How to Extract Legal Citations Using Python (For the Complete Beginner)

Rachael K. Hinkle, University at Buffalo, SUNY

Many approaches to computational text analysis examine a document in its entirety. However, there are also a wide variety of useful techniques that involve extracting certain elements from a text. One such informative element is citations. Legal reasoning in a common law system is based on comparison to previous rulings. This means that the legal sources cited in a judicial opinion or attorney's brief are critical elements of those texts. The purpose of this article is to explain how to use Python to extract citations from legal texts.

Why Python? Much work on citations relies on Shepard's Reports or West KeyCite reports. These sources, available in the subscription services Lexis and Westlaw, provide an exhaustive summary of every citation to a given court opinion. While these reports are incredibly useful, they have some drawbacks. First, not all researchers have access to Lexis or Westlaw. Second, these reports detail whether case B has cited case A, but they do not provide the precise number of times case B cited case A. If you use Python to extract all the citations in case B, you can see not just that case A was cited, but also if it was cited once or a dozen times. In the aggregate, you can compile both how many unique precedents a document cites as well as the overall number of citations. Third, detailed citation reports are not available for some documents, such as briefs. There are an increasingly massive number of raw legal texts freely available to scholars electronically, and many of them contain interesting citations to legal precedent we may want to extract.

With the tools described in this article, the legal citations available in virtually any document can be accessible. Preexisting Python skills are not necessary. I describe all the steps necessary for installing the required software and carefully document the provided Python code so that anyone can use this technique. As someone who had to learn Python from scratch on my own, I understand that it can seem intimidating. But I promise that it is an attainable goal. I have even taught my nieces and nephews (ages 9-12) how to use Python. You can do it too!

The first task is to get all of the required software installed. To make things easier (especially for future Python experts), we will be using a web application called Jupyter Notebook that allows us to run Python code a bit at a time and keep track of notes and comments in a useful way. The main Python library we will be using only works in a slightly older version of Python, Python 3.6. This will complicate our setup a bit. Even if you already use Jupyter Notebook, please follow along from step 2 below.

Steps to install requisite software:

Law and Courts Newsletter, Volume 32, Number 1, Spring 2022. ©The author. Associated data and code available at the *Newsletter* Dataverse.

1. Go to www.anaconda.com and install Anaconda. (Click on "Products" and select "Individual Distribution".) For more information or help on this step, see the file "HowToInstallAnaconda.docx" in the associated Dataverse files.

2. Open the Anaconda Navigator (from your Applications folder).

3. Click on "Environments" (in the left panel).

4. Click on "Create" (at the bottom of the middle panel).

5. Give your new environment a name. For the sake of simplicity, I suggest something like "Python3_6".[1]

6. Under "Packages", check the box for "Python" and select "3.6"[2] from the drop-down menu, then click "Create".

7. With that new environment selected, click on "Home" (in the left panel).

8. Find "Jupyter Notebook" and click the button below it labeled "Install".

Congratulations! You are ready to run Python code. The previous steps only have to be run once on any given computer. Now you are ready to experiment using the code I have provided on Dataverse. The Dataverse files contain a file with an ".ipynb" suffix. This is a Jupyter Notebook file. You cannot open such files with the normal process of simply clicking on them. In order to open these files, use the following steps.[3]

Steps to run code after installing the software:

1. Download the files from Dataverse and save them in a location accessible for your username only, not for all users.[4]

2. Open the Anaconda Navigator (from your Applications folder).

3. Click on "Environments" (in the left panel).

4. Select the environment you named "Python3_6" or similar.

5. Click on "Home" (in the left panel).

[1]Please note that for some operating systems, you may not be allowed to use periods in the environment name.

[2]For some operating systems an additional period and number may appear after the "3.6". For example, it may read "3.6.13". The important thing is to identify the Python version that begins with "3.6."

[3]To run most Jupyter Notebook files you simply have to open Anaconda Navigator and launch Jupyter Notebook. The extra steps here to change the environment are not typically necessary. There are only needed here to use the lexnlp library.

[4]The location of the files is important because Jupyter will show the file structure for files available to your username, not those available to all users on your computer.

6. Locate "Jupyter Notebook" and click on "Launch". This will open a browser tab.

7. Navigate through the file structure to the folder where you have placed the Dataverse files (e.g., Downloads). Click on "Hinkle_LexNLP_Example_Code.ipynb" to open the file.[5]

This Jupyter Notebook file contains a step-by-step explanation of the process of loading text into Python, using lexnlp to extract citations, and exporting citation counts to a .csv file. To execute the code in each cell, place the cursor in the cell and then click the "Run" button. For purposes of clearer explanation, the process is first executed for a single document and then extended to show how to iterate through several documents in a folder.

When using lexnlp, I ran into a minor, but important, wrinkle. The code did not pick up the short references "id." or "ibid." The example code includes the workaround I created to address this concern. Before using lexnlp to extract citations, I first find all "id." and "ibid." strings and replace them with the placeholder "999 U.S. 999". There is no Supreme Court case with such a citation, but it is in a format that lexnlp recognizes as a citation and, therefore, extracts from the text. Since each reference to a citation is extracted in order, I simply convert each "999 U.S. 999" citation to be the legal precedent that immediately precedes it. There are other ways to extract legal citations in Python, and some of them address this issue of short references.[6] However, in my own work, I have found lexnlp to extract the most amount of useful information.[7] It parses citations to provide helpful items such as year and court. I hope this tool proves useful for you as well.

[5]For ease of reference, the Dataverse also contains .pdf versions of the Jupyter Notebook file with both just the code and the code that has been executed in Python.

[6]One such example is the library Eyecite.

[7]For additional information on using the lexnlp library, see "LexNLP: Natural Language Processing and Information Extraction for Legal and Regulatory Texts" by Michael J. Bommarito II, Daniel Martin Katz, and Eric M. Detterman, available at: arxiv.org/pdf/1806.03688.pdf.