

PASTA: Direct Instruction and Summary-Mediated Prompting in LLM-Assisted Code Modification

Ningzhi Tang*, Emory Smith*, Yu Huang[†], Collin McMillan*, Toby Jia-Jun Li*

*{ntang, esmith36, cmc, toby.j.li}@nd.edu, [†]yu.huang@vanderbilt.edu

*University of Notre Dame, Notre Dame, IN, USA [†]Vanderbilt University, Nashville, TN, USA

Abstract—We present PASTA, a JetBrains IDE plugin that supports LLM-assisted code modification through two prompting techniques: (1) **Direct Instruction Prompting**, where developers describe changes explicitly in free-form language, and (2) **Summary-Mediated Prompting**, where changes are made by editing generated summaries of the code. PASTA enables structure-grounded prompting through editable summaries, helping developers specify intent, comprehend code, and control modifications, addressing key challenges in LLM-assisted programming. This demo aims to provoke discussion around new interaction representations for code editing with LLMs, and invites the VL/HCC community to engage with hands-on scenarios that bridge natural language, program structure, and developer intent.

Index Terms—code modification, AI-assisted programming, prompting strategies, summary-mediated interaction

I. INTRODUCTION

Large language models (LLMs) are widely used for code modification, enabling developers to edit existing code using natural language prompts [1]–[5]. However, crafting effective prompts remains challenging due to the inherent open-endedness of natural language, uncertainties in LLM predictions, and the often vague or evolving nature of developer intentions [6]–[8].

A promising approach to easing prompt construction is to scaffold it through editable natural language summaries of existing code, which serve as intermediate representations [9]–[11]. To explore this idea, we designed PASTA¹, a system that supports two prompting techniques for LLM-assisted code modification in general programming languages. As illustrated in Fig. 1, these techniques are:

- **Direct Instruction Prompting:** Freely writing natural language instructions to describe desired changes.
- **Summary-Mediated Prompting:** Editing a natural language summary of existing code to specify new behavior.

PASTA is a study-enabling system prototype developed for our full research paper². It is implemented as a plugin compatible with all JetBrains IDEs (e.g., PyCharm, WebStorm). We will demonstrate PASTA live on Python and JavaScript tasks within JetBrains IDEs and invite attendees to explore it hands-on, reflecting on design opportunities for future AI-assisted development tools. In addition, we will present a poster showcasing our work on leveraging natural language representations for LLM-based programming.

¹Prompt-Assisted Software TrAnsformation



²Tang *et al.*, Exploring Direct Instruction and Summary-Mediated Prompting in LLM-Assisted Code Modification, accepted at VL/HCC 2025.


II. PASTA SYSTEM

The design of PASTA draws on modern LLM coding workflows (e.g., Cursor³, Windsurf⁴), adopting a selection-based prompting interface within the IDE to reflect real-world development practices and integrating full-file context to support code understanding and generation. PASTA is well-documented and open-sourced at <https://github.com/TTangNingzhi/PASTA> to support reproducibility.

A. Interface

Fig. 2 shows the interface of PASTA. Developers begin by selecting a region of code in the editor using the mouse (left panel), then specify their intended modification using one of two prompting methods (right panel).

In summary-mediated prompting, clicking  Retrieve Summary generates a 1-3 sentence natural language description of the selected code. The summary adapts to the length and complexity of the code. The generated summary appears in the top text field and can be freely edited to express the intended change. Developers can click  Diff Summaries to highlight **insertions** and **deletions**, aiding summary revision. In direct instruction prompting, developers write a natural language command in the bottom text field to specify the desired modification directly.

Clicking  Commit Prompt in either mode submits the selected code, the file context, and either the edited summary or the free-form instruction to the LLM. The returned code is shown in a diff view with line- and token-level highlights, enabling developers to inspect, validate, and selectively accept changes. A loading indicator signals that the LLM is processing the prompt, providing users with timely visual feedback.

B. Implementation

PASTA is implemented as a JetBrains IDE plugin using the official IntelliJ SDK⁵. The interface is integrated as a tool window panel, with functional buttons implemented via the `AnAction` interface. The summary diff feature is supported by the Java Diff Utils library⁶, while code diffing leverages the default diff package provided by the SDK. LLM-powered functionality is built on OpenAI’s GPT-4o⁷ chat completions

³<https://cursor.com/>

⁴<https://windsurf.com/>

⁵<https://plugins.jetbrains.com/docs/intellij/welcome.html>

⁶<https://java-diff-utils.github.io/java-diff-utils/>

⁷<https://openai.com/index/hello-gpt-4o/>

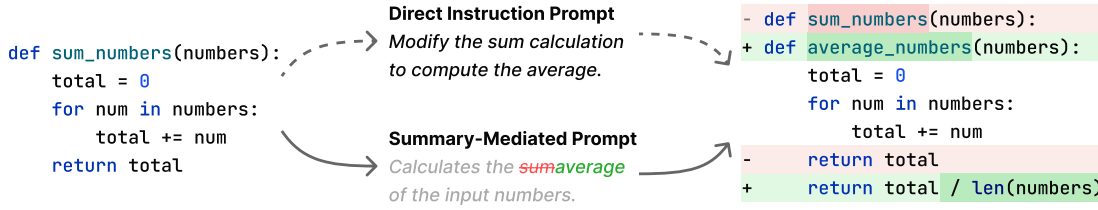


Fig. 1. An illustrative example of direct instruction (dashed arrows) and summary-mediated prompting (solid arrows) for LLM-assisted code modification.

