WASP Software Engineering course
Assignment 1

Sourasekhar Banerjee

July 5, 2022

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

During the individual session, we executed two ipython notebooks in the Pynblinter tool. These notebooks are related to deep learning and federated learning. The results that I received from the linter describe the structural quality of the notebook. The linter produces the statistics of the cells, markdown usage, and code modularization. One typical result for both notebooks is that cells have been executed in a non-linear order. The linter recommends to re-run the notebook from top to bottom to ensure the code is reproducible. Another result is that the notebook is too long, and the suggestion is to divide the notebook into two. The total number of cells should be below the threshold. Other suggestions from the linter are not to keep empty and non-executable cells and to move all import statements to the first cell to make the notebook dependencies more explicit. I think these suggestions are constructive for organizing your notebook, especially before publishing it publicly. The output of the linter can be saved in the JSON format.

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

For my PhD project, which is in the area of federated learning, I am more comfortable using IDEs such as Pycharm-community and Visual Basic for coding. I also use GitLab as a version control system to develop and manage the codes. I only use notebooks to produce results from the files and plot graphs. Perhaps pyblint is more into the static analysis of notebooks, so before showing the results to the supervisors, we can perform static analysis to check if the code is organized correctly or not. It will make the notebook more interpretable. Also, notebooks are handy for showing demonstrations. Before the demo, we can use the pynblint to analyse our notebook. For collaborative projects, the static analysis would also be helpful.

3 Ideas for how the tool could be improved.

The current version of the pynblint focuses on the notebook's structural features and doesn't check the syntax of the code. It is not measuring the quality of the code (e.g., syntax, coding style, indentation); it is measuring how organized the notebook is. One of the authors of that tool acknowledged that in the next version, they would incorporate it. Another drawback I noticed is that the threshold for the number of cells in the notebook is 50, which is very low. So they can extend this threshold from 50 to 80. A few more things can be integrated to make the linter handier, like a spell checker, syntax suggestions, etc.

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

As per my understanding, pynblint is not very useful to me except for organizing the notebook. As pynblint is not looking at the content of the code, it is not helpful to explain and improve the code; only the quality of the notebook will be improved using this tool. Moreover, ML or deep learning depends on code, model, and data. If the work is more data-centric, then the Jupyter notebook is a good option for performing EDA (Exploratory Data Analysis). But again, what kind of data the notebook handles (e.g., images, text, tabular, etc.) can not be explained by the pynblinter. From the coding point of view, the linter needs to provide more details, such as how many cells are processing data and which cells are processing the model. Which cells are producing results? These things need to be present in the statistics.