PDF & Twitter Scraping using Python

Sunbin Cho | Tom Bugdalle | Sergio López | Loukas Sfountouris

Université Paris I Panthéon-Sorbonne

Methods in Empirical Development Economics

Python Project

January 16, 2022

Overview

1. Topic
2. Methodology
   - PDF Scraping + NLP
   - Twitter Scraping + NLP
3. Data visualization
   - PDF Graphs
   - Twitter Graphs
4. Discussion
   - Word-Cloud
5. References

The topic

↪ Public awareness due to climate change: The case of COP26 agenda

↪ Urgency of the subjects discussed on the annual COP, and the implications of the decisions that are taken there (Official statements)

↪ Topic is now subject of public debate in different spheres (e.g. Twitter)

PDF Scraping: Web Crawling

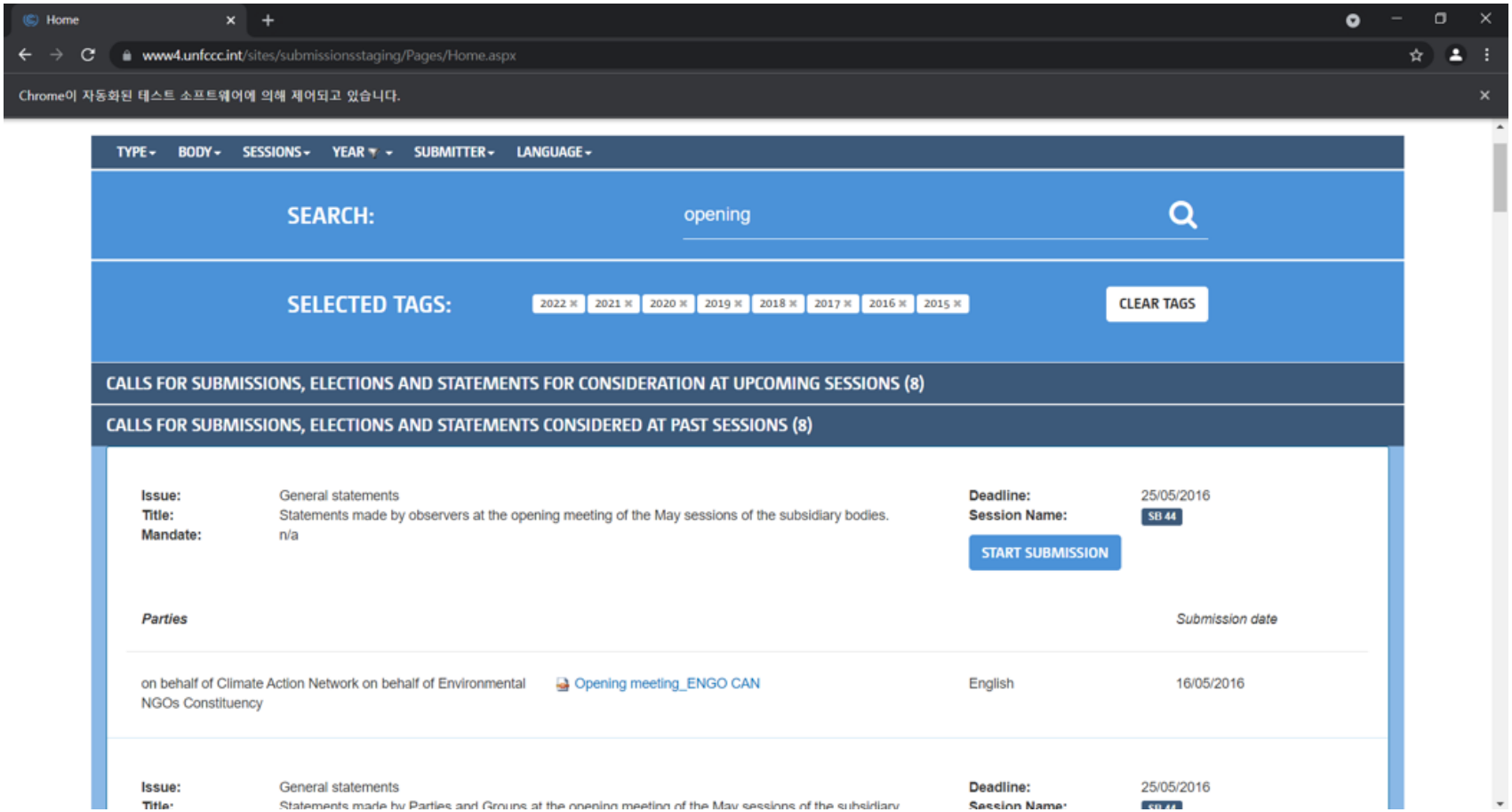
```python
# In[6]:
#pip install selenium
import requests
from bs4 import BeautifulSoup


# In[7]:

from selenium import webdriver
import time



# In[68]:

#Call the web page and maximise
driver = webdriver.Chrome()
web_url = 'https://www4.unfccc.int/sites/submissionsstaging/Pages/Home.aspx'
time.sleep(1)
driver.get(web_url)
driver.maximize_window()
time.sleep(5)
```

