Hinkle_LexNLP_Example_Code

June 6, 2022

[ ]:
```
### How to Extract Legal Citations using Python (for the complete beginner)
### Law and Courts Newsletter
### Rachael K. Hinkle
### June 6, 2022
### Run on macOS 10.15.7

### The only difference in this code for Windows is the way filepaths are set up.
 ↪ They use a different file seperator.
## Macs use a single foward slash; if you are using a mac run the follwing code:
fileSeperator = "/"
## Windows uses a double back slash: if you are using Windows use the following␣
 ↪code,
# Note, to do this delete the "#" at the beginning of the next line.
#fileSeperator = "\\"
```

[ ]:
```
# The following code turns off the autosave feature to avoid inadvertently␣
 ↪saving over a previous file.
# As always, remember to save your work as you go along.
%autosave 0
```

[ ]:
```
## Preliminary Step 1: It helps to verify we are running Python 3.6 before␣
 ↪proceeding

# Python tip: Any text preceeded by a "#" is ignored by Python.

import sys
print(sys.version)
```

[ ]:
```
## Preliminary Step 2: Install the lexNLP package (only need to do this the␣
 ↪first time)
# Note: This step will take a bit of time.  When an asterisk appears in the␣
 ↪square brackets
# to the left of a cell (e.g. "In [*]"), that means the code is running.  A␣
 ↪number will
# appear in the square brackets when the code is done running.
```

[ ]:
```
pip install lexnlp
```

[ ]:

[ ]:
```
## Preliminary Step 3: Import the libraries you will need for this session␣
 ↪(analogous to R packages)

# 'os' provides useful funtions for dealing with the operating system
# 're' provides the ability to use regular expressions
# 'csv' provides the ability to write the results to a .csv file
# 'string' provides useful functions for processing the opinion content
# 'lexnlp' provides the ability to extract citations from natural language

import os, re, csv, string, lexnlp
import lexnlp.extract.en.citations
```

[ ]:

[ ]:
```
## Preliminary Step 4: See the name of your working directory
os.getcwd()
```

[ ]:
```
## Preliminary Step 5: Setup a shortcut reference to your working directory

# Python tip: This just creates a variable with a string.
#             You can name the variable anything you want.

mydir = os.getcwd() + fileSeperator
```

[ ]:

[ ]:
```
### Illustration 1: How to extract citations from a single file
## Note: If you are familiar with Python, you can proceed to Illustration 2
```

[ ]:
```
## Step 1.1: Read in an opinion (or other legal document) in a .txt file
# This code creates a list; each entry of the list is a string that contains the␣
 ↪text
# from one line in the file

# Python tip: The "print()" function is very useful for seeing the contents of␣
 ↪variable
#             to see or double check what the code is doing

# Note: If you get an error, check that the "mydir" variable correctly lists␣
 ↪your directory
# where the file is located, and/or make sure you have downloaded the "myExample.
 ↪txt" file from Dataverse.

tfile = open(mydir + "myExample.txt", "r", encoding="latin-1")
```

```
lines = tfile.readlines()
print(lines)
```

[ ]:

[ ]:
```
## Step 1.2: Process the list from the previous step to create a single string␣
 ↪variable with
#           only the majority opinion content

#create an empty string variable (that we will later fill with the majority␣
 ↪opinion)
opinText = ""

#create a logical operator variable (think of this as a switch set to "off")
opin = False

# loop through each line in the document in turn, after each line it goes to the␣
 ↪next,
# when there are no longer any lines left, the loop will stop
for line in lines:
    # Check to see if the line starts with some or no white space followed by␣
 ↪"OPINION" and then the end of the line
    # If so, turn the switch "on" (by making the logical operator variable equal␣
 ↪True)
    # Once this switch is turned on, it will remain that way until we turn it off
    if re.search("^[\s]*OPINION$", line):
        opin = True

    # Check for various strings that would indicate the end of the majority␣
 ↪opinion
    # If any of these patters of text are found, turn the switch "off" by making␣
 ↪the logical operator variable equal False
    if line and (re.search("^[\s]*DISSENT", line)):
        opin = False
    if line and (re.search("^[\s]*CONCUR", line)):
        opin = False
    if line and (re.search("^[^A-Za-z]*APPENDI", line)):
        opin = False

    # For any line of text that occurs while the switch is "on", that is not␣
 ↪empty, add it
    # to the string variable where we are collecting the majority opinion text.
    if opin and not re.search("^[\s]*$", line):
        opinText = opinText + line

# Now that the loop is complete, and we have the entire opinion, we can process it
```

```
# This code finds any instances of one of more spaces and substitutes a single␣
 ↪space
opinText = re.sub("\s+", " ", opinText)

#Finally, we want to see our final product
print(opinText)

# Python tip: some elements of regular expressions used in the code above:
# ^: beginning of the line
# $: end of the line
# [\s]*: zero or more white spaces
# [\s]+: one or more white spaces
# [^A-Za-z]: Anything that is NOT a capital or lower case letter.
#  (Inside the square brackets "^" means NOT, outside the brackets it means␣
 ↪beginning of the line)
# [A-Z] means any capital letter
# [a-z] means any lower case letter
# [A-z] means any letter
# [0-9] means any numeral
```

[ ]:

[ ]:
```
## Step 1.3: Extract citations from the opinion

#Extract each ciation and put it in a variable
cite_holder = lexnlp.extract.en.citations.get_citations(opinText)

# See the citations extracted by using a loop
# Note: It doesn't work to just print the variable, but you can print (or␣
 ↪otherwise access)
#       each item in it by using a loop.
for cite in cite_holder:
    print(cite)

# From the output we can see that each citation has multiple pieces of␣
 ↪information
# Illustration 2 will show how to get key information into a spreadsheet for␣
 ↪further analysis
```

