SFB 1252
PROMINENCE
IN LANGUAGE

LLM-Assisted Coding

Philip Georgis

July 10, 2025

UNIVERSITY
OF COLOGNE

SFB 1252 PROMINENCE IN LANGUAGE

Common Pitfalls

- **Hallucinations**
  - LLM may reference non-existent libraries or functions that appear to make sense in context, but which will throw errors when running the code, or even introduce malware.
  - Explanations usually sound confident and plausible but might be totally wrong or unsuited for your application, or else not accurately describe what the code does.

- **Complex logic**
  - LLMs may have difficulty generating code for tasks with complex logic, especially if it requires very domain-specific knowledge.

- **Frozen in time**
  - Libraries may have been updated since LLM was trained; may reference outdated documentation or tools.

- **Non-determinism**
  - The same prompt will usually produce different results each time (could be either positive or negative).

General Tips for Coding with LLMs

- **Well-structured prompts** are more successful:
  - Use clear language and avoid or rephrase negative instructions.
  - Include relevant context and details (e.g. purpose of code, examples of data format) to help the LLM understand.
  - If relevant, include snippets of other code with which the code to be generated or analyzed may interact.
  - Clearly delimit input types (instructions, examples, data, code snippets) within prompt.
  - Prime the LLM for success by giving them a persona.
    - “You are a skilled _____ with deep expertise in _____ ...”

General Tips for Coding with LLMs

- **Reframe your task** in terms of what the LLM can do.
  - If your desired task is too complex, perhaps it can be broken down into simpler subtasks which the LLM can handle more successfully.

- **Tell the LLM how you think** the task should be approached.
  - Explicitly mention algorithms, libraries, functions, or other specific methods (with explanations if necessary) that you want to use, which can guide its responses.

- **Ask for options** for how to accomplish or implement your task.
  - Explain your task and request several alternative approaches for how to tackle it.
  - Then use your own judgment to select the method which seems most appropriate for your data and application.

General Tips for Coding with LLMs

- **Always check** that the generated code does what you intended.
  - Ideally, test on a known example and verify that it does what you expect.
  - (!) Very easy for bugs to slip in without validating outputs, or if you are not yet proficient in the programming language.

- **Ask follow-up questions** when you don’t understand or have doubts.
  - This might reveal discrepancies between what you intended and what the model understood.

- **Don’t rely 100% on LLM outputs**.
  - LLMs are very powerful but have their limitations and sometimes make mistakes, like any other tool.
  - Google, StackOverflow, official documentation, and colleagues are still valuable resources in case of doubt.

Use Cases for Coding with LLMs

Well-Suited	Riskier / Less Well-Suited
Common programming languages	Less common or very domain-specific languages
Time-saving tool: Generate code for a straightforward task given clear instructions (e.g. preprocessing data).	**“Magic”:** Perform vague or complex tasks without clear instructions or specified methodology.
Learning tool: Generate examples of how to use a particular function, method, etc.	**Statistical analysis:** LLM doesn’t know about your data’s properties or requirements!
Short code analysis: Explain in natural language what a short code snippet does or why a given method/parameter is required.	**Working with lengthy code:** Mistakes, misinterpretations, and inconsistent behavior are likelier with longer blocks of code due to limited memory.
Optimization: Get feedback on how to improve code efficiency or accomplish your task in another way.	**Rewrite entire code:** May introduce unsolicited changes which could unwittingly impact methods or results.
Troubleshooting: Understand an error	**Domain expertise:** LLM is not a domain