Computional Linguistic Task 5 - report

Jerzy Boksa

December 2025

1 Introduction

The goal of this assignment is to systematically evaluate and compare open-source Large Language Models (LLMs) across diverse task types using different prompting techniques.

I will work with models of two different sizes:

- **Small:** 1–2B parameters
- **Large:** 10–14B parameters, of which at least one is a reasoning-focused model

Through careful prompt engineering and manual evaluation, we aim to analyze how model size, architecture type, and prompting strategy affect performance across 10 diverse tasks.

1.1 Objectives of the Analysis

Analysis will demonstrate:

- The impact of model size on task performance
- Effectiveness of zero-shot versus few-shot prompting
- Benefits of chain-of-thought reasoning for standard models
- Performance differences between standard and reasoning-specialized models
- The importance of prompt engineering in achieving optimal results

1.2 Hardware

All experiments were carried out on the `athena.cyfronet.pl` supercomputing platform [1]. The computations utilized a single NVIDIA A100 GPU equipped with 40 GB of VRAM, providing ample resources for training and evaluating the selected large language models.

2 Methodology

In this section, I provide a detailed description of the models used in our experiments, the evaluation tasks along with their corresponding development and test sets, and the criteria employed to assess model performance.

2.1 Models

In this study, there are employed one smaller language models along with one larger reasoning-focused model. These models differ in size, architecture, and specialization, allowing to evaluate the impact of model scale and reasoning capabilities on task performance.

Models used:

- **Microsoft Phi-3-Mini-4K-Instruct** (3̃.8B parameters) — a small instruction-tuned language model designed for efficient zero-shot and few-shot reasoning [2].
- **deepseek-ai/DeepSeek-R1-Distill-Qwen-14B** (15B parameters) — a high-capacity instruction-tuned model optimized for general reasoning and instruction-following tasks. [3].

The included MiMo-V2-Flash is a new and experimental model. Although it has a very large architecture, we were interested in evaluating its reasoning capabilities and performance on the selected tasks.

All prompts and responses are attached to the `main.ipynb` file.

2.2 Tasks

For each of the ten tasks, two distinct sets of examples were prepared: a development set and a test set. The development set was used iteratively to refine and format prompts, experiment with few-shot or chain-of-thought techniques, and adjust instructions to improve model performance.

The test set, on the other hand, consists of 3-4 carefully selected examples per task, which serve as independent evaluation inputs to assess how well the models perform using the final prompt designs. In this report, only the results obtained with the finalized prompts on the test set are presented, ensuring that the evaluation reflects the true generalization capabilities of each model.

2.2.1 Task 1: Instruction Following (IFEval-style)

This task evaluates the model's ability to follow natural language instructions and generate responses that satisfy the given requirements. It tests understanding, paraphrasing, summarization, translation, and other instruction-based behaviors.

Task Description: The model is provided with an informal English sentence, and its task is to produce a more formal version of the text. The response should be in json format with two fields: before and after for each sentence.

Test Set:

- "They're gonna start the project soon."
- "I'm gonna finish this report tonight."
- "He's gonna call you later."
- "She's gonna join the meeting in a bit."

Evaluation: Since the test set contains only four examples, the model outputs will be manually evaluated. The evaluation will focus on the level of formality and the overall quality of the responses.

2.2.2 Task 2: Creative Writing

This task evaluates the model's ability to perform generative writing. Specifically, it assesses how well the model can produce coherent, imaginative, and contextually appropriate text based on a given prompt. The focus is on creativity, narrative structure, and linguistic fluency, rather than strict factual accuracy, making it a test of the model's expressive and generative capabilities.

Task Description: The goal of this task is to generate a creative six-line song verse based on a given prompt or theme. Each verse should incorporate the requested elements from the prompt and feature some form of alternating rhyme scheme, producing a coherent and expressive lyrical segment.

Test Set: For this task, the test set consists of a single theme: *"A mysterious forest where magical creatures hide"*. The model should generate a six-line song verse inspired by this theme.

Evaluation: The generated verse will be evaluated based on the following criteria:

- Use of rhymes and rhythmic consistency
- Faithfulness to the given theme
- Adherence to the required length of six lines

2.2.3 Task 3: Mathematical Set Theory

In this task, the models' proficiency in mathematical logic and problem-solving will be evaluated. The emphasis is on abstract logical reasoning capabilities, which are expected to pose significant difficulty for both models and likely lead to unsuccessful performance.

