

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  """
4  Retrieve and store local representation of MMWR online with minimal processing,
5  which can include conversion to UTF-8 and basic parsing of HTML.
6
7  @author: chadheilig
8
9  Explore computational resources for mirroring MMWR from online to local,
10 particularly file sizes and processing times in support of
11 mmwr_2-retrieve-and-store.py.
12 """
13
14 %% Import modules and set up environment
15 import os
16 from os.path import join, expanduser, normpath
17 import pickle
18 # from urllib.parse import urlparse, urljoin, urlunparse
19 from bs4 import BeautifulSoup, UnicodeDammit
20 from bs4.formatter import HTMLFormatter
21 import requests
22 import re
23 import multiprocessing
24 from tqdm import tqdm, trange
25 import random
26 import time
27 import pandas as pd
28 pd.set_option('display.expand_frame_repr', False) # show/wrap all DF columns
29
30 os.chdir('/Users/chadheilig/Temp/mmwr-as-corpus/_test')
31 MMWR_BASE_PATH = normpath(expanduser('~/.cdc-corpora/mmwr_temp/'))
32
33 %% Function to experiment with sizes of processed HTML
34 # experiment with different sequences of operations
35 # b0 -> reduce_space -> b1 -> insert_newlines -> b2
36 # b0 -> to_unicode -> u0 -> reduce_space -> u00
37 # u0 -> prettify -> [don't]
38 # u00 -> insert_newlines -> u01
39 # u00 -> prettify -> u02 -> trim_leading -> u03
40 # b1 -> to_unicode -> u1 -> [stop; u1 == u00]
41 # b2 -> to_unicode -> u2 -> [stop; u2 == u01]
42 # b0 -> b1 -> b2; b0 -> u0 -> u00 -> u01; u00 -> u02 -> u03
43 # b0, b1, b2, u0, u00, u01, u02, u03
44 def measure_html_size(html, counter = None):
45     "Given raw HTML, return lengths of several encodings."
46     if counter is not None:
47         print(f'{counter:05d}', end = '')
48         b0 = html # raw HTML
49         u0 = html_to_unicode_b(b0)
50         b1 = html_reduce_space_b(b0) # scrubbed of excess white space
51         u1 = html_to_unicode_b(b1)
52         b2 = html_insert_newlines_b(b1) # no excess space, judicious newlines
53         u2 = html_to_unicode_b(b2)
54
55         # u[0-2] scrub u[0-2] of excess space
56         # u[0-2]1 judiciously insert \n into u[0-2]0

```

```

57     # u[0-2]2 prettify u[0-2]0
58     # u[0-2]3 scrub u[0-2]2 of leading spaces
59     u00 = html_reduce_space_u(u0)
60     u01 = html_insert_newlines_u(u00)
61     u02 = html_prettify_u(u00)
62     u03 = trim_leading_space_u(u02)
63
64     # u10 = html_reduce_space_u(u1)
65     # u11 = html_insert_newlines_u(u10)
66     # u12 = html_prettify_u(u10)
67     # u13 = trim_leading_space_u(u12)
68
69     # u20 = html_reduce_space_u(u2)
70     # u21 = html_insert_newlines_u(u20)
71     # u22 = html_prettify_u(u20)
72     # u23 = trim_leading_space_u(u22)
73
74     if counter is not None:
75         print('.', end = ' ')
76     return dict(
77         b0 = len(b0), u0 = len(u0),
78         u00 = len(u00), u01 = len(u01), u02 = len(u02), u03 = len(u03),
79         b1 = len(b1), # u1 = len(u1),
80     #     u10 = len(u10), u11 = len(u11), u12 = len(u12), u13 = len(u13),
81     b2 = len(b2), # u2 = len(u2),
82     #     u20 = len(u20), u21 = len(u21), u22 = len(u22), u23 = len(u23)
83     )
84
85 def write_html(html, b_path, u_path, counter = None):
86     "Given raw HTML, return lengths of several encodings."
87     if counter is not None:
88         print(f'{counter:05d}', end = ' ')
89     b0 = html # raw HTML
90     u0 = html_to_unicode_b(b0) # convert bytes to UTF-8
91     u00 = html_reduce_space_u(u0) # scrub u0 of excess space (esp. \r)
92     u02 = html_prettify_u(u00) # prettify u00
93     u03 = trim_leading_space_u(u02) # scrub u02 of leading spaces
94     with open(b_path, 'bw') as file_out:
95         file_out.write(b0)
96     with open(u_path, 'w') as file_out:
97         file_out.write(u03)
98     if counter is not None:
99         print('.', end = ' ')
100     return None
101
102 ### Explore computational costs of various processing methods
103
104 ### Explore computational costs of various processing methods
105
106
107 ## Processing times from raw HTML (bytes) to processed HTML
108 ## Store HTML files in local mirror(s)
109
110 # Mirror all files and track byte size
111 # b0: raw, unprocessed HTML file as retrieved from the internet
112 # u3: UTF-8 and lightly reformatted file for local mirror

```

