WASP Software Engineering Course: Assignment 01

Arka Ghosh, PhD student at Department of Computing Science, Umeå University, Sweden, arkag@cs.umu.se

It was a very interactive session with Luigi and I have learned the following things. The pynblint is a static analyzer tool that is tailored to analyse and lint the jupyter notebook (source code) at a very deep level. The pynblint takes a jupyter notebook as input from the user and it returns a detailed report of various types of programmatic, stylistic errors and overall statistics such as several lines, empty cells, unexecuted cells, comments and many more. The command line visuals tend to be black and white but the pynblint's command-line interface is very well decorated with bright colours and is well-spaced. The interface looks good. I have been using the jupyter notebook for quite some time and in my opinion, this tool can help to polish and compact the source code which will promote writing clean code writing, good documentation and code reproducibility. Students and professionals of data science backgrounds will be benefitted immensely from using pynblint.

I am a first-year WASP PhD student and I am still trying to formulate my research in a plausible direction to address the research questions and bridge the existing research gaps in the literature. After theoretical formulation when I will be working on implementation then pynbilnt might come in handy for my project. Thus, it will be too early to determine the applicability of pynblint in the context of my project.

During the interactive session with Luigi, we talked about this. I have proposed an “At a glance” section at the end of the report generated by pynblint, where the most important features or issues of the input source code will be displayed in table format or word cloud format, whichever is convenient. The purpose of this “At a glance” section is to give an overview of the most prominent linting issues to the user. It will help the user to understand the issue in a short amount of time and then the user can take corrective action to modify their source code more efficiently and quickly.

Static analysis tools do not support all programming languages. False positives and false negatives are produced by automated static tools. It does not find vulnerabilities introduced in the runtime environment. Also, static analysis tools can provide a false sense of security that everything is being addressed, which leads to more errors in source code. In a group project where many people are working together, one has to be very careful while using static analysis tools as it may suggest some correction that is not meant to be corrected for testing purposes.