WASP Software Engineering and Cloud Computing Course

Assignment 1 Module 2 - Reflection on pynblint

Simona Gugliermo

What did you learn, in your individual session, about static analysis for ML and the pynblint tool?

The main take away from the meeting with Luigi Quaranta is the discovery of the pynblint tool and more generally of static analysis tools as I was not aware. I learned the basic concepts for designing a notebook and the common mistakes that make notebooks difficult to read and, at worst, not working properly, e.g., when cells are not designed to run in linear order. Moreover, this session motivated me to improve the way I write code in any kind of programming language, not just when using notebooks. I feel that now I am more aware of the importance of complying with code writing "best practices".

Will pynblint be useful to you in your WASP PhD project? Why or why not?

Pynblint could be useful for me during my PhD project because I program in Python yet I do not use notebooks. However, it happens that I come across them for assignments or projects. Moreover, I do not exclude the possibility to use notebooks in the future if, for example, I want to share a repository that is easy for the user to run.

Ideas for how the tool could be improved?

The feedback I immediately shared with Luigi was to add more information about the number of excess lines per cell. In fact, pynblint prints a warning signal if the number of lines in a cell is greater than a user-defined treshold. However, it might be interesting for the user to also know *how many* lines the cell exceeds. For example, if it is only one line then the warning signal can be ignored. On the other hand, if the number of excess lines is high, the user should consider reducing the size of the cell. Another possible improvement is the creation of a more user-friendly interface. Finally, a personal and general comment is that it would be nice to have such a tool working with all programming languages, since the "best practices" of writing code are similar.

What do you see as the limits for static analysis tools in ML? For code, models, and for data?

As far as I understand, static analysis tools focus on the "best practices" of writing code without running the code itself. This might be a limitation when applying them in ML contexts with respect to code, models and data. For example, a Neural Network (NN) can be written correctly in terms of code and compiled without any errors. However, this may not be relevant as the NN may not accurately predict the output thus being of little interest to the user. Moreover, notebooks code writing "best practices" are well known, do "best practices" exist when writing code for ML applications? Are these well established? On the other hand, it would be interesting to use ML to improve static analysis tools thus making them more flexible and user-friendly.