Web Crawling (UNFCCC) for downloading all the opening statements

```python
# In[69]:


#Click and select years
x_path = '//*[@id="searchyear"]/button'
year_button = driver.find_element_by_xpath(x_path).click()
driver.find_element_by_xpath('//*[@id="Open"]').click()
#driver.find_element_by_xpath('//*[@id="2022"]').click()
#driver.find_element_by_xpath('//*[@id="2021"]').click()
driver.find_element_by_xpath('//*[@id="2020"]').click()
driver.find_element_by_xpath('//*[@id="2019"]').click()
driver.find_element_by_xpath('//*[@id="2018"]').click()
driver.find_element_by_xpath('//*[@id="2017"]').click()
driver.find_element_by_xpath('//*[@id="2016"]').click()
driver.find_element_by_xpath('//*[@id="2015"]').click()



# ### Search 'opening'

# In[70]:


search_box = driver.find_element_by_xpath('//*[@id="accordion"]/div[2]/div/div/div[2]/div/div/input')
search_box.click()
search_box.send_keys("opening")
driver.find_element_by_xpath('//*[@id="accordion"]/div[2]/div/div/div[2]/div/div/div/button/i').click()
```

Select all years and search "Opening"

Web Crawling (UNFCCC) for downloading all the opening statements



Download and convert into pdf

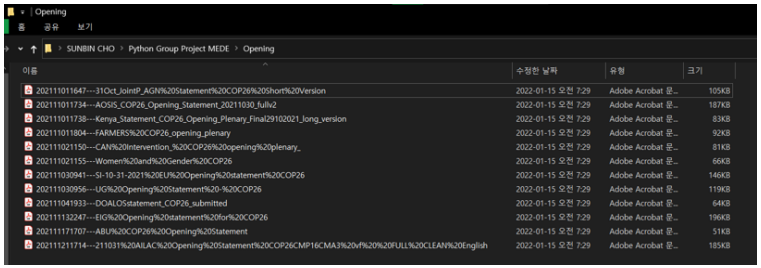
Download all the COP26 opening statement pdf

```
# In[ ]:


pdf_link = []

for a in upcoming_soup.find_all('a', href=True):
    a_string = a['href']
    if a_string.find('.pdf') >= 0:
        if a_string.find('COP26') >= 0:
            pdf_link.append(a_string)

```

Line 136: Read web file and save opening statements into pdf files

```
132 # In[ ]:
133     ## Save opening statements pdf files
134 ## Read web file
135 #pip install urllib3
136 import urllib.request
137
138 for text in sites :
139     report = text.replace(' ','%20')
140     webFile = urllib.request.urlopen(report)
141     pdfFile = open(report.split('/')[-1], 'wb')
142     pdfFile.write(webFile.read())
143     webFile.close()
144     pdfFile.close()
```

From 14 web files, we excluded 2 of spanish version. We'll do the analysis with 12 Opening Statements

