WASP CC&SE Module 2, assignment 1:

Getting to know pynblint

Rasmus Kjær Høier

- What did you learn, in your individual session, about static analysis for ML and the pynblint tool?
    - The tool pynblint is super easy to install and use and provides useful feedback. I mainly received feedback on lack of explanatory markdown text, too long cells and out of order execution. Lack of markdown text and too long cells can be problematic because it makes it harder for collaborators to understand the notebook. Out of sequence cell execution order is a major issue because it can prevent reproducibility. I was aware of the importance of these things beforehand yet I am still inclined to make these same kinds of mistakes again and again, so I think pynblint can help me avoid these kinds of issues in the future.
- Will pynblint be useful to you in your WASP PhD project? Why or why not?
    - I don't use notebooks much in my work, and when I do it is mainly julia Pluto notebooks, so I don't think I will use pynblint much. But if I happen to have a course assignment, which requires me to write a python notebook, then I will use pynblint.
- Ideas for how the tool could be improved?
    - I think the tool is already very good. Short term it would make sense to just add a lot of common sense best practice tips to it, and get feedback from users on which features where helpful. Long term it would be interesting to have a linter that works for multiple Jupyter kernels like Julia, R, Scala, Matlab, Scheme, or even a reactive Python kernel.
- What do you see as the limits for static analysis tools in ML? For code, models, and for data?
    - I see a couple of different limitations. Some are tied to the specific limitations of the Jupyter .ipynb format, and some are more user experience oriented. I don't use notebooks for ML prototyping, training or deployment. I only use them for data visualization and occasionally for coursework.
    - In my opinion the biggest appeal of a notebook linter is the ability to spot when cells have been run out of sequence, which is important to ensure reproducibility. However, perhaps that is just a bandaid on a fundamental design issue with the current Python Jupyter kernel. The last five years a number of reactive notebooks have appeared. I am mostly familiar with the Pluto framework for reactive julia notebooks, but there are quite a few for different reactive notebook frameworks for different languages (including Python). [1] mentions a few of them. In Pluto the kernel keeps track of cell interdependencies, which means updating one cell causes all dependent cells to also update, which entirely prevent issues related to cell execution order. So I think that we will see reactive notebooks become more popular in the future, which will solve the "reproducibility issue of out of sequence cell execution", which in turn will make static linters less important. Another problematic feature of ipynb Jupyter notebooks is that they are not version control friendly, requiring you to install separate tools in order to compare changes between commits within a notebook. This issue doesn't exist in Pluto since the notebook is a simple text file (with .jl extension).
    - A more user experience oriented limitation is that users might make more use of a linter that automatically gives suggestions in real-time as you code. Editors already support various

plugins for simple code completation as well as AI powered tools like github copilot, which give realtime suggestions. For a lot of users that is probably preferable. Another issue is that some users might not be comfortable with command line tools. I think it makes sense to implement this tool as a commandline tool, since this makes it easier to apply it on remote machines, but for some users a GUI might be preferable.

[1] Perkel, 2021, Nature, url: https://www.nature.com/articles/d41586-021-01174-w