Reproducible HTML Notebooks

Computer Science Research Methods
INFO4990
Semester 1, 2023
Assignment 2

Tanaka Chitete

Supervisor:
Dr. Rahul Gopinath

The University of Sydney
School of Computer Science

April 23, 2023

Contents

Chapter 1

Literature review

Reproducibility represents perhaps the most essential aspect in any science discipline and sharing someone's work by means of a Jupyter notebook [source] is meaningful only to the extent which the notebook can be reproduced. With this in mind, we have identified that the high-level process of reproducing a notebook consists of *restoration*, *dissemination*, and *collaboration*. To begin, we enumerate the complexities associated with saving and restoring a notebook. Thereafter, we discuss the complications related to sharing notebooks in a lightweight manner. At last, we evaluate means for facilitating collaboration on a single notebook.

1.1 Restoration

Research Question 1: *How could we achieve the saving and restoration of a partially computed notebook?*

Existing works concerning achieving reproducibility with Jupyter notebooks are either *state-based* [1, 2] or *provenance-based* [3, 4]. At the time of conducting this literature review, the bodies of work for both approaches is quite small.

When examined in detail, there are several challenges associated with reproducing a Jupyter notebook. Not only must the code contained within the notebook be fully replicated, so to must the underlying Python run-time environment. In relation to the run-time environment, more subtle problems arise in replicating referenced data, resolving external library dependencies, and re-initialising variable states [2].

With the aforementioned in mind, Wannipurage, Marru, and Pierce [2] determine that restoring a previously saved notebook can only be made possible through the satisfaction of each of the following requirements:

1. The code within the notebook is portable.
2. Python run-times are identical.
3. Replicate referenced local data.
4. External library dependencies should be replicated or installed if missing.
5. Python run-time variables should be reinitialised.

We note that requirements 1 and 2 are both external states, and thus, do not depend on the execution of the notebook. Requirements 3, 4, and 5, however, are internal states, and in turn, are internal to the Python run-time and the order of execution for the cells in the notebook. In situ, it can be observed that only requirements 1 and 2 are satisfied in most scenarios.

In order to achieve absolute state capture of a notebook, Wannipurage, Marru, and Pierce [2] present their solution–a dockerised customised IPython kernel which captures Jupyter Notebook state [source].

1.2 Dissemination

Research Question 2: *How could we achieve the lightweight sharing of a notebook–including all dependencies–with minimal expectations on the receiver?*

"Dependency hell" is a situation that plagues essentially every piece of software at some point during its lifecycle. Jupyter notebooks are no exception.

Whilst each of these software packages prove to be excellent in their usage, grave problems arise when attempting to execute a notebook which uses some of these packages, on a different system/environment. More often than not, the receiver attempting to use the notebook won't have the complete set of packages installed. Resultantly, they encounter roadblock after roadblock attempting to resolve these dependency issues, if they are even able to do so. With this pain point in mind, finding a means to ameliorate the problem of dependency hell would prove to be crucial in catering to those users without a background in software engineering. In order to resolve dependency management, one such means would be to embed

1.3 Collaboration

Research Question 3: *How could we achieve the collaboration of multiple users on a single notebook?*

Chapter 2

Research outline

Bibliography

(1) M. Juric, S. Stetzler and C. T. Slater, "Checkpoint, Restore, and Live Migration for Science Platforms", 2021.

(2) D. Wannipurage, S. Marru and M. Pierce, "A Framework to capture and reproduce the Absolute State of Jupyter Notebooks", *PEARC 2022 Conference Series - Practice and Experience in Advanced Research Computing 2022 - Revolutionary: Computing, Connections, You*, 2022, DOI: 10.1145/3491418.3530296.

(3) J. Felipe, N. Pimentel, V. Braganholo, L. Murta and J. Freire, "Collecting and Analyzing Provenance on Interactive Notebooks: when IPython meets noWorkflow".

(4) J. F. Pimentel, L. Murta, V. Braganholo and J. Freire, "noWorkflow: a tool for collecting, analyzing, and managing provenance from Python scripts", *Proceedings of the VLDB Endowment*, 2017, **10**, 1841–1844.