IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154115

PROSE: Prompt Refinement, Optimization and Semantic Evaluation

Chetan Kokate[1], Vedant Khadye[2], Shubham Borate[3], Chaitanya Kokate[4]

Dept. of Artificial Intelligence & Data Science, A. C. Patil College of Engineering, Navi-Mumbai, India[1-3]

Dept. of Computer Engineering, A. C. Patil College of Engineering, Navi-Mumbai, India[4]

Abstract: Prompt optimization has emerged as a critical technique for improving the performance, efficiency, and reliability of Large Language Models (LLMs) without modifying their underlying architecture or parameters. Instead of retraining models, optimized prompts guide the model to generate more accurate, consistent, and context-aware responses. This paper presents a systematic approach to prompt optimization by analyzing prompt structures, refinement strategies, and evaluation techniques. The proposed system focuses on improving response relevance, reducing ambiguity, and minimizing token usage through iterative prompt tuning and rule-based optimization. Experimental observations demonstrate that optimized prompts significantly enhance output quality while reducing computational overhead. The study highlights prompt optimization as a cost-effective and scalable solution for real-world AI applications.

Keywords: Semantic Evaluation, Context-Aware Optimization, Natural Language Processing, Automated Prompt Refinement.

1. INTRODUCTION

Large Language Models (LLMs), especially those based on transformer architectures, have brought a major shift in the field of Natural Language Processing (NLP). These models can understand and generate human-like text by learning from vast datasets, enabling them to perform complex language tasks with high accuracy. An LLM can be represented as a function that generates an output response Y for a given input prompt P:

$$Y = f_\theta(P)$$

where f_θ represents the pre-trained model with fixed parameters, P is the prompt, and Y is the generated response. As a result, LLMs are widely used in applications such as chatbots, code generation, document summarization, question answering, recommendation systems, and decision-support tools. However, their effectiveness largely depends on how users interact with them through prompts. Even slight changes in prompt wording can significantly affect the quality and relevance of the generated response.

Traditionally, improving LLM performance involved fine-tuning or retraining using domain-specific data. Although effective, these methods require high computational resources, labelled datasets, and technical expertise. Prompt optimization provides a more practical alternative by improving the input prompt without modifying the model parameters. The objective is to transform an initial prompt P_0 into an optimized prompt P^*:

$$P_0 \rightarrow P^*$$

The optimized prompt can be selected by maximizing a prompt quality function:

$$P^* = \arg\max_{P'} Q(P')$$

where P' represents a candidate prompt and $Q(P')$ measures its quality based on clarity, relevance, semantic preservation, and token efficiency.

Despite its advantages, prompt optimization is often performed manually through trial and error, making it inconsistent and time-consuming. To address this, the proposed framework analyzes, refines, and evaluates prompts using structured templates and rule-based techniques. In this work, prompt quality can be viewed as a combination of semantic similarity, clarity, and efficiency:

$$Q(P') = \alpha S(P_0, P') + \beta C(P') + \gamma E(P')$$

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154115

where $S(P_0, P')$ measures semantic similarity, $C(P')$ represents clarity, $E(P')$ represents efficiency, and α, β, and γ are weighting factors. Thus, the goal is to improve prompt clarity, preserve meaning, and reduce unnecessary token usage without retraining the model.

II. RELATED WORK

A. Literature Survey Summary

Mixture-of-Experts in Prompt Optimization

The paper "Mixture-of-Experts in Prompt Optimization" introduces a novel approach to automated prompt engineering by extending the Mixture-of-Experts (MoE) paradigm to prompt optimization. Traditional prompt optimization techniques often rely on a single instruction prompt, which restricts the model's ability to generalize across diverse and complex task inputs. To overcome this limitation, the author propose the Mixture-of-Prompts (MoP) framework, which divides the task space into multiple semantically homogeneous regions, each handled by a specialized prompt expert. Each expert prompt consists of both optimized instruction and a set of in-context demonstrations, enabling localized task specialization. The optimization process is performed in two phases: demo assignment using semantic clustering and instruction assignment through region-based joint search. Experimental results show that MoP achieves significant performance improvements—up to 43% on benchmark NLP tasks—compared to single-prompt optimization approaches. This work demonstrates that combining structured prompt decomposition with expert specialization greatly enhances prompt coverage and reliability.

Automatic Prompt Optimization via Heuristic Search

Cui et al. present a comprehensive survey of automatic prompt optimization using heuristic-based search methods. The paper systematically categorizes prompt optimization approaches based on where optimization occurs (soft vs. discrete prompts), what is optimized (instruction, examples, or both), and which search strategies are employed. The authors highlight that manual prompt engineering, although effective in some cases, is highly dependent on human intuition and lacks scalability.

The survey emphasizes heuristic search techniques such as beam search, evolutionary algorithms, bandit algorithms, and Monte Carlo search as efficient alternatives to brute-force prompt exploration. These methods iteratively refine prompts based on performance feedback, enabling automated discovery of high-quality prompts. The work also identifies open challenges such as evaluation instability, generalization across tasks, and optimization cost, motivating the development of more structured and efficient prompt optimization frameworks.

Cost-Aware Prompt Optimization

The CAPO framework addresses one of the most critical limitations of prompt optimization: **high computational and token costs**. Zehle *et al.* propose a discrete prompt optimization algorithm that integrates AutoML concepts, evolutionary strategies, and racing mechanisms to reduce unnecessary evaluations. Unlike earlier methods that focus solely on accuracy, CAPO introduces **multi-objective optimization**, balancing task performance and prompt length.

By penalizing excessive prompt length and eliminating underperforming prompt candidates early, CAPO significantly reduces the number of LLM calls required during optimization. Experimental results demonstrate that CAPO outperforms state-of-the-art prompt optimization methods in most benchmark tasks while consuming fewer tokens. This work highlights the importance of efficiency and cost-awareness in deploying prompt optimization for real-world applications.

