



PromptIQ: An AI-Assisted Prompt Analysis and Refinement System Using Chrome Extension, JavaScript and Rule-Based Scoring

Atharv Khumkar¹, Khushi Kathiwale², Anuj Ramteke³, Sanchit Dongre⁴, Prof. Priydarshani Gotmare⁵

¹²³⁴Dept. of CSE(DS) Guru Nanak Institute Of Engineering And Technology Nagpur, Maharashtra, India

⁵Head Of Dept CSE(DS) Guru Nanak Institute Of Engineering And Technology Nagpur, Maharashtra, India

How to Cite this Article:

Khumkar, A., Kathiwale, K., Ramteke, A. & Dongre, S. (2026). PromptIQ: An AI-Assisted Prompt Analysis and Refinement System Using Chrome Extension, JavaScript and Rule-Based Scoring. International Journal of Creative and Open Research in Engineering and Management, <i>10(10)</i>, 1-9. <https://doi.org/10.55041/ijcope.v2i10.068>

License:

This article is published under the terms of the Creative Commons Attribution 4.0 International License (CC BY 4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author(s) and the source are credited.

© The Author(s). Published by International Journal of Creative and Open Research in Engineering and Management.



<https://doi.org/10.55041/ijcope.v2i10.068>

Abstract — Large language models such as ChatGPT, Gemini and Claude are increasingly used for education, software development, data analysis, content creation and decision support. The quality and usefulness of their responses are strongly influenced by how users formulate prompts. However, many users do not have a systematic way to judge whether a prompt is clear, contextual, specific and efficient before submitting it. This paper presents PromptIQ, a browser-based prompt analysis and refinement system designed to provide immediate feedback while a user is interacting with supported AI platforms. The proposed system uses a Chrome extension to detect prompt text through a content script and pass it to a local JavaScript scoring engine. The current scoring model evaluates four dimensions: clarity, context, specificity and efficiency, combined into an overall score on a 0-100 scale. Functional testing of the implemented modules is used to evaluate the reliability of the system. The proposed system demonstrates how browser-based and rule-based technologies can be combined to develop an accessible, transparent prompt-quality feedback platform, with a future version expected to integrate an external language model for explanation and automatic prompt refinement.

Keywords— Prompt Engineering, Large Language Models, Prompt Analysis, Prompt Optimization, Chrome Extension, Natural Language Processing, Generative AI.

I. INTRODUCTION

Large Language Models (LLMs) have changed the way users interact with software systems. Instead of using only fixed menus or commands, users can describe a task in natural language and ask an LLM to generate an answer, explanation, code, summary or plan. Prompt engineering studies how these instructions can be designed to guide model behaviour, and recent surveys show a wide range of prompting techniques, including zero-shot, few-shot, reasoning-oriented, retrieval-based, optimization and self-reflection approaches [1].

A practical problem remains at the user interface level: users often submit prompts without knowing whether the instruction contains enough information. A short prompt may be ambiguous, while a long prompt may contain unnecessary repetition. A user may also omit the intended output format, relevant domain context, constraints or the expected

audience. Existing prompt-engineering research provides many methods for improving model performance, but these methods are generally studied as prompting strategies rather than as a simple end-user quality checker.

PromptIQ addresses this practical gap by acting as a lightweight browser extension. When a user types a prompt into a supported AI website, the extension identifies the prompt text and analyzes it using an interpretable, rule-based scoring engine. The current prototype does not require an external AI API for scoring, which makes the first stage fast, explainable and suitable for local testing. A later stage can use an LLM to explain weaknesses and produce an improved version of the prompt. The main objective of the system is to give users a simple, transparent signal about prompt quality before they submit it, and to provide a foundation for AI-assisted prompt refinement.

II. LITERATURE REVIEW

Previous studies have explored different approaches for improving the design, evaluation and automatic construction of prompts for large language models. These studies highlight the importance of clear, specific and well-structured instructions, while also showing that most existing work is framed as a prompting strategy for the model rather than as a pre-submission quality checker for the end user.

[1] Sahoo et al. (2024) presented a systematic survey categorizing over 41 prompt engineering techniques by application area, covering reasoning, hallucination reduction, optimization and user-intent understanding. The survey demonstrated that prompt engineering is a broad research field with methods spanning zero-shot and few-shot prompting to advanced reasoning and optimization strategies.

[2] Liu et al. (2023) reviewed prompting methods in natural language processing, showing that the choice and structure of a prompt can substantially change model performance without changing the underlying model parameters, motivating the need for tools that help users construct better prompts.

[3] Wei et al. (2022) introduced Chain-of-Thought prompting, showing that suitable step-by-step



demonstrations can significantly improve complex reasoning performance, which highlights specificity and structure as important prompt-quality factors.

[4] Zhou et al. (2022) proposed Automatic Prompt Engineer (APE), an approach that automatically generates and selects effective instructions, indicating that prompt quality can be estimated and optimized in a structured, repeatable way.

[5] Madaan et al. (2023) proposed Self-Refine, an iterative refinement approach where a model critiques and improves its own output, suggesting a natural extension in which a locally scored prompt can be improved through a similar feedback loop.

