



# SwarmPrompt: Swarm Intelligence-Driven Prompt Optimization Using Large Language Models

Thilak Shekhar Shriyan<sup>1</sup>, Janavi Srinivasan<sup>2</sup>, Suhail Ahmed<sup>2</sup>, Richa Sharma<sup>3</sup><sup>a</sup> and Arti Arya<sup>3</sup><sup>b</sup>

<sup>1</sup>*Couchbase, Bangalore, India*

<sup>2</sup>*Hewlett Packard Enterprise, Bangalore, India*

<sup>3</sup>*PES University, Bangalore, India*

{thilak.shriyan43, janavisrinivasan26, suhailahmedvelorum}@gmail.com, {richasharma, artiarya}@pesu.edu

**Keywords:** Swarm Intelligence Algorithms, Prompt Optimization, Prompt Evaluation, Prompt Engineering, Discrete Prompt Optimization, Particle Swarm Optimization, Grey Wolf Optimization.

**Abstract:** The advancement of generative AI and large language models (LLMs) has made developing effective text prompts challenging, particularly for less experienced users. LLMs often struggle with nuances, tone, and context, necessitating precise prompt engineering for generating high-quality outputs. Previous research has utilized approaches such as gradient descent, reinforcement learning, and evolutionary algorithms for optimizing prompts. This paper introduces SwarmPrompt, a novel approach that employs swarm intelligence-based optimization techniques, specifically Particle Swarm Optimization and Grey Wolf Optimization, to enhance and optimize prompts. SwarmPrompt combines the language processing capabilities of LLMs with swarm operators to iteratively modify prompts and identify the best-performing ones. This method reduces human intervention, surpasses human-engineered prompts, and decreases the time and resources required for prompt optimization. Experimental results indicate that SwarmPrompt outperforms human-engineered prompts by 4% for classification tasks and 2% for simplification and summarization tasks. Moreover, SwarmPrompt converges faster, requiring half the number of iterations while providing superior results. This approach offers an efficient and effective alternative to existing methods. Our code is available at SwarmPrompt.

## 1 INTRODUCTION


The art of prompt engineering lies in crafting the right questions to maximize the output of large language models (LLMs). By facilitating direct communication with LLMs through simple natural language commands, prompt engineering enables better responses. However, creating effective text prompts that yield high-quality outputs is a skill with a steep learning curve, especially for novice and non-technical users.


Automated prompt optimization offers key advantages over manual methods, especially in scenarios requiring scalability, efficiency, and precision. Unlike labor-intensive manual approaches prone to human biases, automated methods use algorithms to systematically explore and refine prompts. This data-driven process evaluates numerous variations quickly, enabling faster and more accurate identification of optimal prompts.

Various prompt engineering methods include zero-shot prompting (no examples provided) and few-shot or multi-shot prompting (using multiple examples to guide the model). Techniques like Chain of Thought (COT) prompting help LLMs reason through responses, enhancing performance on complex tasks. However, human expertise remains essential for crafting effective prompts. Prompt engineers require a deep understanding of language syntax, semantics, pragmatics, and model-specific characteristics, often relying on iterative experiments to achieve optimal results. This paper aims to eliminate human dependency by automating the evaluation and optimization of user-provided prompts.

Prompt tuning can be performed in two primary ways to enhance prompt quality:

- **Soft Prompt Tuning:** This approach utilizes gradient descent and requires the computation of internal gradients within LLMs. The resulting prompts are often not human-readable.
- **Discrete Prompt Tuning:** This method modifies concrete tokens from a predefined vocab-

<sup>a</sup> <https://orcid.org/0000-0002-4539-7051>

<sup>b</sup> <https://orcid.org/0000-0002-4470-0311>

ularly, exploring the search space to improve prompts. Discrete prompt optimization, however, is challenging because the prompts are generated through "enumeration-then-selection" heuristics, which may not cover the entire search space.

