Paper ML04

Neither Magic Wand Nor Terminator: Finding the Sweet Spot in AI-Assisted Coding

Josua Böser, Chrestos GmbH, Essen, Germany

Abstract

Remember debugging that cryptic bug until midnight, only to discover a missing semicolon? Now imagine explaining that problem to an AI that alternates between writing flawless algorithms and confidently suggesting you name your variables "myAwesomeString42". Welcome to the bizarre reality of AI-powered coding! While tech prophets either promise coding nirvana or predict developer extinction, most of us just want help with regular expressions without selling our minds to the AI companies. This talk cuts through the hype to show how AI coding tools work in the trenches - when they're brilliant, when they're confusing, and how to tell the difference. We'll demonstrate practical techniques for using AI as your coding co-pilot: one that helps navigate unfamiliar frameworks, suggests better approaches, and occasionally needs its enthusiasm tempered with human wisdom. Learn to harness AI's strengths while retaining the satisfaction of solving problems yourself.

1 Introduction

Moin, dear reader! That's Northern German for "hello."
We are in turbulent times. Almost every month, there's a new update about an AI application that's supposedly going to revolutionize our lives, according to the marketing departments of tech companies. An unbelievable amount of money is being invested in AI, and the hype surrounding Large Language Models (LLMs) is at an all-time high. Living in this age as a fellow human and developer is exciting and unsettling at the same time.

In that spirit I was sitting at the same computer last year trying to figure out the best way to explain prompt engineering to you. But, since then, many techniques have become automated. This sounds like it was a complete waste of time, or? I would say no, as I focused on the underlying principles of how to ask questions, rather than specific tools, which you can also use in your normal life.
This time I want to show you what's possible in the field of AI-assisted coding. This paper will cover available online and offline tools, how to choose your model, and how to use the currently available tools. It is meant to be read after the talk. The paper focuses more on practical applications and examples, while the talk is supposed to instill the basic principles of how you should work with AI. Thinking about trust and safety.
Every tool and model you see in this paper will most likely be subject to change as the field evolves, but if you use the provided websites, you will hopefully find the models and tools of your time. Also, if you take with you the message, that handling trust is the most important thing, you get something out of it for the long run.

I hope you enjoy reading this paper and find it useful. If you have any questions or would like to discuss anything further, please feel free to reach out to me via email or LinkedIn.

Transparency note

For transparency, I used the following models in this paper: Claude Sonnet 4.5 [5] and Gemini 2.5 Pro [19] for formatting and writing assistance. DeepL Write [11] for proof reading, and the extension GitHub Copilot [16] in Visual Studio Code [27] as a framework.

2 What Software can I use?

Have you tried copying and pasting code from ChatGPT, only to wonder if there's a better option? The good news is that there are much more practical tools available. The reality check? They still require you to think critically about what you want to create.
Below, you will find a collection of tools that I have found useful. Keep in mind that the field is moving fast, so what is useful today may be outdated in a few months, weeks, or even days.

What to consider when using agentic tools

You have probably heard something about agents lately. Many companies are adding an agentic mode to their products, which allows the model to perform tasks for you. These tasks can include answering emails, creating folders and files, or managing your calendar. You only need to provide minimal feedback, and the model will take care of the rest. Although this may seem extremely useful, you should know that these models can make costly mistakes and cannot read your mind. They probabilistically attempt to complete your tasks and do not consider the consequences of their actions. Ultimately, you are responsible for what the model does. Always monitor what it is doing, and do not give it more permissions than necessary. You can't trust an agent any more than you can trust a complete stranger.

2.1 Online Tools

The most capable AI tools are online. If you want the best performance, there is no way around this. However, I've learned that despite the marketing promises, the practical differences between the major providers' tools are smaller than you'd expect. Once one provider adds something to their toolkit, it doesn't take long before the tool is available from the others. Google is an exception, offering many different product types. However, they are also known for abandoning their projects. To learn more, take a look at the website killedbygoogle [24]
Online tools fall into two main categories: Those that allow you to use models from many providers, and those that only host their own models. First, I will present two of the multi-model providers that allow you to try different models.

