Wang, Y., Guo, H., Shu, T., . . . , & Hu, Z. (2022). RLPrompt: Optimizing discrete text prompts with reinforcement learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 3369–3391. <https://doi.org/10.18653/v1/2022.emnlp-main.222>
- [7] Loya, M., Sinha, D., & Futrell, R. (2023). Exploring the sensitivity of LLMs’ decision-making capabilities: Insights from prompt variations and hyperparameters. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 3711–3716. <https://doi.org/10.18653/v1/2023.findings-emnlp.241>
- [8] Miah, M. S. U., Kabir, M. M., Sarwar, T. B., Safran, M., Alfarhood, S., & Mridha, M. F. (2024). A multimodal approach to cross-lingual sentiment analysis with ensemble of transformer and LLM. *Scientific Reports*, 14(1), 9603. <https://doi.org/10.1038/s41598-024-60210-7>
- [9] Sclar, M., Choi, Y., Tsvetkov, Y., & Suhr, A. (2024). Quantifying language models’ sensitivity to spurious features in prompt design or: How I learned to start worrying about prompt formatting. In *International Conference on Learning Representations 2024*, 1–29.
- [10] Loss, L. A., & Dhuvad, P. (2025). An LLM-Based genetic algorithm for prompt engineering. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 527–530. <https://doi.org/10.1145/3712255.3726633>
- [11] Loss, L. A., & Dhuvad, P. (2025). Reasoning-guided evolutionary prompt optimization for improved financial problem solving. In *NeurIPS 2025 Workshop: Generative AI in Finance*, 1–13.
- [12] Holland, J. H. (1975). *Adaptation in natural and artificial systems* (7th ed.). USA: University of Michigan Press.
- [13] Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. UK: Addison-Wesley.

- [14] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . , & Polosukhin, I. (2017). Attention is all you need. In *31st Conference on Neural Information Processing Systems*, 6000–6010.
- [15] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Long and Short Papers)*, 1, 4171–4186. <https://doi.org/10.18653/v1/N19-1423>
- [16] Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., . . . , & Sifre, L. (2022). Training compute-optimal large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, 30016–30030.
- [17] OpenAI, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., . . . , & Zoph, B. (2023). *GPT-4 technical report*. arXiv. <https://doi.org/10.48550/ARXIV.2303.08774>
- [18] Li, Y., Wang, S., Ding, H., & Chen, H. (2023). Large language models in finance: A survey. In *Proceedings of the Fourth ACM International Conference on AI in Finance*, 374–382. <https://doi.org/10.1145/3604237.3626869>
- [19] Naveed, H., Khan, A. U., Qiu, S., Saqib, M., Anwar, S., Usman, M., . . . , & Mian, A. (2025). A comprehensive overview of large language models. *ACM Transactions on Intelligent Systems and Technology*, 16(5), 106. <https://doi.org/10.1145/3744746>
- [20] Liu, J., Shen, D., Zhang, Y., Dolan, W. B., Carin, L., & Chen, W. (2022). What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out: The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, 100–114. <https://doi.org/10.18653/v1/2022.deelio-1.10>
- [21] Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., & Neubig, G. (2023). Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9), 195. <https://doi.org/10.1145/3560815>
- [22] Zhang, T., Wang, X., Zhou, D., Schuurmans, D., & Gonzalez, J. E. (2023). *TEMPERA: Test-time prompt editing via reinforcement learning*. arXiv. <https://doi.org/10.48550/arXiv.2211.11890>
- [23] Pryzant, R., Iter, D., Li, J., Lee, Y., Zhu, C., & Zeng, M. (2023). Automatic prompt optimization with “gradient descent” and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 7957–7968. <https://doi.org/10.18653/v1/2023.emnlp-main.494>
- [24] Guo, Y., Liang, Y., Wu, C., Wu, W., Zhao, D., & Duan, N. (2024). Learning to plan by natural language. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, 10062–10098. <https://doi.org/10.48550/arXiv.2304.10464>
- [25] Li, H., & Shi, N. (2022). Application of genetic optimization algorithm in financial portfolio problem. *Computational Intelligence and Neuroscience*, 2022(1), 5246309. <https://doi.org/10.1155/2022/5246309>
- [26] Lehman, J., Gordon, J., Jain, S., Ndousse, K., Yeh, C., & Stanley, K. O. (2024). Evolution through large models. In W. Banzhaf, P. Machado, & M. Zhang (Eds.), *Handbook of evolutionary machine learning* (pp. 331–366). Springer. https://doi.org/10.1007/978-981-99-3814-8_11
- [27] Meyerson, E., Nelson, M. J., Bradley, H., Gaier, A., Moradi, A., Hoover, A. K., & Lehman, J. (2024). Language model crossover: Variation through few-shot prompting. *ACM Transactions on Evolutionary Learning*, 4(4), 27. <https://doi.org/10.1145/3694791>
- [28] Tanaka, H., Mori, N., & Okada, M. (2023). Genetic algorithm for prompt engineering with novel genetic operators. In *2023 15th International Congress on Advanced Applied Informatics Winter*, 209–214. <https://doi.org/10.1109/IIAI-AAI-Winter61682.2023.00047>
- [29] Chen, A., Dohan, D. M., & So, D. R. (2023). EvoPrompting: Language models for code-level neural architecture search. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, 7787–7817.
- [30] Guo, Q., Wang, R., Guo, J., Li, B., Song, K., Tan, X., . . . , & Yang, Y. (2024). Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. In *The Twelfth International Conference on Learning Representations*, 1–24.
- [31] Hsieh, C.-J., Si, S., Yu, F., & Dhillon, I. (2024). Automatic engineering of long prompts. In *Findings of the Association for Computational Linguistics: ACL 2024*, 10672–10685. <https://doi.org/10.18653/v1/2024.findings-acl.634>
- [32] Poza, J., Moreno, V., Fraga, A., & Álvarez-Rodríguez, J. M. (2021). Genetic algorithms: A practical approach to generate textual patterns for requirements authoring. *Applied Sciences*, 11(23), 11378. <https://doi.org/10.3390/app112311378>
- [33] Chouayfati, P., Herbster, N., Sáfrán, Á. D., & Grabmair, M. (2025). GenDLN: Evolutionary algorithm-based stacked LLM framework for joint prompt optimization. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, 4, 1171–1212. *Student Research Workshop*. <https://doi.org/10.18653/v1/2025.acl-srw.92>
- [34] Liu, S., Chen, C., Qu, X., Tang, K., & Ong, Y.-S. (2024). Large language models as evolutionary optimizers. In *2024 IEEE Congress on Evolutionary Computation*, 1–8. <https://doi.org/10.1109/CEC60901.2024.10611913>
- [35] Yang, C., Wang, X., Lu, Y., Liu, H., Le, Q. V., Zhou, D., & Chen, X. (2024). Large language models as optimizers. In *International Conference on Learning Representations*, 1–41.
- [36] Ye, Q., Ahmed, M., Pryzant, R., & Khani, F. (2024). Prompt engineering a prompt engineer. In *Findings of the Association for Computational Linguistics: ACL 2024*, 355–385. <https://doi.org/10.18653/v1/2024.findings-acl.21>
- [37] Loss, L. A., & Dhuvad, P. (2025). From manual to automated prompt engineering: Evolving LLM prompts with genetic algorithms. In *2025 IEEE Congress on Evolutionary Computation*, 1–8. <https://doi.org/10.1109/CEC65147.2025.11043059>
- [38] Wu, C., Luo, W., Zhou, N., Xu, P., & Zhu, T. (2021). Genetic algorithm with multiple fitness functions for generating adversarial examples. In *2021 IEEE Congress on Evolutionary Computation*, 1792–1799. <https://doi.org/10.1109/CEC45853.2021.9504790>
- [39] Sukstrienwong, A. (2023). ANOVA as fitness function for genetic algorithm in group composition. *TEM Journal*, 12(1), 396–405.
- [40] Sam, R., & H, H. I. (2024). Performance analysis of fitness function in genetic algorithm for fetal brain MRI classification. In *2024 International Conference on Advancements in*