Recently, various methods for discrete prompt optimization have emerged, including Reinforcement Learning (Pang and Lee, 2005) and Evolutionary Algorithms (Guo et al., 2023a).

In prior work (Guo et al., 2023b) leading up to our study, discrete prompt optimization is performed using Genetic Algorithms (GA) and Differential Evolution (DE), a subset of Evolutionary Algorithms. Here it proposes to connect evolutionary operators with LLMs, which proves to be very powerful as implementing evolutionary algorithms conventionally will alter tokens individually, without considering the semantics between the tokens, to generate new candidate solutions for the next iterations. Hence, the LLMs act as evolutionary operators and modify the prompts to preserve their semantic meaning, thus yielding human-understandable prompts.

Regarding the choice of algorithms for this paper, GA mimics natural selection by selecting the fittest individuals for reproduction, involving operations like selection, crossover, and mutation. However, GA can suffer from premature convergence to local optima in complex, multi-modal landscapes, primarily due to the loss of population diversity or suboptimal parameter settings. GA also has high computational costs. On the other hand, DE iteratively improves candidate solutions by leveraging differences between randomly selected individuals, proving effective for continuous optimization with fewer parameters to tune compared to GA. Despite its benefits, DE can stagnate and struggle with noisy fitness landscapes.

Swarm intelligence algorithms, inspired by the collective behaviour of social organisms like ant colonies and bird flocks, use interactions between individual agents to solve optimisation problems. Each swarm agent interacts with others and the environment to explore the solution space, guided by swarm operators' rules that dictate interaction, position updates, and the balance between exploration and exploitation.

These algorithms offer fast convergence, robustness to local optima, parallelism, and decentralization. Particle Swarm Optimization (PSO) and Grey Wolf Optimization (GWO) have gained attention for their efficiency. PSO, inspired by particle movement in nature, often outperforms Genetic Algorithms (GA) and Differential Evolution (DE) with faster convergence and fewer parameters. GWO, modelled on grey wolf social behaviour, achieves a strong bal-

ance between exploration and exploitation with simple implementation. While GA and DE are effective, PSO and GWO address their limitations, making them ideal for complex optimization problems.

This paper introduces SwarmPrompt, a framework for discrete prompt optimization leveraging swarm intelligence. Here, swarm agents represent candidate prompts, and swarm operators—implemented via LLMs—iteratively refine prompts while maintaining semantic meaning. This integration ensures human-readable outputs and enhanced optimization performance. The main contributions of this paper are:

- A novel framework called *SwarmPrompt*, which leverages swarm intelligence algorithms such as PSO and GWO to iteratively optimize the population of prompts, ultimately yielding an optimized discrete prompt.
- Conducted multiple experiments on various tasks such as classification, simplification, and summarization on different population sizes and budgets and compared these results with existing methods of prompt optimization, such as Genetic Algorithms (GA) and Differential Evolution (DE).
- In comparison to existing methods, this work achieves superior results for the classification task and comparable results for the simplification and summarization tasks while running for half the iterations, therefore taking less resources and time to converge.

## 2 RELATED WORK

P. Liu et al. (Liu et al., 2023) provides a comprehensive review of prompt-based learning, categorizing key concepts and methodologies. The study covers fundamental techniques, including the creation of prompt templates and formulating prompt responses, alongside advanced strategies such as multi-prompt learning and prompt-aware training. Additionally, the authors discuss practical applications and the impact of different prompting methods on task outcomes.

B. Lester et al. (Lester et al., 2021) introduces *soft prompts*, which are continuous vectors optimized through backpropagation to adapt pre-trained LLMs efficiently. Unlike traditional fine-tuning, soft prompts require fewer parameter updates and demonstrate scalability as model sizes increase. The study also shows that soft prompts generalize well across tasks, supporting domain transferability.

R. Ma et al. (Ma et al., 2024) investigates prompt optimization mechanisms for LLMs, revealing chal-

lenges such as the inability of LLMs to recognize mistakes or generate optimal prompts in a single refinement step. To address this, the authors propose an *Automatic Behavior Optimization* framework that employs resampling-based and reflection-based prompt regeneration techniques to optimize model behavior effectively.

