# La Próxima Dirección de los Editores de Código con IA

Recientemente, estuve trabajando en agregar una canalización de `xelatex` a GitHub Actions.

Me encontré con un problema con el paquete `fontawesome5` en el flujo de GitHub. La solución proporcionada por 4o-mini (instalar TeX Live 2021 y usar `tlmgr install fontawesome5`) no funcionó para mí. Sin embargo, 4o sugirió un enfoque mejor: actualizar a TeX Live 2023 y seguir usando `tlmgr` para instalar `fontawesome5`. Aunque esto no resolvió completamente el problema, cambiar a TeX Live 2023 mejoró significativamente la situación.

Utilicé ChatGPT para ayudar a resolver el problema. Para más detalles, consulta Lo que ChatGPT O1 puede hacer que 4o-mini no puede.

En este punto, no utilicé editores como Cursor o Windsurf, aunque los probé en otro proyecto. El problema con estos editores de código es que solo capturan la salida de las pruebas locales, lo que limita su funcionalidad en entornos en la nube.

En flujos de trabajo como GitHub Actions, trabajos de Jenkins, o cualquier flujo de implementación o prueba de código, los editores de código deben estar mejor integrados. Deberían ofrecer una interacción fluida con la nube y los procesos de CI/CD.

Esta integración también se aplica a otras herramientas de creación de contenido, ya sea para texto, imágenes, audio o video. Estas herramientas deberían estar integradas con sistemas de pruebas A/B. Las herramientas de IA podrían generar contenido, y las herramientas de pruebas A/B podrían proporcionar retroalimentación. Esta dinámica es similar al Aprendizaje por Refuerzo con Retroalimentación Humana (RLHF, por sus siglas en inglés), donde los modelos de IA mejoran con el tiempo basándose en la retroalimentación del mundo real.

Este concepto de extender RLHF más allá de las salidas del modelo—hacia entornos de prueba y despliegue en el mundo real—parece una dirección prometedora para mejorar tanto los editores de código como las herramientas de creación de contenido impulsadas por IA.

La prueba puede ser instantánea o prolongada, y puede ser automatizada o asistida por humanos. Si las pruebas están automatizadas, como las pruebas A/B de usuarios para una herramienta de IA, aún involucran retroalimentación humana, pero el proceso está automatizado. Por ejemplo, podemos permitir que la computadora verifique los resultados todos los días o cada hora basándose en los resultados de las pruebas A/B para mejorar el proceso de creación. De manera similar, para los trabajos de Jenkins o GitHub Actions, podemos permitir que la computadora verifique después de que se completen sus tareas.

Si la asistencia humana está involucrada, la retroalimentación no puede ser completamente comprendida por la máquina y a menudo es algo vaga. Por ejemplo, cuando las herramientas

de IA crean contenido como imágenes o videos, los humanos podrían señalar que el contenido no es lo suficientemente divertido o que un detalle específico debería mejorarse. Las máquinas aún tienen un largo camino por recorrer para lograr que todo sea perfecto, y si algo es “perfecto” a menudo es subjetivo, dependiendo del gusto individual. Es la retroalimentación humana la que ayuda a mejorar las cosas.

En teoría, todas las reglas definidas por humanos pueden escribirse como *prompts*. Existen *prompts* de usuario y *prompts* del sistema. Deberíamos centrarnos en mejorar los *prompts* en lugar de corregir la salida cada vez.