```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Retrieve and store local representation of MMWR online with minimal processing,
which can include conversion to UTF-8 and basic parsing of HTML.

@author: chadheilig

Explore computational resources for mirroring MMWR from online to local,
particularly file sizes and processing times in support of
mmwr_2-retrieve-and-store.py.
"""

#%% Import modules and set up environment
import os
from os.path import join, expanduser, normpath
import pickle
# from urllib.parse import urlparse, urljoin, urlunparse
from bs4 import BeautifulSoup, UnicodeDammit
from bs4.formatter import HTMLFormatter
import requests
import re
import multiprocessing
from tqdm import tqdm, trange
import random
import time
import pandas as pd
pd.set_option('display.expand_frame_repr', False) # show/wrap all DF columns

os.chdir('/Users/chadheilig/Temp/mmwr-as-corpus/_test')
MMWR_BASE_PATH = normpath(expanduser('~/cdc-corpora/mmwr_temp/'))

#%% Function to experiment with sizes of processed HTML
# experiment with differenct sequences of operations
# b0   -> reduce_space    -> b1  -> insert_newlines    -> b2
# b0   -> to_unicode      -> u0  -> reduce_space       -> u00
#                            u0  -> prettify           -> [don't]
# u00  -> insert_newlines -> u01
# u00  -> prettify        -> u02 -> trim_leading       -> u03
# b1   -> to_unicode      -> u1  -> [stop; u1 == u00]
# b2   -> to_unicode      -> u2  -> [stop; u2 == u01]
# b0 -> b1 -> b2; b0 -> u0 -> u00 -> u01; u00 -> u02   -> u03
# b0, b1, b2, u0, u00, u01, u02, u03
def measure_html_size(html, counter = None):
   "Given raw HTML, return lengths of several encodings."
   if counter is not None:
      print(f'{counter:05d}', end = '')
   b0 = html                          # raw HTML
   u0 = html_to_unicode_b(b0)
   b1 = html_reduce_space_b(b0)    # scrubbed of excess white space
#  u1 = html_to_unicode_b(b1)
   b2 = html_insert_newlines_b(b1) # no excess space, judicious newlines
#  u2 = html_to_unicode_b(b2)

   # u[0-2]0 scrub u[0-2] of excess space
   # u[0-2]1 judiciously insert \n into u[0-2]0
   # u[0-2]2 prettify u[0-2]0
   # u[0-2]3 scrub u[0-2]2 of leading spaces
   u00 = html_reduce_space_u(u0)
   u01 = html_insert_newlines_u(u00)
   u02 = html_prettify_u(u00)
   u03 = trim_leading_space_u(u02)

#  u10 = html_reduce_space_u(u1)
#  u11 = html_insert_newlines_u(u10)
#  u12 = html_prettify_u(u10)
#  u13 = trim_leading_space_u(u12)

#  u20 = html_reduce_space_u(u2)
#  u21 = html_insert_newlines_u(u20)
#  u22 = html_prettify_u(u20)
#  u23 = trim_leading_space_u(u22)

   if counter is not None:
      print('.', end = ' ')
   return dict(
      b0 = len(b0), u0 = len(u0),
      u00 = len(u00), u01 = len(u01), u02 = len(u02), u03 = len(u03),
      b1 = len(b1),# u1 = len(u1),
#      u10 = len(u10), u11 = len(u11), u12 = len(u12), u13 = len(u13),
      b2 = len(b2)#, u2 = len(u2),
#      u20 = len(u20), u21 = len(u21), u22 = len(u22), u23 = len(u23)
      )

def write_html(html, b_path, u_path, counter = None):
   "Given raw HTML, return lengths of several encodings."