Task Description: The problem is straightforward for individuals with a basic background in mathematics, such as first-term engineering students. It originates from an introductory, first-year university course in mathematics, specifically within the area of Logic and Set Theory. Assuming familiarity with the standard axiomatic foundations and formal definitions of set theory, the task can be stated as follows:

Determine

$$\bigcap_{t \in T} A_t,$$

where

$$T = \mathbb{N}, \quad A_t = \{x \in \mathbb{R} \mid 0 \leq x \leq (2t+1)^{-1}\}.$$

For this task, no explicit partition into a training set and a test set is defined. The correct answer is: $\bigcap_{t \in T} A_t = \{0\}$.

In addition, a minor modification will be examined: in the definition of the set A_t, the inequality symbol $\leq$ will be replaced by $<$.

Evaluation: The assessment will focus on the following aspects:

- Ability to comprehend highly abstract definitions
- Ability to apply these definitions in rigorous logical reasoning
- Accuracy and correctness of the provided answers
- Identification and avoidance of logical inconsistencies or fallacies

It is generally hypothesized that models with a larger capacity should exhibit superior performance relative to their smaller counterparts.

2.2.4 Task 4: Code Generation

The objective is to evaluate whether the model can correctly interpret instructions and generate high-quality, functional source code.

Task Description: The objective is to substitute the conventional evolutionary operators employed in **Symbolic Discovery** [4] with a language-model-based mechanism. Concretely, the aim is to modify the `update_step` Python function so that it generates novel and distinct variants of itself (or its behavior), thereby enabling the language model to assume the role traditionally played by basic evolutionary operations. However, a dedicated set of functions is available specifically for performing mutation operations:

- **Arithmetic:** `add`, `substract`, `multiply`, `safe_divide`
- **Unary transformations:** `negate`, `square`, `abs`, `sign`
- **Nonlinear functions:** `exp`, `log`, `tanh`, `sigmoid`, `softplus`
- **Stabilization and bounds:** `safe_sqrt`, `clip`
- **Comparison operators:** `maximum`, `minimum`

Development Set: For the development dataset, a basic Stochastic Gradient Descent (SGD) optimization algorithm will be employed.

Test Set: The test set comprises two **state-of-the-art** optimization algorithms, namely `adam` and `adamw`. Furthermore, it is required to synthesize a novel algorithm, which will subsequently be evaluated on a set of standard benchmark functions.

Evaluation: The evaluation will check if the algorithm:

- runs - check if there is posibility to run it
- if only allowed operations are used
- fitness function of a mean log improvement

Fitness function: mean log improvement. Let $f(\cdot)$ denote the objective function of a given optimization problem and let $x^\star$ be its known global optimum, with $f^\star = f(x^\star)$. For each repetition $r = 1, \ldots, R$, let $x_0^{(r)}$ and $x_T^{(r)}$ denote the initial and final solutions, respectively.

The distance to the optimum at the beginning and at the end of the optimization process is defined as

$$\Delta f_0^{(r)} = \left| f\left(x_0^{(r)}\right) - f^\star \right| + \varepsilon, \qquad \Delta f_T^{(r)} = \left| f\left(x_T^{(r)}\right) - f^\star \right| + \varepsilon,$$

where $\varepsilon = 10^{-12}$ is a small constant introduced for numerical stability.

The logarithmic improvement achieved in repetition r is given by

$$I^{(r)} = \log \Delta f_0^{(r)} - \log \Delta f_T^{(r)} = \log \left(\frac{\Delta f_0^{(r)}}{\Delta f_T^{(r)}} \right).$$

The mean logarithmic improvement for a single problem is then computed as

$$\overline{I} = \frac{1}{R} \sum_{r=1}^{R} I^{(r)}.$$

Finally, the overall fitness value, averaged over a set of optimization problems $\mathcal{P}$, is defined as

$$\text{Fitness} = \exp\left(\frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \overline{I}_p \right),$$

where $|\mathcal{P}|$ denotes the number of considered problems.

The model is evaluated on a collection of benchmark optimization problems described in Appendix A, with each problem solved over 30 independent repetitions. The evaluation code was developed as part of the author's master's thesis and has not been publicly released at the time of writing.

