Dockerizing DraCor – A Container-based Approach to Reproducibility in Computational Literary Studies

Boerner, Ingo
ingo.boerner@uni-potsdam.de
University of Potsdam, Germany

Trilcke, Peer
trilcke@uni-potsdam.de
University of Potsdam, Germany

Milling, Carsten
cmil@hashtable.de
University of Potsdam, Germany

Fischer, Frank
fr.fischer@fu-berlin.de
Freie Universität Berlin, Germany

Sluyter-Gäthje, Henny
sluytergaeth@uni-potsdam.de
University of Potsdam, Germany

Reproducibility as a Challenge for Computational Literary Studies.

While it is controversial whether reproducing research is possible at all in the Humanities (Peels and Bouter 2018), the empirical methods of Digital Humanities (DH) are particularly well suited for different types of repetitive research (cf. Schöch 2021 for a typology). Consequently, the calls for considering reproducibility in DH research in general and Computational Literary Studies (CLS) in particular have become louder (O'Sullivan 2019).

At the same time, enabling reproducibility poses a significant challenge for research projects in CLS. Even if, as is the case in best practices like e.g. Underwood (2018), Piper (2018a), and Kestemont et al. (2022), both code and data are published, the actual reproduction of the analyses performed can be almost impossible – as also Da (2019) criticized in some cases. Besides issues of FAIR data and documentation of research software, especially the interplay of the components involved in the research process (code, data, environments, infrastructures, etc.) proves to be a hurdle for reproducing research.

In the following, we report on a use case in infrastructure research conducted in the context of the project CLS INFRA. With our container-based approach, which we exemplarily implemented for a CLS study using Docker, we aim to prototypically present a way to fully reproducible research.

Using Docker to Enhance Reproducibility

Using Docker [1] in the IT industry is motivated by speeding up development cycles and the reduction of overhead when deploying applications (cf. Kane and Matthias 2018; cf. Lampert 2022 for practical DH-use).The "DevOps" approach focuses on the interplay of application development ("Dev") and the processes that are necessary to have a tool run on a server ("Ops" for operations) and emphasizes the importance of communication between the development and the operations team. Thus Docker workflows have been introduced because they not only streamline communication processes by providing a meaningful and executable form of documentation, the *Dockerfile*, that contains the steps necessary to build a highly portable, self-contained digital artifact (*Docker image*) that is easily deployed as a container; but also, Docker workflows shift the responsibility of handling software dependencies to the development team: Docker enables the people actually writing the application to specify the environment in which their software should be run.

In CLS research projects we will rarely find teams of development and operation professionals that are in need of communicating better, but we would like to argue, that the attempt of reproducing research could be framed in a similar sense: On the one side, we have an individual researcher (or team) that produces a study that relies on some application that operates on data and, on the other side, researchers wanting to reproduce or verify the results. It becomes evident that some hurdles in the process of reproducing CLS research exist due to a lack of clear communication on how to run the analysis scripts and a tendency to offload the responsibility of setting up an environment in which the analysis could be executed to the reproducing party. A containerized research environment might circumvent these problems: Instead of claiming that scripts "work, at least as of today" (Piper 2018a: xii) on the machine of the developing researcher, it could be guaranteed that a container created from an image containing a runnable self-contained research environment would allow for a reproduction of the study.

Use Case: Dockerizing DraCor

We exemplify the benefits of a Docker-based research workflow by referring to our study "Detecting Small Worlds in a Corpus of Thousands of Theater Plays" [2]. In this study, we tested different operationalizations of the so-called "Small World" concept based on a multilingual "Very Big Drama Corpus" (VeBiDraCor) of almost 3,000 theater plays. The corpora available on the DraCor platform [3] are 'living' corpora (we are still adding more plays) which poses an additional challenge for reproducing our study. Furthermore, our analysis script (written in R) retrieves metadata and network metrics from the REST API [4] of the "programmable corpus" (Fischer et al. 2019). Thus, we had to devise a way of not only stabilizing the corpus but also this infrastructural component.

DraCor provides *Docker images* for its services (API, frontend, metrics service), which allow for setting up a local DraCor environment. [5] For VeBiDraCor we set up a workflow that spins up a Docker container from a versioned bare Docker image of the Dra-

Cor database and ingests the data of the plays pulled from specified Github commits using a Python script. From this container we created a ready to use *Docker image*. [6] Because the build process is modular and documented in a *Dockerfile*, it is also possible to quickly change the API's base image or the composition of the corpus by editing a manifest file that controls which plays from which repositories at which state are included. In a second step, we also dockerized the research environment by building on a *Docker image* of RStudio [7] and adding the analysis script.