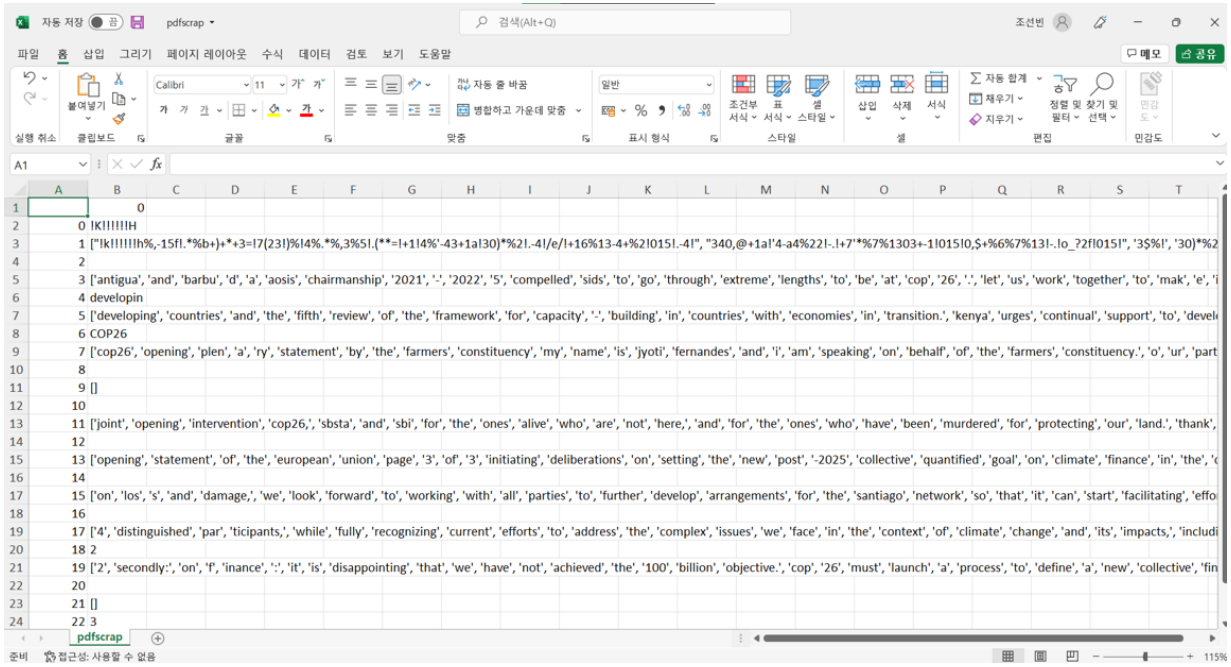


Then we use those ones, to scrap the text inside

```
    ## Scraping PDF

path = r"YOUR DIRECTORY"
dirs = os.listdir(path)
mytext=[]
for item in dirs :
    item_path = os.path.join(path, item)
    with open(item_path, mode='rb') as f:
        reader = PdfFileReader(f)
        for page in reader.pages:
            pass
        text = page.extractText()#.encode('utf-8')
        mytext.append(text)
        print(text)
        text = text.split()
        print(text)
```

Extract/Split text + Word list

Line 144: Extract and Split the text from the Opening statements



Line 175: import CSV file and create word list

Download required packages & Data cleaning (NLP)

Line 187: Download all packages needed

```
187 # In[ ]: From Loukas Download packages for analysis
188     #ANALYSIS
189 import nltk
190 nltk.download('punkt')
191 nltk.download('wordnet')
192 from nltk import sent_tokenize, word_tokenize
193 from nltk.stem.snowball import SnowballStemmer
194 from nltk.stem.wordnet import WordNetLemmatizer
195 from nltk.corpus import stopwords
196 from nltk.tag import pos_tag
197
198 import pandas as pd
199 import numpy as np
200 import re
201 #pip install spacy
202 import spacy
203
204 nlp = spacy.load('en_core_web_lg')
205
206 import spacy
207 from spacy.lang.en.examples import sentences
208
209 nlp = spacy.load("en_core_web_sm")
```

Line 211: Data cleaning

```
211     # Data Cleansing
212 df2=pd.DataFrame(words)
213 df3=df2[0] #To extract the words from the Column 0 (The name of the column that includes words is 0)
214
215 all_sentences = []
216
217 for word in df3:
218         all_sentences.append(word)
219
220 all_sentences
221 #df1 = df.to_string()
222 #df_split = df1.split()
223 #df_split
224
225 lines = list()
226 for line in all_sentences:
227     words = line.split()
228     for w in words:
229         lines.append(w)
230 print(lines)
```

