Setup

In [1]:
```
from bs4 import BeautifulSoup
import json
import logging
from newsapi import NewsApiClient
import numpy as np
import os
import pandas as pd
import random
import re
import requests
import string
import time
```

Source URLs

In [ ]:
```
api_key = os.environ['NewsAPIKey']
newsapi = NewsApiClient(api_key=api_key)

sources = newsapi.get_sources(language='en', country='us')
print(' - '.join([source['id'] for source in sources['sources']]))
```

In [ ]:
```
def news_api_urls(q=None,
                  s=None,
                  d_from='2023-05-01',
                  d_to='2023-05-31',
                  api_lst=[]):
    all_articles = newsapi.get_everything(q=q,
                                          sources=s,
                                          from_param=d_from,
                                          to=d_to,
                                          language='en',
                                          sort_by='relevancy',
                                          page=1)

    for article in all_articles['articles']:
        print('Title:', article['title'])

    source_data01 = [(a['source']['name'],
                      a['author'],
                      a['title'],
                      a['url'],
                      a['publishedAt'],
                      a['content'])
                     for a in all_articles['articles']]

    api_lst.extend(source_data01)
    print(len(api_lst))
```

In [ ]:
```
# Already executed:
# run 'A': 'the-washington-post', '2023-05-30'
# run 'B': 'the-washington-post', '2023-05-20/29'
# run 'C': 'the-washington-post', '2023-05-21/22/23/24/25/26/27/28'
# run 'D': 'the-washington-post', '2023-05-10/11/12/13/14/15/16/17/18/19'
# run 'E': 'the-washington-post', '2023-05-05/06/07/08/09/31', '2023-06-01/02/03'
# run 'F': 'fox-news', '2023-05-24/25/26/27/28'
# run 'G': 'breitbart-news', '2023-05-24/25/26/27/28'
```

```
# run 'H': 'cnn', '2023-05-24/25/26/27/28'

# Last to execute:
#run = 'H'
#source_lst = ['cnn']
#date_lst = ['2023-05-24', '2023-05-25', '2023-05-26', '2023-05-27', '2023-05-28']

q_word_lst = ['justice OR surveillance', 'healthcare OR "health care"',
              '(political AND (bias OR party)) OR republican OR democrat OR election',
              'security AND (social OR national)']
```

In [ ]:
```
api_record_lst01 = []
for s in source_lst:
    print(f'Source: {s}')
    for d in date_lst:
        print(f'Date: {d}')
        for q in q_word_lst:
            print(f'Query word: {q}')
            time.sleep(5 * random.random())
            news_api_urls(q=q,
                          s=s,
                          d_from=d,
                          d_to=d,
                          api_lst=api_record_lst01)
```

Scrape articles

In [ ]:
```
api_df = pd.DataFrame(api_record_lst01, columns=['source_name', 'author', 'title', 'url',
api_df['article_text'] = ''

total_urls = len(api_df)
for i, row in enumerate(api_df.itertuples(), 1):
    print(f'Retrieving url {i} of {total_urls}...', end='')
    response = requests.get(row.url)

    if response.status_code == 200:
        print('; now Scraping...', end='')

        soup = BeautifulSoup(response.content, 'html.parser')

        # The Washington Post
#         try:
#             script_tag = soup.find('script', {'type': 'application/ld+json'})
#             article_json = json.loads(script_tag.string)
#             article_content = article_json['hasPart']['value']
#             api_df.at[row.Index, 'article_text'] = article_content

        # Fox News
#         try:
#             script_tag = soup.find('script', {'type': 'application/ld+json'})
#             article_json = json.loads(script_tag.string)
#             article_content = article_json['articleBody']
#             api_df.at[row.Index, 'article_text'] = article_content

        # Breitbart News
#         try:
#             title_tag = soup.find('h1')
#             if title_tag is not None:
#                 title = title_tag.text
#             content_tags = soup.find_all(['p', 'blockquote'])
#             content = " ".join([tag.text for tag in content_tags if tag.text.strip() !=
#             api_df.at[row.Index, 'article_text'] = content
```

```
        # CNN News
        try:
            script_tag = soup.find('script', {'type': 'application/ld+json'})
            article_json = json.loads(script_tag.string)
            article_content = article_json['articleBody']
            api_df.at[row.Index, 'article_text'] = article_content

        except KeyError:
            print('; missing key in article JSON!', end='')

        time.sleep(5 * random.random())
        print('; done!')
    else:
        print(f'  response is {response.status_code}!')

api_df.to_csv(f'509_final_proj-{run}.csv', index=False)
```