Starting an environment with the *docker-compose* file that documents the "pre-analysis state" would allow a researcher wanting to repeat our analysis to run the script herself/himself (same data, same code) or even change the version of the data used for the analysis, e.g. use another corpus. Thus, different scenarios of repeating research (same code, different data; different code, same data; cf. Schöch 2021) could be implemented.

After we ran our final analysis inside a Docker container, we created a second image of the infrastructure, which can be used to recreate the "post-analysis state" and allows for inspection and verification of the results of our study in the same environment that we used.

Acknowledgement

This work is part of CLS INFRA (https://clsinfra.io), which is funded from the European Union's Horizon 2020 program (grant agreement No. 101004984).

Notes

1. https://www.docker.com.
2. Preprint: https://github.com/dracor-org/small-world-paper/blob/conference-version/Detecting_Small_World_Networks__in_a_Huge_Multilingual_Corpus_of_Theater_Plays.pdf.
3. See https://dracor.org; corpora are hosted on Github: https://github.com/dracor-org.
4. https://dracor.org/doc/api.
5. For a Tutorial see https://github.com/dracor-org/dracor-notebooks/blob/f72a7722a3ce9c3fa35ea856c81a108166c96f4d/docker/local-dracor-with-docker.ipynb.
6. https://github.com/dracor-org/vebidracor; Docker images: https://hub.docker.com/repository/docker/ingoboerner/vebidracor-api.
7. We used an image of the Rocker project: https://rocker-project.org; for a guide of how to use our research environment see https://github.com/dracor-org/small-world-paper/tree/develop.

Bibliography

Fischer, Frank / **Börner, Ingo** / **Göbel, Mathias** / **Hechtl, Angelika** / **Kittel, Christopher** / **Milling, Carsten** / **Trilcke, Peer** (2019): "Programmable Corpora: Introducing DraCor, an Infrastructure for the Research on European Drama", in: *DH2019:#»Complexities«.Book of Abstracts* Utrecht. https://doi.org/10.5281/zenodo.4284002

Da, Nan Z. (2019): "The Computational Case against Computational Literary Studies", in: *Critical Inquiry* 45, 3: 601-639.

Kane, Sean P. / **Matthias, Karl** (2018): *Docker: Up and Running*. O'Reilly.

Kestemont, Mike et al. (2022): "Forgotten books: supplementary materials (data and code)". *Zenodo*, February 2, 2022. https://doi.org/10.5281/zenodo.5947206

Lampert, Marcus (2022): "Introduction to Docker", in *: DHd 2022 Kulturen des digitalen Gedächtnisses. 8. Tagung des Verbands "Digital Humanities im deutschsprachigen Raum" (DHd 2022)*, Potsdam. https://doi.org/10.5281/zenodo.6328077

O'Sullivan, James (2019): "The humanities have a 'reproducibility' problem", in: *Talking humanities*. https://talkinghumanities.blogs.sas.ac.uk/2019/07/09/the-humanities-have-a-reproducibility-problem/ [10/31/2022].

Peels, Rik / **Bouter, Lex** (2018): "The possibility and desirability of replication in the humanities", in: *Palgrave Communications* 4, 1: 1–4. https://doi.org/10.1057/s41599-018-0149-x

Piper, Andrew (2018a): *Enumerations: Data and Literary Study*. University of Chicago Press.

Piper, Andrew (2018b): "Data and code for the book Enumerations: Data and Literary Study (Chicago 2018)". *Github*. https://github.com/piperandrew/enumerations [10/31/2022].

Schöch, Christof (2021): "A Typology of Reproducible Research: Concepts, Terms, Examples". *Seminars on Reproducible Research*, org. Kurt De Belder and Peter Verhaar. Leiden University Library, April 29, 2021. Slides: https://dh-trier.github.io/trr/#/ Video: https://youtu.be/ugzgvWl7nAo [10/31/2022].

Underwood, Ted (2018): "Data and Code to Support Distant Horizons". *Zenodo*, March 25, 2018. http://doi.org/10.5281/zenodo.1206317