[6] Deng et al. (2023) proposed Rephrase and Respond (RaR), showing that allowing a model to rephrase an ambiguous question before answering improves comprehension and response accuracy, supporting the idea that clarity and rephrasing are measurable prompt-quality dimensions.

[7] Yang et al. (2023) proposed Optimization by Prompting (OPRO), using LLMs as optimizers to iteratively improve prompts for accuracy, reinforcing that prompt quality can be treated as an optimizable, quantifiable property.

Based on these studies, PromptIQ does not attempt to replace advanced prompt-optimization research. Instead, it focuses on a lightweight, interpretable, pre-submission analysis layer that estimates clarity, context, specificity and efficiency locally in the browser, and that can later be connected to LLM-based rephrasing and refinement techniques such as those discussed above.

III. PROPOSED METHODOLOGY

The proposed system, PromptIQ, is a browser-based extension designed to simplify the process of judging and improving prompt quality before submission to an AI platform. The system analyzes the text a user types into a supported AI website and returns four interpretable component scores together with an overall quality score. The main purpose of the system is to give users a simple, transparent signal about their prompt so that they can revise it before submitting it to an LLM.

A. Overall System Architecture

PromptIQ follows a client-side, modular architecture. A content script runs on the target AI webpage and detects text entered into standard text areas or content-editable elements. Once new text is detected, the extension passes the prompt to a local JavaScript scoring module. The scoring module performs lexical and rule-based checks and returns a structured object containing the four component scores and

the overall score, which is displayed to the user directly in the browser.

A. OVERALL SYSTEM ARCHITECTURE

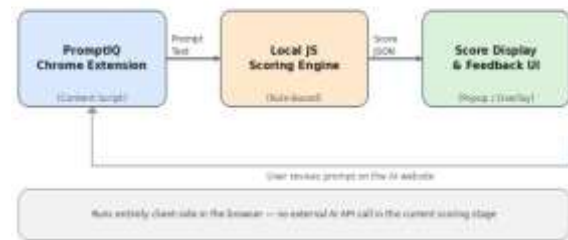


Fig. 1. Overall PromptIQ system architecture.

B. System Workflow

The workflow of the proposed system begins when a user types a prompt into a supported AI website. The content script detects the updated text and sends it to the scoring engine. The scoring engine normalizes the text and evaluates it against the clarity, context, specificity and efficiency rules. The four scores are combined into an overall score, which is returned to the content script and displayed to the user as immediate feedback, before the prompt is submitted to the underlying AI platform.

B. SYSTEM WORKFLOW



Fig. 2. PromptIQ processing workflow.

C. Rule-Based Scoring Module

The scoring module is intentionally deterministic and transparent rather than model-based. Clarity is estimated from action verbs, vague terms, prompt length and sentence structure. Context is estimated from domain, object and purpose terms. Specificity is estimated from technology names, numbers, output-format terms and constraints. Efficiency is estimated from repeated words, filler terms and overall prompt length. The four component scores are combined using an equal-weight arithmetic mean, rounded and constrained to a 0-100 range.



C. RULE-BASED SCORING MODULE



Fig. 3. Rule-based prompt scoring model.

D. Future LLM-Based Refinement Module

The architecture is designed so that a future module can send the prompt and its component scores to an external LLM API, which can generate a natural-language explanation of the weaknesses and an improved version of the prompt. The refined prompt can then be scored again locally, creating an improvement loop that combines the transparency of the local rules with the semantic capability of an LLM.

IV. SYSTEM IMPLEMENTATION

The proposed PromptIQ system is implemented as a Chrome browser extension to provide users with immediate, in-context feedback on prompt quality. The extension integrates detection, scoring and display into a single lightweight client-side application.

A. Chrome Extension Development

The PromptIQ extension is developed using standard Chrome extension APIs (manifest, background script and content script) together with JavaScript. The extension provides a simple, unobtrusive interface through which users can view their prompt-quality score while typing on a supported AI website.

B. Content Script and Prompt Detection

A content script is injected into the target AI webpage and monitors standard text areas and content-editable input elements. Whenever the prompt text changes, the content script captures the current text and forwards it to the scoring engine for evaluation.

C. JavaScript Scoring Engine

The scoring engine performs lexical analysis of the captured prompt text, checking for action verbs, vague words, filler words, domain terms, numerical constraints, output-format terms and repeated terms. Each check contributes to one of the four component scores, without requiring any external API call, keeping the scoring stage fast and fully local.

D. Overall Score Calculation

The four component scores (clarity, context, specificity and efficiency) are combined using an equal-weight arithmetic mean, as summarized in Table I. Equal weighting is used in the prototype because it is simple and interpretable; future validation can test whether different weights better match expert or user judgments.