Tool	Usages	Disadvantages
Fal.ai [13]	Freedom to choose from multiple models and providers. It's good for experimenting with different models without being locked into a platform, and you only pay for the tokens you use. There are no subscription fees. It lets you try all kinds of models, not just LLMs.	It's developed more for developers, so it can be complex. Pricing can vary significantly between models. It is less integrated than single-provider solutions.
Perplexity [33]	It is specialized for internet research, and you can use models from different providers to test them for non-developers.	It focuses on LLMs and internet research. It does this better than its competitors, but the difference is not significant.

Now that you know about some websites for testing different models, the following table shows the main big AI players. This table only contains the most well-known companies, but there are many more out there that usually create more specialized tools for specific tasks. Moondream [29] is one such example. They created a vision-language model that specializes in understanding images. Although it is much smaller than the models of the most famous providers, it performs very well on image understanding tasks. This is a typical example of when bigger is not necessarily better. If you want to explore the available tools, the "There is an AI for that" website [39] provides an overview of the current tools available.

Tool	Usages	Disadvantages
OpenAI		
ChatGPT [31]	Top performer with many integrated tools. It is the first exposure to AI applications for most people and is easy to use.	OpenAI has unpredictable company policies. The GPT-5 rollout caused a backlash when old models were removed, breaking people's workflows (and sometimes their imaginary friends) [42]. They restored the models after receiving complaints, but this highlights the risks of relying on a single provider.
Anthropic		
Claude [3]	This is the preferred option for many programmers who focus on text and coding. Often, people switch to this provider after starting with ChatGPT.	Neither picture nor video creation is possible. It is primarily a tool for text-based tasks. Anthropic suddenly changed their data policy and started collecting data from chats unless you opted out. [2].
Google		
Gemini [18]	It contains nano banana and Veo3, which are part of the best picture and video models currently.	Google, as a company, is problematic in itself. You never know what they're doing with your data. They are no longer using their company motto, "Don't be evil," which is underlined by the fact that they were sued by their own employees for acting against this motto (for which the employees were fired). [45]
Notebooklm [21]	Create summaries of sources and then create a podcast, flash cards, or videos about them. If you are an auditory learner, this is a great way to explain a paper to you.	Always think about the validity of what you are told. In my experience, this method works well, but you have to be careful with the information it provides.
Colab [20]	Online Jupyter notebooks. No installation necessary. Create analyses directly with Gemini.	Do not trust Gemini without understanding what it has done. It is fast, but Gemini is a people pleaser and will not give you an answer that challenges your beliefs.
Ai Studio [17]	Create apps and websites based on HTML, CSS, TypeScript, and React.	You can create apps to play around with, but they are not stable and require a lot of iteration. As an alternative, you can try Replit [35], but it will cost you money. They also recently released their new version replit3 [34].
Mistral		
Le Chat [28]	Mistral is stationed in France and is therefore under European data protection.	The provided models are not the highest performers in the benchmarks.
Deepseek		
DeepSeek AI [12]	Strong and cheap. It was the first to establish the thinking process.	The Chinese legislation creates unique challenges for data privacy and security.
Alibaba		
Qwen [40]	Strong and cheap. Alibaba is the largest provider of open-weight models.	As with DeepSeek, Chinese legislation poses unique challenges regarding data privacy and security.

2.2 Offline Tools

If you want to work without sharing your output, offline tools are your only option. There can be a significant performance trade-off, but offline applications are improving rapidly, and the difference in capabilities between online and offline applications is shrinking. You will need a powerful computer, but if you are working with sensitive data, this is the best option. There are also options to work with cloud providers, which promise better data protection. Ultimately, though, you will still need to trust them with your data.

Tool	Usages	Disadvantages
Open-source		
Ollama [30]	The most basic solution. It is often used to integrate local models into an add-on, such as Cline/GitHub Copilot. It's clean and free of bloatware, and it's minimalistic as a standalone.	There is no structuring of chats. It's not easy to change LLM parameters or integrate system prompts. They started cloud application, which does not fit the local use case.
AnythingLLM [6]	Aims to provide a simple interface for working with large language models. It also provides an app for your phone.	It's still in development and may lack advanced features. You also need a high-end mobile phone to use it satisfactorily.
ComfyUI [9]	It was initially specialized for pictures and videos. It is a visual, node-based programming system that lets you create complex workflows.	It can be overwhelming and complex.
Free but closed		
LM Studio [26]	Gives you lots of ways to customize your LLM	Gets complex in the depths. MCP servers are hard to use at the moment.