- Power, *Communication and Intelligent Systems*, 1–6. <https://doi.org/10.1109/APCI61480.2024.10616722>
- [41] Cheng, D., Huang, S., & Wei, F. (2024). Adapting large language models via reading comprehension. In *International Conference on Learning Representations 2024*, 1–29.
- [42] Hardalov, M., Koychev, I., & Nakov, P. (2019). Beyond English-only reading comprehension: Experiments in zero-shot multilingual transfer for Bulgarian. In *Proceedings of Recent Advances in Natural Language Processing*, 447–459. https://doi.org/10.26615/978-954-452-056-4_053
- [43] Saikh, T., Ghosal, T., Mittal, A., Ekbal, A., & Bhattacharyya, P. (2022). *ScienceQA*: A novel resource for question answering on scholarly articles. *International Journal on Digital Libraries*, 23(3), 289–301. <https://doi.org/10.1007/s00799-022-00329-y>
- [44] Lin, S., Hilton, J., & Evans, O. (2022). TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 3214–3252. <https://doi.org/10.18653/v1/2022.acl-long.229>
- [45] Yuan, L., Wang, J., Fan, S., Bian, Y., Yang, B., Wang, Y., & Wang, X. (2019). Automatic legal judgment prediction via large amounts of criminal cases. In *2019 IEEE 5th International Conference on Computer and Communications*, 2087–2091. <https://doi.org/10.1109/ICCC47050.2019.9064408>
- [46] Nguyen, T., Rosenberg, M., Song, X., Gao, J., Tiwary, S., Majumder, R., & Deng, L. (2016). MS MARCO: A human generated machine reading comprehension dataset. In *5th International Conference on Learning Representations*, 1–10.
- [47] Wang, W. Y. (2017). “liar, liar pants on fire”: A new benchmark dataset for fake news detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 2, 422–426. <https://doi.org/10.18653/v1/P17-2067>
- [48] Casanueva, I., Temčinas, T., Gerz, D., Henderson, M., & Vulić, I. (2020). Efficient intent detection with dual sentence encoders. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, 38–45. <https://doi.org/10.18653/v1/2020.nlp4convai-1.5>
- [49] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E. H., Zhou, D., . . . Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, 24824–24837.
- [50] Guo, C., Wang, S., & Jiao, R. (2026). From evolution to generation: Leveraging LLMs to redefine genetic programming for symbolic regression. In *PRICAI 2025: Trends in Artificial Intelligence: 22nd Pacific Rim International Conference on Artificial Intelligence*, 83–99. https://doi.org/10.1007/978-981-95-7072-0_6