Y. Zhou et al. (Zhou et al., 2022) propose the *Automatic Prompt Engineer* (APE) for generating and selecting task-specific instructions. APE optimizes instruction candidates using zero-shot performance evaluation from another LLM. Results demonstrate that automatically generated instructions surpass human-crafted prompts and existing LLM baselines.

C. Yang et al. (Yang et al., 2023) present an iterative optimization approach where LLMs act as optimizers for generating task solutions. At each stage, new prompts are created based on previous solutions and their evaluations. The approach effectively optimizes tasks like linear regression and combinatorial problems, such as the traveling salesman problem.

X. Wang et al. (Wang et al., 2023) address limitations in prior prompt optimization methods, particularly the underutilization of expert-level prompt knowledge. The authors introduce *PromptAgent*, which formulates prompt optimization as a strategic planning problem and uses Monte Carlo Tree Search (MCTS) to explore high-reward prompts. PromptAgent leverages expert insights, detailed error reflection, and constructive feedback to refine prompts iteratively.

Q. Guo et al. (Guo et al., 2023a) propose *EVO-PROMPT*, an evolutionary algorithm-based approach for prompt optimization. EVOPROMPT begins with an initial population of prompts and iteratively refines them using evolutionary operators and LLM-generated variations. By ensuring coherence and fast convergence without relying on gradients, EVO-PROMPT achieves notable performance gains.

M. Janga Reddy et al. (Janga Reddy and Nagesh Kumar, 2020) highlight the significance of swarm intelligence (SI) algorithms, which leverage collective group intelligence and self-organization to evolve global-level solutions from local interactions. SI methods, often termed behaviorally inspired algorithms, have gained considerable attention, with over 500 metaheuristic algorithms (MAs) introduced to date, including 350 in the last decade (Rajwar et al., 2023).

Although research on swarm intelligence in NLP remains limited, studies suggest that SI algorithms offer substantial benefits for optimization tasks (Bharambe et al., 2024). Specifically, SI techniques

enhance feature selection and parameter tuning, leading to improved performance in tasks such as text clustering (Selvaraj and Choi, 2021). Their inherent parallelism also facilitates scalability, enabling efficient handling of large datasets and complex models (Bharti et al., 2022).

Despite advancements in prompt-based learning and optimization, key challenges persist:

- **Uncontrolled LLM Behavior.** LLMs often produce unpredictable outputs, including deviations from desired structures and irrelevant or redundant information.
- **Resource Constraints.** Existing methods can be computationally expensive, requiring extensive iterations and large budgets to achieve optimal results.

To address these limitations, swarm intelligence algorithms present a promising alternative. By constraining the search space and leveraging structured templates, swarm-based methods can effectively control LLM outputs. Additionally, their ability to converge rapidly enables resource-efficient optimization, achieving high-quality results with fewer iterations and reduced computational costs.

## 3 PROPOSED APPROACH

### 3.1 General Framework of SwarmPrompt

This paper presents an approach that integrates swarm optimization algorithms, specifically PSO and GWO, with LLMs to optimize prompts for tasks like classification, summarization, and simplification. The iterative optimization process enhances prompt quality, achieving significant performance improvements over baseline methods.

Unlike evolutionary algorithms, swarm intelligence algorithms maintain a constant population size, with in-place mutations. As shown in Fig. 1, the framework consists of three main components: prompt instantiation, prompt evolution, and evaluation and update.

The initial prompt set combines human-engineered and LLM-generated prompts to ensure diversity and incorporate prior knowledge. These prompts are paraphrased using an LLM, and the new candidates are evaluated using task-specific metrics. Their effectiveness is validated as discrete prompts for the corresponding task, with the LLM's performance serving as a measure of prompt quality.

