Pynblinter Reflection

During the session with Luigi we used the Pynblinter tool on two of my ipython notebooks. One was a notebook I use to teach a class on computer vision and another was a notebook for personal use for testing an idea. I learned about how it works and got a more defined idea of what is considered "good practice" for a python notebook, for example keeping all the imports in one cell at the beginning and keeping cells below a certain length.

I really liked the idea of Pynblinter and I thought the presentation was quite clear and understandable. I think it will be very useful for python notebooks that I am planning to show and share with other people, for example if I want to use the notebook as part of a class. However, at least for machine learning specifically I think I will be ignoring the cell size comments sometimes. Model definitions cannot really be split apart and are usually far longer than other cells. I also prefer to have a long cell at the end with all the relevant code in one place that can be easily transferred to a python script as an example or reference to be used. Another limitation that I can think of is that I often use python notebooks for quickly prototyping ideas and comparing results. I don't think I will be using Pyblinter at all for these types of notebooks because running cells out of order or having extremely long ones is purposeful instead of an accident.

I am not sure how to further improve Pyblinter for prototyping notebooks since I am not familiar with what is possible to detect. It could be nice if it could detect cells that have not been run in a very long time relative to the others in case they are forgotten cells that need to be cleaned up. Another would be finding overloaded variables that are defined in one cell and later overwritten in another. I think it would be very useful if it could track every instance a certain variable is used or changed. The most useful change I can think of is if Pyblinter could be integrated into the notebook and act like PyCharm, but that is probably more difficult to accomplish.

From my point of view there's one big limit preventing static analysis tools in from becoming more widespread in machine learning. Static analysis needs some sort of widely agreed on coding standard and in my experience, there is a huge variety of coding styles for machine learning. Some people are coding extremely pythonically, others code like it could be straight out of MatLab, and there's everything in between. Unlike industry, I am not sure if there is a standard which prevents a static analysis tool from being able to give deeper analysis and advice.

I think in the future I am looking forward to using Pyblinter for notebooks that I will be presenting to other people. While that might not be very often, I am definitely looking forward to future improvements.