2.3 Integrated Coding Tools

All of the above applications can be used for coding, but they are primarily used for copying and pasting code snippets or for using their own restricted canvas system. To move beyond that, you need tools integrated into your coding environment. These tools can significantly reduce the hassle of copying and pasting code snippets and trying to make them work. They also allow the model to test its own code, otherwise you would have to do it yourself. Honestly, without integrated tools, AI-assisted coding just isn't practical. Everyone should work this way.

Tool	Usages	Disadvantages
IDE		
GitHub Copilot [16]	Autocompletes code as you type, suggests functions based on context. Provides an agentic mode which became really smart, especially with Claude Sonnet 4.5, and uses less tokens than Cline.	It doesn't understand your specific data context as good as Cline, which is the tradeoff for using less tokens.
Cline [8]	A better coding agent with great context awareness. It can use local models with Ollama or LMStudio, if you have powerful hardware.	It eats up your tokens like crazy. Since it shares a lot of context about your project area, it quickly fills up the context windows of your chat.
Continue [10]	An alternative to Cline, which is even more focused on local uses.	Has a smaller community.
Terminal		
Claude Code [4]	Command-line tool for delegating coding tasks.	Even more focused on developers

3 How to Choose a Model

After deciding which tool to use, the next step is selecting a model that meets our needs. Different models excel at different tasks, and AI companies are realizing that general-purpose models often lack specialization. Below is an overview of the models I found most useful as of October 2025. If you want to explore the models yourself, you can use the LM Arena [25], Artificial Analysis [7], and coding-specific benchmarks like SWE-bench [38] or Aider [1] to get an idea of what's out there. However, it should be noted that these models are now trained to perform well on these benchmarks, so it's not wise to rely solely on them. There are also websites in development, such as Risk Rubric [36], which aim to help you assess the risks associated with different AI models.

3.1 Online Models

Online models usually represent the latest advances in AI capabilities. These models are extremely large and expensive, and they are often designed to be jacks of all trades. While you can do nearly anything with them, it's often more cost-efficient to use something specialized.

Your personal favorite model

People often get attached to a specific model, like an old friend with some quirks. For example, I'm a fan of Claude Sonnet 4.5 because it has provided me with the most stable coding assistance so far. After adjusting to the change from version 4 to 4.5, I now understand how to interact with the model. The way you interact with a model can differ widely. Try out different models and find your favorite.

Model	Provider	Advantages	Disadvantages
Claude Sonnet 4.5	Anthropic	First choice for a lot of coders. Preferred by Cursor and many IDEs.	More expensive than alternatives.
Gemini 2.5 Pro	Google	Strong multimodal capabilities. Has a good coding performance according to the benchmarks	One of the models that lied the most to me, while trying to code with it. A real people pleaser.
GPT-5 (high)	OpenAI	Current benchmark leader. Universal with a large context window	Unpredictable company politics. The model could be removed suddenly.
Magistral Medium 1.2	Mistral AI	Best model currently out of the EU	Not as widely adopted or tested as the others and less performant

3.2 Offline Models

Offline models are ideal if you work somewhere with strict data policies, handle proprietary code, or want to avoid monthly subscription fees. These models run entirely on your hardware, giving you complete control over your data with no recurring costs, though you'll have to pay upfront for the hardware.

Model	Provider	Advantages	Disadvantages
GPT os 20B	OpenAI	Good balance of performance and resource needs.	Was likely published as OpenAI was pushed by the public releases of the Alibaba models.
Qwen3 30B 25072	Alibaba	Also in a lot of different sizes available. Used by many in the local community.	Has unique Chinese challenges.
NVIDIA Nemotron Nano 9B V2	NVIDIA	Good performance for its size.	Will be overwhelmed with complex tasks.

How to adjust the behavior of AI models?

If you don't like how a model behaves, you can adjust its behavior using a system prompt. However, it will never be as effective as when used in its standard way. It's like asking a professional singer to act like a comedian. They might be decent at it, but it's not what they were trained to do. If you want to learn more about system prompts, check out these guides [37] [41].

