Reproducibility review of: Automated Extraction of Labels from Large-Scale Historical Maps

Daniel Nüst

2021-06-07

This report is part of the reproducibility review at the AGILE conference. For more information see https://reproducible-agile.github.io/. This document is published on OSF at https://osf.io/anv9r/. To cite the report use

> Nüst, D. (2021, May 6). Reproducibility review of: Automated Extraction of Labels from Large-Scale Historical Maps. https://doi.org/10.17605/osf.io/anv9r

Reviewed paper

> Schlegel, I.: Automated Extraction of Labels from Large-Scale Historical Maps, AGILE GIScience Ser., 2, 12, https://doi.org/10.5194/agile-giss-2-12-2021, 2021.

Summary

The provided workflow could be partially reproduced, notably the text detection and string similarity calculations could be executed and provide sufficiently similar results. The author does a good job of providing all input and output data for each step, however, some manual steps were included which could not be reproduced, as well as a final step due to long execution time. The provided computational environment documentation was a little bit incomplete, but together with the detailed instructions it was usable. The software stack was pretty challenging to install and the workflow is rather rough around the edges, but I have no doubt that all code and data used by the author is shared in the provided repositories and can work as advertised if the computing environment is more closely recreated.

Reproducibility reviewer notes

I started with the repository https://doi.org/10.5281/zenodo.4721174 mentioned in the Data and Software Availability section, and was restarted with the updated version (`v1.1.2` on GitHub) from https://doi.org/10.5281/zenodo.4738397. The Zenodo record links to the GitHub repository https://github.com/IngaSchl/Label-Extraction. Let's get the data:

```
wget -O Label-Extraction-v1.1.2.zip https://zenodo.org/record/4738397/files/IngaSchl/Label-Extraction-v1.1.2.zip?download=1
unzip Label-Extraction-v1.1.2.zip

cd IngaSchl-Label-Extraction-33eae48/01\ Data/
wget https://maps.princeton.edu/download/file/harvard-g6299-h3-1853-l5-geotiff.tif
```

The manual cropping to the area of interest of the data file cannot be reproduced, but the downloaded file seems to include the used data file.

Looking at the the required software set-up of tools that suggest they are installed with `install.sh` scripts, I decide to create a `Dockerfile` to not work in my own system. The `Dockerfile` is published alongside this report. As the used version of Strabo is not mentioned, I use the default (`1.1`).

```
docker build --tag agile-2021-005 .
```

To get have OpenCV 4.0.0.21, I need Python 3.6, and tesseract only installed from the regular sources on Ubuntu 20.04. Let's see if I'm lucky. First try error: "ImportError: libSM.so.6: cannot open shared object file: No such file or directory", adding `libsm6` and others to Dockerfile helped.

```
docker run --rm -it -v $(pwd)/IngaSchl-Label-Extraction-33eae48/:/labelextraction agile-2021-005

# in the container:
python3.6 /strabo/strabo-text-recognition-deep-learning/run_command_line.py --checkpoint-path /strabo/east_icdar2015_resnet_v1_50_rbox \
  --image /labelextraction/02\ Text\ detection/data/input/G6299_H3_1853_L5_subset.png \
  --config /strabo/strabo-text-recognition-deep-learning/configuration.ini

# ImportError: Python version mismatch: module was compiled for version 3.8, while the interpreter is running version 3.6. > fixed in Dockerfile
```

This command creates a number of `.png` files with the labels in the current working directory, which seem to match the files used for text recognition. The `geoJson1.json` created by me mostly matches the one provided by the authors, as the following snippet shows. Most importantly the `geometry` data is the same.

```
reproduced_geojson <- sf::read_sf("IngaSchl-Label-Extraction-33eae48/02\ Text\ detection/data/input/G6299_H3_1853_L5_subset.png_cee5616e-ae62-11eb-bfdb-0242ac110002/geoJson1.json")
original_geojson <- sf::read_sf("IngaSchl-Label-Extraction-33eae48/02\ Text\ detection/data/output/geoJson1.json")
waldo::compare(original_geojson, reproduced_geojson)
```

```
##      old$inclination     | new$inclination
## [51] 87.66269414087618   | 87.66269414087618   [51]
## [52] -125.70669140060288 | -125.70669140060288 [52]
## [53] 55.12467165539782   | 55.12467165539782   [53]
## [54] -121.12247019679201 - 58.87752980320802   [54]
## [55] -60.36161631184866  | -60.36161631184866  [55]
## [56] -124.47221388003645 | -124.47221388003645 [56]
## [57] -96.00900595749452  | -96.00900595749452  [57]
##
##      old$inclination     | new$inclination
## [58] -96.65442504600659  | -96.65442504600659  [58]
## [59] -126.07601155636860 | -126.07601155636860 [59]
## [60] -92.48955292199916  | -92.48955292199916  [60]
## [61] 76.67546873810923   - 76.67546873810922   [61]
## [62] -124.28687697720898 | -124.28687697720898 [62]
## [63] -13.39249775375110  | -13.39249775375110  [63]
## [64] -88.93908830973577  | -88.93908830973577  [64]
```