TABLE I. PROMPTIQ SCORING DIMENSIONS

Dimension	Key Indicators	Effect
Clarity	Action verbs, vague terms, length	Higher with action verbs; lower with vague words
Context	Domain terms, purpose terms	Higher when domain/purpose stated
Specificity	Tech names, numbers, format terms	Higher with explicit format/constraints
Efficiency	Repeats, filler terms, length	Lower with repetition/filler

E. System Features

- Real-time detection of prompt text on supported AI websites
- Local, interpretable scoring of clarity, context, specificity and efficiency
- Combined overall prompt-quality score (0-100 scale)
- No dependency on an external AI API for the current scoring stage
- Modular design allowing future integration with an LLM-based refinement module

V. RESULTS AND DISCUSSION

The PromptIQ prototype was evaluated using example prompts to check the behaviour of the rule-based scoring engine. For the prompt "Create a Power BI dashboard for sales analysis," the engine identifies an action verb ("create"), domain terms ("Power BI," "dashboard," "sales," "analysis") and an intended output type ("dashboard"). The rule engine therefore has evidence for clarity, context and specificity, and because the prompt is concise with little repetition, efficiency is also expected to remain relatively high. The exact numerical score depends on the implemented rule set and should be treated as a heuristic indicator rather than a scientifically validated measure of prompt quality.

The major advantage of the current approach is interpretability: a user or developer can inspect why a score changed because the rules are explicit, and the system is lightweight because scoring does not require an API call. However, rule-based analysis cannot fully understand semantics, user intent, factual correctness or task-specific quality, so a prompt can contain many domain keywords and



still be poorly designed. To validate PromptIQ further, a planned evaluation will compare system scores with human judgments across prompts from education, programming, data analysis, writing and general question-answering domains, using multiple independent reviewers and correlation-based agreement metrics.

Research on automatic prompt generation and iterative refinement suggests a natural extension of the current results: a future PromptIQ version can first identify weaknesses locally and then ask an LLM to produce an improved prompt, which can be scored again to create an improvement loop combining the transparency of local rules with the semantic capabilities of an LLM.

VI. CONCLUSION

The proposed PromptIQ system provides a simple and convenient solution for judging and improving prompt quality before submission to a large language model. The extension detects prompts directly in the browser and evaluates them using four interpretable dimensions: clarity, context, specificity and efficiency, combined into a single overall score.

A major feature of the system is its fully local, JavaScript-based scoring engine, which provides fast, transparent feedback without depending on an external AI API. This modular design also provides a foundation for a future AI-assisted refinement feature.

The system is developed using Chrome extension APIs, JavaScript and a rule-based scoring engine, providing effective, real-time feedback within the user's existing AI-chat workflow. Overall, PromptIQ offers an accessible, user-friendly bridge between prompt-engineering research and everyday use of large language models. In the future, the system can be enhanced with a visual score panel, an AI-powered "Improve Prompt" feature, before-and-after score comparison, domain-specific scoring and human validation studies.

ACKNOWLEDGMENT

We would like to express our sincere gratitude to [Institute Name] for providing us with the opportunity and resources to complete this project. We would also like to thank the faculty members of the Department of Computer Science and Engineering for their valuable guidance, support and encouragement throughout the development of the PromptIQ project. Finally, we would like to thank our team members and everyone who directly or indirectly contributed to the successful completion of this project.

REFERENCES

- [1] P. Sahoo, A. K. Singh, S. Saha, V. Jain, S. Mondal, and A. Chadha, "A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications," arXiv:2402.07927v2, 2025.
- [2] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing," *ACM Computing Surveys*, vol. 55, no. 9, 2023.
- [3] J. Wei, X. Wang, D. Schuurmans, B. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24824-24837, 2022.
- [4] Y. Zhou, A. I. Muresanu, Z. Han, K. Paster, S. Pitis, H. Chan, and J. Ba, "Large Language Models Are Human-Level Prompt Engineers," arXiv:2211.01910, 2022.
- [5] A. Madaan et al., "Self-Refine: Iterative Refinement with Self-Feedback," arXiv:2303.17651, 2023.
- [6] Y. Deng, W. Zhang, Z. Chen, and Q. Gu, "Rephrase and Respond: Let Large Language Models Ask Better Questions for Themselves," arXiv:2311.04205, 2023.
- [7] C. Yang, X. Wang, Y. Lu, H. Liu, Q. V. Le, D. Zhou, and X. Chen, "Large Language Models as Optimizers," arXiv:2309.03409, 2023.
- [8] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafraan, K. Narasimhan, and Y. Cao, "ReAct: Synergizing Reasoning and Acting in Language Models," *International Conference on Learning Representations*, 2023.
- [9] P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459-9474, 2020.
- [10] T. B. Brown et al., "Language Models are Few-Shot Learners," *Advances in Neural Information Processing Systems*, vol. 33, 2020.
- [11] A. Radford et al., "Language Models are Unsupervised Multitask Learners," *OpenAI Technical Report/Blog*, 2019.
- [12] S. Diao, P. Wang, Y. Lin, and T. Zhang, "Active Prompting with Chain-of-Thought for Large Language Models," arXiv:2302.12246, 2023.
- [13] X. Wang et al., "Self-Consistency Improves Chain of Thought Reasoning in Language Models," arXiv:2203.11171, 2022.
- [14] H. S. Zheng et al., "Take a Step Back: Evoking Reasoning via Abstraction in Large Language Models," arXiv:2310.06117, 2023.