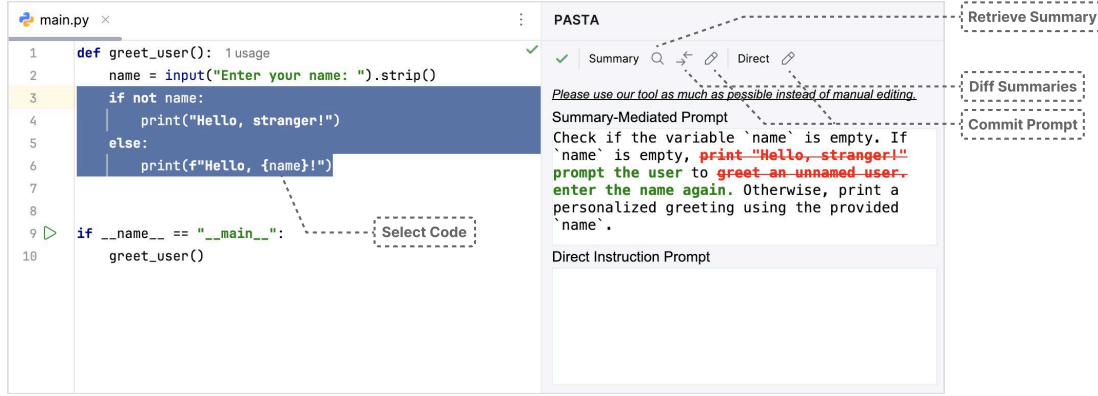


Fig. 2. Interface of PASTA, integrated into JetBrains IDEs (e.g., PyCharm, WebStorm).

API. To improve in-context learning and ensure output consistency, we include a set of few-shot examples in each prompt.

ACKNOWLEDGMENT

This research was supported in part by an AnalytiXIN Faculty Fellowship, an NVIDIA Academic Hardware Grant, a Google Cloud Research Credit Award, a Google Research Scholar Award, and NSF grants CCF-2211428, CCF-2315887, and CCF-2100035. Any opinions, findings, or recommendations expressed here are those of the authors and do not necessarily reflect the views of the sponsors.

REFERENCES

- [1] J. T. Liang, C. Yang, and B. A. Myers, “A large-scale survey on the usability of ai programming assistants: Successes and challenges,” in *Proceedings of the 46th IEEE/ACM international conference on software engineering*, 2024, pp. 1–13.
- [2] N. Tang, M. Chen, Z. Ning, A. Bansal, Y. Huang, C. McMillan, and T. J.-J. Li, “Developer behaviors in validating and repairing llm-generated code using ide and eye tracking,” in *2024 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 2024, pp. 40–46.
- [3] L. Zheng, W.-L. Chiang, Y. Sheng, T. Li, S. Zhuang, Z. Wu, Y. Zhuang, Z. Li, Z. Lin, E. P. Xing *et al.*, “Lmsys-chat-1m: A large-scale real-world llm conversation dataset,” *arXiv preprint arXiv:2309.11998*, 2023.
- [4] F. Cassano, L. Li, A. Sethi, N. Shinn, A. Brennan-Jones, J. Ginesin, E. Berman, G. Chakhnashvili, A. Lozhkov, C. J. Anderson *et al.*, “Can it edit? evaluating the ability of large language models to follow code editing instructions,” *arXiv preprint arXiv:2312.12450*, 2023.
- [5] N. Tang, “Towards effective validation and integration of llm-generated code,” in *2024 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 2024, pp. 369–370.
- [6] L. Tankelevitch, V. Kewenig, A. Simkute, A. E. Scott, A. Sarkar, A. Sellen, and S. Rintel, “The metacognitive demands and opportunities of generative ai,” in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2024, pp. 1–24.

- [7] C. Chen, S. Feng, A. Sharma, and C. Tan, “Machine explanations and human understanding,” *arXiv preprint arXiv:2202.04092*, 2022.
- [8] J. Zamfirescu-Pereira, R. Y. Wong, B. Hartmann, and Q. Yang, “Why johnny can’t prompt: how non-ai experts try (and fail) to design llm prompts,” in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, 2023, pp. 1–21.
- [9] Y. Tian, Z. Zhang, Z. Ning, T. Li, J. K. Kummerfeld, and T. Zhang, “Interactive text-to-sql generation via editable step-by-step explanations,” in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 16 149–16 166.
- [10] Y. Tian, J. K. Kummerfeld, T. J.-J. Li, and T. Zhang, “Sqlucid: Grounding natural language database queries with interactive explanations,” in *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, 2024, pp. 1–20.
- [11] M. X. Liu, A. Sarkar, C. Negreanu, B. Zorn, J. Williams, N. Toronto, and A. D. Gordon, ““what it wants me to say”: Bridging the abstraction gap between end-user programmers and code-generating large language models,” in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, 2023, pp. 1–31.