Manual steps for "03 Additional adjustments" could not be reproduced, so I continued with the "04 Text recognition" step *in the container*:

```
tesseract 04\ Text\ recognition/data/input/ID_24.tif 04\ Text\ recognition/data/output-reproduction/ -l deu --psm 8
#Tesseract Open Source OCR Engine v4.1.1 with Leptonica
#Error in pixReadFromTiffStream: tiled format is not supported

# Can fix files with the following command, but text extraction still empty
# tiffcp 04\ Text\ recognition/data/input/ID_17.tif 04\ Text\ recognition/data/input/ID_17striped.tif
```

So unfortunately I don't have a Windows system, so I seem to encounter a quirk here in the accepted input files with the TIFF library under Linux. I can fix the error message by converting the file, but then the text is not recognised. I have very little doubt though that the instructions provided by the authors don't work as advertised for them.

For the "05 String Similarity" calculations I start a local Jupyter Lab instance with the required packages:

```
mkvirtualenv agile-005
# following steps in the virtual environment

# need to comment out msys2-conda-epoch==20160418, not available on my Linux system
pip install -r IngaSchl-Label-Extraction-33eae48/requirements.txt
pip install jupyterlab

# missing dependencies
pip install xlrd openpyxl

jupyter-lab
```

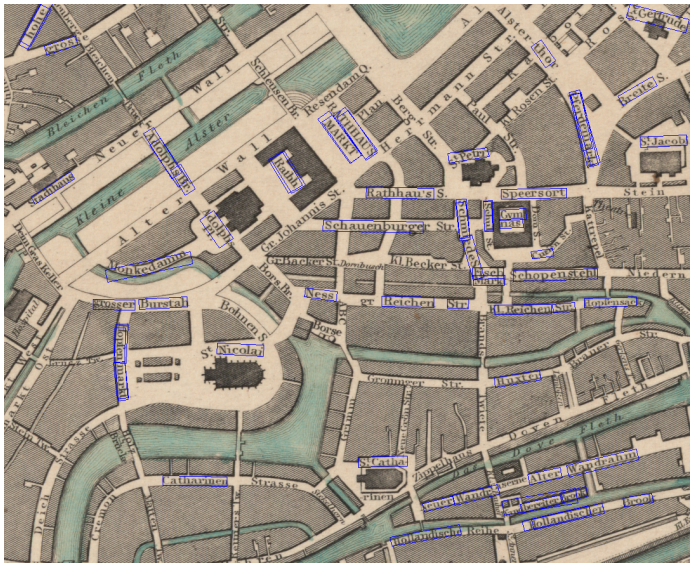

Figure 1: One output file from text recognition run.

I could run all cells in `String_Similarity_by_Levenshtein_Distance.ipynb`, and found the documentation to be extensive and a little bit raw, though very transparent, including tests by the author while developing the workflow etc. At first I got an error reading the `OCR_results.xlsx file`: `XLRDError: Excel xlsx file; not supported`, so I changed the data loading to use `openpyxl`.

```
# via https://stackoverflow.com/questions/65254535/xlrd-biffh-xlrderror-excel-xlsx-file-not-supported
ocr_results = pd.read_excel("data/OCR_results.xlsx", sheet_name='Lindley', header=1, engine='openpyxl')
```

Figure 6 seems to be created using QGIS, but no project file or georeferenced version of the base map was included in the repository.

I did not run the final notebook of step “06 Approximate georeferencing”, because of the advertised run time, but noticed that the shapefile only is included in a ZIP archive and thus the first few cells already don’t work out of the box.

Comments to the authors

- Please consider properly citing the used libraries and tools (see AGILE Reproducible Paper Guidelines for a start); your work would not be possible without `strabo`, `fuzzywuzzy` and many other packages
- The License link in the right hand sidebar of the Zenodo record is broken, but good job properly depositing the GitHub repo on Zenodo.
- It is a bit confusing that you switch between Linux and Windows command lines; suggest to work around that in the future, e.g., by putting all required code in a container
- “Computing time my be extensive: please provide your experiences (”on my machine with 4 CPUs this took 3 hours"), which is more helpful for readers/users.
- Suggest to provide a files for the required dependencies for the Jupyter Notebooks, and to provide instructions on how to use the notebooks - not all readers of your paper will be familiar with Jupyter!
- A list/table in the README file which notebook/scripts creates which figure in the paper is helpful for readers.
- You can probably quickly make your repository Binder-ready by adding GDAL to `apt.txt`
- Don’t just mention where the output files are stored, but what output files I should expect.
- It was a bit confusing to me that some data is using pixel coordinates, while other data is properly georeferenced - maybe that can be documented better.
- `.xlsx` files are a lot less accessible than plain text based formats such as CSV, and are not properly tracked by git - reconsider in next workflow. The same for Shapefile - the data can probably just as well be stored as GeoCSV.

- Consider creating all Figures, especially maps, with code; would have been useful to have access to the georeferenced old map.