UNGA Speech Web Scraper Walkthrough

[IMPORTANT: For the PDF to TXT to work at the end none of the filenames can have spaces in them]

1. **Getting links from the UN website to the country pages, which each hold the statements as PDFs.** You will need to change certain variables, URLs, and ranges in the Python script to be able to access the PDFs and download properly.
   - Load **ungaspeech_scraper.py** into the IDE
   - Change the sections marked in red boxes
     1. Session number of the debate
     2. URL to the UN webpage with the links to the country pages (e.g. for session 70 the URL is: https://gadebate.un.org/en/sessions-archive/70). For older sessions, all of the countries are listed on one webpage. For the latest session there is a webpage for each date of the General Debate.
     3. Subset the *links* list only for the links to country pages. Do this manually by clicking on the *links* list under variable explorer and finding the indices/range. [NOTE: make sure the ending number in the range is one greater than the index of the last country page to ensure you capture all of them

```
##############################################################################
#
#SCRAPE PDFs FROM UN WEBSITE
#
##############################################################################

##ENTER SESSION NUMBER HERE##
session = 70  1
yr = str(1945+session)

##This section collects the links to the country pages, which each hold the speech
from bs4 import BeautifulSoup
from urllib.request import Request, urlopen

#Enter the URL to the UN page that has the links to the country pages for speeches
#There should be one for each session, or one for each day of each session
req = Request("https://gadebate.un.org/en/sessions-archive/70")  2
html_page = urlopen(req)

soup = BeautifulSoup(html_page, "lxml")

#Get all of the links on the page
links = []
for link in soup.findAll('a'):
    links.append(link.get('href'))

#subset to only the country page links ####THIS MAY NEED TO CHANGE###
links = links[24:217]  3
```

OUPUT FROM THIS SECTION:

links – list of links to each country's page, which holds their country statemen

2. **Import converter to convert country names into ISO codes**

This is important for naming the files later. This section imports each line of the country code converter csv into a string value in a list named *iso_cc.*

- Change the file path in the red box to the file path for the converter on your computer. This file is slightly different from the original iso_to_country.csv (which has the official UN country names). This is because a few of the names are slightly different than their official UN name. [Note: Mismatch of names is something that can be solved pretty easily later if they aren't all perfect]

OUTPUT FROM THIS SECTION:

iso_cc - a list of strings in which each index holds a country code and its corresponding country (e.g. index 0 holds 'AFG,Afghanistan')

```
import csv
iso_cc = [] #create a blank list which will hold the conversions

#open the csv and write each row to an index of the iso_cc list. This creates
    #a standardized list with the first three letters as the iso3 code and
    #the end of each string in the list with the official UN country name
with open('./Documents/UNGA Speeches/iso_to_country.csv', newline='') as csvfile:
    iso_to_country = csv.reader(csvfile, delimiter=' ', quotechar='|')
    for row in iso_to_country:
        iso_cc.append(' '.join(row))
        #print(' '.join(row))
```

3. **Parse through the country pages and get the PDFs and meta data about the speeches**

```
import requests
#opens a blank text file to write the information about each speech to
with open('./Documents/UNGA_Speeches/%s/unga_%s_meta' % (session, session), 'w', newline='')
    unga_meta = csv.writer(f1, delimiter='|')
    #loop through each link to each country page
    for i in range(0, len(links)):
        page = requests.get("https://gadebate.un.org" + links[i])
        soup2 = BeautifulSoup(page.content, 'html.parser')
        #print(soup2.prettify())

        #first get the data about speaker, country, and date of the speech
        meta = str(soup2.find_all(id="statement-speaker-and-title")).split(">")
        country = meta[2][0:-4]
        speaker = meta[4][0:-4]
        #remove special letters from speaker names which can't be put into the csv
        if "š" in speaker:
            speaker = speaker.replace("š", "s")
        if "ć" in speaker:
            speaker = speaker.replace("ć", "c")
        if "ğ" in speaker:
            speaker = speaker.replace("ğ", "g")
        if "ē" in speaker:
            speaker = speaker.replace("ē", "e")
        if "Č" in speaker:
            speaker = speaker.replace("Č", "C")
        if "á" in speaker:
            speaker = speaker.replace("á", "a")
        if "č" in speaker:
            speaker = speaker.replace("č","c")
        if "ṣ" in speaker:
            speaker = speaker.replace("ṣ", "s")
        if "ạ" in speaker:
            speaker = speaker.replace("ạ", "a")

        #get the date - need to do this because there are blank spaces in the date
        f = len(meta[6]) - len(meta[6].lstrip())
        date = meta[6][f:-meta[6].index(yr)-1]
```

1

2

3

This section iterates through the country pages (each index of *links*) and pulls the PDFs and metadata about the speeches (country name, speaker, date, language(s)). The metadata is saved in a CSV which can be imported into Excel for manipulation and analysis.