To gather all the words from the word list

```
Console 1/A
"'stakeholder',", "'collaboration',", "'and',", "'a',", "'source',", "'of',",
"'innovation',", "'and',", "'new',", "'ideas',", "'for',", "'climate',", "'action.',",
"'14.',", "'a',", "'great',", "'opportunity',", "',',", "'not',", "'just',", "'to',",
"'richly',", "'excha',", "'nge',", "'of',", "'views',", "',',", "'but',", "'to',", "'m',",
"'ak',", "'e',", "'bold',", "'decisions',", "'that',", "'will',", "'spur',",
"'accelerated',", "'climate',", "'action',", "'and',", "'enable',", "'us',", "'achieve',",
"'the',", "'goals',", "'of',", "'the',", "'paris',", "'agreement.',", "'thank',", "'you',",
"'all',", "'.']", 'COP26', 'opening', 'plen', 'a', 'ry', 'Statement', 'by', 'the', 'Farmers',
'Constituency', 'My', 'name', 'is', 'Jyoti', 'Fernandes', 'and', 'I', 'am', 'speaking', 'on',
'behalf', 'of', 'the', 'farmers', 'constituency.', 'O', 'ur', 'partic', 'i', 'pation', 'in',
'this', 'process', 'is', 'instrumental', 'in', 'raising', 'global', 'climate', 'ambition',
'for', 'our', 'sector', ',', 'untapping', 'the', 'great', 'potential', 'that', 'a',
'griculture', 'and', 'its', 'people', 'ha', 've', 'in', 'the', 'fight', 'against', 'climate',
'change.', 'W', 'e', 'want', 'to', 'see', 'the', 'Paris', 'Agreement', 'implemented', 'in',
'full', 'so', 'that', 'the', 'necessary', 'enabling', 'framework', 'and', 'architectures',
'move', 'us', 'forward', 'towards', 'a', 'better', 'fu', 'ture', 'for', 'the', 'hundreds',
'of', 'millions', 'of', 'farmers', 'across', 'the', 'world', 'who', 'are', 'on', 'the',
'front', 'line', 'of', 'this', 'global', 'climate', 'emergency', 'and', 'for', 'us', 'all',
'because', 'we', 'depend', 'on', 'the', 'food', 'they', 'produce.', 'We', 'want', 'to',
'see', 'inclusive', 'and', 'ambitious', 'NDCs', 'that', 'deliver', 'on', ';', 'th', 'at',
'lead', 'to', 'investment', 'in', 'research', ',', 'innovation', 'and', 'tec', 'hnology',
'transfer,', 'in', 'extension', 'services', 'and', 'capacity', 'building', ',', 'and', 'to',
'ambitious', 'financial', 'frameworks', 'to', 'support', 'a', 'just', 'transition', 'and',
'a', 'dynamic', 'agricultural', 'sector.', 'O', 'nly', '15%', 'of', 'climate', 'finance',
'has', 'been', 'allocated', 'to', 'agriculture', '.', 'This', 'imbalance', 'must', 'be',
'addressed.', 'This', 'means', 'that', 'decisions', 'must', 'be', 'made.', 'Farmers',
'understand', 'the', 'difficulty', 'in', 'making', 'decisions', 'in', 'the', 'face', 'of',
'complexity', '.', 'They', 'do', 'this', 'every', 'day', '.', 'They', 'have', 'to',
'because', 'they', 'know', 'that', 'the', 'weather', 'and', 'the', 'planet', 'wait', 'for',
'no', 'person.', 'We', 'are', 'the', 'key', 'to', 'unlocking', 'inclusive,', 'equitable,',
'sustainable', 'and', 'resilient', 'food', 'systems', 'and', 'economies', 'with', 'f',
'armers', 'of', 'all', 'shapes', 'and', 'sizes', 'at', 'their', 'heart', 'and', 'which',
'leave', 'no', 'farmer', 'behind', '.', 'F', 'armers', 'organi', 's', 'ations', 'and',
'agricultural', 'cooperatives', 'provide', 'a', 'strong', 'foundation', 'on', 'which', 'to',
```

N.B. We installed spacy sm and lg model before (source: `https://anaconda.org/conda-forge/spacy-model-en_core_web_sm`)

From line 288: Data Visualization (NLP)

1. Counting words

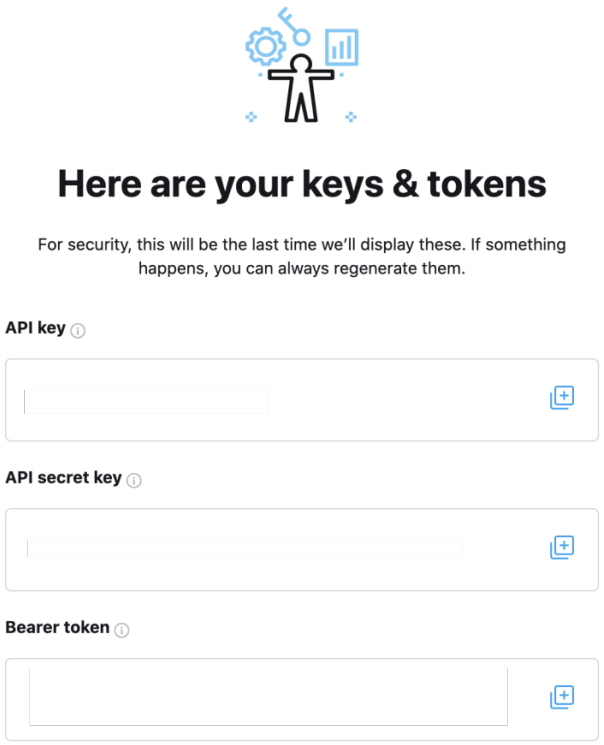
```
288  # In[ ]: VISUALIAZTION
289
290  #count words again with nltk
291
292  df = df2[0].value_counts()
293
294  #df
295  #df['freq'] = df.groupby(0)[0].transform('count')
296  #df['freq'] = df.groupby(0)[0].transform('count')
297  #df.sort_values(by = ('freq'), ascending=False)
298  #This will give frequencies of our words
299
300  from nltk.probability import FreqDist
301
302  freqdoctor = FreqDist()
303
304  for words in df:
305      freqdoctor[words] += 1
306
307  freqdoctor
```