   if counter is not None:
      print(f'{counter:05d}', end = '')
   b0 = html                         # raw HTML
   u0 = html_to_unicode_b(b0)        # convert bytes to UTF-8
   u00 = html_reduce_space_u(u0)     # scrub u0 of excess space (esp. \r)
   u02 = html_prettify_u(u00)        # prettify u00
   u03 = trim_leading_space_u(u02) # scrub u02 of leading spaces
   with open(b_path, 'bw') as file_out:
       file_out.write(b0)
   with open(u_path, 'w') as file_out:
       file_out.write(u03)
   if counter is not None:
      print('.', end = ' ')
   return None

#%% Explore computational costs of various processing methods

#%% Explore computational costs of various processing methods


## Processing times from raw HTML (bytes) to processed HTML
#%% Store HTML files in local mirror(s)

# Mirror all files and track byte size
# b0: raw, unprocessed HTML file as retrived from the internet
# u3: UTF-8 and lightly reformatted file for local mirror
```

```
MMWR_BASE_PATH_b0 = normpath(expanduser('~/cdc-corpora/mmwr_b0/'))
create_mmwr_tree(MMWR_BASE_PATH_b0)
MMWR_BASE_PATH_u3 = normpath(expanduser('~/cdc-corpora/mmwr_u3/'))
create_mmwr_tree(MMWR_BASE_PATH_u3)

# Write original raw HTML and UTF-8/prettified HTML,
# while tracking file sizes
for count, (html, path) in enumerate(zip(mmwr_raw_html, mmwr_dframe.path)):
   write_html(html, MMWR_BASE_PATH_b0 + path, MMWR_BASE_PATH_u3 + path, count)

## File sizes
# /mmwr/preview/mmwrhtml/ss5704a1.htm # 13325
# mmwr_dframe.iloc[13325]
measure_html_size(mmwr_raw_html[13325])
start_time = time.time()
html_sizes = [measure_html_size(html, count)
   for count, html in enumerate(mmwr_raw_html)]
print(f"\nTime elapsed: {int((time.time() - start_time) // 60)} min {round((time.time() -
start_time) % 60, 1)} sec")
# ~80 minutes with 18 measures; ~30.5 minutes with 8
pickle.dump(html_sizes, open('html_sizes.pkl', 'wb'))
html_sizes_df = pd.DataFrame(html_sizes)
html_sizes_df.to_excel('html_sizes.xlsx', engine='openpyxl')
html_sizes_df.to_csv('html_sizes.csv')

## Processing times from raw HTML (bytes) to processed HTML

# randomly select 1000 articles
random.seed(24601)
samp_1000 = random.sample(range(13792), 1000)
samp_html = [mmwr_raw_html[i] for i in samp_1000]
x_b1 = [html_reduce_space_b(html) for html in samp_html]
x_b2 = [html_insert_newlines_b(html_reduce_space_b(html)) for html in samp_html]
x_u0 = [html_to_unicode_b(html) for html in samp_html]
x_u00 = [html_reduce_space_u(html_to_unicode_b(html)) for html in samp_html]
x_u01 = [html_insert_newlines_u(html_reduce_space_u(html_to_unicode_b(html)))
   for html in samp_html]
x_u02 = [html_prettify_u(html_reduce_space_u(html_to_unicode_b(html)))
   for html in samp_html]
x_u03 = [trim_leading_space_u(html_prettify_u(html_reduce_space_u(
   html_to_unicode_b(html)))) for html in samp_html]

start_time = time.time()
%timeit -n 1 -r 10 x_b1 = [html_reduce_space_b(html) for html in samp_html]
%timeit -n 1 -r 10 x_b2 = [html_insert_newlines_b(html_reduce_space_b(html)) for html in
samp_html]
%timeit -n 1 -r 10 x_u0 = [html_to_unicode_b(html) for html in samp_html]
%timeit -n 1 -r 10 x_u00 = [html_reduce_space_u(html_to_unicode_b(html)) for html in
samp_html]
%timeit -n 1 -r 10 x_u01 =
[html_insert_newlines_u(html_reduce_space_u(html_to_unicode_b(html))) for html in
samp_html]
%timeit -n 1 -r 10 x_u02 = [html_prettify_u(html_reduce_space_u(html_to_unicode_b(html)))
for html in samp_html]
%timeit -n 1 -r 10 x_u03 =
[trim_leading_space_u(html_prettify_u(html_reduce_space_u(html_to_unicode_b(html)))) for
html in samp_html]
print(f"\nTime elapsed: {int((time.time() - start_time) // 60)} min {round((time.time() - start_time) % 60, 1)} sec")