4 How to program with AI

4.1 Our Software Setup

In order to start working with AI, we need a setup. At the time of writing, I recommend the setup shown in the following table. It is not the best, but it is easy to set up. I currently prefer Visual Studio Code as my IDE because it has a large ecosystem of extensions and integrates well with GitHub Copilot. If you prefer open-source software and don't want Microsoft to track your usage data, you can use VSCodium [43] as an alternative. Note that some extensions may not be available. To effectively use GitHub Copilot, you will need a GitHub account and an active subscription. This costs 10 dollars a month but is the most convenient way to integrate AI directly into your workflow. If you want to start creating anything serious, using version control is essential. Whether generated by AI or humans, code will be faulty, so it's important to have a safety net to revert to a working state if something breaks. Git is the de facto standard here. The last step in the setup process is to select an AI model. As discussed in the previous section, there are many options available. For this setup, I recommend using Claude Sonnet 4.5 because it is one of the most popular models among developers.

Software	Description
Visual Studio Code [27]	Integrated development environment (IDE) for coding and editing.
GitHub Copilot [16]	AI assistant integrated into VS Code, provides inline code suggestions, chat interface, and agent mode.
Git [15]	Version control system for tracking changes and managing code history. Like having safes in a computer game.
Claude Sonnet 4.5 [5]	One of the most liked model among developers, as it works well with the Github copilot and Cline.

4.2 How to Set Up a Project

Now that our setup is ready, we can start a new project. First, create a folder for your project and open it in Visual Studio Code. Next, initialize a new Git repository in the folder by running git init in the terminal. This will allow you to track changes to your code as you work. You can also use the built-in Git features in VS Code to manage your repository. Then, create a virtual environment for your project. This lets you manage dependencies and avoid conflicts with other projects. For R, the equivalent is to use renv. If you don't know how to perform these tasks, simply ask the AI to perform them for you. These are standard tasks that the AI can easily handle. Just make sure to review the generated terminal commands before executing them.

Setting up copilot instructions

GitHub Copilot will prompt you to create a file called "copilot-instructions.md" in the root directory of your project. This file helps the AI understand your project's structure and provides coding guidelines. Initially, you can provide a brief description of your project and the programming language you are using. As your project progresses, add more details about your coding style, architectural decisions, and other relevant information. This will help the AI generate code that better aligns with your project's needs. You can also ask Copilot to create this file based on your project description and update it as you go. Note that the longer the file becomes, the more tokens it will use from your context window.

What is a context window?

The context window refers to the amount of information that the AI can "see" and consider at once when generating code or responses. This information is measured in tokens, or chunks of text. If your instructions file is very long, more of this limited space will be used up, leaving less room for relevant code or comments.

Step	Action
Project Folder	Create a new folder for your project and open it in Visual Studio Code.
Git	Run `git init` in the terminal to enable version control and track code changes.
Environment	Set up isolated dependency management (Python: `venv`, R: `renv`) to avoid conflicts with other projects.
Instructions file	Add project description, programming language, coding style, and architecture guidelines to help AI understand your project context. The `copilot-instructions.md` file will be used by GitHub Copilot to provide better suggestions.
Review Setup	Verify all components are working before starting to code.

Once you have your project set up, you can start coding.

4.3 Lets start Programming

Now that everything is set up, we can start coding. For this project, I suggest creating a small Shiny web app because they are easy to develop and demonstrate functionality quickly. Of course, you can choose a different project if you prefer. In this case, we will use AI as a learning partner and coding assistant. This means that the code created is not usable for production. AI projects often fail in the long run because people assume that code created by AI is production-ready when it is not. You must review, test, and optimize it as you would with any other code. From the beginning, you need to consider architecture, safety, and performance. To start coding, ask the AI to generate a basic structure for your Shiny application. Currently, you would first create a plan for your app and then ask the AI to generate a skeleton based on that plan.

Starting prompts

1. `Create a plan for a Shiny app in R that allows users to upload a CSV file containing blood pressure measurements, and then visualize the data in a plot.`
2. `Review and modify the created plan by asking me questions about the app.`
3. `Create the code for a Shiny app based on the created plan.`

Don't worry about the details of a small, shiny app. However, when you start building larger projects involving multiple components, you need to be aware of every decision in the plan. Therefore, the second prompt is crucial for refining the app's requirements and ensuring that all aspects are considered. A great project always starts with a solid plan and clear goals. After some back-and-forth with the AI, Claude Sonnet 4.5 makes simple apps work right out of the box. Now, you have a functioning, shiny app, but you don't understand the code.