Prompt Optimization with Human Feedback

Lin et al. explore **Prompt Optimization with Human Feedback (POHF)**, addressing scenarios where numeric evaluation scores are unavailable or unreliable. Instead of relying on automated metrics or validation datasets, the proposed method collects **pairwise human preference feedback** between responses generated by different prompts. Inspired by dueling bandit theory, the authors introduce the **Automated POHF (APOHF)** algorithm, which efficiently selects prompts for comparison and learns a latent utility function.

The approach significantly reduces the amount of human feedback required while still converging to high-quality prompts. This work is particularly relevant for interactive systems and real-world deployments where user satisfaction is more meaningful than predefined metrics. POHF demonstrates that human-in-the-loop prompt optimization can be both practical and effective for black-box LLMs.

APO-CF (Confusion Matrix Feedback)

Choi proposes a prompt optimization approach based on **confusion matrix feedback (APO-CF)**, targeting efficient refinement in relevance evaluation tasks. Unlike traditional methods that rely on iterative resampling or multi-step

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154115

feedback loops, APO-CF leverages the full distribution of model predictions—both correct and incorrect—through a confusion matrix to guide prompt updates. This enables a **single-step optimization mechanism**, significantly reducing computational overhead and the number of required model evaluations.

The method demonstrates competitive performance in information retrieval scenarios while improving efficiency and scalability. By incorporating both positive and negative prediction signals, APO-CF achieves more balanced prompt refinement compared to error-only feedback approaches. However, the approach is primarily designed for **task-specific settings**, particularly query–passage relevance evaluation, which may limit its generalization to broader NLP tasks and diverse prompt optimization scenarios.

Paper Name	Method	Limitations
Heuristic-based Prompt Optimization Survey	Uses heuristic search algorithms to iteratively refine prompts	Mostly theoretical; lacks practical implementation
CAPO: Cost-Aware Prompt Optimization	Evolutionary + AutoML with multi-objective optimization (cost + performance)	Requires many LLM calls; still computationally expensive
Mixture-of-Prompts (MoP)	Mixture-of-Experts approach with clustering and joint optimization	Complex design; depends on optimal clustering
APO-CF (Confusion Matrix Feedback)	Uses confusion matrix feedback for single-step refinement	Task-specific; limited generalization
APOHF (Human Feedback Optimization)	Uses human preference feedback via dueling bandits	Requires human input; not fully automated

B. Research Gap

Prompt optimization is becoming increasingly important with the growing use of Large Language Models, yet current approaches remain largely manual, unstructured, and inconsistent. Existing techniques such as heuristic-based optimization and Mixture-of-Prompts improve performance but often introduce high computational complexity and lack real-time applicability. Cost-aware methods reduce token usage but may compromise optimization depth, while human feedback-based approaches are effective but slow and not scalable.

Most existing solutions focus on a single aspect of optimization, such as accuracy, cost, or feedback, rather than providing a unified multi-objective framework that balances clarity, efficiency, semantic preservation, and response quality. Additionally, there is a lack of automated systems that can perform real-time adaptive prompt optimization while maintaining low latency and high semantic consistency.

Another important gap is the limited transparency and explainability of existing prompt optimization methods. Many systems generate optimized prompts without clearly showing whether the original meaning was preserved, how token usage changed, or why a particular optimization strategy was selected. Furthermore, several approaches provide limited support for domain-aware prompt refinement, which reduces their effectiveness across different application areas such as coding, education, healthcare, finance, and legal tasks.

There is a clear need for a structured, scalable, and automated prompt optimization framework that can refine prompts efficiently, reduce token usage, preserve meaning, compare multiple optimization strategies, and adapt to different user requirements in real time. Such a framework should seamlessly integrate into real-world AI applications without requiring manual intervention, repeated trial-and-error, or model retraining.

III. PROPOSED SYSTEM

The proposed system introduces **PROSE (Prompt Refinement, Optimization and Semantic Evaluation)**, an automated prompt optimization framework designed to enhance the quality, efficiency, and reliability of prompts used in Large Language Models. Unlike traditional prompt engineering approaches that rely on manual trial-and-error, the proposed system operates through a structured pipeline that refines prompts systematically while preserving their original intent.

The framework processes raw user prompts by analyzing their structure, intent, and linguistic patterns. It applies

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154115

semantic refinement techniques to eliminate ambiguity, improve clarity, and restructure instructions in a more precise and interpretable form. By identifying redundant or vague components in the input, the system ensures that the optimized prompt communicates the task more effectively to the language model.

In addition to semantic refinement, the system incorporates context-aware optimization strategies that adapt the prompt structure based on the nature of the task, such as explanation, summarization, or instruction-based queries. The framework also integrates rule-based optimization techniques to enhance prompt readability and consistency, while an optional LLM-based module further improves prompt quality using advanced language understanding capabilities.

To ensure efficiency, the system performs cost-aware optimization by minimizing token usage without compromising semantic meaning. It evaluates prompts using multiple quality metrics, including clarity, readability, token efficiency, and semantic similarity, ensuring that the optimized output remains faithful to the original intent while being more concise and effective.

All processing stages are integrated into a modular pipeline architecture that supports real-time execution through web interfaces, APIs, and command-line tools. The system generates optimized prompts, alternative prompt variations, and detailed evaluation metrics, enabling users to make informed decisions. By combining semantic analysis, rule-based refinement, and adaptive optimization techniques, the proposed system provides a scalable and efficient solution for improving human–AI interaction without requiring model retraining or manual prompt engineering.