# 19.8 s ± 207 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
# 28.3 s ± 458 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
# 240 ms ± 7.47 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
# 19.2 s ± 388 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
# 32.8 s ± 3.94 s per loop (mean ± std. dev. of 7 runs, 1 loop each)
# 1min 46s ± 6.54 s per loop (mean ± std. dev. of 7 runs, 1 loop each)
# 1min 40s ± 6.71 s per loop (mean ± std. dev. of 7 runs, 1 loop each)

# 19.7 s ± 2.05 s per loop (mean ± std. dev. of 10 runs, 1 loop each)
# 23.2 s ± 1.07 s per loop (mean ± std. dev. of 10 runs, 1 loop each)
# 177 ms ± 9.8 ms per loop (mean ± std. dev. of 10 runs, 1 loop each)
# 15 s ± 145 ms per loop (mean ± std. dev. of 10 runs, 1 loop each)
# 22.9 s ± 1.61 s per loop (mean ± std. dev. of 10 runs, 1 loop each)
# 1min 9s ± 4.55 s per loop (mean ± std. dev. of 10 runs, 1 loop each)
# 1min 9s ± 2.46 s per loop (mean ± std. dev. of 10 runs, 1 loop each)
# 37:50.5 total run

#%% Store HTML files in local mirror(s)

# Mirror all files and track byte size
# b0:     length of requests.get(url).content (unprocessed HTML file)
# u1, b1: length of UnicodeDammit(b0,.).unicode_markup
# u2, b2: length of re.sub(r'\s+', ' ', u1)
# u3, b3: length of BeautifulSoup(u2, 'lxml').prettify()
MMWR_BASE_PATH_b0 = normpath(expanduser('~/cdc-corpora/mmwr_b0/'))
create_mmwr_tree(MMWR_BASE_PATH_b0)
MMWR_BASE_PATH_u3 = normpath(expanduser('~/cdc-corpora/mmwr_u3/'))
create_mmwr_tree(MMWR_BASE_PATH_u3)

# Write original raw HTML and UTF-8/prettified HTML,
# while tracking file sizes
start_time = time.time()
for count, (html, path) in enumerate(zip(mmwr_raw_html, mmwr_dframe.path)):
   write_html(html, MMWR_BASE_PATH_b0 + path, MMWR_BASE_PATH_u3 + path, count)
print(f"\nTime elapsed: {int((time.time() - start_time) // 60)} min {round((time.time() - start_time) % 60, 1)} sec")
# ~24 minutes

#%% Restore HTML from local mirror

start_time = time.time()
mmwr_html_pkl = pickle.load(open('mmwr_raw_html.pkl', 'rb'))
print(f"\nTime elapsed: {int((time.time() - start_time) // 60)} min {round((time.time() - start_time) % 60, 1)} sec")

start_time = time.time()
mmwr_html_b0 = [read_raw_html(MMWR_BASE_PATH_b0 + path)
                     for path in mmwr_dframe.path]
print(f"\nTime elapsed: {int((time.time() - start_time) // 60)} min {round((time.time() - start_time) % 60, 1)} sec")

start_time = time.time()
```

```
mmwr_html_u3 = [read_uni_html(MMWR_BASE_PATH_u3 + path)
                     for path in mmwr_dframe.path]
print(f"\nTime elapsed: {int((time.time() - start_time) // 60)} min {round((time.time() - start_time) % 60, 1)} sec")

# mmwr_raw_html == mmwr_html_pkl # True
# mmwr_raw_html == mmwr_html_b0 # True

start_time = time.time()
%timeit -r 10 mmwr_html_pkl = pickle.load(open('mmwr_raw_html.pkl', 'rb'))
print(f"\nTime elapsed: {int((time.time() - start_time) // 60)} min {round((time.time() - start_time) % 60, 1)} sec")
1.64 s ± 489 ms per loop (mean ± std. dev. of 10 runs, 1 loop each)
Time elapsed: 0 min 24.3 sec

start_time = time.time()
%timeit -r 10 mmwr_html_b0 = [read_raw_html(MMWR_BASE_PATH_b0 + path) for path in mmwr_dframe.path]
print(f"\nTime elapsed: {int((time.time() - start_time) // 60)} min {round((time.time() - start_time) % 60, 1)} sec")
21.4 s ± 4.63 s per loop (mean ± std. dev. of 10 runs, 1 loop each)
Time elapsed: 4 min 3.3 sec

start_time = time.time()
%timeit -r 10 mmwr_html_u3 = [read_uni_html(MMWR_BASE_PATH_u3 + path) for path in mmwr_dframe.path]
print(f"\nTime elapsed: {int((time.time() - start_time) // 60)} min {round((time.time() - start_time) % 60, 1)} sec")
23.6 s ± 4.9 s per loop (mean ± std. dev. of 10 runs, 1 loop each)
Time elapsed: 4 min 13.9 sec
```