Pynblint: a Static Analyzer for Python Jupyter Notebooks

Luigi Quaranta
University of Bari
Bari, Italy
luigi.quaranta@uniba.it

Fabio Calefato
University of Bari
Bari, Italy
fabio.calefato@uniba.it

Filippo Lanubile
University of Bari
Bari, Italy
filippo.lanubile@uniba.it

ABSTRACT

Jupyter Notebook is the tool of choice of many data scientists in the early stages of ML workflows. The notebook format, however, has been criticized for inducing bad programming practices; indeed, researchers have already shown that open-source repositories are inundated by poor-quality notebooks. Low-quality output from the prototypical stages of ML workflows constitutes a clear bottleneck towards the productization of ML models. To foster the creation of better notebooks, we developed Pynblint, a static analyzer for Jupyter notebooks written in Python. The tool checks the compliance of notebooks (and surrounding repositories) with a set of empirically validated best practices and provides targeted recommendations when violations are detected.

CCS CONCEPTS

• Software and its engineering → Software maintenance tools;
  Software configuration management and version control systems;
• Human-centered computing → Collaborative and social computing systems and tools.

KEYWORDS

Computational notebooks, lint, software quality, data science, machine learning

1 INTRODUCTION

The massive adoption of AI-based technologies in the modern software industry is raising new intriguing challenges, many of which concern the shift of machine learning (ML) model prototypes into production-ready software components. Often, a variety of factors contribute to rendering this shift difficult and costly to achieve; these range from technical matters – like the complexity of reproducing lab model performances in live systems – to human aspects, arising from the coexistence of varied backgrounds and perspectives in multidisciplinary teams [4].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CAIN’22, May 16–24, 2022, Pittsburgh, PA, USA
© 2022 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-9275-4/22/05...$15.00
https://doi.org/10.1145/3522664.3528612

In this context, a prominent role is played by the tools that practitioners adopt at the various stages of ML workflows. Tool misuse can hinder the quality of team collaboration and constitute a bottleneck towards the productization of ML models. Notably, computational notebooks represent a prime example. In the last few years, they have established themselves as the tool of choice of many data scientists for activities comprised in the early stages of ML workflows, from data exploration to model prototyping. Their most popular implementation, Jupyter Notebook,1 combines code, documentation, and multimedia output in an interactive narrative of computations, providing unparalleled support for fast experimental iterations and lightweight documentation of experiments. However, besides the evident benefits they bring, Jupyter notebooks have been criticized for inducing bad programming habits and offering limited built-in support for software engineering best practices [1, 2]. Indeed, researchers have already shown that notebooks often contain poor quality code and that – in spite of their original vocation – they typically end up being messy and scarcely documented [5, 7].

Commonly, poor-quality and hardly-reproducible notebooks, written by data scientists in the early stages of ML workflows, get in the way of the model productization process. Indeed, transitioning from ML model prototypes to production-ready ML components often entails, in practice, the consolidation of experimental code from notebooks into structured and tested codebases [3]. Under such circumstances, low-grade notebooks might represent an expensive bottleneck and a potential source of technical debt.

2 PYNBLINT

In our previous work [6], we collected and validated a catalog of 17 best practices for professional collaboration with computational notebooks. Our guidelines foster a use of notebooks aware of software engineering best practices while preventing potential drawbacks. In the light of these findings, we have developed Pynblint,2 a static analysis tool for Jupyter notebooks written in Python.

Besides being the most popular notebook platform to date, Jupyter Notebook has inspired the design of most modern computational notebook implementations; the majority of them currently adopts the .ipynb JSON-encoded format and offers the same core functionalities as Jupyter (e.g., Google Colaboratory3). Furthermore, most Jupyter notebooks are written in Python. For these reasons we chose Python Jupyter notebooks as the target of our static analysis tool.

1https://jupyter.org
2https://github.com/collab-uniba/pynblint
3https://colab.research.google.com
2.1 Definition of the linting rules

Pynblint implements a set of checks to assess the quality of notebooks and the code repositories they belong to. We derived each check (hereafter referred to as linting rule) as an operationalization of the best practices from our catalog. These best practices range from recommendations for better traceability and reproducibility of computations (e.g., “Use version control” and “Manage project dependencies”) to hints for code quality enhancements (e.g., “Stick to coding standards” and “Test your code”), and clues for a more consistent use of notebooks narrative capabilities (e.g., “Leverage Markdown headings to structure your notebook”).

Not all best practices could be fully operationalized. For instance, we found no practical way to verify compliance with the recommendation “Distinguish production and development artifacts” (e.g., “poetry lock files”).

As a static analyzer, Pynblint can be also leveraged in the context of CI/CD pipelines. For instance, once installed on a CI/CD server (e.g., GitHub Actions), it can be invoked alongside other quality assurance tools at build time. According to user preferences, a build might fail if the linting process reveals potential notebook defects.

This feature sets Pynblint apart from Julynter [5], which is to provide cue to the best of our knowledge – the only alternative Jupyter linting tool. Julynter was developed as a plug-in of Jupyter Lab, the evolution of the Jupyter Notebook IDE; it performs real-time checks on the quality and reproducibility of Jupyter notebooks while also providing improvement recommendations. However, Julynter can only be executed as a live assistant within Jupyter Lab and cannot be leveraged off-line, as a standalone module. Therefore, while useful during notebook writing, Julynter cannot be integrated into CI/CD pipelines and cannot be adopted as a per-commit hook. Moreover, being tied to the Jupyter Lab environment, it cannot be used to analyze notebooks produced with different platforms (e.g., Google Colab or the Kaggle Notebooks IDE), even if they comply with the standard Jupyter format (.ipynb).