Congratulations! You are now a Vibecoder. (Or maybe the new type of project manager.)

But as developers, or aspiring developers, the hard part is just beginning. You will need to understand, test, and optimize the code. Since we don't know the code, we can ask the AI to explain it to us.

Documentation prompt

```
Write an in depth documentation for the the app.R file.  Explain every function and
every line of code in detail so that a junior R developer can understand it.
```

Now is the time to understand the code, thoroughly test it, and modify anything you don't like. Never trust code. Always review it. Always test it. Always optimize it. It doesn't matter if the creator was human or AI.

I will leave that part to you. Now, we will focus on additional use cases, as well as tips and tricks to help you get the most out of AI-assisted coding.

5 Tips and Use Cases for Programmers

Utilize the system prompt The most common problem when using AI to code is that the generated code does not fit the project. This may be due to misinterpreted requirements, missing context, or insufficient details in the prompts. The system prompt is your biggest tool to influence every aspect of the AI's behavior. Make sure to generate it as clearly as possible. But remember, longer does not mean better. Be precise and concise.

What is a system prompt?

A system prompt establishes the general rules for how an AI assistant behaves throughout a conversation. It is always added to your prompt when you interact with the model, ensuring the model remembers your important context. Most users do not interact directly with system prompts in online tools, and you can only affect them indirectly. In GitHub Copilot, you can use the copilot-instructions.md file, mentioned in chapter 4.2 to achieve something similar.

Embrace the fact that AI is probabilistic. All AI models are probabilistic. This means they don't always provide the same answer to the same question. This also means that they do not always provide the correct answer; instead, they provide a probable one. Therefore, if the generated code is not what you expected, try asking again. Sometimes it will work on the second try.

Think about code architecture. When working on larger projects, having a solid architecture in place is crucial. AI can assist with this by generating diagrams, suggesting design patterns, and providing best practices. However, you must review and adapt these suggestions to fit your project's specific requirements. To review the AI's suggestions for R shiny apps, read the books "Engineering Production-Grade Shiny Apps" [14] and "R Packages (2e)" [44]. Depending solely on AI to create good architecture will most likely result in a messy codebase that is hard to maintain.

Use AI for double programming. Double programming is the foundation of validating clinical tables, figures, and listings. Two programmers create programs independently and without communicating initially, then compare their results. If the results differ, the programmers discuss them to identify the source of the discrepancy. AI naturally fits into this workflow as a double programming tool. Given the nature of double programming, in which two developers compare results, replacing one developer with AI is a simple process. Currently, the process works best when you:

- Share the expected table structure and corresponding SAP section with the AI
- Let the AI generate a code plan first and then have the AI generate detailed, commented code based on that plan
- Let a junior programmer review, test, and optimize the code

Experience with AI double programming

Six months ago, using older models such as the Claude Sonnet 3.5 and GPT-3.5 was a tedious process. However, the Claude Sonnet 4.5 model allows junior programmers to develop adequate programs with AI assistance. Nevertheless, complex statistical analyses still require expert support. In my experience, human developers are still necessary to guide the AI. Nevertheless, this provides junior programmers with an excellent opportunity to develop their skills.

Training the next generation

Who will maintain your code in five years if you don't raise the next generation of programmers? Perhaps AI? But who will teach AI your company's specific standards and requirements? AI again? While there are many uncertainties, one thing is certain: you need knowledgeable people, even in an AI-driven world. AI-assisted double programming is an ideal training ground. Juniors can learn by reviewing and refining, and with AI's help, they can take on more complex tasks.

The following is a list of common use cases in which AI can assist programmers. Some are repeats from previous sections, but I want to provide an overview of the use cases that I found valuable.

Generating code After all, this is kind of a no-brainer. In my opinion, software development is the most significant use case of AI that delivers monetary value. Helping you translate your thoughts into code can be a real benefit, but always remember, when you only use online solutions, this help can be taken from you at any time. Therefore, make sure you understand the generated code, test it thoroughly, and optimize it to truly learn from your work.