[ ]:

[ ]:
```
### Illustration 2: How to extract citations from all files in a subfolder
###                 and output the data to a spreadsheet with one row per citation
```

[ ]:
```
## Step 2.1: Create and open a .csv file that we will write information into
fout = open(mydir + "myNewSpreadsheet.csv", "w", newline="")
outfilehandle = csv.writer(fout,
```

```
                            delimiter=",",
                            quotechar='"',
                            quoting=csv.QUOTE_NONNUMERIC)

# Create a row that contains the variables names and write that to the first row␣
 ↪of the .csv file
localrow = []
localrow.append("filename")
localrow.append("volume")
localrow.append("reporter")
localrow.append("page1")
localrow.append("page2")
localrow.append("court")
localrow.append("year")
localrow.append("raw citation")
#localrow.append("numCites")
#localrow.append("numIds")
outfilehandle.writerow(localrow)

print(localrow)
```

[ ]:
```
## Step 2.2: Prepare information for where to read text files from

#create a filepath to the folder where all text files are located
dirname = mydir + "sampleOpinions" + fileSeperator
# Create a list of all files in a given folder
dirlist = os.listdir(dirname)
# See the list of all files to make sure things are working
print(dirlist)
```

[ ]:
```
## Step 2.3: Loop through all documents, process text, extract ciations, process␣
 ↪citations,
##           and export to spreadsheet

for entry in dirlist: ## each entry is a separate txt file with a single opinion
    infilehandle = open(dirname + entry, encoding='latin-1')
    txtlines = infilehandle.readlines()

    # This is useful for watching the progress of the code, espcially with many␣
 ↪documents
    print(entry)

    # Create empty variables (each time through the loop it is important to make␣
 ↪sure
    # all variables are empty so values from the previous document do not carry␣
 ↪over)
```

```
    opin_string = ""
    #opin_string2 = ""
    #opin_string3 = ""
    #opin_string4 = ""
    sent_holder = []
    cite_holder = []
    localrow = []
    sct = 0
    op_line = False

    # Loop through each line in the document
    for txtline in txtlines:

        # Find the beginning and end of the opinion
        if re.search("^[^A-Za-z]*OPINION[\s]*$", txtline):
            op_line = True
        if line and (re.search("^[\s]*DISSENT", line)):
            opin = False
        if line and (re.search("^[\s]*CONCUR", line)):
            opin = False
        if line and (re.search("^[^A-Za-z]*APPENDI", line)):
            opin = False

        # Create a string with the majority opinion and process the text
        if op_line:
            opin_string = opin_string + txtline

    ## After looping through all lines in the document, do the following steps␣
↪once per document

    ## Process text in opinion string

    # Get rid of newline characters
    opin_string = re.sub("\n", " ", opin_string)
    # Replace short citations with the placeholder "999 U.S. 999" that will be␣
↪recognized by lexnlp
    opin_string = re.sub("[\s](I|i)d\.[\s]", " 999 U.S. 999 ", opin_string)
    opin_string = re.sub("[\s](I|i)bid\.[\s]", " 999 U.S. 999 ", opin_string)

    # Extract citations
    cite_holder = lexnlp.extract.en.citations.get_citations(opin_string)

    # Change "cite_holder" to a list object
    cite_holder = list(cite_holder)

    # Create a second variable that replaces short references with the full␣
↪citation
```

```
   #Pyton tip: Python indexes begin with zero.  So to extract the first element␣
↪of the
   #           variable "cite_holder", we can use "cite_holder[0]"

   cite_holder2 = []
   # Since we need to refer back to the previous element of the loop for this␣
↪task,
   # we use a different kind of loop that uses an index, "i"
   for i in range(0, len(cite_holder)):
       if cite_holder[i][0] == 999: #and cite[1] == "U\.S\." and cite[3] == 999:
           cite_holder2.append(cite_holder2[i-1])
       else:
           cite_holder2.append(cite_holder[i])

   # Create empty variables that will be used for exporting information to␣
↪spreadsheet
   vol = ""
   rep = ""
   page1 = ""
   page2 = ""
   court = ""
   year = ""

   # Put content in each variable that will be written to the spreadsheet
   for cite in cite_holder2:
       vol =str(cite[0])
       rep = str(cite[1])
       page1 = str(cite[3])
       page2 = str(cite[4])
       court = str(cite[5])
       year = str(cite[6])

       #The lexnlp function does not perform perfectly, here is some code to␣
↪clean up
       # some of the issues by getting rid of info that is not correct
       # Note: the "raw citation" column still has the full information for␣
↪each cite
       if len(court) > 20:
           court = ""
       if court == "None":
           court = ""
       if page2 == "None":
           page2 = ""
       if year == "None":
           year = ""
       if re.search("U\.S\.", rep):
```

```
            court = "SCOTUS"
        if re.search("L\.[\s]*Ed\.", rep):
            court = "SCOTUS"
        if re.search("S\.[\s]*Ct\.", rep):
            court = "SCOTUS"

        # Write a row to the spreadsheet for each ciation
        if vol != "":
            localrow = []
            localrow.append(entry)
            localrow.append(vol)
            localrow.append(rep)
            localrow.append(page1)
            localrow.append(page2)
            localrow.append(court)
            localrow.append(year)
            localrow.append(cite)
            outfilehandle.writerow(localrow)

# Close the files you are reading from and to
infilehandle.close()
fout.close()
```

[ ]: