UDC 004.8:808.1

B. M. Matsenko
V. V. Chopliak

PROMPT ENGINEERING AS A NEW FORM OF TECHNICAL WRITING

Vinnytsia National Technical University

Анотація

Зі стрімким розвитком великих мовних моделей (LLM) парадигма взаємодії людини та комп'ютера зазнала значних змін. Ця стаття розглядає інженерію підказок (prompt engineering) не просто як інструмент комунікації з алгоритмами, а як новітню форму технічного письма. На відміну від традиційної технічної документації, яка пояснює роботу систем для людей, інженерія підказок перекладає людські наміри у детерміновані інструкції для нейромереж. У дослідженні аналізуються ключові паралелі між розробкою програмного забезпечення та структуруванням підказок, підкреслюється важливість синтаксису, контексту та ітеративного вдосконалення. Стверджується, що фахівці з програмної інженерії мають природну перевагу в цій галузі завдяки навичкам алгоритмічного мислення та оптимізації процесів.

Ключові слова: Інженерія підказок, технічне письмо, штучний інтелект, великі мовні моделі, взаємодія людини та комп'ютера, розробка програмного забезпечення, алгоритмічна комунікація.

Abstract

With the rapid evolution of Large Language Models (LLMs), the paradigm of human-computer interaction has fundamentally shifted. This paper examines prompt engineering not merely as a tool for communicating with algorithms, but as an emerging form of technical writing. Unlike traditional technical documentation, which explains system mechanics to human users, prompt engineering translates human intent into deterministic instructions for neural networks. The research analyzes key parallels between software engineering and prompt structuring, emphasizing the critical role of syntax, context formulation, and iterative refinement. It argues that software engineering specialists possess a natural advantage in this domain due to their foundational skills in algorithmic thinking and process optimization.

Keywords: Prompt Engineering, Technical Writing, Artificial Intelligence, Large Language Models, Human-Computer Interaction, Software Engineering, Algorithmic Communication.

Introduction

For decades, technical writing has served as the vital bridge between complex technological systems and their end-users. Its primary objective has always been clarity: distilling intricate software architectures, hardware manuals, or API documentation into accessible human language. However, the advent of generative Artificial Intelligence (AI) and Large Language Models (LLMs) has inverted this paradigm. Today, the challenge is not only explaining machines to humans but accurately explaining human intent to machines. This has given rise to "Prompt Engineering" – a discipline that requires the exactitude of coding and the linguistic nuance of technical writing. This paper explores how prompt engineering constitutes a modern evolution of technical communication, requiring a rigorous, structured approach to natural language.

The syntax of machine interaction

Traditional technical writing relies on standardized frameworks: headings, bullet points, and linear logic. Prompt engineering demands a similar, if not more rigid, architectural approach to language. A well-engineered prompt is rarely a simple question; it is a structured command set consisting of context, instruction, input data, and output constraints. When communicating with an LLM, natural language acts as the compilation code. Ambiguity in a prompt leads to "hallucinations" or suboptimal outputs, much like a syntax error in traditional programming leads to a compilation failure. Therefore, the technical writer of the AI era must construct prompts with explicit constraints, defining the persona, tone, format, and boundaries of the expected response.

Parallels with software engineering

The methodology of prompt engineering closely mirrors the lifecycle of software development. As software engineering moves toward increasingly high-level abstractions, the ability to write effective prompts becomes analogous to writing clean, maintainable code. Key overlaps include:

- **Modularity.** Breaking down complex tasks into a sequence of smaller, chained prompts (Prompt Chaining), similar to writing discrete functions or microservices.
- **Debugging and iteration.** Analyzing the output of an LLM to identify where the instruction was misunderstood and refining the prompt accordingly.
- **Version control.** Maintaining repositories of successful prompt templates that yield consistent results across different model versions. Students and professionals in software engineering are uniquely positioned to excel in this new field. The algorithmic mindset – understanding conditional logic (if-then statements) and edge cases – translates directly into anticipating how an LLM will process a given set of instructions.

Strategies for effective prompt design

1. **Zero-shot and few-shot prompting.** Providing the model with specific examples of the desired output (Few-Shot) significantly reduces variance. This is akin to providing code snippets in API documentation.
2. **Chain-of-thought (CoT) prompting.** Instructing the model to "think step-by-step." This forces the LLM to generate an explicit logic trail, making the output more accurate and easier to audit.
3. **Context encapsulation.** Creating a closed environment within the prompt to prevent the model from pulling irrelevant data from its pre-training weights, ensuring strictly domain-specific answers.

Conclusion

Prompt engineering is rapidly maturing from a niche experimental skill into a formalized branch of technical writing and software engineering. As AI systems become integrated into every facet of digital infrastructure, the ability to predictably and safely control these models via natural language will be a critical engineering competency. By applying the principles of algorithmic design, clear syntax, and iterative debugging to human language, prompt engineers ensure that AI tools function as reliable, deterministic instruments rather than unpredictable novelties. Ultimately, mastering this new form of technical writing is essential for building the next generation of intelligent software solutions.

REFERENCES

1. White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., ... & Schmidt, D. C. (2023). A prompt pattern catalog to enhance prompt engineering with chatgpt. *arXiv preprint arXiv:2302.11382*.
2. Budiu R. Mobile User Experience: Patterns to Make Sense of it All / R. Budiu, J. Nielsen. – Fremont : Nielsen Norman Group, 2020. – 198 p. *(Note: Retained a foundational UX/UI text as prompt engineering heavily influences user experience).*
3. Reynolds, L., & McDonell, K. (2021). Prompt programming for large language models: Beyond the few-shot paradigm. *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*.
4. Norman D. The Design of Everyday Things: Revised and Expanded Edition / D. Norman. – New York : Basic Books, 2013. – 368 p.
5. Lo, L. S. (2023). The CLEAR path: A framework for enhancing information literacy through prompt engineering. *The Journal of Academic Librarianship*, 49(4), 102720.

Маценко Богдан Михайлович – студент групи 2ПІ-25б, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця, e-mail: bmacenko222@gmail.com.

Науковий керівник ***Чопляк Вікторія Володимирівна*** – викладач англійської мови, кафедра іноземних мов, Вінницький національний технічний університет, м. Вінниця, e-mail: nikavnuchkova@gmail.com.

Bohdan M. Matsenko – a student of 2SI-25b, Faculty of Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: bmacenko222@gmail.com.

Scientific supervisor ***Victoria V. Chopliak*** – teacher of English, Foreign Languages Department, Vinnytsia National Technical University, Vinnytsia, e-mail: nikavnuchkova@gmail.com.