Reflections on static analysis tools for Machine Learning

The session provided a practical, first-hand introduction to static analysis tools and code linting for ML. It was made clear that their purpose in general is to check the code—and in this case also the structure of the notebook itself—for known mistakes and unclarities. All this without compiling the code. More specifically, during the tutorial session, I learned how to use the Pynblint-tool on my own Jupyter notebooks. The suggestions made by Pynblint were highly relevant, and I learned that static analysis tools for notebooks can be immensely valuable if one is to share it with co-workers or external parties. Also, the option to add custom rules may be useful to enforce certain stylistic and functional standards to notebooks.

I strongly believe that Pynblint will be useful in my PhD-studies for two reasons. Firstly, I use notebooks extensively to convey results from data analysis and Machine Learning (ML) to my project co-workers—even if they do not have a data science background. For this reason, it is imperative that the notebooks have good "style" and that they can be run top to bottom without the need for modifications. A static analysis tool such as Pynblint has potential to make this process more structured for me as a programmer, as its suggestions can serve as a checklist before the notebook is shared. Secondly, my own personal notebooks tend to get messy—and that it is important to keep this in check. While it may sometimes be beneficial to write code quickly without regards to readability for, e.g., rapid prototyping, it is also essential to be able to keep a codebase neat and orderly. Pynblint can make this process less painful, by making it possible for me to quickly check each individual notebook for style issues.

One way to potentially improve Pynblint is through the creation of a web-based user interface, or alternatively through direct integration into Jupyter notebooks. While I am a little ambivalent on the former, the latter seems to be the better solution. The benefit for me lies in the fact that I can check each notebook as I work on them, rather than having to open another application. There are also issues with using a web-based interface, such as the need to maintain a server-side connection to access the Pynblint tool. Users may also be hesitant to upload their notebooks to a remote server for security reasons, as they may not want sensitive information to be exposed to anyone outside the user's network. Another way to improve Pynblint is to add features to the current version, such as support for other languages often used in notebooks, such as Scala. Also, the emerging use of multilingual notebooks could restrict Pynblint's functionality, as the existing languages may not be fully supported.

In general, I see the role of static analysis- and linting tools in ML as an interesting and useful addition to the toolbox of ML developers. They can add new tools and features to prevent ML developers to make common mistakes with both models and data. However, I think that static analysis tools should not only be regarded as a means to improve the performance of ML systems, but also as a valuable tool for learning about ML. In a sense, a static analysis tool could carry a large part of the collective knowledge that we have gathered about practical ML problems. They could then be used to teach new developers about ML techniques with which they are currently unfamiliar.

On the other hand, a limitation of static analysis tools in ML is that they do not allow for any meaningful way of checking for bugs since these tools are not designed to deal with errors in the code. The issue is compounded by the fact that ML programs cannot be properly analysed

without having a full knowledge of the underlying ML engine and of the data in the ML system. To be able to perform analysis, the static analysis tool must not only know about the source code and its data, but also the internals of the ML algorithm. As such, static analysis tools do not offer any sort of practical assistance to ML developers who cannot be sufficiently trained to do such analysis. As a result, I would say that static analysis tools are a valuable contribution to ML, but they should not be regarded as a panacea for ML errors and problems.