Using the OEIL app to "webscrape" data on EU legislation

Michele Scotto di Vettimo *

Updated November, 2020

1 What is the OEIL app?

This app is a translation into a stand-alone executable file of a Python script created for web scraping the European Parliament's Legislative Observatory (OEIL) and EUR-Lex websites. The main purpose of the app is, therefore, to collect information on EU legislation from multiple institutional sources, using EUR-Lex to fill the gaps of the OEIL database and viceversa. Furthermore, the app extracts also the text of the summaries of the legislative acts and store it in a spreadsheet along with other act-level variables. These texts are stored in UTF-8 format and can be easily used by the most common packages for quantitative text analysis.

The app has been created using PyInstaller (version 4.1), whereas the original script has been written using Python 3.8.2. The transformation of the Python script into a .exe file allows users to use the webscraper even without having Python installed on their machine.

* Department of Political Economy, King's College London, michele.scotto_di_vettimo@kcl.ac.uk.

2 What does the app do?

The app gathers various information about European Union's legislative proposals and store it into a spreadsheet. As the app relies primarily on the OEIL database, only information about proposals concluded from 1994 onwards can be retrieved. Nonetheless, along with what is available on the OEIL repository, the script collects other information about the legislative proposals also from the EUR-Lex website. The following list provides an overview of the various variables collected:

- Act's general information:
  - Title
  - Inter-institutional procedure number
  - Legal instrument (e.g., Regulation, Directive, Decision, etc.)
- Proposal's details:
  - Proposal number
  - Date of the proposal
  - Celex number of the proposal
  - Commission Directorate-General responsible for the proposal
  - Dummy recording whether the proposal has been revised or not
  - Revised proposal number (if appropriate)
  - Revised proposal date (if appropriate)
- Legislative process:
  - Status of the proposal (i.e., adopted, pending, withdrawn or rejected)
  - Legislative procedure used
  - Treaty legal basis
  - Date of conclusion (if not pending)
  - Celex number of final act (if adopted)

- Institutional involvement:
  - Number and dates of European Parliament readings
  - European Parliament committee responsible for the file
  - Number of European Parliament committees asked for opinion
  - Details of the rapporteur (name, party and nationality)
  - Number of discussions in the Council of Ministers
  - Type of item on Council agenda (i.e., 'A' or 'B')
- Content-related variables:
  - Summary of the legislative proposal and of the final act (the latter only if adopted)
  - List of EUROVOC descriptors
  - List of Directory Codes
  - List of Subject Matter
  - Number of recitals of the proposal

3 Usage

The app has three main functions: (1) downloading all the legislative proposals; (2) downloading a specific set of proposals; and (3) combining the various spreadsheets downloaded into one single database. In all cases, the first step is, of course, to "unzip" the compressed folder and run the exe file by double-clicking on it. If necessary, allow antivirus to make an exception and execute the file. In the future developments, all the instructions will be passed to the app through a more user-friendly interface. At the moment, however, the app still relies on textual information given in sequence. The next paragraphs offer some guidance on how to use the app.

3.1 Downloading all legislative proposals

Once opened, the app will prompt some user inputs to figure out what you want to do. To download all the legislative proposals, skip the first two questions by pressing ENTER. Then, when asked to enter the year of interest, you can enter a range starting from 1994 to the current year (e.g., 1994-2020) and press ENTER to continue. Again, press ENTER at the following questions and the app will start the download.

While downloading, the app will save temporary spreadsheets every 100th iteration, if you open one of these files please make sure that you close it again before the app tries to overwrite it! Note that the download speed depends on the quality of your internet connection. In general, however, downloading all the years in a row can take quite long. Yet, there's a trick. You can launch the app multiple times simultaneously and each time download just a couple of years.

3.2 Downloading a specific set of proposals

You can operate a more targeted download by providing few more inputs when launching the app. The above instructions imply also that it is possible to download all the proposals from only a restricted range of years or from one specific year alone. To do so, follow the steps described above and enter the appropriate range (or year) when prompted. Additionally, it is possible to download a specific range of proposals within a year (or a set of years). To do so, enter the desired range when prompted. Finally, it is possible to download even a single legislative proposal. To do so, skip the first question about merging files and, when asked if you want to download a specific act, press Y and then enter the inter-institutional procedure number. Note that the app does not want to know about the legislative procedure. Hence, if you want to download the file 1999/0123(COD) you have to type just 1999/0123. If you like more complicated things, you can also select a specific year and then a specific procedure number. This, in effect, will lead to the same result as giving the exact inter-institutional procedure number.

3.3 Combining various spreadsheets

The app generates spreadsheets for each year covered during the download, regardless of the number of acts downloaded. These spreadsheets are saved in a folder named "Extractions", which is created automatically when a sheet is ready to be saved. It is possible to ask the app to combine the sheets from different years, at the conditions that these are saved in the "Extractions" folder and that the folder is in the same directory of the app. To do so, press Y at the first question to continue.

4 Troubleshooting

This section deals with the possible problems that can be encountered when running the app. Keep in mind that by no means this is an exaustive list. So, if you experience a different problem, please feel free to get in touch with me.

Script fails to execute

This problem may occur when you double-click on the exe file and you get an error message saying something like "Fatal error detected: failed to execute the script...". In that case, you should check whether the antivirus is blocking the execution of the file. If yes, add an exception.

Script crashes at the very beginning

If the app opens but crashes at the very beginning without giving you any error message, check whether you have typed the required input information in the appropriate format. In any case, the app should alert you before accepting any incorrectly written input. If the crash is instead due to a faulty code in the app, an error message should inform you about the nature of the problem, and you can get in touch with me so that I can address it. Note that crashes due to a faulty code are also possible if there are changes to the structure of the webpages from which the app gathers the data.

No data is collected

This applies when the app runs, you see it progressing through the range of proposals that you wanted to download, but eventually no data is collected. In this case, I would check whether the OEIL and/or EUR-Lex website are accessible (e.g., they might be under maintenance or unavailable for other reasons). If they are, double-check that the act you wanted to collect actually exists and its page is available.

Incorrect data is collected

This applies when, technically, the data collection concludes without problems, but the collected data is wrong. In this case, you should try to figure out whether the data on the OEIL or EUR-Lex website is correct in the first place. There are (very few) situations where the two portals record inconsistent information. For instance, because one of the two is not updated. Another reason for the gathering of incorrect information could be that the Python code contains some errors. If you think it might be the case, by all means drop me an email!