2. Creating plot with top overall words

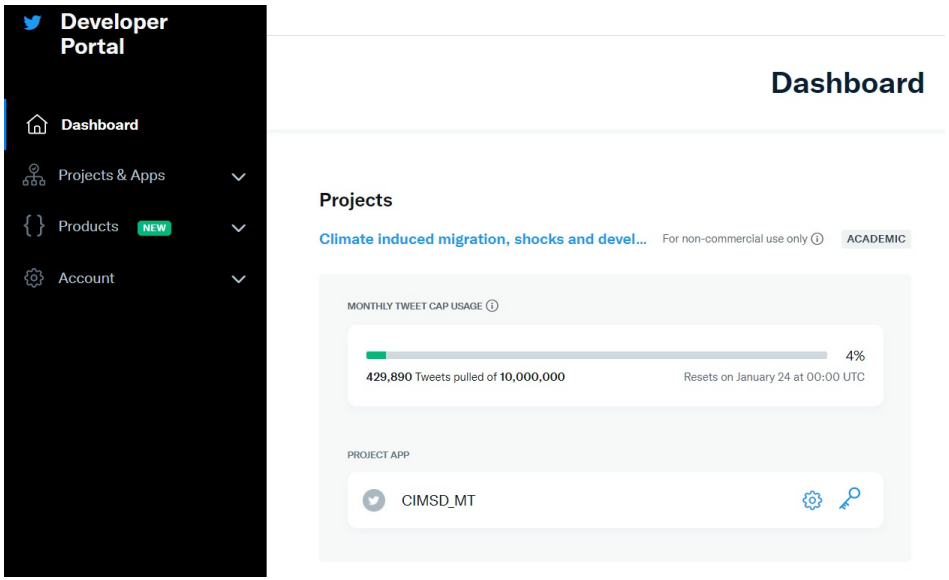
```
309  ##plot
310
311  import matplotlib.pyplot as plt; plt.rcdefaults()
312  import numpy as np
313  import matplotlib.pyplot as plt
314  import seaborn as sns
315
316  #This is a simple plot that shows the top 20 words being used
317  #df.plot(20)
318
319  df = df[:20,]
320  plt.figure(figsize=(10,5))
321  sns.barplot(df.values, df.index, alpha=0.8)
322  plt.title('Top Words Overall')
323  plt.ylabel('Word from Opening Statements', fontsize=12)
324  plt.xlabel('Count of Words', fontsize=12)
325  plt.show()
```

Twitter Scraping: Pre-requisites

To extract tweets you need to apply at *https://developer.twitter.com/en/products/twitter-api/academic-research* and present your project



Which will give you a bearer token, that will allow you to program Python queries to obtain information from Twitter



Packages and Environment variables

Required Packages:

```
# -- coding: utf-8 --
import pandas as pd
import requests
import time
```

Line 24: Put the previously obtained bearer token The 'headers' object uses the token and returns headers we will use for scraping.

```
23  #BEARER TOKEN
24  bearer_token = "-"
25  #CREATE URL
26  search_url = "https://api.twitter.com/2/tweets/search/all"
27  #CREATE HEADERS
28  headers = {"Authorization": "Bearer {}".format(bearer_token)}
29
```

Scraping

By using the requests package, execute queries on the desired keywords, and set specific days to concentrate on days of the COP meeting, among other settings

```
query_params = {'query': 'lang:en (cop26)',
            'start_time': {},
            "end_time": yesterday.isoformat("T") + "Z",
            'max_results': 500,
            'expansions': 'author_id,in_reply_to_user_id,geo.place_id',
            'tweet.fields': 'id,text,author_id,in_reply_to_user_id,geo,conversation_id,created_at
            'user.fields': 'id,name,username,created_at,description,public_metrics,verified',
            'place.fields': 'full_name,id,country,country_code,geo,name,place_type'}
```