How to Cite: Loss, L. A., & Dhuvad, P. (2026). Toward Fully Automated Prompting: An LLM-Powered Genetic Algorithm for Prompt Generation, Optimization, and Self-Evaluation. *Journal of Data Science and Intelligent Systems*. <https://doi.org/10.47852/bonviewJDSIS62028440>

Appendix A. Sensitivity and Ablation Studies

This appendix reports sensitivity and ablation studies designed to characterize the robustness of our method and its sensitivity to key design choices. These analyses improve transparency around stochastic variability and practical reproducibility, but they are not intended to constitute a full formal significance-testing framework.

Zero generations: We used GPT-4o-mini and Mistral Large to generate 2000 prompts for the Topic Classification dataset. This is approximately five times the total number of prompts produced by a single run of our LLM-powered GA with 20 generations and a population size of 20. The goal of this study was twofold: (1) to ablate GA-driven evolution and (2) to test whether our GA’s performance could be explained simply by generating a large number of prompts. The results are reported below:

LLM	Dataset	Reported LLMed GA	Zero gen.
GPT-4o-mini	Topic Classification	0.788	0.607
Mistral Large	Topic Classification	0.773	0.466

These results reveal that random prompt generation does not reliably achieve performance comparable to our approach. Multiple runs of this experiment with different models exhibited the same pattern. This experiment can be replicated using our shared code by setting the number of generations to 0 and the population size to 2,000.

Zero mutation: We analyzed the impact of removing the mutation operator by setting the mutation probability to 0 in our shared code. We used GPT-4o and GPT-4o-mini on the Topic Classification dataset. The results were as follows:

LLM	Dataset	Reported LLMed GA	Zero mutation
GPT-4o-mini	Topic Classification	0.788	0.732
GPT-4o	Topic Classification	0.772	0.741

These results reveal a small but meaningful reduction in the best-individual fitness. In other experiments and datasets, the effect varied but consistently indicated that removing mutation increases the GA’s sensitivity to the initial population. This experiment can be replicated by setting the mutation probability to 0 in our shared code.

Multiple reruns: We analyzed variability in the best individual across repeated runs. To this end, we tracked three of our models on three distinct datasets while performing 10 independent runs of our LLM-powered GA. This study is intended to characterize stochasticity and robustness, not to replace a formal significance-testing framework. Here, we fixed the random seed (thus keeping the in-context and validation samples fixed) while allowing the LLM to generate different prompts as usual. The results were as follows:

LLM	Dataset	Reported LLMed GA	Multiple runs
GPT-4o-mini	Topic Classification	0.788	0.787 ± 0.038
GPT-4o-mini	Financial Math	0.708	0.707 ± 0.047
GPT-4o	Topic Classification	0.772	0.772 ± 0.031
Mistral Large	ScienceQA	0.802	0.804 ± 0.044

These results reveal an average variation of 4% across the best individuals produced by the tested models under the same initial conditions. In separate design experiments, we observed similar variability for larger models, while smaller models sometimes varied more widely. For example, our worst performer in that regard, Llama-3.2-11B, exhibited up to a 14% difference between runs (based on a 5-run experiment). This experiment can be replicated by re-running our shared code multiple times without modifying the seed or other input parameters.

Generations vs. population: We analyzed the impact of the number of generations and the population size using GPT-4o-mini on the Topic Classification dataset. We varied the population size in {10, 20, 30, 40} and the number of generations in {10, 20, 30, 40}. The results were as follows:

These results reveal the expected GA behavior: larger populations and more generations increase the number of evaluated prompts, improving the probability of finding better solutions. They also suggest diminishing returns at larger sizes, with performance approaching saturation. This experiment can be replicated by modifying these input parameters in our shared code.

Cost in fitness: We investigated the impact of incorporating generation cost (input + output) into our fitness formulation. We used GPT-4o-mini and Mistral Large on the Topic Classification dataset. At the time of experimentation, pricing was \$0.15/million input

Gen. Pop.	10	20	30	40
10	0.606	0.708	0.744	0.778
20	0.652	0.788	0.801	0.808
30	0.654	0.789	0.801	0.804
40	0.696	0.788	0.804	0.804

tokens and \$0.60/million output tokens. Using the fitness function, $\text{Fitness} = (\alpha \times \text{matching} - \times \text{cost})$, we set $\alpha = 0.7$ and $\beta = 0.3$. The results were as follows:

LLM	Dataset	Reported LLMed GA	Cost-constrained fitness
GPT-4o-mini	Topic Classification	0.788	0.780
Mistral Large	Topic Classification	0.773	0.775

These results reveal that penalizing fitness by cost did not substantially change the best-individual fitness. Although we expected shorter prompts and/or shorter elicited responses, we did not observe a consistent relationship. Quantitatively:

LLM	Dataset	Avg. LLMed GA cost	Avg. cost- constrained fitness
GPT-4o-mini	Topic Classification	\$0.000058 \pm \$0.000010	\$0.000057 \pm \$0.000009
Mistral Large	Topic Classification	\$0.000773 \pm \$0.000013	\$0.000773 \pm \$0.000013

where “avg. cost” is the average cost of running the best individual across all 500 evaluation samples. Overall, these experiments do not support the claim that adding cost (or length-related terms) to the fitness function reliably induces a strong cost/length reduction without affecting quality. This experiment can be replicated by modifying the coefficients in our shared code.