3 Results

This section presents the outcomes of evaluating the selected language models across all tasks. For each task, results are reported for both small and large models under different prompting strategies, including Zero-Shot, Few-Shot, and Chain-of-Thought (where applicable).

The purpose of this section is to systematically compare model performance and highlight the impact of:

- Model size (small vs. large)
- Model architecture and reasoning capabilities
- Prompting strategy (Zero-Shot, Few-Shot, CoT)
- Compare quality of response between models

Each task subsection provides a detailed analysis of the model outputs and observarions. Differences in performance across models and prompting approaches are discussed, providing insights into the strengths and limitations of each model in relation to specific task types.

3.1 Task 1: Instruction Following (IFEval-style)

The objective is to reformulate sentences in a more formal register and to provide them in the specified JSON format. The corresponding results for each model and prompting strategy are presented therein.

3.1.1 Model Ph3-Mini

For each prompt strategy, the model was able to correctly formalize the test examples. The "after" outputs are more formal. Differences between prompting strategies can be observed in the raw responses, without json text extracting.

Before	**Zero-Shot After**	**Few-Shot After**	**CoT After**
Could you like, maybe check this for mistakes?	Could you please review this for any errors?	Could you please review this for any errors?	Could you please review this for any errors?
OMG, that was such a bad idea!	That was an unfortunate decision.	That was an unfortunate decision.	That was an unfortunate decision.
I guess we should, you know, start soon.	It seems advisable that we commence promptly.	It seems advisable to commence promptly.	It seems advisable that we commence promptly.
Lemme know if you need any help with that.	Please inform me if assistance is required.	Please inform me if assistance is required for that task.	Please inform me if assistance is required.

Table 1: Example outputs of Phi-3 Mini on the IFEval task under different prompting strategies.

The model was able to successfully formalize the informal sentences across all prompting strategies, producing valid JSON outputs. The extracted "after" sentences are largely similar, which is expected given that the same model was used throughout the task.

3.1.1.1 Zero-shot prompting strategy With the Zero-Shot strategy, the model completed the task successfully, but the raw responses often contained extra text after the JSON output. While this is not critical—since the JSON can be easily extracted—it may lead to higher token consumption and increased costs when using external LLMs.

3.1.1.2 Few-shot prompting strategy The Few-Shot strategy also produced correct formalizations, but overall token usage was significantly lower compared to Zero-Shot. For the training sentences, some additional text was generated, whereas for the test set, only the JSON array was returned. The JSON included the few-shot example sentences, which can be easily filtered if necessary.

3.1.1.3 Chain-of-Thought (CoT) strategy The Chain-of-Thought (CoT) strategy yielded results very similar to Zero-Shot. Although the raw responses contained extra tokens beyond the expected JSON, the sentences were formalized correctly. Overall, all strategies were effective in producing formalized sentences, with minor differences in verbosity and token efficiency.

Differences between prompting strategies are mainly observed in token usage and the presence of additional text beyond the expected JSON array.

3.1.2 Model deepseek-ai/DeepSeek-R1-Distill-Qwen-14B

For this model, all prompting strategies were successfully executed. The resulting formalized sentences are summarized in the table below, providing a clear comparison across Zero-Shot, Few-Shot, and Chain-of-Thought approaches.

Before	**Zero-Shot After**	**Few-Shot After**
Could you like, maybe check this for mistakes?	Could you please review this for any errors?	Could you please review this document for any errors?
OMG, that was such a bad idea!	That was an unfortunate decision.	That was an unfortunate decision.
I guess we should, you know, start soon.	It is advisable that we begin promptly.	It is advisable that we begin promptly.
Lemme know if you need any help with that.	Lemme know if you need any help with that.	Please let me know if assistance with that is required.

Table 2: Example outputs of Phi-3 Mini on the IFEval task under different prompting strategies.

ref

As we look at those example, the sentences are more formal. Unfortunately, there is no clear solution for that model. Despite the fact all prompt strategeis used json format, all responses contain some irrelevant text.

3.1.2.1 Zero-shot prompting strategy The strategy successfully parsed sentences in formal way and with json desired output. Before the json ouput however, there are some sentences indicating that the model is clearly in reasoning mode.

3.1.2.2 Few-Shot Prompting Strategy With the larger model using the Few-Shot prompting strategy, the desired output was successfully produced in the correct JSON format. However, to ensure the full JSON response was included, it was necessary to increase `max_new_tokens` from 512 to 1024. This is due to the reasoning mechanism of the model, which requires additional tokens to generate complete outputs.