- Needs to change (marked in **red**)
    - File path for the csv that the code will create that has information on each speech
- May need to change (marked in **blue**)

  These are things that may need to change from session to session, though they most likely are the same from the original version.
    1. This is the initial part of the link before every country page ending (which are found in the *links* list). Barring major change to the UN's website this shouldn't change.
    2. This is an id within the HTML which we pull in order to access the speaker, country name, and date. This information is put into a list of strings called *meta*.
        a. The arrows above (lines 67, 68, 92) pull the meta data from the *meta* list

        If this needs to be changed, use Inspect on Google Chrome or Beautiful Soup to identify the changed id name to be able to pull the information. If not only the id is changed but also the format of the info pulled from that id is changed, you may need to change the index and the subset in lines 67, 68, and 92.
    3. Depending on the format of the info pulled from the HTML and the spacing in what is pulled for the date, the subset and index may need to be changed
- This middle section also converts special/accented character letters in the speakers' names to normal letters. This usually is an issue for Eastern European/Balkan countries. The special letters become an issue when we try to put the names into the CSV, which is why I've coded some that have come up to convert to basic letters. Check if a speaker's name has an odd character if you get the error:

  UnicodeEncodeError: 'charmap' codec can't encode character '\u0107' in position 48: character maps to <undefined>

  Then, add a couple lines of code to convert it to a basic letter, like I've done.

OUTPUT FROM THIS SECTION:

unga_meta – (or whatever you named it) described fully at the end of the next section

meta – list that holds the meta data of speaker, country, and date along with some other symbols. We parse through this and subset some values to get what we need.

4. Finding PDF links and more metadata

This section pulls the PDFs from the country pages, formats some more metadata and saves all of the meta data to the *unga_meta* csv (or its counterpart with your naming).

1. We use the ISO3-to-country-name converter to find the ISO3 code for each speech. This will be used for the metadata file and the downloaded PDF filenames. The Boolean *changed* checks if there is a match between the scraped country name and one in the converter. We loop through every name in the converter and if the scraped country name matches, then we save *cou* as the corresponding ISO3 code. Because it is a match we mark *changed* as true. If there is no match,

we still need something that we can identify later and change to the correct ISO3, so line 105 takes the first three letters of the scraped country name and assigns that to *cou.*

2. This part finds all of the PDF links on the page and puts them into the list *pdf_links*. It does this by iterating through all instances of 'a' tags in the HTML code, pulling the PDFs, and appending them to the *pdf_links* list.

```
97        changed = False
98
99        #loop through the converter
100       for c in iso_cc:
101           if country == c[4:]: #if the country name is in the converter
102   1           cou = str(c[0:3]) #save the country code for that name
103               changed = True
104       if changed == False: #if there was no match
105           cou = country[0:3] #use the first three letters as the country code
106
107       #create a blank list for all of the pdf files. This will be between 0 if
108           #a country doesn't have a posted statement to however many statements
109           #there are based on the number of languages available
110       pdf_link = ""
111   2   pdf_links = []
112       for link in soup2.findAll('a'): #find all of the pdfs and add them to the list
113           if ".pdf" in str(link):
114               pdf_links.append(str(link))
115
```

OUTPUT FROM THIS SECTION:

pdf_links – list of all PDFs on the page, which will be all of the country statements

5. Identifying the PDF to download and saving information on the languages

This section identifies if there is a PDF in English and downloads that. It identifies which languages are available. It saves all of this information to a csv file for every speech. [NOTE: You may have to import *unga_meta* (the csv) into Excel after every run of the code (for every date there are speeches) to save all of the metadata in one file. You wouldn't have to do this for the archives because there is only one webpage we are pulling country-page links from]

1. We set the *pdf_link* variable, which is the one we will download as the PDF, as the PDF tagged with "English" so we always the get the English version if available.
2. If no English version is available and there is only one language available then pull that statement by setting *pdf_link* equal to that link. If there are multiple languages available (e.g. Arabic and French) then pull one PDF with the language hierarchy: 1) French, 2) Spanish, 3) Russian, 4) Arabic. Set this PDF's link to *pdf_link*. The variable *pdf_link* will be left empty if no statement is provided.
3. Here all of the languages which can be extracted from the *pdf_links* list and are concatenated into one string called *language*, which we add to the metadata. Like the code in Section C marked in blue, this may need to be changed if the format of the PDF links are changed or the HTML code is changed because the subset may need different indices.
4. Finally, part 4 writes all of the meta data to a csv. It prints out the counter every time, so you can see the progress of the loop.

```
1
english = False
#pull the one in English (if it exists) because this is what we download
for j in pdf_links:
    if "English" in j:
        pdf_link = str(j)
        english = True

2
if english == False:
    if len(pdf_links) == 1 and "English" not in pdf_links[0]:
        pdf_link == str(pdf_links[0])
    else:
        for j in pdf_links:
           if "Arabic" in j:
               pdf_link = str(j)
        for j in pdf_links:
           if "Russian" in j:
               pdf_link = str(j)
        for j in pdf_links:
           if "Spanish" in j:
               pdf_link = str(j)
        for j in pdf_links:
           if "French" in j:
               pdf_link = str(j)

3
language = "" #empty string to record the languages available
#concatenate into one string the different languages available
for k in pdf_links:
    l = str(k).split("\"")[2][1:-4]
    language = language + l + " "

4
#save session #, country name, country code, speaker, date, and language(s)
    #into the csv we opened at the top of this section
unga_meta.writerow([session, country, cou, speaker, date, language])
print(i) #counter to see progress
```