A. Algorithm:

```
Input:
  P0      // raw input prompt
  M       // optimization mode: cost or context
  Gmax    // maximum number of genetic algorithm generations
ε       // convergence threshold

Output:
  P*      // final optimized prompt
  E*      // final evaluation metrics

Begin

  if P0 is empty then
    return Error("Prompt cannot be empty")
  end if


Xp ← Preprocess(P0, M)
   // Normalize input prompt and mode

  Pr ← SemanticRefiner(Xp)
  // Remove ambiguity, redundancy, and weak phrasing

  A ← ContextAnalyzer(Pr)
  // Detect domain, intent, examples, and mode

  D ← DomainTermExpansion(Pr, A)
  // Extract seed terms and related domain-specific terms

  Pe ← Enrich(Pr, A, D)
  // Add goal, context, constraints, examples, and metadata

  R ← RuleBasedOptimizer(P0, M, D)
  // Generate deterministic rule-based optimized prompt

  F ← FeatureExtractor(Pe)
  // Extract token length, complexity, ambiguity, domain, intent, and reasoning features
```

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154115

```
S ← StrategyPredictor(F)
// Predict suitable optimization strategies

ML ← MLGuidedOptimizer(R, M, S, D)
// Generate ML-guided optimized prompt

G0 ← InitializePopulation(Pe, R, ML, S)
// Create initial population for genetic optimization

best_previous ← 0

for g ← 1 to Gmax do

   for each candidate Pi in Gg-1 do
      Fitnessi ← Fitness(Pi, P0, M)
   end for

   best_current ← maximum Fitnessi in Gg-1

   if |best_current - best_previous| < ε then
      break
   end if

   Parents ← SelectTopCandidates(Gg-1)
   // Selection step

   Offspring ← Crossover(Parents)
   // Crossover step

   Mutants ← Mutation(Offspring, S, D)
   // Mutation step

   Gg ← Parents ∪ Offspring ∪ Mutants

   best_previous ← best_current

end for
GA ← candidate Pi in final population Gg with    maximum Fitnessi
// Best genetic algorithm output after convergence or maximum generations

C ← {R, ML, GA}
// Store all candidate optimized prompts

for each Pi in C do
   Ei ← Evaluator(P0, Pi, M)
end for

Pb ← candidate Pi in C with maximum Ei.improvement_score
// Select the best optimized candidate

P* ← PromptFormatter(Pb, M, D)
// Format into Role, Task, Context, Constraints, Output, and Example sections

P* ← CleanPrompt(P*)
// Remove extra spaces, duplicate lines, and formatting artifacts

E* ← Evaluator(P0, P*, M)
// Compute final token count, semantic score, output quality, and improvement score
```

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154115

return P*, E*

End

Mathematical convergence conditions:

$$| F_{best}^{(g)} - F_{best}^{(g-1)} | < \epsilon$$

where:

- $F_{best}^{(g)}$ is the best fitness score at generation g
- $F_{best}^{(g-1)}$ is the best fitness score at the previous generation
- ϵ is the convergence threshold

This means the genetic algorithm stops early if the improvement in the best fitness score becomes very small. If convergence is not reached, the algorithm continues until the maximum number of generations G_{max} is completed. The best candidate selection step is:

$$P_b = \arg \max_{P_i \in C} ImprovementScore(P_i)$$

where:

$$C = \{R, ML, GA\}$$

IV.SYSTEM ARCHITECTURE

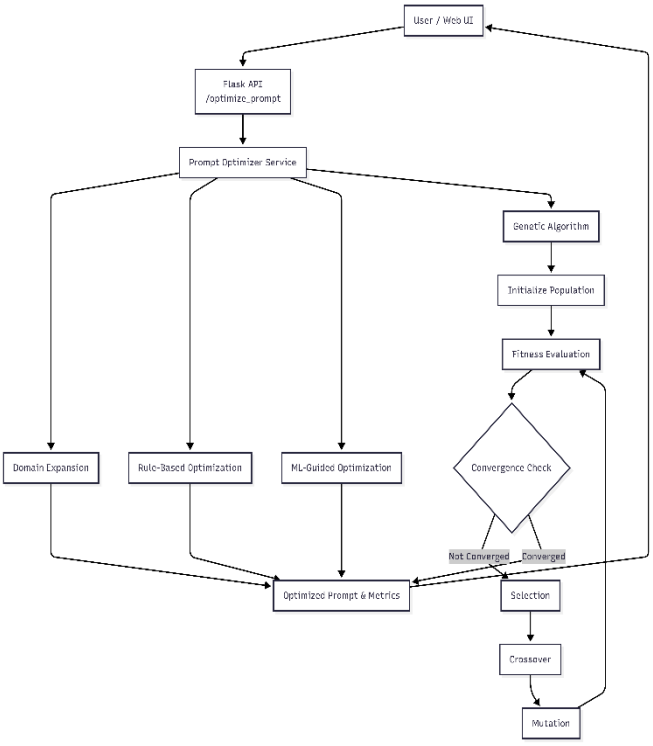

Fig.1. System Architecture

PROSE framework that integrates multiple optimization strategies, including rule-based methods, machine learning, and evolutionary algorithms. The architecture follows a service-oriented approach where each component contributes to refining the input prompt and generating an optimized output along with evaluation metrics.

Key Components

1. User / Web UI

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154115

The User Interface (UI) acts as the entry point of the system, allowing users to input raw prompts. It provides an interactive environment where users can submit queries and receive optimized prompts along with performance metrics. The UI communicates with the backend through API calls.

2. Flask API (/optimize_prompt)

The backend is implemented using a Flask-based REST API, specifically the /optimize_prompt endpoint. This component is responsible for:

- Receiving user input prompts
- Forwarding requests to the optimization service
- Returning optimized results back to the UI

It ensures seamless communication between the frontend and the core optimization engine.

3. Prompt Optimizer Service

This is the core orchestration layer of the system. It manages the execution of different optimization techniques and coordinates their outputs. The service:

- Distributes the input prompt to multiple optimization modules
- Aggregates results from each module
- Selects or combines the best optimized prompt

This modular design improves flexibility and allows easy integration of additional optimization techniques in the future.

4. Domain Expansion

The Domain Expansion module enhances the input prompt by enriching it with context-specific keywords and related terms. This improves semantic coverage and ensures better understanding by downstream models.

5. Genetic Algorithm (GA)

The Genetic Algorithm module applies evolutionary techniques such as:

- **Selection**
- **Crossover**
- **Mutation**

to iteratively generate improved prompt variations. The process begins by creating an initial population of candidate prompts from the refined prompt, rule-based output, and ML-guided output. Each candidate prompt is then evaluated using a fitness function, and the highest-scoring candidates are selected for further evolution.

In the proposed system, the fitness function is designed to support both cost-aware and context-aware optimization. For a candidate prompt P_i, the fitness score is defined as:

$$Fitness(P_i) = \begin{cases} \alpha \cdot \dfrac{1}{T(P_i)} + \beta \cdot Q(P_i), & \text{if mode = cost} \\ \alpha \cdot SR(P_i) + \beta \cdot A(P_0, P_i), & \text{if mode = context} \end{cases}$$

where $T(P_i)$ represents the token count of the candidate prompt, $Q(P_i)$ represents the output quality score, $SR(P_i)$ represents the semantic richness score, and $A(P_0, P_i)$ represents the accuracy or intent preservation score between the original prompt P_0 and the candidate prompt P_i. The constants α and β are weighting factors used to balance the optimization objectives. In this system, $\alpha = 0.55$ and $\beta = 0.45$.

For cost-aware optimization, the fitness function rewards prompt that use fewer tokens while maintaining acceptable output quality. For context-aware optimization, the fitness function rewards prompt that contain richer semantic structure while preserving the original intent. Based on these scores, the GA selects the best-performing candidates, applies crossover to combine useful prompt structures, and performs mutation to introduce variations such as simplification, enrichment, example addition, or domain-specific vocabulary injection. This iterative process ensures progressive optimization and helps the system identify the most effective prompt variation.

6. ML-Guided Optimization

This module leverages machine learning models and NLP techniques to evaluate and refine prompts. It analyses prompt-level features such as token length, complexity, ambiguity, domain, intent, and reasoning requirements to recommend suitable optimization strategies. These strategies help improve the structure, clarity, contextual relevance, and efficiency of prompts. The module focuses on:

- Clarity and readability scoring

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154115

- Semantic similarity
- Context understanding
- Strategy prediction based on prompt features
- Cost-aware prompt analysis

A key evaluation component in this module is semantic similarity, which measures how closely the optimized prompt preserves the meaning of the original prompt. In the proposed system, the original prompt P_0 and the optimized prompt P_i are converted into TF-IDF vector representations.

Semantic similarity is calculated using cosine similarity:

$$Sim(P_0, P_i) = \frac{\overrightarrow{P_0} \cdot \overrightarrow{P_i}}{\| \overrightarrow{P_0} \| \| \overrightarrow{P_i} \|}$$

where $\overrightarrow{P_0}$ represents the TF-IDF vector of the original prompt, $\overrightarrow{P_i}$ represents the TF-IDF vector of the optimized prompt, $\overrightarrow{P_0} \cdot \overrightarrow{P_i}$ denotes the dot product between the two vectors, and $\| \overrightarrow{P_0} \|$ and $\| \overrightarrow{P_i} \|$ denote their vector magnitudes.
The module also supports cost-aware analysis by measuring token efficiency.

Token efficiency compares the token count of the original prompt and the optimized prompt:

$$Eff(P_i) = \frac{T(P_0) - T(P_i)}{T(P_0)}$$

where $T(P_0)$ represents the token count of the original prompt and $T(P_i)$ represents the token count of the optimized prompt. A higher value of $Eff(P_i)$ indicates better token reduction, while a value close to zero indicates similar prompt length.

These semantic similarity and token efficiency measures ensure that the optimized prompt preserves the user's original intent while reducing unnecessary token usage. Since the fitness function has already been defined in the Genetic Algorithm module, these metrics are used as supporting evaluation measures for strategy prediction, prompt refinement, and final candidate comparison. The ML-guided approach therefore provides data-driven improvements by selecting suitable optimization strategies, maintaining semantic consistency, improving clarity, and enhancing the overall quality of prompts.

7. Rule-Based Optimization

The Rule-Based module applies predefined linguistic and structural rules to improve prompts. These include:

- Grammar correction
- Sentence restructuring
- Keyword enhancement

This ensures baseline quality and correctness before advanced optimization techniques are applied.

8. Optimized Prompt & Metrics

All optimization modules feed into a final aggregation layer that produces:

- The optimized prompt
- Associated evaluation metrics (e.g., clarity score, readability, effectiveness)

This output is sent back to the user via the API and displayed in the UI.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154115

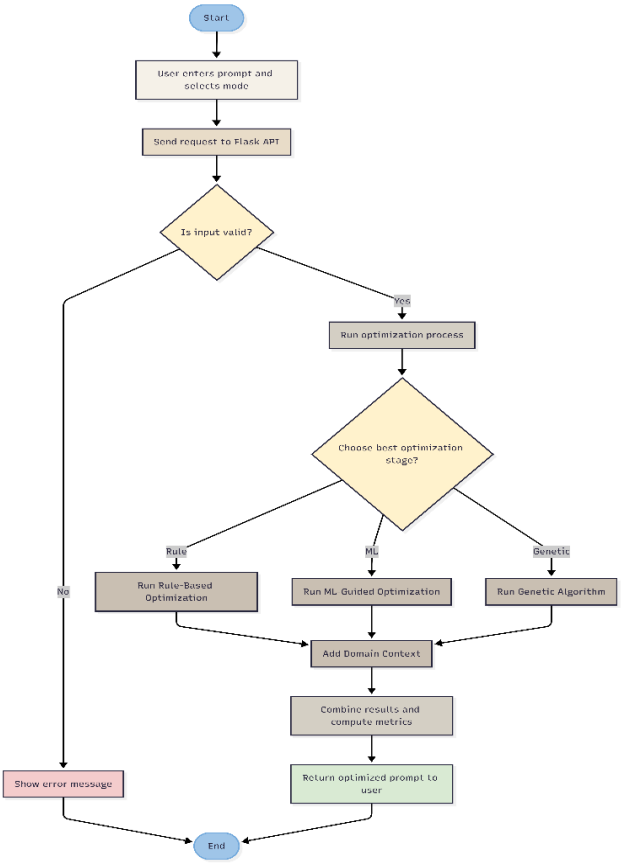

Fig.2. System Workflow

1. Start and User Input

The process begins when the user enters a raw prompt through the interface and selects a preferred optimization mode (Rule-based, ML-based, or Genetic Algorithm). This input serves as the initial data for processing.