3.1.3 Comparison

Overall, for this specific task, the smaller model performed better, particularly when using the Few-Shot prompting strategy. Not only was it able to accurately formalize sentences and produce the desired JSON output, but it also returned the test set without any additional, unnecessary data. It is worth noting, however, that this may not always be the case, and in some scenarios the smaller model could still generate extra text. Additionally, the smaller model was faster in processing.

In contrast, the larger model clearly demonstrated stronger reasoning capabilities. For each prompt, it provided a detailed analysis of the sentences to be formalized. While it was able to return valid JSON, it consistently generated extra tokens as part of its reasoning process and was unable to skip this analysis to return only the JSON output.

It appears reasonable to conclude that the larger model derives fewer benefits from the Few-Shot prompting strategy. This observation will be further explored in subsequent tasks.

3.2 Task 2: Creative Writing

Both models were able to generate short poetic verses on the specified topic. However, they differed in their rhyme schemes and in the accuracy of their rhyming. Furthermore, all models complied with the prescribed verse length of six lines.

3.2.1 Model Ph3-Mini

Overall, that model adhered to the specified constraints, generating coherent text that remains thematically consistent and is composed of six lines. However, under different strategy settings, not all of the rhymes are accurate, both with respect to lexical choice and to the prescribed ordering. The results are below:

Zero-shot

```
In the heart of the
   forest, where
   shadows play, (A)

Lies a realm
   untouched by the
   sun's fiery ray.
   (B)

Creatures of magic,
   in their secret
   lair, (A)

Whispering secrets,
   in the cool, damp
    air. (B)

Through the ancient
   trees, their eyes
    gleam with
   delight, (A)

In the mysterious
   forest, they hide
    from sight. (B)
```

Few-shot

```
In the heart of the
    woods where
   shadows play
The whispers of the
    trees tell
   tales of the
   night
Creatures of myth
   and legend come
   out to stay
Beneath the moon's
   soft glow, they
   dance in delight
Their magic weaves
   through the air,
    a spellbinding
   sight
In this enchanted
   realm, where
   dreams take
   flight
```

CoT

```
In the heart of the
    forest, where
   shadows play,
Creatures of magic
   dance in the
   light of day.
Whispers of wonder,
    secrets untold,
A realm of
   enchantment,
   where dreams
   unfold.

Through the trees,
   a symphony of
   life,
A world of mystery,
    where magic is
   rife.
```

Figure 1: Results on the test for different prompting strategies.

For each prompt strategy, model were albe to create at least some "poem" that is realted to the given test subject.

3.2.1.1 Zero-shot prompting strategy Under this prompting strategy, the generated output is of high quality. All rhymes are contextually appropriate, and the thematic content appears to be accurate. The structural organization of the text is largely satisfactory; however, the occurrence of atypical markers "(A)" and "(B)" at the ends of certain lines is noted, which may represent an artifact introduced by the prompt itself.

In addition, the rhyme scheme is not ABAB as might be expected, but instead follows an AABB pattern.

3.2.1.2 Few-Shot Prompting Strategy The few-shot prompting strategy exhibited superior performance, primarily because it did not produce lines ending with the token “(A)”. While the rhymes generated under this setting are of good quality, their overall structural pattern more closely approximates an ABAB rhyme scheme when compared to the zero-shot condition.

3.2.1.3 Chain-of-Thought (CoT) Strategy The Chain-of-Thought (CoT) prompting strategy also incorporates rhyme, although not in the canonical ABAB rhyme scheme. Furthermore, the overarching thematic coherence and structural organization of the prompt are preserved accurately.

3.2.2 Model deepseek-ai/DeepSeek-R1-Distill-Qwen-14B

The larger model performed worse on that task. Overall rhymes are lessa ccurate epsecially for the few-shot. Despite the only text, the response contains a lot of reasoning text. which was expected as it is larger reasoning targeted model.

Below there are both short verses. The only poems are presented, the reasoning part is skipped and can be seen in `main.ipynb`.

Zero-shot

```
Whispering pines and ancient trees,
Where secrets linger in the breeze,
Eyes that glow, a fleeting form,
In the shadows, they transform.
The forest hums with hidden lore,
A realm where magic evermore.
```

Few-shot

```
Whispers of wonder where the veil gives way,
Eyes like embers in the gathering day,
Ancient giants guard secrets untold,
Whispers of wings on the air,
Glowing moss paints paths unseen,
Where starlight weaves the magic we've never seen.
```

Figure 2: Results on the test for different prompting strategies.

Both responses align with the designated topic; however, their rhyme schemes do not adhere to the prescribed ABAB pattern, and multiple segments fail even to maintain an AABB configuration. Nonetheless, the overall macro-structural organization is appropriate.

The principal difference between the responses is that the model employing the few-shot strategy appears to engage in more extensive analysis and deliberation. However, this does not translate into higher or lower accuracy; it merely results in increased token usage, which is suboptimal.

3.2.3 Comparison

For this task, the smaller model appeared to yield superior performance. It not only consumed fewer tokens, thereby reducing cost, but also adhered more consistently to the specified rhyming constraints. The perceived aesthetic quality or "beauty" of the generated verses is inherently subjective and was therefore not considered a primary evaluation criterion.

3.3 Task 3: Mathematical Set Theory

Overall, the task is relatively straightforward for human participants; however, despite requiring only a limited degree of abstract reasoning, the models were still able to perform basic task successfully.

All prompts and their corresponding responses are documented in `main.ipynb`, specifically in Section 4, entitled "Task 3." An illustrative response generated by Phi using a few-shot strategy is presented in the table below.

Table 3: Phi3-Mini response with few-shot strategy

For $t = 1$: $A_1 = \{x \in \mathbb{R} \mid 0 \leq x \leq (2 \cdot 1 + 1)^{-1}\} = \{x \in \mathbb{R} \mid 0 \leq x \leq \frac{1}{3}\}$
For $t = 2$: $A_2 = \{x \in \mathbb{R} \mid 0 \leq x \leq (2 \cdot 2 + 1)^{-1}\} = \{x \in \mathbb{R} \mid 0 \leq x \leq \frac{1}{5}\}$
For $t = 3$: $A_3 = \{x \in \mathbb{R} \mid 0 \leq x \leq (2 \cdot 3 + 1)^{-1}\} = \{x \in \mathbb{R} \mid 0 \leq x \leq \frac{1}{7}\}$
As $t \to \infty$, the upper bound $(2t + 1)^{-1} \to 0$.
To be in all sets A_t, x must satisfy all inequalities simultaneously.
Therefore, the generalized intersection:

$$\bigcap_{t \in \mathbb{N}_1} A_t = \{0\}$$

Final answer:

$$\bigcap_{t \in \mathbb{N}_1} A_t = \{0\}$$

3.3.1 Model Ph3-Mini

Overall, the model successfully accomplished the original task, generating both correct solutions and coherent chains of reasoning for each of the two strategies. However, a slight alteration in the task specification—namely, replacing a non-strict inequality with a strict one—led the model to systematically fail on the modified task.

3.3.1.1 Zero-shot prompting strategy Unexpectedly, the model employing a zero-shot strategy with provided definitions was able to solve the task and produce the correct answer. However, a minor modification in the inequality led the model to an error. Specifically, instead of selecting the smallest value of the upper bound given by the limit as $t \to \infty$, the model incorrectly chose the smallest finite value of t. Overall, its performance can be evaluated as only partially successful, achieving approximately half of the expected outcomes.

3.3.1.2 Few-Shot Prompting Strategy & Cot The CoT part is included in the few-shot example.

The few-shot prompting strategy performed better than the zero-shot setting. When provided with an illustrative example, the model tended to follow the demonstrated reasoning pattern, yielding a higher-quality analysis and a more concise, accurate response.

However, the model again failed when presented with a slight modification of the inequality. In this case, the model produced the answer $[0, 0)$, which corresponds to the empty set. This is mathematically incorrect: when limits are involved in inequalities, the transition from a non-strict inequality ($\leq$) to a strict one ($<$) typically weakens the inequality in such a way that, for this task, the solution set should remain $\{0\}$ regardless of whether the problem is posed with $\leq$ or $<$.

This behavior suggests that the model lacks an accurate internalization of the concept of weakening inequalities in the context of limits, and therefore fails to preserve the correct solution set under small formal changes to the inequality.