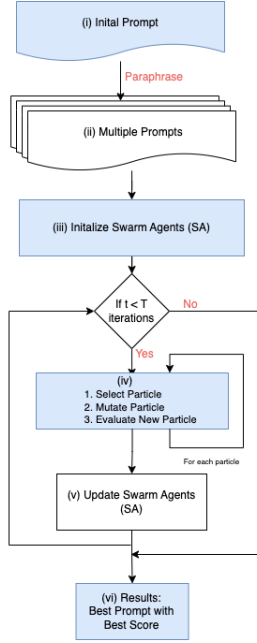


Figure 1: General workflow of swarm intelligence algorithms for optimizing prompts.

This iterative process refines prompts to achieve task-specific goals effectively.

Swarm agents are initialized with algorithm-specific parameters. For PSO, this includes population size, inertia weight, and cognitive and social coefficients; for GWO, parameters include the number of wolves, convergence factor, and hunting behavior. Since LLMs handle the updates, explicit coefficient adjustments are not performed.

The evolutionary process runs for a fixed number of iterations ( $T$ ), also referred to as the budget (Guo et al., 2023b). Each iteration involves the following steps:

- **Selection.** A particle (prompt) is selected from the population.
- **Mutation.** The particle undergoes mutation using a swarm intelligence mutation function. In PSO, this involves updating velocity and position based on cognitive and social components. In GWO, positions are updated relative to the alpha, beta, and delta wolves.
- **Evaluation.** The mutated particle is evaluated using fitness metrics: accuracy for classification, ROUGE-n for summarization, and SARI for simplification.
- **Update.** Swarm agents are updated based on particle performance. In PSO, global and personal best positions are updated; in GWO, the wolf positions adjust based on the top three solutions.

For simplicity, the LLM adjusts parameters dynamically during optimization using a predefined template, as detailed in the experimental setup.

The objective is to evolve prompts using PSO and GWO for optimal task performance, rather than making minor tweaks. This iterative update mechanism steers the search towards improved solutions by leveraging swarm intelligence principles, enabling effective exploration of the search space and generating high-quality, diverse prompts.

The process terminates after  $T$  iterations, identifying the best-performing prompt with the highest score as the output. Fig. 1 illustrates the entire workflow.

### 3.2 SwarmPrompt Using Particle Swarm Optimization

In PSO, we begin with a population of  $N$  prompts, iteratively refining them based on a fitness function evaluated on the development set. SwarmPrompt using PSO follows these steps:

- **Evolution.** Each prompt is modified using the LLM, acting as the PSO operator, to move closer to the global best prompt and its own previous best prompt.
- **Update.** The new prompts are evaluated on the development set. The global best prompt is updated to the highest-performing prompt, and each prompt’s personal best is updated accordingly.

The process terminates after a fixed number of iterations, returning the global best prompt from the final iteration. The detailed algorithm is provided in Algorithm 1.

---

Algorithm 1: Optimizing Prompts using Particle Swarm Optimization (PSO).

---

**Require:** Initial set of prompts, population size, development dataset

- 1: Initialize: evaluate initial prompts on the dataset
  - 2: Initialize personal best  $P_{\text{best}}$  for each prompt and global best  $P_{\text{global}}$
  - 3: **for** each iteration **do**
  - 4:   **for** each prompt in the population **do**
  - 5:     Generate a new prompt using  $P_{\text{best}}$ ,  $P_{\text{global}}$ , and the current prompt
  - 6:     Evaluate the new prompt on the dataset
  - 7:     Update  $P_{\text{best}}$  if the new prompt performs better
  - 8:   **end for**
  - 9:   Update  $P_{\text{global}}$  with the best  $P_{\text{best}}$  prompt
  - 10: **end for**
  - 11: **return**  $P_{\text{global}}$  as the best prompt
-

### 3.3 SwarmPrompt Using Grey Wolf Optimization

Algorithm 2: Optimizing Prompts using Grey Wolf Optimization (GWO).