Evaluation set: To account for varying dataset sizes, we analyzed sensitivity to evaluation-set size. We used Mistral NeMo on the China Law and MS MARCO datasets (hundreds of thousands to millions of samples). We fixed the number of in-context samples and varied the evaluation-set size in {100, 500, 1000, 5000}. The results were as follows:

Eval. Dataset	100	500	1000	5000
China Law	0.964	0.910	0.906	0.901
MS MARCO	0.580	0.418	0.420	0.416

These results reveal no substantial impact on achieved fitness. This may be because, despite their size, these datasets are relatively homogeneous, allowing the fixed in-context samples to capture sufficient characteristics for effective generalization. This experiment can be replicated by changing the evaluation-set size parameter in our shared code.

LLM parameters: During development, we tested various intrinsic and extrinsic parameters. The most relevant results are reported above; many other configurations were either less consequential or too numerous to include here. In particular, we examined LLM parameters such as temperature, top-p, and top-k. We observed that the magnitude (and sometimes the direction) of their effects varied by model, including among models from the same vendor (e.g., GPT-4o vs. GPT-4o-mini), making it difficult to isolate consistent correlations with best-individual quality or other GA intermediates. Moreover, some of these parameters are being deemphasized or deprecated in newer model APIs.

Accordingly, we focused less on LLM sampling parameters and used temperature = 1.0, top-p = 0.95, and max tokens = 4096. All other parameters were left at each model/vendor default. Note that LLM parameters are not exposed as input parameters in our shared code; however, readers can supply them directly through the model API to replicate these experiments.

Optimization by self-refinement: Alternative approaches propose that prompt optimization can be achieved by having a single prompt iteratively refine itself in a closed loop toward optimality. Our intuition suggests that relying on a single initial prompt (however strong a random initialization may be) to capture all the ingredients necessary to solve a complex problem, while introducing new elements at random, is unlikely to outperform an optimization strategy in which a pool of prompt candidates exchanges “tested” information and progressively and robustly captures more information about the problem. To test this hypothesis, we evaluated self-refinement using one of the most popular publicly available prompt-optimization frameworks: DSPy (<https://dsp.ai>). The results were as follows:

LLM	Dataset	LLMed GA	DSPy self-refine
	Topic Classification	0.788	0.140
GPT-4o	Bulgarian Q&A	0.996	0.136
Mini	Financial Math	0.708	0.624
	ScienceQA	0.970	0.334
	Topic Classification	0.772	0.654
GPT-4o	Bulgarian Q&A	0.944	0.956
	Financial Math	0.971	0.772
	ScienceQA	0.938	0.462
	Topic Classification	0.773	0.400
Mistral	Bulgarian Q&A	0.996	0.188
Large	Financial Math	0.523	0.110
	ScienceQA	0.802	0.092

These results reveal underwhelming performance from the self-refinement approach on most of the tested datasets. Some datasets were evaluated multiple times to rule out “unlucky” initializations. Even so, in our tests, self-refinement could not consistently produce optimal solutions.

LLM-as-a-judge: We investigated a possible concern that using an LLM in the fitness function may bias evaluation toward that model’s own style, despite the LLM not being asked to judge its own prior reasoning or generation process, as explained in Section 3.3. To test whether this setup nevertheless introduced self-preference, we compared runs in which the same LLM family was used both as task and fitness evaluator against runs in which a different LLM family served as fitness evaluator. We performed this analysis using GPT-4o-mini and Mistral Large on Topic Classification, Bulgarian Q&A, Financial Math, and ScienceQA. The results were as follows:

Task	Dataset	Fitness	
		GPT-4o-mini	Mistral Large
LLM	Topic Classification	0.788	0.773
GPT-4o	Bulgarian Q&A	0.996	0.996
mini	Financial Math	0.708	0.747
	ScienceQA	0.970	0.961
	Topic Classification	0.773	0.799
Mistral	Bulgarian Q&A	0.996	0.996
Large	Financial Math	0.523	0.608
	ScienceQA	0.802	0.861

These results did not reveal a consistent pattern, suggesting that a model was systematically more favorable to its own outputs than to outputs produced by the other model.

Deceptive prompts: We investigated whether LLMs could produce prompts that exploit the fitness function rather than genuinely improve task performance. Intuitively, because of the restricted roles assigned to each LLM in our algorithmic setup, this risk is strongly constrained: (1) the prompting LLM sees only a single input and produces an output; it has no access to the optimization state, downstream evaluation, or even the existence of an evaluator; and (2) the fitness LLM is invoked in a separate stateless call that sees only the prediction and the ground truth and returns a yes/no match. Because these calls are isolated and unaware of each other, collusion to game the fitness function is operationally impossible.

We also manually reviewed all the best prompts from the multiple rerun experiments above, as well as large samples of intermediate prompt candidates across different generations. We did not observe any prompt suggesting an attempt to manipulate the fitness evaluator or otherwise exploit the pipeline. While broader deployment settings may still require guardrails, we found no evidence that deceptive prompting contributed to the gains reported here.

Appendix B. Examples of Generated Prompts

This appendix is reserved for examples of prompts generated by our LLM-powered GA. Tables B1, B2, and B3 present examples of prompts autonomously generated by our approach at different stages of optimization: during initialization and by various genetic operations.

Table B1

Initialization examples. Prompts generated by the LLM-powered GA during initialization contrasted with the best prompt produced by the final generation. Initial Xa, Xb, and Xc are prompts produced during initialization, while Best X is the prompt with the highest fitness after n generations

Example	Prompt
Example 1.	GPT-4o-mini (Topic Classification)
Initial 1a.	You are provided with a series of inputs that represent various computational problems or questions across different domains. Your task is to classify each input by identifying the core problem it represents and assigning it to the appropriate category. Respond with the category name that corresponds to the problem without elaboration. [Examples]
Initial 1b.	You are tasked with analyzing various computational and mathematical problems. For each input provided, identify the underlying concept or domain of the problem and provide a succinct response that categorizes it. The expected output should be a single keyword or phrase that encapsulates the core topic. Please follow this format: INPUT: [Insert the problem statement here] OUTPUT: [Provide the relevant keyword or phrase that represents the concept]
Initial 1c.	You are tasked with generating the output type that categorizes the given input based on the underlying concepts or domains related to computer science, mathematics, or logic. Use the provided input examples to determine the most appropriate category based on the primary focus of the question. Here is the structure of your output: - INPUT: [the user's input] - OUTPUT: [the appropriate category based on the understanding of the input]. Ensure that your output directly reflects the essence of the question being asked and the fundamental principles involved. Aim for a concise yet accurate categorization that encompasses the main idea conveyed in the input.
Best 1.	You are a specialized expert in mathematics, computer science, and logic. Your task is to analyze user queries, identify the primary concepts or categories, and provide clear and relevant explanations in a structured format. Please follow these steps: 1. Identify the main concept or category related to the user's question. 2. Provide a concise explanation of the foundational principles associated with that concept. 3. Utilize precise, domain-specific terminology to enhance clarity and accuracy. Format your response as follows: - INPUT: [the user's input] - OUTPUT: [the identified category or concept, along with a succinct explanation of its core principles] Categorization Guidelines: - For mathematical inquiries, focus on concepts such as "Calculus," "Algebra," "Probability," or "Statistics." - For computer science questions, clarify topics like "Data Structures," "Algorithms," "Complexity Analysis," or "Database Normalization." - For logical reasoning, specify concepts such as "Boolean Logic," "Deductive Reasoning," or "Inductive Reasoning." - For programming-related inquiries, address subjects like "Recursion," "Data Types," "Control Structures," or "Object-Oriented Programming." Examples: 1. INPUT: "What is the time complexity of merging two sorted arrays?" OUTPUT: "O(n + m), where n and m are the lengths of the two arrays. Category: Time Complexity" 2. INPUT: "How do you find the height of a binary tree?" OUTPUT: "The height of a binary tree is the number of edges on the longest path from the root to a leaf node, typically calculated recursively. Category: Binary Tree" 3. INPUT: "Explain dynamic programming." OUTPUT: "Dynamic programming is a method for solving complex problems by breaking them down into simpler overlapping subproblems and storing their solutions to optimize efficiency. Category: Algorithm Design"
Example 2.	Llama-3.2-11B (ScienceQA)
Initial 2a.	Given a passage and a question, extract the relevant information from the passage to answer the question. The passage may contain technical terms, complex sentences, and abstract concepts. The question may require understanding of the context, identifying key phrases, or making inferences. Provide a concise and accurate answer based on the information presented in the passage. [Examples]
Initial 2b.	Given a passage and a question related to the passage, extract the relevant information from the passage and provide a concise answer to the question. The question may require understanding of specific concepts, identifying relationships between entities, or recognizing patterns in the text. Use the context provided in the passage to inform your response. This prompt instructs the model to: 1. Read and understand the passage. 2. Identify the relevant information related to the question. 3. Extract the necessary details to answer the question. 4. Provide a concise and accurate response based on the passage.

(Continued)

Table B1
(Continued)