Combine output files (*only at end of iterative process*)

In [ ]:

```
# Combine all the output files
master_df = pd.DataFrame()
for letter in string.ascii_lowercase:
    file_name = f'509_final_proj-{letter.upper()}.csv'

    if os.path.isfile(file_name):
        df = pd.read_csv(file_name)

        master_df = pd.concat([master_df, df], ignore_index=True)
    else:
        break
print(master_df.info())

# Get rid of what appear to be very cluttered (misread) article text rows
pattern = re.compile('\n{3,}')
rows_list = []
for index, row in master_df.iterrows():
    try:
        processed_row = row
        if not pattern.search(str(processed_row[6])):
            rows_list.append(processed_row)
    except Exception as e:
        print(f'Error processing row {index}: {e}')
print(f'Rows processed = {len(rows_list)}')
new_df = pd.concat(rows_list, axis=1).transpose()
new_df.columns = master_df.columns

# Save the new file
new_df.to_csv('509_final_proj.csv', index=False)
```

Combine all final files

In [2]:

```
df0 = pd.read_csv('../data/509_final_proj.csv')
df1 = pd.read_csv('../data/data_parsed_amc.csv')
df2 = pd.read_csv('../data/News_API_FOX_CNN_Breitbert_May18_23_May31_3.csv')
```

In [3]:

```
print(df0.info())
print(df1.info())
print(df2.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1594 entries, 0 to 1593
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   source_name   1594 non-null   object
 1   author        1564 non-null   object
 2   title         1593 non-null   object
 3   url           1593 non-null   object
 4   publishedAt   1593 non-null   object
 5   content       1593 non-null   object
 6   article_text  1561 non-null   object
dtypes: object(7)
memory usage: 87.3+ KB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3390 entries, 0 to 3389
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   source_name     3390 non-null   object
 1   author          3375 non-null   object
 2   title           3390 non-null   object
 3   url             3390 non-null   object
 4   publish_date    3390 non-null   object
 5   article_parsed  3375 non-null   object
dtypes: object(6)
memory usage: 159.0+ KB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1302 entries, 0 to 1301
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   Unnamed: 0.1    1302 non-null   int64 
 1   Unnamed: 0      1302 non-null   int64 
 2   Source          1302 non-null   object
 3   Author          1302 non-null   object
 4   Title           1302 non-null   object
 5   URL             1302 non-null   object
 6   date            1302 non-null   object
 7   content         1302 non-null   object
 8   article_parsed  1295 non-null   object
dtypes: int64(2), object(7)
memory usage: 91.7+ KB
None
```

In [4]:

```
df0 = df0.rename(columns={'publishedAt': 'publish_date'})

df1 = df1.rename(columns={'article_parsed': 'article_text'})
df1['content'] = np.nan

df2 = df2.drop(columns=['Unnamed: 0.1', 'Unnamed: 0'])
df2 = df1.rename(columns={'Source': 'source_name',
                          'Author': 'author',
                          'Title': 'title',
                          'URL': 'url',
                          'date': 'publish_date',
                          'article_parsed': 'article_text'})

df = pd.concat([df0, df1, df2], ignore_index=True)

df = df.drop_duplicates(subset='article_text')
df.info()
```

```
df.to_csv('../data/master.csv', index=False)
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 4509 entries, 0 to 4983
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   source_name   4509 non-null   object
 1   author        4472 non-null   object
 2   title         4509 non-null   object
 3   url           4509 non-null   object
 4   publish_date  4509 non-null   object
 5   content       1158 non-null   object
 6   article_text  4508 non-null   object
dtypes: object(7)
memory usage: 281.8+ KB
```

In [ ]: