WASP PhD Course on Software Engineering and Cloud Computing - `pynblint` reflection

Robert Gieselmann, robgie@kth.se

2022

During the session with Luigi Quaranta I learned about `pynblint`, a static analysis tool for python jupyter notebooks. With this new tool I analyzed some of my own notebooks and evaluated them in terms of coding style and overall cleanliness (comments, empty spaces, ...).

What did you learn, in your individual session, about static analysis for ML and the pynblint tool? In the single session, I learned that static analysis tools can indeed be useful to promote good software development practices even in prototyping environments like Jupyter notebooks. The pynblint tool was easy to use and quickly generates a summary of high-level properties that can be useful to assess the quality of a notebook in terms of coding and documentation. Examples include the number of cells, the length of cells, the number of comments, or the occurrence of empty cells. It also provides customization, such as setting a maximum number of cells allowed before a warning is triggered in the summary report.

Will pynblint be useful to you in your WASP PhD project? Why or why not? My research is at the intersection of robotics and machine learning. I program mainly in Python and C++, but rarely use Jupyter notebooks. Instead, I usually use standard Python scripts within an IDE such as `PyCharm`/ `Clion`. This is mainly because I find it cumbersome to organize larger projects with many files using Juypter notebooks. Therefore, at this time, I don't think `pynblint` will be useful to me as part of my WASP PhD project. Nonetheless, I think Juypter notebooks are great for visualizing and sharing preliminary ideas within a team. I might use them and tools like pynblint in my future projects.

Ideas for how the tool could be improved? It could be improved by providing an add-on that integrates `pynblint` into the Jupyter notebook interface. Time constraints are probably one of the main reasons preventing the widespread use of code analysis tools. By reducing the number of steps required to perform a static analysis, I believe more researchers would integrate the tools into their workflow.

What do you see as the limits for static analysis tools in ML? For code, models, and for data? First of all, the same limitations apply to static analysis tools applied to ML (code, models, data) as to any other code in general. Since static analysis does not execute the code, it is very difficult to ensure that the desired runtime behavior is correct. Probably this problem is exacerbated in the ML/deep learning environment where models are often complex and e.g. neural networks are considered black box models. Static analysis of data could also be difficult because data sets vary widely and quality criteria can be very problem-specific. Useful tools may therefore require a certain level of domain knowledge.