**Require:** Initial set of prompts, population size, development dataset

- 1: Initialize: evaluate initial prompts on the dataset
- 2: Initialize alpha  $P_\alpha$ , beta  $P_\beta$ , and delta  $P_\delta$  prompts
- 3: **for** each iteration **do**
- 4:   **for** each prompt in the population **do**
- 5:     Generate a new prompt using  $P_\alpha$ ,  $P_\beta$ ,  $P_\delta$ , and the current prompt
- 6:     Evaluate the new prompt on the dataset
- 7:   **end for**
- 8:   Update  $P_\alpha$  (best prompt),  $P_\beta$  (second-best), and  $P_\delta$  (third-best)
- 9: **end for**
- 10: **return**  $P_\alpha$  as the best prompt

In GWO, we start with a population of  $N$  prompts, refining them iteratively based on a fitness function evaluated on the development set. SwarmPrompt using GWO follows these steps:

- **Evolution.** Each prompt is modified using the LLM, acting as the GWO operator, to move closer to the alpha, beta, and delta prompts (the best, second-best, and third-best prompts).
- **Update.** New prompts are evaluated on the development set, and the alpha, beta, and delta prompts are updated as the top three performing prompts.

The process terminates after a fixed number of iterations, returning the alpha prompt as the best prompt. The detailed algorithm is provided in Algorithm 2.

## 4 EXPERIMENTAL SETUP

### 4.1 Datasets

PSO and GWO algorithms were evaluated on three datasets, corresponding to distinct tasks: classification (cls), simplification (sim), and summarization (sum). The datasets used are as follows:

- **MR Dataset** (Pang and Lee, 2005). Developed by Pang and Lee, it consists of movie reviews labeled on a multi-point scale (1–5 stars), enabling nuanced sentiment classification beyond binary analysis.
- **ASSET Dataset** (Alva-Manchego et al., 2020). Created by Alva-Manchego et al., it includes mul-

iple simplified versions of complex sentences, supporting diverse simplification tasks such as lexical replacement and sentence splitting.

- **SAMSum Dataset** (Gliwa et al., 2019). Designed by Gliwa et al., this dataset comprises human-annotated dialogues for abstractive summarization, enabling concise conversational text summarization.

The optimized prompts are evaluated as discrete inputs to the LLM for each task, and the LLM’s performance serves as the metric for assessing the optimization process.

### 4.2 Experiments Conducted

- Population sizes of 4, 6, 8, and 10 were tested with a budget of 5 iterations for classification, simplification, and summarization tasks.
- Additional experiments with population sizes of 4 and 8 were conducted with a budget of 10 iterations for classification and simplification tasks.

### 4.3 Templates

Templates for prompt mutation in PSO and GWO are provided to the LLM during each iteration.

- **PSO Template.** Fig. 2 shows the template for PSO. Here, *Prompt 1* refers to the global best prompt, *Prompt 2* to the personal best prompt, and *Prompt 3* to the current prompt being modified. The example task in Fig. 2 is classification.

Query
Please follow the instructions step-by-step to generate a better prompt. <ol style="list-style-type: none"> <li>1. Crossover the following prompts and generate a new prompt such that the new prompt is a mix of 70 percent of Prompt 1, 20 percent of Prompt 2 and 10 percent of Prompt 3:                Prompt 1: Your task is to classify the comment as one of the following categories: terrible, bad, okay, good, great.                Prompt 2: In this task, you are given sentences from movie reviews. The task is to classify a sentence as one of the following categories: terrible, bad, okay, good, great.                Prompt 3: Determine the sentiment of the provided statement.</li> <li>2. Take the prompt generated in Step 1 and append &lt;prompt&gt; to the beginning of it and &lt;/prompt&gt; to the end of it.</li> </ol>
Response
<ol style="list-style-type: none"> <li>1. Crossover Prompt: In this task, you are given comments from movie reviews. Your task is to classify each comment as one of the following categories: terrible, bad, okay, good, great.</li> <li>2. &lt;prompt&gt;In this task, you are given comments from movie reviews. Your task is to classify each comment as one of the following categories: terrible, bad, okay, good, great.&lt;/prompt&gt;</li> </ol>

Figure 2: Template for PSO-based prompt mutation.