Auto completion The auto-completion features in IDEs like Visual Studio Code and RStudio have been significantly enhanced by AI. These features can suggest entire functions, classes, or code snippets based on the current context. These features speed up development and help you adhere to best practices by suggesting standard patterns.

Writing documentation Most developers love writing code but hate writing documentation. AI can help you generate this necessary documentation from your codebase. This is my favorite application of AI.

Commenting This is directly related to documentation and deserves its own mention. Well-placed comments can greatly improve code readability, yet most developers are either too lazy to write them (because the code should be self-explanatory, right?) or use them excessively. After working on another project for one month, I tend to forget why I wrote a complex function the way I did or who told me to use a specific selection in my code. AI lessens the hassle of writing comments by helping you organize your thoughts and identify areas of your code that need clarification. Comments can also be a great way to guide the AI when you want it to modify or extend your code.

Testing Although we all need to test our code, writing tests is often tedious and time-consuming. AI can generate standard tests that are sufficient most of the time. However, for complex testing scenarios, you still need to rely on your expertise to ensure adequate coverage. To learn more about testing in R, read the Testing chapter in R Packages (2nd ed.). [44]. There are also ways to use AI to automatically test websites and applications, but that is beyond the scope of this paper.

Debugging We all hate bugs because they highlight our inadequacies as programmers. Using an IDE with AI support can help you identify and fix bugs faster by analyzing error messages, stack traces, and code context. However, for complex bugs, you still need to rely on your debugging skills to trace the root cause.

Code reviewing Although every code needs to be reviewed, having a second pair of eyes is not always feasible. AI can assist with code reviews by checking for adherence to coding standards, potential security vulnerabilities, and performance issues. If you want an extra set of eyes on your code, AI is a good place to start.

Refactoring Refactoring is the process of restructuring existing computer code without altering its external behavior. This can include changing the programming language of a codebase or improving its structure and readability. Many people in the industry hope that AI will eliminate their technical debt by automatically refactoring their code (a significant portion of the financial industry still relies on COBOL, and its legacy systems are difficult and expensive to change). While AI can suggest refactorings, it's important to remember that refactoring requires a deep understanding of the codebase. Therefore, be cautious when relying solely on AI for this purpose. Perhaps in the future, but not yet.

Optimization Who doesn't want faster code? If you have slow code that needs optimization, AI can help you identify a faster way to achieve the same result. This doesn't always work, but it's helpful to have a rubber duck to talk to.

Learning This is the golden goose of using AI as a programmer! Learning new programming languages, frameworks, and libraries has never been easier. With AI, learning is like having a personal tutor available 24/7. It may not be perfect, but getting started with something new has never been easier.

6 Data Security

I hope I made it clear that, while AI is a great tool to assist programmers, big companies are trying to access your data in every way possible. They claim to do so to improve their models, but in the long run, no one knows what they or others will do with all this data. Perhaps I am culturally biased as a German because our history shows what can happen when certain entities have free access to data. I hope this is understandable.
For example, during World War II, occupying forces in the Netherlands used population records maintained by the Dutch government to identify and deport Jewish citizens. This serves as a stark reminder of how misusing data can have devastating consequences. At the time the data was collected, those responsible were unaware it could be used in this way. Nevertheless, the data was ultimately used to commit atrocities. [46]

As a general rule, you can assume that every interaction with an online AI is a donation of your data to the company providing the AI service. It doesn't matter if someone claims that they don't use your data to train their models. These models exist because companies collected data through web scraping [32], which always poses authorization problems. Anthropic also suddenly changed their privacy policy terms to include pro users' chats in their training unless they opt out [2].

Rule of Thumb for Online AI use

Online AI Use = Data Donation

Healthcare is an industry where data is sacred. This doesn't mean we can't use AI; it just means we need to be extra careful when we do. If you're creating something, always use made-up data, which is now extremely easy to generate. Anonymize your data as much as possible, and use local applications when possible.

7 Conclusion

The goal of this paper was to provide practical insights into how programmers can use AI tools in their daily work. Through the talk and this paper, I hope to have encouraged healthy skepticism and critical thinking so that you will use modern software development approaches to review the output and code of both AI and human programmers (and, again, use Git). If you want to further explore the topic, I encourage you to look at the provided links. As we know, AI topics evolve rapidly, so staying up to date is challenging.