```

113 MMWR_BASE_PATH_b0 = normpath(expanduser('~cdc-corpora/mmwr_b0/'))
114 create_mmwr_tree(MMWR_BASE_PATH_b0)
115 MMWR_BASE_PATH_u3 = normpath(expanduser('~cdc-corpora/mmwr_u3/'))
116 create_mmwr_tree(MMWR_BASE_PATH_u3)
117
118 # Write original raw HTML and UTF-8/prettified HTML,
119 # while tracking file sizes
120 for count, (html, path) in enumerate(zip(mmwr_raw_html, mmwr_dframe.path)):
121     write_html(html, MMWR_BASE_PATH_b0 + path, MMWR_BASE_PATH_u3 + path, count)
122
123 ## File sizes
124 # /mmwr/preview/mmwrhtml/ss5704a1.htm # 13325
125 # mmwr_dframe.iloc[13325]
126 measure_html_size(mmwr_raw_html[13325])
127 start_time = time.time()
128 html_sizes = [measure_html_size(html, count)
129     for count, html in enumerate(mmwr_raw_html)]
130 print(f"\nTime elapsed: {int((time.time() - start_time) // 60)} min {round((time.time() -
130 start_time) % 60, 1)} sec")
131 # ~80 minutes with 18 measures; ~30.5 minutes with 8
132 pickle.dump(html_sizes, open('html_sizes.pkl', 'wb'))
133 html_sizes_df = pd.DataFrame(html_sizes)
134 html_sizes_df.to_excel('html_sizes.xlsx', engine='openpyxl')
135 html_sizes_df.to_csv('html_sizes.csv')
136
137 ## Processing times from raw HTML (bytes) to processed HTML
138
139 # randomly select 1000 articles
140 random.seed(24601)
141 samp_1000 = random.sample(range(13792), 1000)
142 samp_html = [mmwr_raw_html[i] for i in samp_1000]
143 x_b1 = [html_reduce_space_b(html) for html in samp_html]
144 x_b2 = [html_insert_newlines_b(html_reduce_space_b(html)) for html in samp_html]
145 x_u0 = [html_to_unicode_b(html) for html in samp_html]
146 x_u00 = [html_reduce_space_u(html_to_unicode_b(html)) for html in samp_html]
147 x_u01 = [html_insert_newlines_u(html_reduce_space_u(html_to_unicode_b(html)))
148     for html in samp_html]
149 x_u02 = [html_prettify_u(html_reduce_space_u(html_to_unicode_b(html)))
150     for html in samp_html]
151 x_u03 = [trim_leading_space_u(html_prettify_u(html_reduce_space_u(
152     html_to_unicode_b(html)))) for html in samp_html]
153
154 start_time = time.time()
155 %timeit -n 1 -r 10 x_b1 = [html_reduce_space_b(html) for html in samp_html]
156 %timeit -n 1 -r 10 x_b2 = [html_insert_newlines_b(html_reduce_space_b(html)) for html in
156 samp_html]
157 %timeit -n 1 -r 10 x_u0 = [html_to_unicode_b(html) for html in samp_html]
158 %timeit -n 1 -r 10 x_u00 = [html_reduce_space_u(html_to_unicode_b(html)) for html in
158 samp_html]
159 %timeit -n 1 -r 10 x_u01 =
159 [html_insert_newlines_u(html_reduce_space_u(html_to_unicode_b(html))) for html in
159 samp_html]
160 %timeit -n 1 -r 10 x_u02 = [html_prettify_u(html_reduce_space_u(html_to_unicode_b(html)))
160 for html in samp_html]
161 %timeit -n 1 -r 10 x_u03 =
161 [trim_leading_space_u(html_prettify_u(html_reduce_space_u(html_to_unicode_b(html)))) for

```