- **GWO Template.** Fig. 3 illustrates the template for GWO. Here, *Prompt 1* is the alpha prompt, *Prompt 2* the beta prompt, *Prompt 3* the delta prompt, and *Prompt 4* the current prompt being modified. The example task in Fig. 3 is simplification.

Query
<p>Please follow the instructions step-by-step to generate a better prompt.</p> <ol style="list-style-type: none"> <li>Crossover the following prompts and generate a new prompt such that the new prompt is a mix of 30 percent of Prompt 1, 25 percent of Prompt 2, 25 percent of Prompt 3 and 20 percent of Prompt 4:  Prompt 1: Rewrite the input text into simpler text.  Prompt 2: Rewrite my complex sentence in simpler terms, but keep the meaning.  Prompt 3: Simplify my complex sentence while maintaining its meaning.  Prompt 4: Make the input text easier to understand.</li> <li>Take the prompt generated in Step 1 and append &lt;prompt&gt; to the beginning of it and &lt;/prompt&gt; to the end of it.</li> </ol>
Response
<ol style="list-style-type: none"> <li>Crossover Prompt: Rewrite the complex text into simpler text while keeping its meaning.</li> <li>&lt;prompt&gt;Rewrite the complex text into simpler text while keeping its meaning.&lt;/prompt&gt;</li> </ol>

Figure 3: Template for GWO-based prompt mutation.

## 4.4 Hyperparameters

To conduct experiments on PSO and GWO algorithms, prompts are modified using predefined templates. In PSO, adjustments are based on the personal best and global best prompts, while in GWO, they depend on the Alpha, Beta, and Delta prompts. These adjustments correspond to key parameters: inertia weight, cognitive and social coefficients (PSO), and hunting behavior coefficients  $a$ ,  $A$ , and  $C$  (GWO).

For simplicity, the parameters are hardcoded into template files. Figures 2 and 3 illustrate the hyperparameters: in PSO, 10% of the current prompt is retained, modified by 70% of the global best and 20% of the personal best. For GWO, 20% of the current prompt is retained, adjusted by 30% of the Alpha, and 25% each of the Beta and Delta prompts. A higher weight is allocated to the global best (PSO) and Alpha (GWO) prompts, as they guide the mutation towards better-performing prompts.

The hyperparameters were selected empirically to balance performance and computational efficiency. While they provided satisfactory results, they can be refined or dynamically adjusted in future work to better align with specific experimental objectives.

## 5 RESULTS & DISCUSSION

### 5.1 SwarmPrompt Using PSO

In the SwarmPrompt (PSO) framework, each particle tracks its personal best position across iterations, while the global best prompt is updated and used for subsequent mutations. Our observations highlight that with a smaller budget and larger population, PSO efficiently explores the search space, converging rapidly to near-optimal results.

### 5.2 SwarmPrompt Using GWO

In the SwarmPrompt (GWO) approach, the optimization process relies on three primary swarm agents: Alpha, Beta, and Delta, along with one particle undergoing mutation. Notably, the minimum population size for GWO is four. Experimental results show that increasing the population size expands the search space, which can negatively impact accuracy. However, increasing the budget allows GWO to refine its search more effectively over iterations.

Table 1: Model Accuracy for Different Tasks and Optimization Techniques.

Method	Classification (MR)	Simplification (ASSET)	Summarization (SAMSum)
Human-Engineered Prompts	88.75	43.03	26.25
Evoprompt (DE)	90.22	46.21	29.62
Evoprompt (GA)	90.07	46.43	28.67
SwarmPrompt (PSO)	91.5	44.81	28.23
SwarmPrompt (GWO)	92.2	45.06	28.95

Table 1 shows that SwarmPrompt outperforms Evoprompt by 2% and human-engineered prompts by 4% in the classification task (MR dataset). In the simplification task (ASSET dataset), Evoprompt slightly exceeds SwarmPrompt by 1%, but SwarmPrompt still outperforms human-engineered prompts by 2%. For summarization (SAMSum dataset), Evoprompt (DE) achieves the highest accuracy, followed by SwarmPrompt (GWO), with SwarmPrompt outperforming human-engineered prompts by 2%.