Store the results for the analysis

```
        #Creating a list of responses from the scrapping
        response = requests.request("GET", search_url, headers=headers, params=query_params)
        #Technical threshold to avoid potential errors in the process
        if response.status_code == 429:
            time.sleep(60)
            response = requests.request("GET", search_url, headers=headers, params=query_params)
        #Falsification like text (Automatically python sends back an error if response code!=200)
        if response.status_code != 200:
            raise Exception(response.status_code, response.text)
        #The "clean" responses are saved in a different list (in JSON format)
        r = response.json()
        #Saving the scrapped data as text
        for e in r["data"]:
            #print(e["text"])
            mytext.append(e["text"])
    #Dataframing the text, with the help of Panda, in order to be saved in an editing format whether in csv or in xlsx
    df = pd.DataFrame(r["data"]) #correct one
    #Exporting data in an Excel and/or csv
    df.to_excel("twitterscraping.xlsx")
    df.to_csv("twitterscraping.csv")
```

NLP-Step 1: Importing packages and loading spacy model

Required packages:

```
#ANALYSIS
import nltk
nltk.download('punkt')
nltk.download('wordnet')
from nltk.stem.snowball import SnowballStemmer
import pandas as pd
import re
import spacy
nlp = spacy.load('en_core_web_lg')
```

Split the tweets into lines...

Inde	Type	Size	Value
0	str	127	The latest Architects News from piqu! https://t.co/W1ZTwuloQ0 Thanks t ...
1	str	195	Caroline May and Charles Winch contributed an article published in Law ...
2	str	223	If @ScotGovFM was THAT CONCERNED as she claims. The Louisa Jordan woul ...
3	str	278	Tracking today: US @CommerceGov holds civil nuclear trade advisory com ...
4	str	143	The latest Journal of #Sustainability #CSR & #Marketing! https://t ...
5	str	101	Syria: How an ecological disaster can unfold in mere decades.

NLP-Step 2: Stemming + Removing Stop words

Remove RT (Retweet acronyms)

```python
lines = [re.sub(r'[^A-Za-z0-9]+', '', x) for x in lines]

lines

lines2 = []

for word in lines:
    if word != '':
        lines2.append(word)

#Removing RT (Retweet condition within tweet)
for word in lines:
    if word == 'RT':
        lines2.remove(word)

```

Stem words to their root

```python
#This is stemming the words to their root
from nltk.stem.snowball import SnowballStemmer

# The Snowball Stemmer requires that you pass a language parameter
s_stemmer = SnowballStemmer(language='english')

stem = []
for word in lines2:
    stem.append(s_stemmer.stem(word))

stem

#Part 3
#Removing all Stop Words
import spacy
nlp = spacy.load('en_core_web_lg')
stem2 = []

for word in stem:
    if word not in nlp.Defaults.stop_words:
        stem2.append(word)

```

Remove all Stop words

```python
import spacy
nlp = spacy.load('en_core_web_lg')
stem2 = []

for word in stem:
    if word not in nlp.Defaults.stop_words:
        stem2.append(word)

stem2
```