2. Request Transmission

The input prompt is sent to the backend via the **Flask API**, which acts as a communication bridge between the user interface and the optimization system.

3. Input Validation

The system checks whether the input is valid (e.g., non-empty, properly formatted).

- **If invalid:** The system generates an error message and terminates the process.
- **If valid:** The workflow proceeds to the optimization stage.

4. Optimization Process Initiation

Once validated, the system initiates the prompt optimization process. At this stage, the system determines the most suitable optimization approach based on the selected mode.

5. Selection of Optimization Technique

A decision node selects one of the following optimization strategies:

Rule-Based Optimization:

Applies predefined linguistic rules such as grammar correction, sentence restructuring, and keyword enhancement.

ML-Guided Optimization:

Uses machine learning models to evaluate prompt quality (e.g., clarity, readability, semantic relevance) and improve it accordingly.

Genetic Algorithm Optimization:

Generates multiple prompt variations and iteratively improves them using evolutionary operations such as mutation, crossover, and selection based on a fitness function.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154115

6. Domain Context Enhancement

Regardless of the chosen method, the system enriches the optimized prompt by adding **domain-specific context**, improving its relevance and effectiveness for downstream tasks.

7. Result Aggregation and Evaluation

The system combines outputs from the optimization process and computes evaluation metrics such as:

- Clarity score
- Readability
- Quality or effectiveness

This ensures that the final prompt meets performance standards.

8. Output Generation

The optimized prompt, along with its evaluation metrics, is returned to the user via the interface.

9. End of Process

The workflow concludes after delivering the output or displaying an error message in case of invalid input.

V. RESULTS AND DISCUSSIONS

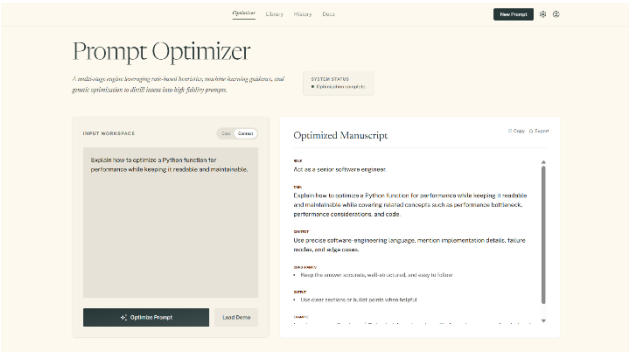

Fig.3. Prompt Optimizer Interface (Context-Aware Mode)

This figure illustrates the PROSE system interface operating in context-aware optimization mode. The input prompt is transformed into a structured and detailed optimized prompt containing role, task, context, constraints, and output sections. The system enhances prompt richness by adding domain-specific details and improving clarity. This mode prioritizes completeness and semantic depth over token efficiency. The interface also highlights the final optimized manuscript as the primary output.

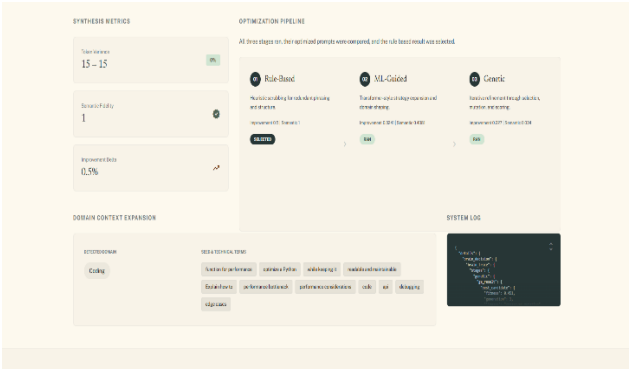

Fig.4. Optimization Dashboard (Context-Aware Results)

This figure illustrates the optimization dashboard for context-aware prompt refinement. The results show a significant increase in token count (15 → 130), reflecting enriched prompt structure and added contextual details. Although semantic fidelity slightly decreases, the improvement score is higher due to enhanced clarity and completeness. The pipeline again selects rule-based optimization as the final stage. This highlights the trade-off between prompt wealth and token efficiency.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154115

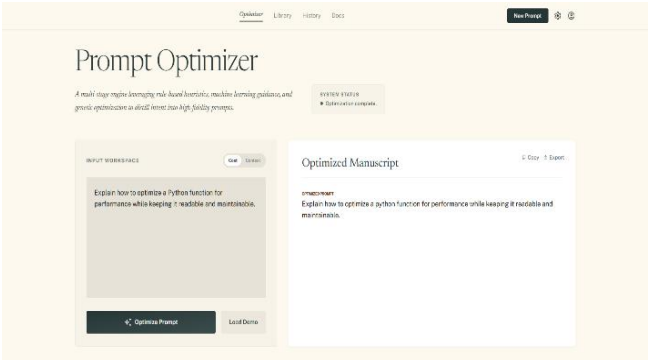

Fig.5. Prompt Optimizer Interface (Cost-Aware Mode)

This figure shows the PROSE system functioning in cost-aware optimization mode. The optimized prompt retains the original intent while minimizing token usage and avoiding unnecessary expansions. Unlike context-aware mode, this approach produces a concise and efficient output. It is particularly useful for reducing computational costs and API usage. The interface demonstrates how prompt simplification is achieved without compromising meaning.