3 CONCLUSION AND FUTURE WORK

We implemented Pynblint, a static analyzer for Python Jupyter notebooks. To refine the tool, we are currently conducting a formative study with experienced Jupyter users; at the same time, we are developing a web front-end for Pynblint, to make it easily accessible by novice data scientists with limited or no command-line experience. Furthermore, we will validate the tool with a field study, involving data science professionals from multiple companies. As a result of the validation process, we expect to expand the core set of available linting rules and to make the tool proactive, i.e., capable of automatically fixing a selected set of violations.

REFERENCES

[1] Soumya Chattopadhyay, Ishita Prasad, Austin Z Henley, Anita Sarma, and Titus Barik. 2020. What’s wrong with computational notebooks? Pain points, needs, and design opportunities. In Proc. of the 2020 CHI conference on human factors in computing systems. 1–12. https://doi.org/10.1145/3313831.3376729

[2] Joel Grus. 2018. I don’t like notebooks. https://www.youtube.com/watch?v=7JpDExbUQ3i&ts=0. Further. We implement Pynblint, a static analyzer for Python Jupyter notebooks. To refine the tool, we are currently conducting a formative study with experienced Jupyter users; at the same time, we are developing a web front-end for Pynblint, to make it easily accessible by novice data scientists with limited or no command-line experience. Furthermore, we will validate the tool with a field study, involving data science professionals from multiple companies. As a result of the validation process, we expect to expand the core set of available linting rules and to make the tool proactive, i.e., capable of automatically fixing a selected set of violations.

REFERENCES

[1] Soumya Chattopadhyay, Ishita Prasad, Austin Z Henley, Anita Sarma, and Titus Barik. 2020. What’s wrong with computational notebooks? Pain points, needs, and design opportunities. In Proc. of the 2020 CHI conference on human factors in computing systems. 1–12. https://doi.org/10.1145/3313831.3376729

[2] Joel Grus. 2018. I don’t like notebooks. https://www.youtube.com/watch?v=7JpDExbUQ3i&ts=0. Further. We implement Pynblint, a static analyzer for Python Jupyter notebooks. To refine the tool, we are currently conducting a formative study with experienced Jupyter users; at the same time, we are developing a web front-end for Pynblint, to make it easily accessible by novice data scientists with limited or no command-line experience. Furthermore, we will validate the tool with a field study, involving data science professionals from multiple companies. As a result of the validation process, we expect to expand the core set of available linting rules and to make the tool proactive, i.e., capable of automatically fixing a selected set of violations.

REFERENCES

[1] Soumya Chattopadhyay, Ishita Prasad, Austin Z Henley, Anita Sarma, and Titus Barik. 2020. What’s wrong with computational notebooks? Pain points, needs, and design opportunities. In Proc. of the 2020 CHI conference on human factors in computing systems. 1–12. https://doi.org/10.1145/3313831.3376729

[2] Joel Grus. 2018. I don’t like notebooks. https://www.youtube.com/watch?v=7JpDExbUQ3i&ts=0. Further. We implement Pynblint, a static analyzer for Python Jupyter notebooks. To refine the tool, we are currently conducting a formative study with experienced Jupyter users; at the same time, we are developing a web front-end for Pynblint, to make it easily accessible by novice data scientists with limited or no command-line experience. Furthermore, we will validate the tool with a field study, involving data science professionals from multiple companies. As a result of the validation process, we expect to expand the core set of available linting rules and to make the tool proactive, i.e., capable of automatically fixing a selected set of violations.

REFERENCES

[1] Soumya Chattopadhyay, Ishita Prasad, Austin Z Henley, Anita Sarma, and Titus Barik. 2020. What’s wrong with computational notebooks? Pain points, needs, and design opportunities. In Proc. of the 2020 CHI conference on human factors in computing systems. 1–12. https://doi.org/10.1145/3313831.3376729

[2] Joel Grus. 2018. I don’t like notebooks. https://www.youtube.com/watch?v=7JpDExbUQ3i&ts=0. Further. We implement Pynblint, a static analyzer for Python Jupyter notebooks. To refine the tool, we are currently conducting a formative study with experienced Jupyter users; at the same time, we are developing a web front-end for Pynblint, to make it easily accessible by novice data scientists with limited or no command-line experience. Furthermore, we will validate the tool with a field study, involving data science professionals from multiple companies. As a result of the validation process, we expect to expand the core set of available linting rules and to make the tool proactive, i.e., capable of automatically fixing a selected set of violations.

REFERENCES

[1] Soumya Chattopadhyay, Ishita Prasad, Austin Z Henley, Anita Sarma, and Titus Barik. 2020. What’s wrong with computational notebooks? Pain points, needs, and design opportunities. In Proc. of the 2020 CHI conference on human factors in computing systems. 1–12. https://doi.org/10.1145/3313831.3376729

[2] Joel Grus. 2018. I don’t like notebooks. https://www.youtube.com/watch?v=7JpDExbUQ3i&ts=0. Further. We implement Pynblint, a static analyzer for Python Jupyter notebooks. To refine the tool, we are currently conducting a formative study with experienced Jupyter users; at the same time, we are developing a web front-end for Pynblint, to make it easily accessible by novice data scientists with limited or no command-line experience. Furthermore, we will validate the tool with a field study, involving data science professionals from multiple companies. As a result of the validation process, we expect to expand the core set of available linting rules and to make the tool proactive, i.e., capable of automatically fixing a selected set of violations.