```

    html in samp_html]
162 print(f"\nTime elapsed: {int((time.time() - start_time) // 60)} min {round((time.time() -
162 start_time) % 60, 1)} sec")
163
164 # 19.8 s ± 207 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
165 # 28.3 s ± 458 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
166 # 240 ms ± 7.47 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
167 # 19.2 s ± 388 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
168 # 32.8 s ± 3.94 s per loop (mean ± std. dev. of 7 runs, 1 loop each)
169 # 1min 46s ± 6.54 s per loop (mean ± std. dev. of 7 runs, 1 loop each)
170 # 1min 40s ± 6.71 s per loop (mean ± std. dev. of 7 runs, 1 loop each)
171
172 # 19.7 s ± 2.05 s per loop (mean ± std. dev. of 10 runs, 1 loop each)
173 # 23.2 s ± 1.07 s per loop (mean ± std. dev. of 10 runs, 1 loop each)
174 # 177 ms ± 9.8 ms per loop (mean ± std. dev. of 10 runs, 1 loop each)
175 # 15 s ± 145 ms per loop (mean ± std. dev. of 10 runs, 1 loop each)
176 # 22.9 s ± 1.61 s per loop (mean ± std. dev. of 10 runs, 1 loop each)
177 # 1min 9s ± 4.55 s per loop (mean ± std. dev. of 10 runs, 1 loop each)
178 # 1min 9s ± 2.46 s per loop (mean ± std. dev. of 10 runs, 1 loop each)
179 # 37:50.5 total run
180
181 ### Store HTML files in local mirror(s)
182
183 # Mirror all files and track byte size
184 # b0: length of requests.get(url).content (unprocessed HTML file)
185 # u1, b1: length of UnicodeDammit(b0,.).unicode_markup
186 # u2, b2: length of re.sub(r'\s+', ' ', u1)
187 # u3, b3: length of BeautifulSoup(u2, 'lxml').prettify()
188 MMWR_BASE_PATH_b0 = normpath(expanduser('~/.cdc-corpora/mmwr_b0/'))
189 create_mmwr_tree(MMWR_BASE_PATH_b0)
190 MMWR_BASE_PATH_u3 = normpath(expanduser('~/.cdc-corpora/mmwr_u3/'))
191 create_mmwr_tree(MMWR_BASE_PATH_u3)
192
193 # Write original raw HTML and UTF-8/prettified HTML,
194 # while tracking file sizes
195 start_time = time.time()
196 for count, (html, path) in enumerate(zip(mmwr_raw_html, mmwr_dframe.path)):
197     write_html(html, MMWR_BASE_PATH_b0 + path, MMWR_BASE_PATH_u3 + path, count)
198 print(f"\nTime elapsed: {int((time.time() - start_time) // 60)} min {round((time.time() -
198 start_time) % 60, 1)} sec")
199 # ~24 minutes
200
201 ### Restore HTML from local mirror
202
203 start_time = time.time()
204 mmwr_html_pkl = pickle.load(open('mmwr_raw_html.pkl', 'rb'))
205 print(f"\nTime elapsed: {int((time.time() - start_time) // 60)} min {round((time.time() -
205 start_time) % 60, 1)} sec")
206
207 start_time = time.time()
208 mmwr_html_b0 = [read_raw_html(MMWR_BASE_PATH_b0 + path)
209                 for path in mmwr_dframe.path]
210 print(f"\nTime elapsed: {int((time.time() - start_time) // 60)} min {round((time.time() -
210 start_time) % 60, 1)} sec")
211
212 start_time = time.time()

```

```
213 mmwr_html_u3 = [read_uni_html(MMWR_BASE_PATH_u3 + path)
214                       for path in mmwr_dframe.path]
215 print(f"\nTime elapsed: {int((time.time() - start_time) // 60)} min {round((time.time() -
215 start_time) % 60, 1)} sec")
216
217 # mmwr_raw_html == mmwr_html_pkl # True
218 # mmwr_raw_html == mmwr_html_b0 # True
219
220 start_time = time.time()
221 %timeit -r 10 mmwr_html_pkl = pickle.load(open('mmwr_raw_html.pkl', 'rb'))
222 print(f"\nTime elapsed: {int((time.time() - start_time) // 60)} min {round((time.time() -
222 start_time) % 60, 1)} sec")
223 1.64 s ± 489 ms per loop (mean ± std. dev. of 10 runs, 1 loop each)
224 Time elapsed: 0 min 24.3 sec
225
226 start_time = time.time()
227 %timeit -r 10 mmwr_html_b0 = [read_raw_html(MMWR_BASE_PATH_b0 + path) for path in
227 mmwr_dframe.path]
228 print(f"\nTime elapsed: {int((time.time() - start_time) // 60)} min {round((time.time() -
228 start_time) % 60, 1)} sec")
229 21.4 s ± 4.63 s per loop (mean ± std. dev. of 10 runs, 1 loop each)
230 Time elapsed: 4 min 3.3 sec
231
232 start_time = time.time()
233 %timeit -r 10 mmwr_html_u3 = [read_uni_html(MMWR_BASE_PATH_u3 + path) for path in
233 mmwr_dframe.path]
234 print(f"\nTime elapsed: {int((time.time() - start_time) // 60)} min {round((time.time() -
234 start_time) % 60, 1)} sec")
235 23.6 s ± 4.9 s per loop (mean ± std. dev. of 10 runs, 1 loop each)
236 Time elapsed: 4 min 13.9 sec
```