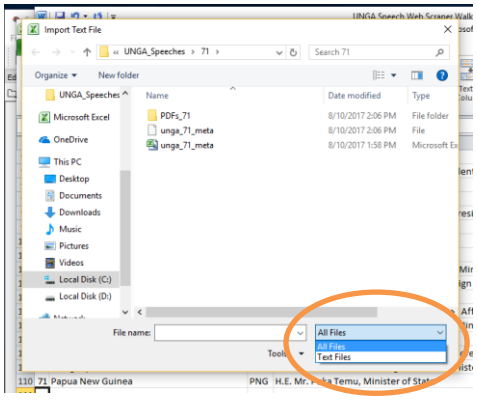
OUTPUT FROM THIS SECTION:

unga_meta – (or however you named it) holds session number, country, ISO3, speaker, date, and language(s) of the speeches. Below is the csv after importing into Excel. It is delimited with "|"

	A	B	C	D	E	F
1	Session	Country	ISO3	Speaker	Date	Language(s)
2	70	Secretary-General of the United Nations		H.E. Mr. Ban Ki-moon, Secretary-General	9/28/2015	English
3	70	President of the General Assembly (opening)		H.E. Mr. Mogens Lykketoft, President of the 70th	9/28/2015	English
4	70	Brazil	BRA	H.E. Mrs. Dilma Rousseff, President	9/28/2015	English
5	70	Antigua and Barbuda	ATG	H.E. Mr. Gaston Alphonso Browne, Prime Ministe	10/1/2015	English
6	70	Poland	POL	H.E. Mr. Andrzej Duda, President	9/28/2015	English
7	70	China	CHN	H.E. Mr. XI Jinping, President	9/28/2015	English

TO EXPORT CSV DATA INTO EXCEL:

1. Open a new Excel spreadsheet then Data > From Text
2. Choose to look at All File types
3. Double click on the unga_meta data file (here labeled unga_71_meta)
4. Press "Next" on the Text Import Wizard that pops up (ensuring that "delimited" is bubbled in. For delimiters choose "Other" and enter "|". Click Next and then Finish
5. Add the data either to the top of the spreadsheet or append it onto data that already exists.



Importing to Excel is something you'll want to do after every run of the code that scrapes the PDFs

pdf_links – a list of all of the pdf links on the webpage. We get this to be able to pull the one that is in English or to see what other languages are available

pdf_link – a string for the link that is in English. If English is unavailable it pulls either the only PDF available or the easiest language to translate available. Left empty if no available statement

6. Download the PDF

- We check if *pdf_link* is empty, which happens if there is no statement provided. If this is true then the loop iterates to its next value, skipping the current country
- We then isolate only the URL from the string *pdf_link* and save that as *pdf_link.* The way line 163 finds the link may need to be changed if the HTML is ever changed.
- The last part downloads the PDF, either into the main PDF folder (PDFs_session-number, here it'll be PDFs_70) if the statement is in English, or into a Need_Translation folder within the session folder (e.g. /70) if the statement is in another language.
- Change the file paths marked in red for where you want to save the PDFs.
    - The way this file path/naming for each PDF works is that it is based on the session and country name. For example, for Afghanistan in session 70, the PDF is saved in ./Documents/UNGA_Speeches/70/PDFs_70/AFG_70.pdf.
    - The %s's substitute in the variables that are in the parentheses at the end

```
156         #soup2.findAll('a')[22]
157
158         #if there is no statement, then iterate the loop to the next value
159         if pdf_link == "":
160             continue
161
162         #subset to the lnik itself
163         pdf_link = str(pdf_link).split("\"")[1]
164
165         #get the pdf using the link
166         response = requests.get(pdf_link)
167         if(english == True):
168         #save the pdf using the session number and country code as the file name
169             with open(('./Documents/UNGA_Speeches/%s/PDFs_%s/%s_%s.pdf'
170                        %(session,session,cou,session)), 'wb') as f:
171                 f.write(response.content)
172         else:
173             with open(('./Documents/UNGA_Speeches/%s/PDFs_%s/Need_Translation/%s_%s.pdf'
174                        %(session,session,cou,session)), 'wb') as f:
175                 f.write(response.content)
176
```