3.3.2 Model deepseek-ai/DeepSeek-R1-Distill-Qwen-14B

The larger model demonstrated superior performance in this setting, successfully solving the task for both the strict inequality ($<$) and the non-strict inequality ($\leq$).

However, it was necessary to increase the maximum number of new tokens generated by the model from 1024 to 2048, which substantially increased the computational time.

An intriguing aspect of the model's behavior is that it engages in a form of self-verification by generating provisional hypotheses and subsequently evaluating them. For example, it may reason as follows: `"Wait, but is 0 the only element? Let me think again. Suppose` x `is a negative number. However, all` A_t `sets have lower bound 0, so` x `must be greater than or equal to 0. Therefore, negative numbers are excluded."`

3.3.2.1 Zero-shot prompting strategy There are no issues with either of the inequalities: the reasoning is correct, and it demonstrates an appropriate recognition that, in the context of limits, it is necessary to relax (weaken) the inequality. Although this is not stated explicitly, the following sentence provides indirect evidence of such understanding: *"So,* x *must satisfy* $0 \leq x < \frac{1}{2t+1}$ *for every* t*. But as* t *increases,* $\frac{1}{2t+1} \to 0$*. So, the only* x *that is less than all these upper bounds is* $x = 0$*."* This argument shows that the underlying conceptual framework has been correctly understood.

3.3.2.2 Few-Shot Prompting Strategy The behavior closely resembles the zero-shot setting and successfully passed both evaluations. The fundamental concept of the generated output remains similar; however, a larger number of validation steps are present, consistent with those highlighted in the few-shot example.

3.3.3 Comparison

Overall, the larger model demonstrated substantially superior performance, as it was able to appropriately generalize the underlying concept in the presence of a minor modification, namely a change in the inequality sign (from $\leq$ to $<$). This modification should not alter the final outcome, because inequalities typically become weaker when passing to the limit. The larger model successfully captured this nuance, whereas the smaller model failed to do so.

Another notable difference is that the larger model required a significantly higher number of newly generated tokens. This is attributable to the fact that it produced a more extensive and detailed analysis and, even after presenting a final answer, it proceeded to verify the correctness of its own solution.

3.4 Task 4: Code Generation

In the context of code generation, as with the mutation of different optimization algorithms, the models were initially able to produce code that appeared syntactically and structurally sound. However, some models only modified constant values without introducing any substantive changes to the underlying logic or algorithmic structure, which was not the intended outcome.

3.4.1 Model Ph3-Mini

For the model phi, none of the prompting strategies were able to substantially modify or improve the algorithmic behavior.

3.4.1.1 Zero-shot prompting strategy

For the zero-prompt setting, only the constants were modified in the AdamW algorithm used for testing. Below, the corresponding “before” and “after” configurations are presented. Moreover, this modification resulted in a reduction of the fitness score by the same amount. For the training examples, it is evident that the change affects only the identifiers (variable names) within the code.

	Before	After
fitness	9.22605836	0.57701782

It is worth noting that the fitness value can be redefined or adjusted to accommodate different sets of objective functions, both in this context and for the subsequent algorithms.

3.4.1.2 Few-shot prompting strategy

In the few-shot setting, the model performed even worse than in the zero-shot configuration. Its output did not exhibit substantive variation; instead, it increasingly reproduced essentially the same code example. This behavior is unexpected, particularly given that, for the training example, the model was able to derive the momentum optimization algorithm starting from stochastic gradient descent (SGD). Since there are no differences between the algorithms, the evaluation based on the fitness score is omitted.

3.4.1.3 Chain-of-Thought (CoT) strategy

The COT prompt strategy were able to actually modify te algorithm very noticeable - not only constants.

```
def update_step(x, g, v):
   b2=0.999
   eps=1e-08
   lr=0.01
   wd=0.01
   one=1.0
   v1=multiply(b2, v)
   one_minus_b2=substract(one,
       b2)
   g2=square(g)
   v2=multiply(one_minus_b2,
      g2)
   v_new=add(v1, v2)
   sqrt_v=safe_sqrt(v_new)
   denom=add(sqrt_v, eps)
   update=safe_divide(g, denom
      )
   step=multiply(lr, update)
   decay=multiply(wd, x)
   total_step=add(step, decay)
   x_new=substract(x,
      total_step)
   return x_new, v_new
```

