Reflection

Awad distinguishes between different “types” of AI. What classification scheme does the paper use and why do these types matter for scientific research?

Early on in the paper we see a general categorization of ai models as their purposes and how they’re refined for said purposes, whether they’re accessory to each other, and how they handle privacy constraints. In the many examples Awad gave us in terms of how Ai can be used to experiment and stimulate gene editing and protein synthesis, we can at least understand that we don’t want a Natural Language Model to evaluate our synthesized protein experiment. (/data represented as a graph or a sequence of numbers is better understood by models calibrated to it rather than a language model)

Does Awad make a clear distinction between AI as a tool and AI as a scientific collaborator? If so, what are the differences and what are some examples given to support the differences? Do these examples suggest a real shift in how science is conducted, or mostly an extension of existing methods?

Bayesian Optimization, where a model is used as a tool to select an experiment based on configurations – that tell it whether this experiment will produce more interesting results – , is an example of an AI being used as a tool. And CNN models being used to observe and identify patterns in experiments related to genome sequencing, are examples of a collaborator.

What are some limitations or risks of using AI in science? How do these relate to issues such as interpretability, bias, reproducibility, or theory formation?

There was the Galatica model that fabricated citations and generated claims based on the training data. The consequences are similar to someone who has no knowledge of medicine writing a paper or making a public declaration that lysol can be used to kill covid. “They risk spreading authoritative-sounding misinformation.”

According to Awad’s arguments, is AI more likely to accelerate scientific discovery or to reshape the scientific method itself? Do you agree or disagree?

It’s not so much as changing the scientific method itself, but more so enhancing the visualization of data during the experiment and the rate that one completes it following the scientific method. I agree that it could accelerate the rate of scientific discovery, but since I have a smaller understanding of life sciences like biology or chemistry, I’m hesitant to say if the efficacy of an experiment based on synthesized data will yield the same results as an experiment based on data gathered over time.

Comparison

Which tool(s) did you use?

I used vs-code to create and edit the program files. person.py, person_factory.py, and family_tree.py

If you used an LLM, what was your prompt to the LLM?

When I didn't want to go through the trouble of writing long while loops or for loops for functions like Decade, Duplicate Name, and Count, I would start writing the function and hit tab when github copilot generated a suggestion. I eventually disabled the extension and all related settings because it got in the way of debugging. When it came to issues related to my understanding of the project specs, between misreading or misremembering, I used chatgpt to help rephrase and give me a better understanding, as well as when I ran into issues with the previously suggested copilot edits.

What differences are there between your implementation and the LLM?

The LLM had a more current understanding of python and python styled coding. I could already see in my classes and functions a lot of them were more like C or Java, especially when it came to managing the class variables and where to use self versus not using self.somefunc() or __init__().

What changes would you make to your implementation in general based on suggestions from the LLM?

Probably more stylistic changes, I know I made changes to class names, file names, function names, based on snake versus camel casing that's more typical in other languages, and also how to better use self.some_func().

What changes would you refuse to make?

If there's an extreme amount of imports, especially ones I'm not familiar with that might require a pip install. Sure there are plenty of libraries that would probably make things more convenient, but implementing a large number of libraries and how they interact with each other bothers me and from what I hear about overusing such implementations is that debugging and explaining how the code works becomes a nightmare.