Code-Dokumentation

Gute Maintainer-Praxis, Instandhaltung und Wartung, Qualitätsstandards

- Link collection code documentation
- Working with an ethical code assistant - interview with Mark Padgham (Urban Analyst)

□□ Link collection code documentation

To read, listen & watch

- *How to Write Good Documentation*: https://guides.lib.berkeley.edu/how-to-write-good-documentation
- *How to write effective documentation for your open source project*: https://opensource.com/article/20/3/documentation
- *Software Documentation Best Practices [With Examples]*: https://helpjuice.com/blog/software-documentation
- *Ship it!* by Jared Richardson and Will Gwaltney, Jr: https://pragprog.com/titles/prj/ship-it/
- *The Pragmatic Programmer* (Thomas/Hunt, 2020) is a good general introduction to professional software development: Keeping code readable and easy-to-change, learning requirements, utilizing unit- and property tests, basics of securing applications: https://pragprog.com/titles/tpp20/the-pragmatic-programmer-20th-anniversary-edition/
  - There is also a German translation (*Der pragmatische Programmierer*): https://www.hanser-fachbuch.de/fachbuch/artikel/9783446463844
- *Weniger schlecht Programmieren* (Passig/Jander) is similar to "Pragmatic Programmer", but focuses more on code and less on projects: https://dpunkt.de/produkt/weniger-schlecht-programmieren/
- *Command Line Interface Guidelines* are an open-source guide to help you write better command-line programs: https://clig.dev/

Methods & tools

- *Tracer Bullet Development* is a method to get a working version of your software quickly. This allows to evaluate code architecture and user interface early. The method is described on this website: https://growsmethod.com/practices/TracerBullets.html
- *Refactoring Guru* has a lot of resources on common ways (patterns) of how to approach particular problems and how to improve existing code to make it easier to read and to change (refactoring): https://refactoring.guru/
- *12 Factor app* is less about code than about running the code in a way that it can be easily and safely configured, ran and monitored. When you create your project based on modern frameworks (i.e. Laravel) it will often follow such principles: https://12factor.net/

□□ Working with an ethical code assistant - interview with Mark Padgham (Urban Analyst)

Code assistants can be incredibly helpful and save developers a lot of time, but it can take a lot of patience to find the one that works for you. Mark Padgham, our former grantee, was looking for a code assistant that was not only useful but also ethical. Here he explains what he was looking for exactly - and why - and what he ended up using.

What project are/were you working on?

https://urbananalyst.city, based on all software components in https://github.com/UrbanAnalyst

Why were looking for a code assistant?

I needed to code complex routines in rust and compile to web assembly (WASM) to integrate within a Typescript front-end. My usual computer languages are C++ and R, not rust or Javascript. I knew a code assistant could (1) help translate my code ideas from C++ to rust syntax, and (2) help with the WASM integration into Javascript, with which I had no experience at that stage. I wanted a code assistant for the first bit because I thought translating ideas would be faster than me writing everything directly in rust. I needed a code assistant for the second bit, because I really had no idea how to do it.

Are there code assistants that are not based on AI?

I only note that I don't use the term "AI". In this context, there is a distinction between large language models (LLMs), and the equivalent trained on code, which are commonly referred to as "large code models." Simply referring to all as "AI" erases these kinds of important distinctions. (And there are many other, much more important reasons to avoid using that term.) LLMs are also very good at code, and I know many people who use ChatGPT as a code assistant. The model I ended up using is built on CodeLLama (https://huggingface.co/codellama). But it's probably reasonable to suspect that there are no code assistants in the form we know know them that are not based on LLMs or LCMs.

Why was it important to you to use ethical AI? What are the benefits of en ethical AI code assistant?

I have never used GitHub Copilot, because it was obvious to me and many other people from the very beginning that that was trained by scraping the entire contents of GitHub without any consideration of, or respect for, the open source license conditions of code used in that training. https://githubcopilotlitigation.com/ traces the history of the main legal case against Copilot, which is still ongoing. This behaviour is unethical, and I refuse to use any tools trained in such ways.

More general arguments against this kind of wholesale "theft" (generally expressed in legal contexts as "unlawful use") of material for training LLMs have emerged in the case brought by the New York Times against OpenAI: https://theconversation.com/how-a-new-york-times-copyright-lawsuit-against-openai-could-potentially-transform-how-ai-and-copyright-work-221059

How did you conduct your research?

Primarily through reading "model cards" on huggingface.com, and scouring articles they referred to for details of training data. The original Code Llama model is described in the paper linked from https://ai.meta.com/research/publications/code-llama-open-foundation-models-for-code/, with the model card for the current version at https://huggingface.co/codellama/CodeLlama-70b-hf. I followed all links to understand exactly how the training data were obtained, and was satisfied that Code Llama was about as ethically acceptable as you get these days.

Which tool did you end up using?

phind.com. At the time I started using it, it was based on CodeLlama-34B, described in the Phind blog at https://www.phind.com/blog/code-llama-beats-gpt4, with the CodeLlama model card at https://huggingface.co/Phind/Phind-CodeLlama-34B-v2. The Phind model uses additional training on "an internal Phind dataset" of 1.5B tokens. There used to be a mention elsewhere that suggested that they purchased that dataset commercially. The ethics of such datasets and their procurement can of course never be ascertained, but both the initial development of CodeLlama, and the subsequent development of Phind, certainly seem to have been done with far greater ethical awareness than anything displayed by OpenAI or Microsoft. And the majority of training data both for CodeLlama and Phind remain datasets that have been obtained in full respect of open source licensing conditions.

Do you have any other recommendations for developers who are looking for good tools to help them do their work?

Read the enormous number of articles that have been written on the motivations of the major companies who develop LLMs to understand their very specific agendas. And then act against them by refusing to use their tools in favour of alternatives that respect open source licensing conditions. The current state of code assistants, LLMs, and associated hype is insightfully analysed in https://blog.pragmaticengineer.com/the-ai-developer/. To quote that article, the hype surrounding

this technology is, "a well-choreographed show."