# Max Woolf highlights the potential and limitations of large language models in code optimisation



Max Woolf, a senior data scientist at Buzzfeed, has recently published findings on the efficacy of large language models (LLMs) in code optimisation, showcasing their capacity to generate and refine programming solutions. The experiment is of particular interest to those in the tech and software development sectors, revealing the nuanced relationship between user expertise and AI performance.

Woolf's investigation centred around the ability of LLMs, such as Anthropic's Claude, to enhance coding tasks upon request. The experiment involved directing Claude to write Python code capable of determining the difference between the smallest and largest numbers from a selection of one million random integers whose digits sum to 30. The LLM initially produced a solution that, although functional, resembled what one might expect from a novice programmer, executing the task in an average time of 657 milliseconds on an Apple M3 Pro MacBook Pro.

Upon receiving a prompt to "write better code," Claude responded by substantially improving the efficiency of the code, achieving a performance increase of 2.7 times. Subsequent prompts resulted in further optimisations, with the final iterations yielding enhancements that included multithreading techniques, ultimately attaining a speed increase of an impressive 99.7 times over the initial version. Notably, however, these improvements were accompanied by new errors that required manual intervention to rectify.

Woolf explored the concept of "prompt engineering," which involves meticulously crafting queries to provide LLMs with specific instructions on expectations and desired outcomes. He states, "Although it's both counterintuitive and unfun, a small amount of guidance asking the LLM specifically what you want, and even giving a few examples of what you want, will objectively improve the output of LLMs more than the effort needed to construct said prompts."

While Woolf's findings indicate that iterative requests can lead to significant code improvements, he acknowledges that the refinement process introduces complexity and potential bugs. He concludes by asserting that LLMs are unlikely to usurp the role of software engineers in the foreseeable future, as a foundational understanding of software engineering principles remains essential for evaluating code quality and managing domain-specific constraints.

Woolf's findings are echoed in a recent research paper from a team of computer scientists affiliated with Northeastern University, Wellesley College, and Oberlin College. The study, entitled "Substance Beats Style: Why Beginning Students Fail to Code with LLMs," posits that the content of prompts—rather than their structure—plays a critical role in the quality of LLM responses. The authors, including Francesca Lucchetti and Zixuan Wu, conclude that a solid grasp of the subject matter is imperative for successful interaction with LLMs, reinforcing the notion that experienced developers will consistently yield superior results than novices when utilising LLMs for code assistance.

The advancements in LLM capabilities present both excitement and caution for businesses, as the technology continues to evolve, promising improved productivity while highlighting the indispensable human touch required to effectively harness such tools.

Source: [Noah Wire Services](https://www.noahwire.com)

## Bibliography

1. <https://www.aibase.com/news/14583> - This article details Max Woolf's experiment with Anthropic's Claude, showing how repeated requests to improve code led to significant performance enhancements, from 657 milliseconds to 6 milliseconds, and discusses the importance of prompt engineering.
2. <https://www.theregister.com/2025/01/07/ai_can_write_improved_code_research/> - This article describes Woolf's experiment where Claude was tasked with optimizing Python code, resulting in a 99.7 times speed increase over the initial version, and highlights the role of prompt engineering in improving code quality.
3. <https://www.heise.de/en/news/AI-as-a-programming-aid-more-efficient-code-but-more-errors-10231932.html> - This article discusses Woolf's findings on how LLMs can generate more efficient code but introduce errors, emphasizing the need for precise instructions and manual intervention to rectify these errors.
4. <https://www.theregister.com/2025/01/07/ai_can_write_improved_code_research/> - This source corroborates the concept of 'prompt engineering' and its impact on the quality and efficiency of the code generated by LLMs, as well as the introduction of errors in highly optimized code.
5. <https://www.aibase.com/news/14583> - This article explains that despite significant performance improvements, human developers are essential for validating solutions and troubleshooting, as AI-generated code cannot be used directly without human oversight.
6. <https://www.heise.de/en/news/AI-as-a-programming-aid-more-efficient-code-but-more-errors-10231932.html> - This source supports the idea that while LLMs can generate faster code, they are not designed to handle complex optimizations without introducing errors, and thus require human intervention.
7. <https://www.theregister.com/2025/01/07/ai_can_write_improved_code_research/> - This article highlights that the refinement process of code through LLMs introduces complexity and potential bugs, reinforcing the need for a foundational understanding of software engineering principles.
8. <https://www.aibase.com/news/14583> - This article discusses how the language model approaches the code with a fresh perspective with each new request, allowing for continual improvement, and the importance of precise prompt engineering for significant performance enhancements.
9. <https://www.heise.de/en/news/AI-as-a-programming-aid-more-efficient-code-but-more-errors-10231932.html> - This source emphasizes that LLMs are inventive and provide valuable suggestions, but they cannot replace human workers due to the need for specialist knowledge to recognize good ideas and resolve problems.
10. <https://www.theregister.com/2025/01/07/ai_can_write_improved_code_research/> - This article notes that LLMs will write better code if developers know how to ask effectively, which limits the utility of AI code help for novices and underscores the importance of software development experience.
11. <https://www.aibase.com/news/14583> - This article mentions that subtle changes in the way questions are asked can significantly affect the output of the LLM, highlighting the nuanced relationship between user expertise and AI performance.
12. <https://www.theregister.com/2025/01/07/ai_can_write_improved_code_research/> - Please view link - unable to able to access data