Listing 1: Algorithm Before

```
def update_step(x, g, v, t):
    b1=0.9
    b2=0.999
    eps=1e-08
    lr=0.01
    wd=0.01
    one=1.0
    t1=multiply(b1, t)
    t2=multiply(b2, t)
    v1=multiply(t1, v)
    one_minus_b1=substract(one
       , t1)
    g2=square(g)
    v2=multiply(t2, g2)
    v_new=add(v1, v2)
    sqrt_v=safe_sqrt(v_new)
    denom=add(sqrt_v, eps)
    update=safe_divide(g,
       denom)
    step=multiply(lr, update)
    decay=multiply(wd, x)
    total_step=add(step, decay
       )
    clip_step=clip(total_step,
        -1, 1)
    x_new=substract(x,
       clip_step)
    v_new=multiply(
       one_minus_b1, g)
    return x_new, v_new
```

Listing 2: Modified Algorithm

However, there is not much text explaining the reasoning. In addition, the input of the function has been modified in an unexpected manner. Consequently, the function cannot be properly evaluated without incorporating my modifications.

	Before	After
fitness	9.22605836	2.30487705

This modification also led to a decrease in the fitness value. At least, the program executed without errors after setting t to 1.

3.4.2 Model deepseek-ai/DeepSeek-R1-Distill-Qwen-14B

The model and the resources utilized were likely disproportionate to the complexity of the task. It was unable to reach the relevant code section, despite the parameter ‘max_new_tokens‘ being set to 2048, which can be considered relatively large for this purpose. In addition, the generation of the response was highly time-consuming.

3.4.2.1 Zero-shot prompting strategy The model initiated an analysis of the provided algorithm and began to internally generate potential additional features and operations. Although it adhered to the specified set of permissible operations, it was unable to produce the complete target algorithm within the limit of 2048 tokens.

3.4.2.2 Few-shot prompting strategy The behavior of the model under the few-shot prompting configuration was analogous to that observed in the zero-shot setting.

3.4.3 Comparison

The experimental results suggest that the `phi` model is likely too small to achieve satisfactory performance on the given tasks, or alternatively, that more extensive prompt engineering may be required to elicit better outputs. A direct quantitative comparison between the available models is not feasible, as the larger model cannot be evaluated under the same conditions due to its inability to fit within the expected new-token context window. Nevertheless, an examination of the model's behavior indicates that it may still be capable of producing a viable algorithmic solution.

In conclusion, additional prompt-engineering efforts appear necessary for this task, both to improve the performance of the smaller model and to encourage the larger model to generate more efficient and concise code examples.

3.5 Tasks 5–10: Remaining Work

The remaining tasks (5–10) will be presented in a more concise manner and subjected to a less comprehensive analysis, due to the substantial amount of time that would be required for an in-depth examination.

4 Conclusion

References

[1] Academic Computer Centre Cyfronet AGH. *Athena Supercomputer Documentation.* `https://docs.hpc.cyfronet.pl/supercomputers/athena/`. Accessed: 2025-12-27. Cyfronet AGH, 2025. URL: `https://docs.hpc.cyfronet.pl/supercomputers/athena/`.

[2] Hugging Face and Microsoft. *Microsoft Phi-3-Mini-4K-Instruct Model.* `https://huggingface.co/microsoft/Phi-3-mini-4k-instruct`. Accessed: 2025-12-27. Hugging Face, 2024. URL: `https://huggingface.co/microsoft/Phi-3-mini-4k-instruct`.

[3] DeepSeek-AI et al. *DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning.* 2025. arXiv: `2501.12948 [cs.CL]`. URL: `https://arxiv.org/abs/2501.12948`.

[4] Xiangning Chen et al. “Symbolic Discovery of Optimization Algorithms”. eng. In: (2023).

A Benchmark Optimization Problems

This appendix lists all benchmark optimization problems used in the evaluation. The *Instance ID* denotes different translations and/or rotations of each problem.

Table 4: Benchmark problems.

Name	**Dim**	**Instance ID**
Sphere	5	1
Sphere	10	5
Sphere	25	10
Rastrigin	5	1
Rastrigin	10	5
Rastrigin	25	10
Rosenbrock	5	1
Rosenbrock	10	5
Rosenbrock	25	10
Ackley	5	1
Ackley	10	5
Ackley	25	10
Griewank	5	1
Griewank	10	5
Griewank	25	10