These results indicate that SwarmPrompt excels in classification but is less effective for simplification and summarization, likely due to the higher complexity and dimensionality of these tasks, where PSO and GWO struggle to maintain diversity. In contrast, Differential Evolution (DE) in Evoprompt performs better in high-dimensional spaces.

Notably, Evoprompt’s results were achieved with a budget of 10, while SwarmPrompt achieved similar accuracy with a budget of just 5, demonstrating its greater resource efficiency.

### 5.3 Impact of Population Size on Task Accuracy

Figs. 4, 5, and 6 provide detailed insights into the relationship between population size and accuracy for SwarmPrompt across classification, simplification, and summarization tasks, respectively, under a constant budget of 5.

- **Classification Task (MR Dataset).** As shown in Fig. 4, both PSO and GWO achieve peak accuracy with a population size of 4. Further increases in population size result in stagnation of accuracy.
- **Simplification Task (ASSET Dataset).** Fig. 5 indicates that PSO achieves maximum accuracy with a population size of 4, while GWO performs best with a population size of 6.
- **Summarization Task (SAMSum Dataset).** In Fig. 6, GWO consistently outperforms PSO across all population sizes, demonstrating its superior performance for summarization tasks.



Figure 4: Accuracy trends for PSO and GWO with Population Size for Classification.

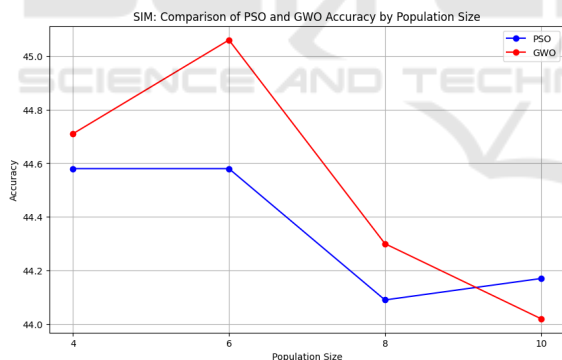


Figure 5: Accuracy trends for PSO and GWO with Population Size for Simplification.

### 5.4 Impact of Budget on Task Accuracy

Table 2 presents results for classification and simplification tasks with a higher budget (10) for population sizes of 4 and 8. The findings suggest that increasing the budget has minimal impact on accuracy, reiterating that SwarmPrompt achieves effective results even with a smaller budget.



Figure 6: Accuracy trends for PSO and GWO with Population Size for Summarization.

Table 2: Accuracy for PSO and GWO with Budget 10 on CLS and SIM Tasks.

Method	CLS Accuracy		SIM Accuracy	
	Population Size 4	Population Size 8	Population Size 4	Population Size 8
SwarmPrompt (PSO)	89.6	89.6	44.81	44.58
SwarmPrompt (GWO)	90.0	89.6	27.96	44.31

These results confirm that SwarmPrompt achieves stable accuracy with reduced computational resources, making it a highly efficient optimization method.

## 6 CONCLUSION & FUTURE SCOPE

In this study, we proposed SwarmPrompt, leveraging swarm intelligence algorithms (PSO and GWO) as optimization operators for LLM prompt optimization. Our experiments demonstrate that SwarmPrompt consistently outperforms human-engineered prompts and achieves competitive performance against existing evolutionary methods like Evoprompt, but with significantly lower resource requirements.

Key observations include:

- SwarmPrompt excels in classification tasks but shows minor limitations for high-dimensional tasks like simplification and summarization.
- Smaller population sizes (4–6) and lower budgets (5) are sufficient to achieve optimal results, ensuring computational efficiency.