Example	Prompt
Initial 2c.	Given a passage and a question about the passage, generate an answer to the question based on the information provided. Passage: [Insert passage here] Question: [Insert question here] Answer: [Generate answer based on passage]
Best 2.	Given a passage and a corresponding question, identify key concepts, entities, and relationships in the text, including specialized vocabulary, intricate concepts, and subtle notions. To answer the question, insert the passage and question into the format: "Given a passage, answer the following question: [insert question here]. The passage is: [insert passage here]." Provide a concise response that directly answers the question, using specific key phrases, sentences, or paragraphs from the passage, making inferences or drawing logical conclusions as necessary, and ensure the answer is grounded in the provided text, avoiding speculation or external knowledge.
Example 3.	Mistral NeMo (Financial Reasoning)
Initial 3a.	<p>Given a table of financial data with columns for "Years Ended December 31," "Revenue," "Operating Income," and "Operating Margin," calculate the change in "Operating Margin" between two consecutive years and express it as a percentage of the value in the first year. Additionally, calculate the sum of a specific value over a period of two years and provide the result. Example table: Years Ended December 31 Revenue Operating Income Operating Margin </p> <p> - - - - 2014 4264 485 0.114 2013 4057 318 0.078 </p> <p>The change in "Operating Margin" between 2014 and 2013 is 3.6%, and the sum of the "Operating Margin" in 2014 and 2013 is 0.192. Given another table:</p> <p> Years Ended December 31 </p> <p> </p> <p> 2015 </p> <p> 2014 </p> <p> 2013 </p> <p>Calculate the change in "Operating Margin" between 2015 and 2014 and express it as a percentage of the value in 2015. Additionally, calculate the sum of a specific value over the years 2015 and 2014 and provide the result.</p>
Initial 3b.	<p>**Instructions:**</p> <p>You are an AI that helps people solve problems by extracting and calculating relevant information from given texts. You will receive a text that contains financial or numerical data, and you will be asked to perform specific calculations or extractions based on that data.</p> <p>**Input:**</p> <p>You will be provided with a text that includes various numerical data points, tables, or financial information. The input will also include specific questions or tasks that need to be addressed based on the provided text.</p> <p>**Output:**</p> <p>Your response should be a clear and concise answer to the question or task, based on the information provided in the text. The output should be a numerical value or a set of numerical values, formatted as instructed. Do not include any additional commentary or explanations.</p> <p>**Examples:**</p> <p>[Examples]</p>
Initial 3c.	<p>You are an AI that helps people solve problems by extracting relevant information from a given text and performing calculations or comparisons based on that information. Below is a text that contains numerical or textual data. Your task is to:</p> <ol style="list-style-type: none"> 1. Identify and extract the relevant information needed to answer the query. 2. Perform any necessary calculations or comparisons using the extracted information. 3. Format the output as a single numerical or textual value, depending on the query. <p>Text: {input_text}</p> <p>Query: {query}</p> <p>Output: {output}</p>

(Continued)

Table B1
(Continued)

Example	Prompt
Best 3.	<p>**Instructions:** Given a text input containing numerical data and a specific query, extract the relevant numerical values and perform the necessary calculations to produce a numerical output. Provide the output in a structured format without any additional comments. **Input Format:** Text: [input text containing numerical data] Query: [specific query or calculation to be performed] **Output Format:** Output: [calculated numerical value or values separated by commas] **Steps:** 1. Identify and Extract the Relevant Numerical Values: - Read the input text and query carefully to identify and extract the numerical values relevant to the query. 2. Perform the Required Calculations: - Use the extracted numerical values to perform the necessary calculations as specified in the query. **Examples:** [Examples]</p>
Example 4.	Mistral Large
Initial 4a.	<p>You are an AI that helps people solve problems by extracting the correct answer from a given question. Given the input question, identify the correct answer from the provided options. The input will be a question in Bulgarian, followed by several options. Your task is to extract and return only the correct answer. Input Format: INPUT: <Question in Bulgarian> OUTPUT: <Correct Answer> Example: INPUT: АТФ се синтезира в: ендоплазмената мрежа, рибозомите, хлоропластите, апарата на Голджи? OUTPUT: хлоропластите Rules: 1. Read the input question carefully. 2. Identify the correct answer from the given options. 3. Return only the correct answer without any additional comments or explanations. Begin: INPUT:</p>
Initial 4b.	<p>Task: Multiple-Choice Question Answering Input: A question or statement followed by a list of options. Output: The single correct answer from the list of options. Instructions: 1. Read the question or statement carefully. 2. Evaluate each option based on the information provided in the question or statement. 3. Select the one option that best answers the question or completes the statement. Examples: INPUT: АТФ се синтезира в: ендоплазмената мрежа, рибозомите, хлоропластите, апарата на Голджи? OUTPUT: хлоропластите INPUT: Кой договор е подготвен от Парижката мирна конференция от 1919 г.: Рапалски, Версайски, Локарнски? OUTPUT: Версайски [Additional examples follow the same format] Now, provide the input for the next question: INPUT: [Next question or statement with options] OUTPUT:</p>
Initial 4c.	<p>You are given a question or problem followed by a list of options. Your task is to identify and return the correct or most relevant answer from the given options. Input: [Question or problem followed by a list of options] Output: [The correct or most relevant answer from the given options] Examples: [Examples]</p>

(Continued)

Table B1
(Continued)

Example	Prompt
Best 4.	<p>You are an AI that helps people solve problems. Given a multiple-choice question in Bulgarian that includes a question and several options, your task is to identify and provide the correct answer(s) from the given options. Format your response as follows: Format: QUESTION: (the multiple-choice question in Bulgarian) OPTIONS: - A) (option A) - B) (option B) - C) (option C) - D) (option D)</p> <p>- (additional options if available) ANSWER: (the correct answer(s) from the options, separated by a comma if there are more than one, include the option letter and the actual answer) Example: QUESTION: Какво е главният град на България?? OPTIONS: A) Пловдив B) Варна C) София D) Бургас ANSWER: C) София</p>

