

A bitter lesson for the ‘AI skillset’

In the 2019 essay “[The Bitter Lesson](#)”, professor and scientist Richard Sutton writes about how “general methods that leverage computation are ultimately the most effective [approach to machine learning tasks]”. Since 2019, with the development and continued improvements of Large Language Models (LLMs), a new bitter lesson has cropped up in the community. Researcher Daniel Miessler [writes](#) that “we should avoid poisoning AI’s native capabilities with our supposedly superior guidance, because it’s not actually superior”.

As models reach the capability necessary to do novel mathematics and physics research, our community must grapple with how to use these tools now, and how to train future researchers to use them later. What is the most effective way to get results out of the latest models and harnesses¹? What should we teach present and future graduate students to help them find effective ways to get results from future models and harnesses?

A common response² is that we need an ‘AI skillset’: the knack for writing a clever set of skills and tools for our agent, designing a multi-agent verification pipeline, scaffolding a protocol for research-relevant tasks using our unique insight into what research looks like, Alternatively, the ‘AI skillset’ is a collection of prompting tricks, enhancing an otherwise unruly agent by telling them how the job should be done. Perhaps such skillsets are useful temporarily, when a particular model’s quirks can be understood and worked around to the benefit of the user. Yet the models are now developed faster than we can learn their quirks, and certainly faster than we can design and update a curriculum that teaches how to circumvent them.

The bitter lesson of the ‘AI skillset’, in Miessler’s words, is that we must “Be extremely specific about what [we] want, [...] and let [the LLM] figure out how to execute [on its own]”. The ‘AI skillset’ is not a set of techniques for humans interacting with an LLM, but the skill of asking the genie in a data-center what we want in a loophole-free way, and the skill of being humble enough not to tell it how to make it happen.

Then the question becomes, how do we develop these skills? One way is practice in the traditional way, as a taste for knowing to recognize what is the desirable outcome can come from firsthand experience of producing one; I often find myself realizing what I really want only partway through solving the wrong problem myself. The second is to design non-traditional approaches, where the exercise itself is not to reach the desired outcome, but to practice specifying what it is. Neither way requires LLM use.

¹ A “harness” is the software wrapping an LLM in a user-friendly way, giving it access to a file structure and the tools to read and write, as well as others like searching the web. (e.g. Claude Code and Codex)

² This is a common response in my bubble. Perhaps I am arguing against a strawman and most already know this page’s conclusion, yet I will write it anyway to make sure it is written.