While our work explored a fixed set of hyperparameters, future research can investigate dynamic hyperparameter tuning and other swarm intelligence techniques like Ant Colony Optimization. Additionally, this study focused on Alpaca, an open-source

LLM. Future experiments can extend to closed-source LLMs such as GPT to further validate the approach.

In conclusion, integrating Swarm Intelligence Algorithms with LLMs has proven to be a powerful method for prompt optimization, offering substantial performance gains with reduced resource consumption. This work lays the foundation for further advancements in efficient LLM optimization techniques.

## ACKNOWLEDGEMENTS

We would like to extend our thanks to Guo Qingyan, who collaborated on the paper (Guo et al., 2023b) with Microsoft and provided essential insights and guidance on the feasibility of extending this project to further ventures.

## REFERENCES

- Alva-Manchego, F., Martin, L., Bordes, A., Scarton, C., Sagot, B., and Specia, L. (2020). Asset: A dataset for tuning and evaluation of sentence simplification models with multiple rewriting transformations. *arXiv preprint arXiv:2005.00481*.
- Bharambe, U., Ramesh, R., Mahato, M., and Chaudhari, S. (2024). Synergies between natural language processing and swarm intelligence optimization: A comprehensive overview. In *Advanced Machine Learning with Evolutionary and Metaheuristic Techniques*, pages 121–151.
- Bharti, V., Biswas, B., and Shukla, K. (2022). Swarm intelligence for deep learning: Concepts, challenges and recent trends. In *Advances in Swarm Intelligence: Variations and Adaptations for Optimization Problems*, pages 37–57. Springer International Publishing.
- Gliwa, B., Mochol, I., Biesek, M., and Wawer, A. (2019). Samsun corpus: A human-annotated dialogue dataset for abstractive summarization. *arXiv preprint arXiv:1911.12237*.
- Guo, Q., Wang, R., Guo, J., Li, B., Song, K., Tan, X., Liu, G., Bian, J., and Yang, Y. (2023a). Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. *arXiv preprint arXiv:2309.08532*.
- Guo, Q., Wang, R., Guo, J., Li, B., Song, K., Tan, X., Liu, G., Bian, J., and Yang, Y. (2023b). Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. *arXiv preprint arXiv:2309.08532*.
- Janga Reddy, M. and Nagesh Kumar, D. (2020). Evolutionary algorithms, swarm intelligence methods, and their applications in water resources engineering: a state-of-the-art review. *H2Open Journal*, 3(1):135–188.
- Lester, B., Al-Rfou, R., and Constant, N. (2021). The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.
- Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., and Neubig, G. (2023). Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35.
- Ma, R., Wang, X., Zhou, X., Li, J., Du, N., Gui, T., Zhang, Q., and Huang, X. (2024). Are large language models good prompt optimizers? *arXiv preprint arXiv:2402.02101*.
- Pang, B. and Lee, L. (2005). Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. *arXiv preprint cs/0506075*.
- Rajwar, K., Deep, K., and Das, S. (2023). An exhaustive review of the metaheuristic algorithms for search and optimization: Taxonomy, applications, and open challenges. *Artificial Intelligence Review*, 56(11):13187–13257.
- Selvaraj, S. and Choi, E. (2021). Swarm intelligence algorithms in text document clustering with various benchmarks. *Sensors*, 21(9):3196.
- Wang, X., Li, C., Wang, Z., Bai, F., Luo, H., Zhang, J., Jovic, N., Xing, E., and Hu, Z. (2023). Promptagent: Strategic planning with language models enables expert-level prompt optimization. *arXiv preprint arXiv:2310.16427*.
- Yang, C., Wang, X., Lu, Y., Liu, H., Le, Q., Zhou, D., and Chen, X. (2023). Large language models as optimizers. *arXiv preprint arXiv:2309.03409*.
- Zhou, Y., Muresanu, A., Han, Z., Paster, K., Pitis, S., Chan, H., and Ba, J. (2022). Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*.