If you want to stay up to date, I encourage you to follow people online whose job it is to stay up to date with AI. My favorite German YouTube channels are Digitale Profis [23] and ct3003 [22]. If you prefer another language, use your preferred search engine to find creators in your language. But be careful if someone wants money from you. There are a lot of people who try to profit from the hype and in my experience people tend to over-complicate things, so that they can sell you a more expensive product.

I hope you enjoyed the paper and found it useful.

References

[1] *Aider Leaderboard: AI Coding Assistants Comparison*. Leaderboard comparing various AI coding assistants on multiple metrics. URL: https://aider.chat/docs/leaderboards/ (visited on 09/16/2025).

[2] Anthropic. *Anthropic's Terms Update*. Updates to the terms of service for Anthropic's AI products. 2025. URL: https://www.anthropic.com/news/updates-to-our-consumer-terms.

[3] Anthropic. *Claude*. 2025. URL: https://claude.ai/ (visited on 09/16/2025).

[4] Anthropic. *Claude Code*. 2025. URL: https://claude.com/product/claude-code (visited on 09/18/2025).

[5] Anthropic. *Claude Sonnet 4.5*. https://www.anthropic.com/news/claude-sonnet-4-5. Language model. 2025.

[6] *AnythingLLM*. 2025. URL: https://anythingllm.com/ (visited on 09/18/2025).

[7] *Artificial Analysis: AI Model Evaluations*. In-depth evaluations and comparisons of AI models. URL: https://artificialanalysis.ai/ (visited on 09/16/2025).

[8] *Cline*. 2025. URL: https://cline.bot/ (visited on 09/18/2025).

[9] ComfyAnonymous. *ComfyUI: A powerful and modular stable diffusion GUI*. Open-source software. 2025. URL: https://github.com/comfyanonymous/ComfyUI.

[10] *Continue*. 2025. URL: https://github.com/continuedev/continue (visited on 09/18/2025).

[11] DeepL SE. *DeepL Write: AI Writing Assistant*. AI-powered writing improvement and grammar correction tool. 2025. URL: https://www.deepl.com/en/write (visited on 09/18/2025).

[12] DeepSeek. *DeepSeek Chat: AI Conversational Assistant*. 2025. URL: https://chat.deepseek.com (visited on 09/16/2025).

[13] fal.ai. *fal.ai: Fast and Scalable AI Inference*. Platform for running AI models with optimized inference. 2025. URL: https://fal.ai/ (visited on 09/18/2025).

[14] Colin Fay et al. *Engineering Production-Grade Shiny Apps*. Chapman and Hall/CRC, 2021. URL: https://engineering-shiny.org/.

[15] Git Development Community. *Git*. Distributed version control system for tracking changes in source code. 2025. URL: https://git-scm.com/ (visited on 09/18/2025).

[16] GitHub, Inc. *GitHub Copilot: Your AI Pair Programmer*. https://github.com/features/copilot. AI-powered code completion and suggestions integrated with major IDEs. 2025.

[17] Google. *AI Studio*. 2025. URL: https://aistudio.google.com (visited on 09/16/2025).

[18] Google. *Gemini*. 2025. URL: https://gemini.google.com/ (visited on 09/16/2025).

[19] Google. *Gemini 2.5 Pro*. Large Language Model. Accessed on September 2, 2025. 2025. URL: https://deepmind.google/models/gemini/pro/.

[20] Google. *Google Colaboratory*. https://colab.research.google.com/. Free cloud-based Jupyter notebook environment with GPU access. 2025.

[21] Google. *Notebooklm*. URL: https://notebooklm.google.com/ (visited on 09/16/2025).

[22] Heise Medien GmbH and Co. KG. *ct3003*. ct3003 YouTube channel. 2025. URL: https://www.youtube.com/@ct3003 (visited on 09/18/2025).

[23] Johannes Ruof and Timothy Meixner. *Digitale Profis YouTube Channel*. YouTube channel covering digital tools and AI technologies. 2025. URL: https://www.youtube.com/@DigitaleProfis (visited on 09/18/2025).