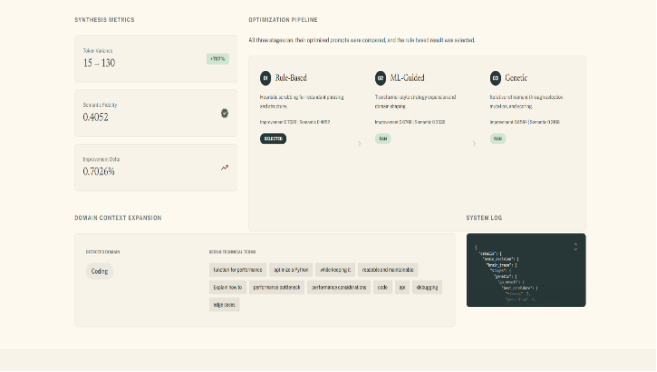

Fig.6. Optimization Dashboard (Cost-Aware Results)

This figure presents the optimization dashboard for cost-aware prompt refinement. The metrics indicate zero token variation (15 → 15), high semantic fidelity (1.0), and minimal improvement delta. The pipeline visualization shows that all three stages executed, with rule-based optimization selected as the final output. This demonstrates the system's ability to maintain efficiency while preserving semantic accuracy. The dashboard also includes domain detection and system logs for transparency.

A. Model Evaluation.

A. Evaluation Metrics

To assess the effectiveness of the proposed PROSE system, multiple quantitative metrics were used to evaluate the quality of optimized prompts. These metrics ensure that the system not only improves prompt structure but also maintains semantic integrity.

Clarity Score: Measures how clearly the prompt conveys the intended task.
Readability Score: Indicates the ease with which the prompt can be understood.
Semantic Similarity (Intent Score): Ensures that the optimized prompt preserves the original meaning.
Token Variation (%): Evaluates efficiency by comparing prompt length before and after optimization.
Processing Time: Measures the time required for prompt optimization.
Improvement Score: Represents overall optimization gain based on system evaluation metrics.

B. Experimental Setup

The evaluation was conducted using real-time prompt optimization outputs generated from the PROSE system interface. Since no standard ready-made dataset was available for the proposed prompt optimization task, a custom prompt evaluation dataset was created specifically for this study. The dataset was manually curated to represent

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154115

common real-world Large Language Model usage scenarios and to evaluate the system across different prompt types, domains, complexity levels, and output requirements.

The dataset consists of **200 prompts** distributed across **8 domains**, with **25 prompts per domain**. The selected domains include General, Education, Coding, Writing, Summarization, Reasoning, Technical/Data, and Business/Finance. The prompts cover multiple task types such as explanation, summarization, rewriting, code generation, debugging, reasoning, data analysis, business planning, and content generation.

Each prompt was annotated with metadata fields such as prompt ID, raw prompt, domain, task type, expected optimization mode, complexity level, ambiguity level, expected output style, and notes. The expected mode field indicates the recommended mode for a prompt; however, for experimental comparison, every prompt was tested under both optimization modes.

The system was tested under two configurations:

- **Cost-Aware Optimization Mode:** Focuses on minimizing token usage while preserving the original meaning.
- **Context-Aware Optimization Mode:** Focuses on enriching prompt structure, adding relevant context, and improving response completeness.

Since each of the 200 prompts was evaluated under both configurations, the total number of evaluation runs was:

$$200 \times 2 = 400$$

where 200 represents the total number of prompts and 2 represents the two optimization modes. This setup enabled a comparative analysis of how the same prompt behaves under cost-focused and context-focused optimization objectives.

The implementation was carried out using a Flask-based modular pipeline architecture with integrated evaluation metrics and pipeline visualization. Each prompt was evaluated before and after optimization using system-generated metrics such as token variation, semantic fidelity, output quality, and improvement score.

The dataset details are shown in Table 1.

Parameter	**Description**
Dataset Type	Custom prompt evaluation dataset
Dataset Source	Manually curated prompts based on common LLM usage scenarios
Number of Prompts	200
Number of Domains	8
Prompts per Domain	25
Total Evaluation Runs	400
Evaluation Interface	PROSE web interface
Optimization Modes	Cost-Aware and Context-Aware
Metadata Fields	Prompt ID, raw prompt, domain, task type, expected mode, complexity level, ambiguity level, expected output style, notes
Evaluation Metrics	Token variation, semantic fidelity, output quality, improvement score

The domain-wise distribution of the dataset is shown in Table 2.

Domain	**Number of Prompts**	**Example Task Types**
General	25	Explanation, comparison, short notes, general analysis
Education	25	Concept explanation, tutoring, revision notes, lesson planning
Coding	25	Code generation, debugging, optimization, API design

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154115

Domain	Number of Prompts	Example Task Types
Writing	25	Rewriting, email writing, academic writing, content generation
Summarization	25	Article summaries, meeting summaries, report summaries, TLDR generation
Reasoning	25	Problem solving, decision analysis, cause-effect analysis, trade-off reasoning
Technical/Data	25	Data analysis, ETL, dashboard metrics, ML evaluation, data quality
Business/Finance	25	Business proposals, financial explanation, SWOT analysis, pricing strategy
Total	**200**	—

C. Results and Analysis

The PROSE system demonstrated distinct optimization behavior under cost-aware and context-aware modes. The system interface results (Fig. 3 and Fig. 5) highlight how optimization strategies affect prompt structure, token usage, and semantic richness.