Table B2
Mutation examples. Prompts generated by the LLM-powered GA during the mutation operation. Highlighted lines indicate the edits introduced by mutation, while the remaining text aligns with the unchanged lines from the original input

#	Input	Output
1.	<p>Given the passage: [Insert passage text] Answer the question below by identifying and extracting the specific phrase, term, or concept from the passage that directly addresses the context and intent of the following question: [Insert question] Output: [Provide only the most relevant phrase, term, or concept from the passage.]</p>	<p>Given the passage: [Insert passage text] Identify and extract the specific phrase, term, or concept from the passage that directly addresses the context and intent of the following question: [Insert question] Output: [Provide only the most relevant phrase, term, or concept from the passage.]</p>
2.	<p>Analyze the problem statement to determine its core concept and select the most suitable category from the list of topics in computer science, mathematics, and logic: [List of Topics]</p>	<p>Evaluate the given problem statement to identify its core concept. Based on your analysis, select the most suitable category from the list of topics in computer science, mathematics, and logic: [List of Topics]</p>
3.	<p>Instructions: Given a text input containing numerical data and a specific query, identify and extract the relevant numerical values, and perform the necessary calculations to produce a numerical output. Provide the output in a structured format without any additional comments. Input Format: Text: [input text containing numerical data] Query: [specific query or calculation to be performed] Output Format: Output: [calculated numerical value or values separated by commas]</p> <p>Steps: 1. Identify and Extract the Relevant Numerical Values: - Read the input text and query carefully to identify and extract the numerical values relevant to the query. 2. Perform the Required Calculations: - Use the extracted numerical values to perform the necessary calculations as specified in the query.</p>	<p>Instructions: Given a text input containing numerical data and a specific query, your task is to: 1. Extract the relevant numerical values based on the query. 2. Perform the necessary calculations using the extracted values. 3. Return the results in a structured format without any additional comments. Input Format: Text: [input text containing numerical data] Query: [specific query or calculation to be performed] Output Format: Output: [calculated numerical value or values separated by commas]</p> <p>Steps: 1. Identify and Extract the Relevant Numerical Values: - Read the input text and query carefully to identify and extract the numerical values relevant to the query. 2. Perform the Required Calculations: - Use the extracted numerical values to perform the necessary calculations as specified in the query.</p>

Table B3

Crossover examples. Prompts generated by the LLM-powered GA during the crossover operation. Inputs Xa and Xb present example prompts selected for crossover, and Output X shows the resulting crossed-over prompt, with lines aligned according to their similarity

#	Input	Output
1a.	Classify the given problem based on the underlying topic it represents, using one of the following categories: [Categories] Input Problem: [Place the problem text here] Output Category: [Provide the relevant category]	Classify the given problem statement by identifying its main topic or category. Use the list of categories provided to determine the most appropriate label for the problem. [Categories] Input Problem: [Place the problem text here] Output Category: [Provide the relevant category]
1b.	You will be given an input question or problem statement... Identify the main topic or category. [Examples:] Your task is to read the input question and determine which category or topic it is addressing, then provide the corresponding label.	[Examples:]
2a.	As a knowledgeable assistant, select the correct answer from the provided options using factual information. Ensure clarity in both the question and options. FORMAT: QUESTION: [Insert question here] OPTIONS: 1. [Option 1] 2. [Option 2] 3. [Option 3] 4. [Option 4] OUTPUT: [Correct Option] QUESTION: “АТФ се синтезира в: ендоплазмената мрежа, рибозомите, хлоропластите, апарата на Голджи?” OPTIONS: 1. ендоплазмената мрежа 2. рибозомите 3. хлоропластите 4. апарата на Голджи OUTPUT: хлоропластите	As an informed assistant, determine the correct answer from the given options using accurate knowledge. Ensure clarity in presenting both the question and the options.
2b.	Instruction: Identify the correct answer from the options provided, using historical, scientific, or logical knowledge. Question: [question] Choices: 1. [choice1] 2. [choice2] 3. [choice3] 4. [choice4] Correct Choice:	Instruction: Identify the correct answer from the options provided, using factual information. Question: [Insert question here] Options: 1. [Option 1] 2. [Option 2] 3. [Option 3] 4. [Option 4] Correct Option: [Correct Option]
3a.	Extract the explicitly stated information from the provided passage to directly and accurately answer the question. Focus solely on the relevant part of the passage, ensuring your response is precise, concise, and entirely based on the text without introducing assumptions, interpretations, or unnecessary details. Format: Passage: [Insert passage text] Question: [Insert question text] Answer: [Provide precise answer based solely on passage.]	Extract the specific information explicitly stated in the provided passage to accurately answer the question. Focus solely on the relevant portion of the text, ensuring your response is precise, concise, and strictly adheres to the passage without introducing assumptions, interpretations, or irrelevant details. Format: Passage: [Insert passage text] Question: [Insert question text] Answer: [Provide answer.]
3b.	Carefully analyze the provided passage to extract the specific information explicitly required to answer the given question. Your response must be clear, concise, and strictly grounded in the passage, avoiding interpretations, assumptions, or irrelevant details. Focus solely on the relevant portion of the text to ensure accuracy and alignment with the question's intent. Instructions: 1. Read the passage thoroughly to understand its context and content. 2. Locate the exact section of the passage that directly addresses the question. 3. Provide a clear and concise response that strictly adheres to the question's phrasing and intent, using only the information explicitly present in the passage. Template: Passage: [Insert passage text] Question: [Insert question text] Answer: [Provide the precise answer based solely on the passage.]	Instructions: 1. Read the passage carefully to understand its context and content. 2. Identify the exact section of the passage that directly addresses the question. 3. Provide a clear and accurate response using only the explicitly stated information, either as a direct quote or an exact paraphrase, aligned strictly with the question's intent. Template: Passage: [Insert passage text] Question: [Insert question text] Answer: [Insert precise response based solely on the passage.]

