Pynblint session with Luigi Quaranta

Viktor Nilsson

June 2022

1 The session

My session with Luigi Quaranta started with us discussing and documenting my familiarity and use of notebook based coding, primarily with Jupyter notebooks. We also made an assessment of my own perceived software engineering skills and experience. Then we moved on to a hands-on trial of Luigi's static analyzer **pynblint**. First, he walked me through how to install pynblint in a new venv-environment. Then, in order to try the tool out, we run the analysis on two different Jupyter notebooks from my work. The tool reported a handful of problems detected in them, which we went through one by one.

Finally, we talked about whether a static analyzer for Jupyter notebooks would be of benefit to my team. Also, Luigi asked what I thought about the tool, how likely I was to use it, etc. I consider the effort of creating a static analyzer very important, since I find that many of the luxuries from modern coding are lost in the notebook setting. If the tool delivers what it promises, I am thereby very likely to use it almost as often as I use notebooks. However, this still depends on how intelligently the problems are detected; if naïve methods are used, the warnings might be more clutter than help.

Finally, I gave some suggestions about what improvements I would like to see. These were:

- Integration into IDEs like VSCode and JupyterLab for smoother, automatic detections and suggestions.
- Detect bad figures, such as those that are too small.
- Leave out warning for the final empty code cell that Jupyter tends to generate automatically.

2 Takeaways

Before the activity, I did not know about the pynblint project, and it was nice to see that it could give useful advice about my notebooks. I had seen some static analysis of notebooks before, in IDEs, but it was a welcome surprise to see that this is becoming even an academic discipline, as Luigi is a PhD student. I also learned that pynblint can indeed detect a large base of desired best practices such as: version control usage, code modularization, utilization of markdown for comments and documentation, data availability, etc.

I believe that static analysis and other types of automatized help will improve and get increasingly employed by data scientists using notebooks. Simultaneously, I think that it will never be as clean as static analysis for "classic" programming. One reason is that the purpose of notebooks is being inherently interactive. One wants to modify small portions of code while doing experiments, which means that code does not usually execute sequentially, only after the notebook is finished is this done, taking away some of the benefit.