WASP LINT Reflection Assignment

Jonas Hansson

September 2022

In my session I got a chance to test the quality of my notebooks. For my session I chose to test a notebook where I had prototyped an NLP agent. At my session I realised that my notebooks tend to be long, partly because I develop, explore, evaluate, and test in the same notebook which can make them excessive. I tend to split the notebooks into chapters, but it might still be better practice to actually split the notebooks up and make them slightly shorter. For instance, the exploration phase is not necessary in the the testing phase. More specific to my ML code was a tendency to have too long code snippets and too few MD blocks which I think is due to my mindset being primarily set on developing ideas for myself. Therefore, the LINT serves a purpose of reminding me of writing my notebooks with more emphasis on my coauthors.

In the session I also found out that I leave quite a few deadcode blocks, which once again is caused of my very explorative way of coding. I was also reminded of the importance of virtual environments for reproducability which is very important for myself as a scholar.

Based on my session it is quite apparent that LINT is very useful for me. Unfortunately, pynblint is made for Python notebooks and I mainly work with Julia notebooks. I think the takeaways are the same, and I can use the feedback from pynblint to improve my Julia code too, but the tool pynblint will not be my LINT:er of choice. But a similar tool, targeting the Julia standards would be of great help for me and my PhD project.

Occasionally I write python notebooks, primarily when I do ML. For these cases I think there are a couple of features that I would like to see in pynblint. Namely: 1) A better userinterface. For instance, one could make the LINT more like an app or editor plug-in so one can more easily see where the conventions are broken, rather then getting which lines in the notebook that are troublesome. 2) In the same awe, one could also add interactive fixes where the LINT proposes updates and where these can be accepted like when you do a git-merge.

When it comes to developing ML code in notebooks there are many common pitfalls that a LINT tool could catch. Like splitting data set into test, validation, and training sets. In my own experience I feel that most coding issues when doing ML are related to the used package which are inherently hard for a LINT tool to fix. But there are many standards when doing ML, like how to perform transfer learning where a LINT tool could be used to recommend the standard methods of performing the tasks. In the transfer learning case it is easy to train too many layers which is very hard to catch purely based on the results. I am sure there are many more situations where there are good standards which a LINT could promote to make the ML code better.