VSC Users Day --- October 24, 2023

Scientific Programming in The Age of AI

## Intro:

Frederik De Ceuster, postdoc at the Institute of Astronomy (KU Leuven).  
I usually work on simulations of stars, but am also interested in the technological bigger picture, for instance, in the question:  
“How will our beloved field of scientific programming change in The Age of AI?”  
For that, I am collaborating with VSC and CECI,  
but also with DiRAC, the British equivalent of VSC.

## The Age of AI:

I think it is safe to say that we do live in The Age of AI.

Over the years we have seen artificial intelligence beating us at several “typically human” tasks, going far beyond distinguishing cats and dogs.

In 2011, IBM’s Watson beat humans in the American TV Quiz Jeopardy!  
In 2016, DeepMind’s AlphaGo beat humans at the board game go.

And, earlier this year, a creation from OpenAI’s DALL-E2 won a prestigious photography contest.

There is, of course, much more than fun and games, with AI and AI-related technologies facilitating and accelerating many research areas.

We see here, for instance, the Nature covers presenting a new matrix multiplication algorithm and the work on protein folding by AlphaFold.

However, the watershed moment, and the reason why we are here today talking about AI, was probably November 30, 2022, when OpenAI released its AI chat bot, called ChatGPT, to the public.

With more than 100 million users in less than 2 months, it was then the fastest growing app in history, and it demonstrated the power of the underlying technology to a huge audience.  
Among the many users are also students and researchers who use it to write papers, (so universities and journals suddenly had to revise their policies), but they also use it to write software!

For instance, I had a student, working on his Master thesis, who had a prototype, written in Python, that was meant to be integrated in a large code base, written in Fortran.

In less than a week, and heavily guided by ChatGPT, he managed to port his prototype to Fortran and integrate it in the large code base.

To me, this was completely unexpected, and it was the moment I realized that I needed to get involved in this.

## Large Language Models (LLMs)

To understand it’s power, we first need to know how it works.

ChatGPT is what we call a Large Language Model or LLM.

These are deep artificial neural networks that are essentially trained to repeatedly predict the next word in a sequence of text.

The sudden success of LLMs is not just because nowadays vast amounts of computing resources and data are available, but mainly because of the invention of a new architecture of these networks, called the transformer. This provides an attention mechanism that can weigh the importance of different words with respect to each other and as such can create a context for each word. Moreover, this architecture allows to train the model in a parallel way, such that the vast amounts of compute and data can be leveraged efficiently.

Training here means fitting the billions or even trillions of free parameters of the model to best perform this next-word prediction.

The training data consists of vast collections of text gathered from the internet.

## Large Language Models (LLMs): training data

A good thing about this is that the internet contains plenty of open-source software: think about GitHub, Bitbucket, and also stack overflow!

However, scraping the internet for data also creates some issues, for instance, with undesirable biases which might be picked up, or intellectual property rights on the data.

Don’t get me wrong, I’m not saying that you should not make your projects open source, by all means, please do so! But make sure to use a proper license so that future policies can hopefully protect your intellectual property.

## LLMs for code

Next to ChatGPT, several AI-tools based on LLMs are being developed specifically to generate code, and I’ll just list a few here for reference.  
Geert Jan and Frédéric will demonstrate how you can use then in a minute.

But before that, I would just like to take a moment to envision what the future of AI-assisted software development might look like.

## The future of AI-assisted software development:

In the short term, we will probably see AI increasing developer productivity by taking over low-level tasks and writing boilerplate code, documentation, or tests.

The use of AI will furthermore help to democratize software, lowering the barrier to entry, allowing more people to develop software.

I already see this with more students that can solve more problems, simple by asking ChatGPT for guidance.

The downside, of course, is that some of them don’t really understand how the code that “they” came up with works…

Therefore, it’s a good thing that ChatGPT, for instance, also explains the code it presents, and AI-tools can in general be used to explain code and help you understand (as long as you are willing to ask for and read the answers).

Another thing is that the translation capabilities of AI, will help us to be more technology agnostic. For instance, IBM and Intel are working on projects that aim to translate critical legacy software in the financial industry, written in COBOL, that is currently very difficult to maintain.  
One can easily imagine that similar technologies can be used to translate CPU to GPU code.

In the long term, it seems that AI will create another language abstraction, in which AI will work as a kind of compiler to translate a natural language specification into executable code. A key difference will be the development process can be much more interactive, the AI might, for instance, ask questions about aspects that aren’t well defined.

Overall, we can see the role of software developer shift more towards that of a development manager, with the actual development being done by AI. As AI keeps getting better in writing code (whereas the overall human ability to write code is probably rather constant), we might hope that this evolution will result in better, more robust, and correct software.

Nevertheless, although AI might take care of more and more aspects of the development process, knowledge about the underlying technology, the low level software, and the hardware, will still help you to create better code, just like a good supervisor needs to understand the work that they supervising.