The Next Direction of AI Code Editors

Recently, I was working on adding an `xelatex` pipeline to GitHub Actions.

I encountered an issue with the `fontawesome5` package in the GitHub flow. The solution provided by 4o-mini (installing TeX Live 2021 and using `tlmgr install fontawesome5`) didn't work for me. However, 4o suggested a better approach: upgrading to TeX Live 2023 and still using `tlmgr` to install `fontawesome5`. While this didn't completely fix the problem, switching to TeX Live 2023 significantly improved the situation.

I used ChatGPT to help figure out the problem. For more details, check out What ChatGPT O1 Can Do That 4o-mini Cannot.

At this point, I didn't use editors like Cursor or Windsurf, though I did try them in another project. The issue with these code editors is that they only capture local test output, which limits their functionality in cloud environments.

In workflows like GitHub Actions, Jenkins jobs, or any code deployment or testing flow, code editors need to be better integrated. They should provide seamless interaction with the cloud and CI/CD processes.

This integration also applies to other content creation tools—whether for text, images, audio, or video. These tools should be integrated with A/B testing systems. AI tools could generate content, and A/B testing tools could provide feedback. This dynamic is similar to Reinforcement Learning from Human Feedback (RLHF), where AI models improve over time based on real-world feedback.

This concept of extending RLHF beyond just model outputs—into real-world testing and deployment environments—seems like a promising direction for improvement in both code editors and AI-driven content creation tools.

The test can be either instant or long, and it can be either automated or assisted by humans. If the tests are automated, such as user A/B testing for an AI tool, it still involves human feedback, but the process is automated. For example, we can let the computer check results every day or every hour based on A/B testing outcomes to improve the creation process. Similarly, for Jenkins or GitHub Actions jobs, we can let the computer check after their tasks are completed.

If human assistance is involved, the feedback cannot be fully understood by the machine and is often somewhat vague. For instance, when AI tools create content like images or videos, humans might point out that the content isn't funny enough, or that a specific detail should be improved. Machines still have a long way to go in making everything perfect, and whether something is “perfect” is often subjective, depending on individual taste. It is human feedback that helps make things better.

In theory, all human-defined rules can be written as prompts. There are user prompts and system prompts. We should focus on improving the prompts instead of fixing the output every time.