Appendix C. Examples of Fitness Matches

This appendix provides examples of positive and negative matches as adjudicated by our LLM-based fitness function.

Table C1 presents examples of positive matches across different scenarios, including differences in numerical format and precision between the gold answer and the model output, multilingual outputs, and domain-specific synonyms. Table C2 provides examples of negative matches across different models and datasets.

Table C1
Examples of POSITIVE matchings by the LLM-based fitness evaluation. Columns “Golden,” “Output,” and “Model” show the expected answer, the LLM’s actual output, and the used model, respectively

#	Golden	Output	Model
1	B Tree	Databases: Indexing, B Trees	Mistral Large
2	Network Flow	Networking: TCP/IP, Network Layering, Packet Fragmentation	Mistral Large
3	Database Normalization	Databases: Normalization, Normal Forms (1NF, 2NF, 3NF, BCNF)	Mistral Large
4	Flip Flop	Digital Logic: Flip-Flops (D flip-flop, JK flip-flop), Sequential Circuits, Propagation Delays	Mistral Large
5	темперамента	Correct Choice: 1. темперамента	GPT-4o
6	килийни, елино-български, взаимни, класни	взаимни, килийни, класни, елино-български	GPT-4o
7	1. “张某” 2. “走私、贩卖、运输、制造毒品”	1. Suspect’s Full Name: “张某” 2. Primary Crime: “Drug Trafficking”	GPT-4o
8	1. “王某某” 2. “拒不执行判决、裁定”	1. Suspect’s Full Name: “王某某” 2. Primary Crime: “Obstruction of Justice”	GPT-4o
9	Horn ALCHIF	The problem is 2EXPTIME-complete in Horn-ALCHIF (and also in Horn-ALC and in ELI).	Llama-3.3
10	The number of additional variables that are added to original stratified bases	The value of additional variables corresponds exactly to the number of priority levels existing in the base.	Llama-3.3
11	Our method Polyanya	Our method, Polyanya, extends and generalizes Anya.	Llama-3.3
12	Mostly-true	Mostly-true: The statement likely captures a real trend or observation regarding ... but may lack full context or accurate figures.	Mistral NeMo
13	Pants-fire	Pants-fire: The statement is incorrect and not supported by the factual evidence of national debt increases under each president.	Mistral NeMo
14	Barely-true	Barely-true: The statement is misleading and lacks significant factual support and context, leading to the classification of “barely-true.”	Mistral NeMo
15	17951.0	Net Change = 99,832 - 81,881 = 17,951 million	GPT-4o-mini
16	16.0	\$16 million	GPT-4o-mini
17	0.7446	74.46%	GPT-4o-mini
18	0.03353	Percentage change: $(116 / 3460) \times 100\% = 3.34\%$	Llama-3.2-11B
19	2430.0	Based on the calculation, we can report the following key metric: Statutory net income for Bermuda subsidiaries in 2010: \$2430 million	Llama-3.2-11B
20	Meta Interpretive Learning	Machine Learning → Inductive Logic Programming (ILP) → Meta-Interpretive Learning	Llama-3.2-11B

Table C2

Examples of NEGATIVE matchings by the LLM-based fitness evaluation. Similarly to Table C1, columns “Golden,” “Output,” and “Model” show the expected answer, the LLM’s actual output, and the used model, respectively

#	Golden	Output	Model
1	Concurrency	Computer Science: Algorithms, Memory Management	Mistral Large
2	System of equations	Mathematics: Linear Algebra	Mistral Large
3	-0.09709	(\$10 million / \$103 million) * 100 \approx 9.71%	GPT-4o
4	A neuromorphic architecture	Neuromorphic chips	GPT-4o
5	Many systems	Distributed systems	GPT-4o-mini
6	ML and statistics	Optimization is a vital ingredient in machine learning	GPT-4o-mini
7	POSITIVE	**Analysis: ** “NEUTRAL”	Mistral NeMo
8	1. “陈 3 某,” “陈 5 某” 2. “组织、领导传销活动”	1. **Suspect’s Full Name: ** “陈某” 2. **Primary Crime: ** “Leading a Criminal Syndicate”	Mistral NeMo
9	Comal and Guadalupe	Comal County	Mistral NeMo
10	Логодаж, Златарево	Гранични контролно-пропускателни пунктове на границата на България със Северна Македония са: Кулата, Златарево (Струмица), Илинден (Делчево) и Логодаж (Делчево), както и ГКПП Станке Лисичково (селско стопанство) и ГКПП Златарево (жп)	Llama-3.3