NLP-Step 3: Count words and Visualization

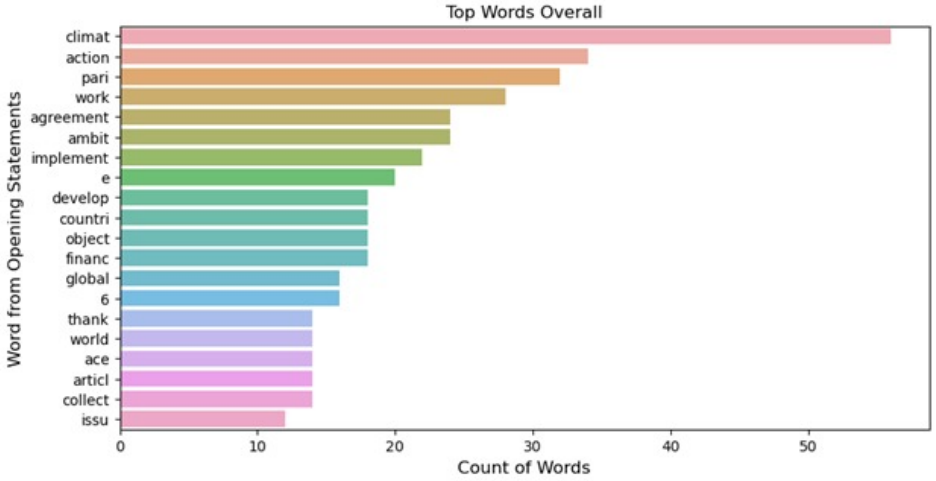
Count word frequency

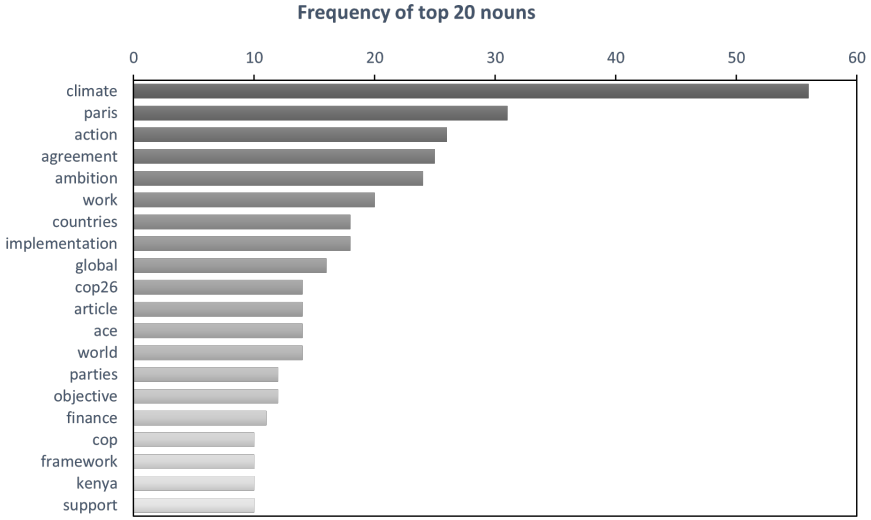
```
from nltk.probability import FreqDist

freqdoctor = FreqDist()

for words in df:
    freqdoctor[words] += 1

freqdoctor
```

Create and label graphs

```
df7 = df7['Word'].value_counts()
df = df7[:20,]
plt.figure(figsize=(10,5))
sns.barplot(df.values, df.index, alpha=0.8)
plt.title('Top Organizations Mentioned')
plt.ylabel('Word from Tweet', fontsize=12)
plt.xlabel('Count of Words', fontsize=12)
plt.show()
```