[24] *Killed by Google: A Graveyard of Discontinued Google Products*. Community-maintained list of products and services discontinued by Google. URL: https://killedbygoogle.com/ (visited on 09/16/2025).

[25] *LM Arena: Benchmarking Large Language Models*. Platform for evaluating and comparing large language models across various tasks. URL: https://lmarena.ai/ (visited on 09/16/2025).

[26] *LM Studio*. 2025. URL: https://lmstudio.ai/ (visited on 09/18/2025).

[27] Microsoft. *Visual Studio Code*. Free, open-source code editor with extensive extension support. 2025. URL: https://code.visualstudio.com/ (visited on 09/18/2025).

[28] Mistral AI. *Mistral Chat: AI Assistant*. 2025. URL: https://chat.mistral.ai (visited on 09/16/2025).

[29] *Moondream*. Vision language model for image understanding. 2025. URL: https://moondream.ai/ (visited on 09/18/2025).

[30] *Ollama*. 2025. URL: https://ollama.ai/ (visited on 09/18/2025).

[31] OpenAI. *ChatGPT*. 2025. URL: https://chatgpt.com/ (visited on 09/16/2025).

[32] Organisation for Economic Co-operation and Development. *Intellectual property issues in artificial intelligence trained on scraped data*. OECD Artificial Intelligence Papers 33. Paris: OECD Publishing, Feb. 2025. DOI: 10.1787/d5241a23-en.

[33] Perplexity AI. *Perplexity: AI-Powered Search and Chat*. 2025. URL: https://www.perplexity.ai (visited on 09/16/2025).

[34] Replit, Inc. *Replit Agent 3: AI-Powered Development Assistant*. 2025. URL: https://replit.com/agent3 (visited on 09/18/2025).

[35] Replit, Inc. *Replit: Build and Deploy Apps Quickly*. 2025. URL: https://replit.com (visited on 09/02/2025).

[36] *Risk Rubric: AI model Risk Assessment*. Tool for assessing risks associated with AI models. URL: https://riskrubric.ai/ (visited on 09/16/2025).

[37] Simplr. *Mastering System Prompts for LLMs*. Guide to crafting effective system prompts for large language models. 2025. URL: https://dev.to/simplr_sh/mastering-system-prompts-for-llms-2d1d (visited on 09/18/2025).

[38] *SWE-bench: Software Engineering Benchmark for LLMs*. Benchmarking large language models on software engineering tasks. URL: https://swe-bench.github.io/ (visited on 09/16/2025).

[39] *There's An AI For That: AI Tools Directory*. Comprehensive directory of AI tools and applications. URL: https://theresanaiforthat.com/ (visited on 09/16/2025).

[40] Tongyi Lab. *Qwen3-235B-A22B-2507: A Large Language Model*. https://github.com/QwenLM. Accessed: 2025-09-03. 2025. URL: https://chat.qwen.ai.

[41] Towards Data Science Inc. *Towards Data Science*. https://towardsdatascience.com. Leading publication for data science, machine learning, and AI professionals. 2025.

[42] understandingai. *GPT-5 Release Announcement*. Discussion on the capabilities and impact of GPT-5. 2025. URL: https://www.understandingai.org/p/is-gpt-5-a-phenomenal-success-or.

[43] VSCodium Community. *VSCodium*. Open-source builds of Visual Studio Code without Microsoft branding and telemetry. 2025. URL: https://vscodium.com/ (visited on 09/18/2025).

[44] Hadley Wickham and Jennifer Bryan. *R Packages*. 2nd. O'Reilly Media, 2023. URL: https://r-pkgs.org/.

[45] Wikipedia. *Google's "Don't Be Evil" Motto Controversy*. Discussion on the implications of Google's "Don't Be Evil" motto. URL: https://en.wikipedia.org/wiki/Don%27t_be_evil (visited on 09/16/2025).

[46] Wikipedia. *The Holocaust in the Netherlands*. Overview of the Holocaust events in the Netherlands during World War II. URL: https://en.wikipedia.org/wiki/The_Holocaust_in_the_Netherlands (visited on 09/16/2025).

8 Contact

Name: Josua Böser
Company: Chrestos GmbH
Address: Girardetstraße 1-5, Essen, Germany, 45131
Email: josua.boeser@chrestos.de