OUTPUT FROM THIS SECTION:

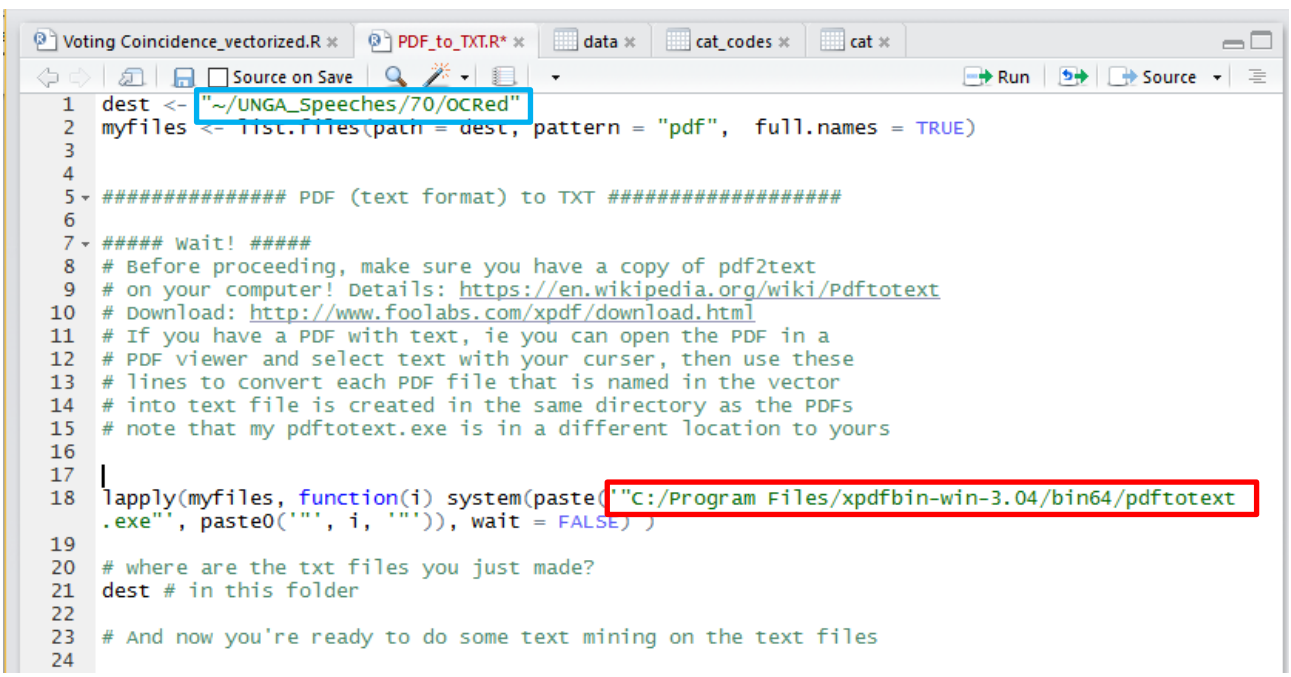
The country's PDF in the specified folders.

7. **Converting PDFs to TXTs**

In this section, we move from Python to RStudio to convert the PDFs to TXT files. This will only work for the PDFs that are "smart" PDFs (i.e. the text can be highlighted and is not just an image).

[NOTE: For this section to work none of your filenames can have spaces in them]

- Getting PDF to Text on your paper
    1. Download pdf2text on your computer: http://www.foolabs.com/xpdf/download.html
    2. This will download as a compressed folder. Note where it is saved on your computer or move it and change the file path to what I have below. Unzip the compressed folder (or click on pdftotext in the bin64 folder) and move it to Program Files on the Local Disk. Copy the file path and enter it where the red box below indicates.
- Begin converting
    1. Change the variable *dest* to the location of the PDFs, this is also where the converted txt files will be stored. This is outlined in blue below.
    2. Run the code then check if it created the txt files.
    3. Some of the PDF files may not be "smart" PDFs, and after this point you will easily be able to identify those because the TXT files are normally 1 KB because they're pretty much empty.



    4. Filter the folder with the PDFs and TXTs to just TXT files then sort by size. Identify the countries with the 1 KB TXT files and pull the corresponding PDFs into a folder because these will need OCR. You can delete the empty TXT files.

8. **OCR**
    - Compress the folder of PDFs that need OCR and email it to yourself on the low-side, which has Adobe Acrobat Pro and can do the OCR
    - Open the files in Adobe Acrobat Pro then Tools > Text Recognition > In Multiple Files then upload your files/the folder and press OK.
    - Email these back to yourself on the DIN and run them through Step 6 to convert to TXT files and you'll have a full corpus.