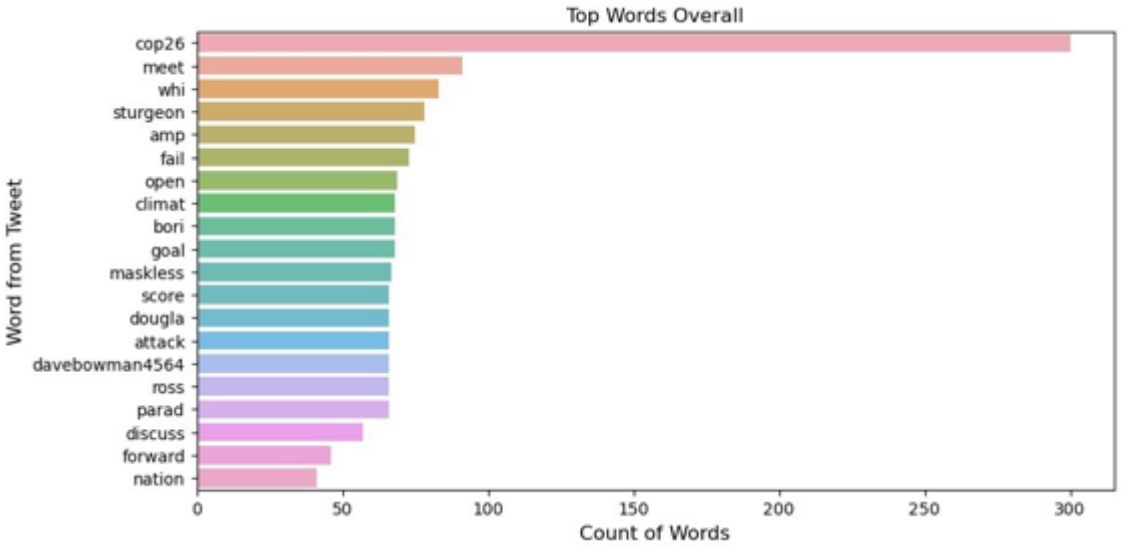
PDF Graphs

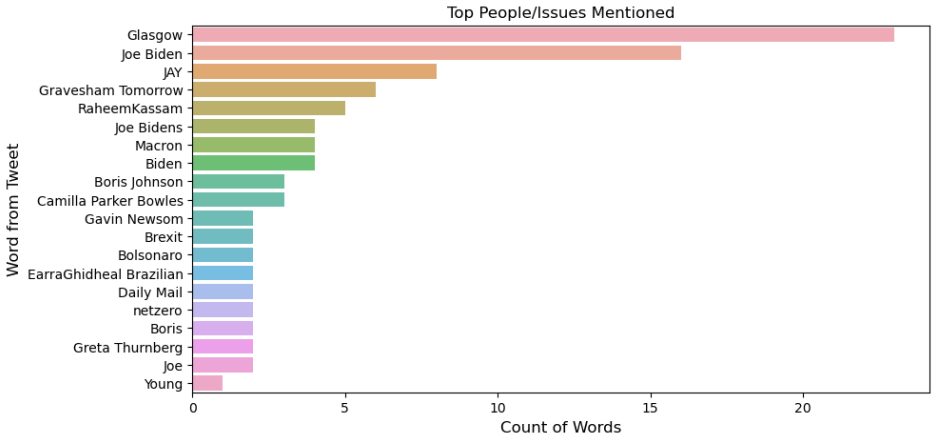


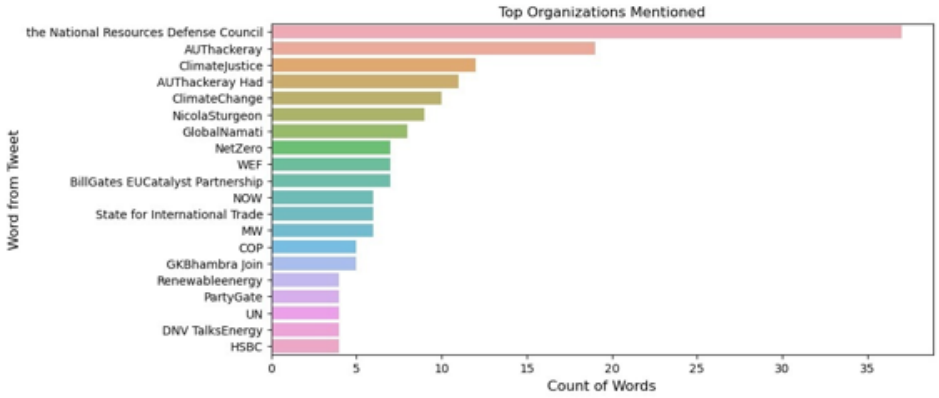


Note: Excel post-editing

Twitter Graphs







Analysis

PDF Scraping

- Climate (56) is the most used noun with a lead of 25 words, which is in line with one's expectantions for speeches at the COP26
- Top words like action (26), agreement (25), implemenation (18) and article (14) characterize, at least formally, a strongly solution-oriented approach
- Negative words like devastating (4), damage (4), emergency (2), disaster (2) or instability (2) are used very rarely indicating a reluctance of producing unpleasant emotions
- The relatively frequent use of the country name Kenya (10) seems to be idiosyncratic, as the next country name in the ranking is India (2) with a much lower frequency

Twitter Scraping

- At first glance, we observe a diverse composition of apparently unrelated and incomplete Twitter keywords
- Presumably, this is a result of the high variety of differing motives behind each and every individual tweet
- While top words like "fail" and "attack" can typically characterize a complaining Twitter user, they also create the impression of a rather high degree of pessimism most likely deriving from low expectations for the outcome of cop26
- The relatively frequent use of the word "maskless" in relation to cop26, suggests that twitter users might observe and even criticize a loose enforcement of mask rules among cop26 participants

Word-Cloud: A duality ? (using wordart.com)

PDF

Twitter

Merci!

References

1. Fenniak, M., (2006) PyPDF, All rights reserved
2. Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585, 357–362. https://doi.org/10.1038/s41586-020-2649-2
3. Honnibal, M., Montani, I. (2017). spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing.
4. McKinney, W., et al. (2010). Data structures for statistical computing in python. In Proceedings of the 9th Python in Science Conference (Vol. 445, pp. 51–56)
5. Porter, M. F. (2001). Snowball: A language for stemming algorithms Published online (Accessed 11.03.2008, 15.00h)
6. Richardson, L. (2007). Beautiful soup documentation. April.
7. Van Rossum, G. (2020). The Python Library Reference, release 3.8.2. Python Software Foundation.