Table 1: Performance Comparison (Cost vs Context Optimization)

Metric	Cost-Aware Optimization	Context-Aware Optimization
Token Count	15 → 15	15 → 130
Token Variation (%)	0%	+767%
Semantic Fidelity	1.00	0.4052
Improvement Score	0.5%	0.7026
Processing Time	< 1 sec	< 1 sec

- **Metric Improvement. (Bar Graph)**

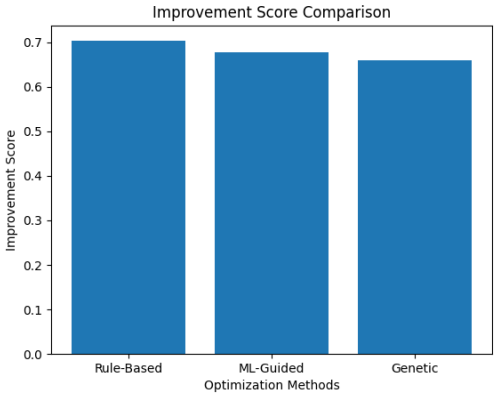

Fig.6. Optimization Methods

Observation:
Context-aware optimization significantly increases prompt richness and structural depth.
Cost-aware optimization maintains minimal token usage while preserving original meaning.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154115

- **Token Reduction. (Pie Chart)**

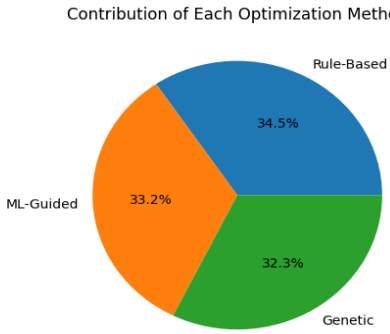

Fig. 7. Contribution of each method

Observation:

Cost-aware mode results in no token increase (0%), making it highly efficient.

Context-aware mode increases tokens significantly (+767%) to enhance contextual completeness and instruction clarity.

- **Optimization Pipeline Comparison**

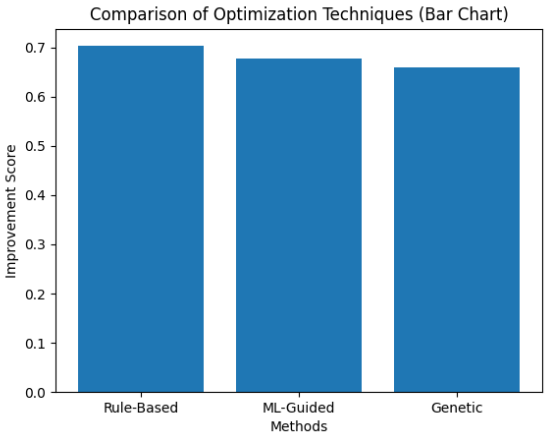

Fig.7. Comparison between approaches

Method	Improvement	Semantic Score	Status
Rule-Based	0.7026	0.4052	Selected
ML-Guided	0.6768	0.3536	Ran
Genetic	0.6584	0.3168	Ran

Observation:

Rule-based optimization was selected as the final stage due to higher semantic alignment and structured output.
ML-guided and genetic approaches contributed to refinement but were not selected as optimal.

D. Comparative Analysis

Compared to traditional prompt optimization approaches, the updated PROSE system:

- Supports **dual optimization modes (Cost vs Context aware)**
- Provides **dynamic trade-off between efficiency and richness**

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026
DOI: 10.17148/IJARCCE.2026.154115

- Enables **real-time pipeline evaluation and selection**
- Maintains **high semantic fidelity in cost mode**
- Enhances **prompt completeness in context mode**

Unlike earlier approaches, the system does not rely on a single optimization objective but dynamically adapts based on user-selected mode.

E. Observations

- Cost-aware optimization ensures **maximum efficiency with zero token increase**
- Context-aware optimization significantly improves **prompt structure and completeness**
- Rule-based stage often provides the **most stable and interpretable output**
- Semantic fidelity remains high in cost mode but slightly reduced in context expansion
- System operates efficiently in **real-time (<1 sec)**

Statistical Validation

To further validate the effectiveness of the proposed PROSE framework, statistical analysis was performed on key evaluation metrics across multiple prompt optimization runs. The analysis focuses on the mean and standard deviation of performance indicators to assess consistency and reliability.

Mean Analysis

The mean values of evaluation metrics indicate the overall performance of the system across different optimization modes.

- The **mean semantic fidelity** in cost-aware mode remains close to **1.0**, indicating strong preservation of original intent.
- The **mean improvement score** is higher in context-aware mode, reflecting enhanced prompt structure and completeness.
- The **mean token variation** remains minimal in cost-aware mode, confirming high efficiency, while it increases significantly in context-aware mode due to added contextual richness.

These results demonstrate that the system effectively balances efficiency and performance depending on the selected optimization mode.

Standard Deviation Analysis

The standard deviation is used to measure the variability and consistency of the optimization results.

- A **low standard deviation in semantic fidelity (cost mode)** indicates consistent preservation of meaning across different prompts.
- A **moderate standard deviation in context-aware mode** reflects variability due to dynamic context expansion and structural enrichment.
- The **low variation in processing time** confirms stable real-time performance of the system.

Overall, the low standard deviation values in key metrics indicate that the proposed system produces **reliable and consistent optimization results**.

The statistical consistency of the system confirms its robustness and suitability for real-world deployment, as it maintains stable performance across varying prompt inputs.

F. Limitations

1. Context-aware optimization may lead to significant token expansion, increasing computational cost.
Semantic score may slightly decrease when excessive contextual details are introduced.
2. Rule-based optimization relies on predefined heuristics, which may limit adaptability to diverse prompt types.
3. Genetic and ML-guided optimization stages introduce additional computational complexity.
4. Evaluation metrics may not fully capture qualitative improvements in prompt richness and contextual depth.
5. The system exhibits **domain dependency**, where optimization performance may vary across different application domains due to limited domain-specific knowledge.
6. The optimization process may lead to **overfitting to specific prompt patterns**, reducing generalization to unseen or diverse prompt structures.
7. Repeated optimization strategies may introduce **pattern bias**, where prompts follow similar structural templates, limiting creativity and diversity.
8. The system may inherit **LLM bias**, as optimization decisions influenced by language models can reflect underlying biases present in training data.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154115

CONCLUSION

This paper introduced **PROSE (Prompt Refinement, Optimization and Semantic Evaluation)**, a structured and automated framework for improving prompt quality in Large Language Models. The proposed system moves beyond conventional trial-and-error prompt engineering by implementing a systematic optimization pipeline that integrates semantic refinement, rule-based structuring, and adaptive optimization strategies.

The key contribution of this work lies in the design of a **hybrid, multi-stage optimization framework** capable of enhancing prompt clarity, reducing ambiguity, and balancing efficiency with contextual richness. By supporting both cost-aware and context-aware optimization modes, the system enables flexible adaptation based on user requirements, offering a practical trade-off between token efficiency and semantic completeness.

The results demonstrate that the framework effectively improves prompt usability while maintaining high semantic fidelity and low processing latency. This highlights its suitability for real-time applications and interactive AI systems. Furthermore, the modular architecture ensures scalability and allows seamless integration with existing AI pipelines.
Overall, PROSE provides a **scalable, efficient, and application-oriented solution** for automated prompt optimization. The framework contributes to advancing human–AI interaction by reducing reliance on manual prompt design and enabling more reliable and structured communication with language models.

Future work will focus on enhancing the optimization capability of the PROSE framework by incorporating **adaptive learning mechanisms**, enabling the system to continuously improve prompt refinement strategies based on past optimization outcomes and user interactions. This will allow the framework to dynamically adjust its optimization approach for different task types and domains.

Additionally, the integration of **reinforcement-based (RC-based) optimization techniques** can further improve decision-making within the pipeline. By introducing reward-driven learning, the system can iteratively refine prompts based on performance feedback, leading to more optimal and context-aware prompt generation over time.

Further improvements can be achieved through deeper **LLM integration**, where advanced language models are more tightly coupled with the optimization pipeline for enhanced semantic understanding, reasoning, and context generation. This would enable the system to handle more complex, ambiguous, and domain-specific queries with higher accuracy.

Moreover, expanding evaluation metrics and incorporating domain-specific prompt tuning strategies will improve the robustness and generalization capability of the system across diverse real-world applications.

ACKNOWLEDGMENT

I would like to express my sincere gratitude to my guide, Prof. (Mrs.) S. P. Bansu, for her valuable guidance, constant support, and insightful feedback throughout this project. I also thank the Head of the Department, which is also our guide Prof. S. P. Bansu, and the faculty of the AI-DS department at A.C. Patil College of Engineering for providing the necessary resources and a supportive learning environment. I am grateful to my friends and peers for their encouragement and cooperation, and to my family for their continuous motivation and unwavering support. This project has been a valuable learning experience, and I am thankful to everyone who contributed to its successful completion.

REFERENCES

[1] X. Lin, Z. Dai, A. Verma, S.-K. Ng, P. Jaillet, and B. K. H. Low, "Prompt Optimization with Human Feedback," *arXiv preprint arXiv:2405.17346*, 2024. [Online]. Available: https://arxiv.org/abs/2405.17346

[2] W. Cui, J. Zhang, Z. Li, H. Sun, D. Lopez, K. Das, B. Malin, and S. Kumar, "Automatic Prompt Optimization via Heuristic Search: A Survey," *arXiv preprint arXiv:2502.18746*, 2025. [Online]. Available: https://arxiv.org/abs/2502.18746

[3] J. Choi, "Efficient Prompt Optimization for Relevance Evaluation via LLM-Based Confusion Matrix Feedback," *Applied Sciences*, vol. 15, no. 9, p. 5198, 2025. [Online]. Available: https://doi.org/10.3390/app15095198

[4] E. Akadal, "Reducing Artificial Intelligence Costs in Business through Prompt Optimization," *International Journal of Management and Data Analytics*, vol. 5, no. 1, pp. 111–123, 2025. [Online]. Available: https://doi.org/10.5281/zenodo.15331028

[5] T. Zehle, M. Schlager, T. Heiß, and M. Feurer, "CAPO: Cost-Aware Prompt Optimization," *AutoML Conference*, 2025. [Online]. Available: https://github.com/finitearth/capo

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026
DOI: 10.17148/IJARCCE.2026.154115

[6] A. Sabbatella, A. Ponti, I. Giordani, A. Candelieri, and F. Archetti, "Prompt Optimization in Large Language Models," *Mathematics*, vol. 12, no. 6, p. 929, 2024. [Online]. Available: https://doi.org/10.3390/math12060929

[7] R. Pryzant, D. Iter, J. Li, Y. T. Lee, C. Zhu, and M. Zeng, "Automatic Prompt Optimization with 'Gradient Descent' and Beam Search," *arXiv preprint arXiv:2305.03495*, 2023. [Online]. Available: https://arxiv.org/abs/2305.03495

[8] Z. Wu, X. Lin, Z. Dai, W. Hu, Y. Shu, S.-K. Ng, P. Jaillet, and B. K. H. Low, "Prompt Optimization with EASE: Efficient Ordering-aware Automated Selection of Exemplars," in *Proc. NeurIPS*, 2024.

[9] Anonymous, "Mixture-of-Experts in Prompt Optimization," *ICLR Conference Submission*, 2024. [Online]. Available: https://arxiv.org/abs (under review).

[10] W. Cui, Z. Li, H. Sun, D. Lopez, K. Das, B. Malin, S. Kumar, and J. Zhang, "A Survey of Automatic Prompt Optimization with Instruction-focused Heuristic-based Search Algorithm," *arXiv preprint arXiv:2502.18746*, 2025. [Online]. Available: https://arxiv.org/abs/2502.18746

