```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Extract and organize metadata and text of EID

@author: chadheilig


Main product:
"""

#%% Import modules and set up environment
# import from 0_cdc-corpora-header.py

import time
from bs4 import SoupStrainer

os.chdir('/Users/cmheilig/cdc-corpora/_test')
EID_BASE_PATH_u3 = normpath(expanduser('~/cdc-corpora/eid_u3/'))

# EID DataFrame, articles only
eid_dframe = pickle.load(open('pickle-files/eid_dframe.pkl', 'rb'))
eid_art_dframe = eid_dframe.loc[eid_dframe.level == 'article', ].reset_index(drop=True)
# [11211 rows x 8 columns]

#%% Read HTML from mirror into list of strings

# alternatively, restore from eid_uni_html.pkl and take subset

# eid_art_html = [read_uni_html(EID_BASE_PATH_u3 + path)
#                       for path in tqdm(eid_art_dframe.mirror_path)]
# 10640/10640 [01:00<00:00, 174.56it/s]
# pickle.dump(eid_art_html, open('eid_art_html.pkl', 'wb'))
# eid_art_html = pickle.load(open('eid_art_html.pkl', 'rb'))

eid_art_html = [html_reduce_space_u(read_uni_html(EID_BASE_PATH_u3 + path))
                      for path in tqdm(eid_art_dframe.mirror_path)]
# 11211/11211 [02:07<00:00, 88.07it/s]
# pickle.dump(eid_art_html, open('eid_art_html.pkl', 'wb'))
# eid_art_html = pickle.load(open('eid_art_html.pkl', 'rb'))

#%% Count frequency of various elements across EID corpus

x = BeautifulSoup(eid_art_html[9000], 'lxml')
y = [tag.name for tag in x.find_all(True)]
z = { w: y.count(w) for w in sorted(set(y)) }
# repr(z)
#  {'a': 122, 'address': 1, 'body': 1, 'br': 7, 'button': 11, 'circle': 8,
#   'div': 95, 'em': 3, 'footer': 1, 'form': 3, 'g': 12, 'h3': 1, 'h4': 2,
#   'h5': 3, 'head': 1, 'header': 1, 'hr': 2, 'html': 1, 'i': 3, 'img': 1,
#   'input': 13, 'li': 84, 'link': 15, 'main': 1, 'meta': 35, 'nav': 6,
#   'noscript': 1, 'ol': 1, 'p': 14, 'path': 206, 'polygon': 27, 'rect': 11,
#   'script': 23, 'span': 70, 'strong': 4, 'style': 1, 'svg': 34, 'symbol': 50,
#   'table': 1, 'td': 3, 'th': 3, 'title': 14, 'tr': 3, 'ul': 20, 'use': 10}

def count_tags(html):
```

```
   tag_list = [tag.name for tag in BeautifulSoup(html, 'lxml').find_all(True)]
   tag_dict = { tag: tag_list.count(tag) for tag in sorted(set(tag_list)) }
   return tag_dict

start_time = time.time()
eid_art_tags = [count_tags(html) for html in tqdm(eid_art_html)]
print(f"\nTime elapsed: {int((time.time() - start_time) // 60)} min {round((time.time() -
start_time) % 60, 1)} sec")
# 13:22.0
eid_art_tags_df = pd.DataFrame(eid_art_tags).fillna(0) # fill in 0s; converts to float64
eid_art_tags_df.to_excel('eid_art_tags_df.xlsx', engine='openpyxl')

pd.concat(
   [eid_art_tags_df.quantile(axis=0, q=pct/100) for pct in range(0, 110, 10)],
   axis=1
   ).to_excel('eid_art_tags_quantiles.xlsx', engine='openpyxl')
# [116 rows x 11 columns]

# rewrite to operation on soup rather than HTML
# which allows applying function to subsoup objects
def count_tags(soup):
   tag_list = [tag.name for tag in soup.find_all(True)]
   tag_dict = { tag: tag_list.count(tag) for tag in sorted(set(tag_list)) }
   return tag_dict

# count_tags(BeautifulSoup(eid_art_html[9000], 'lxml'))

start_time = time.time()
eid_art_tags = [count_tags(BeautifulSoup(html, 'lxml'))
                for html in tqdm(eid_art_html)]
print(f"\nTime elapsed: {int((time.time() - start_time) // 60)} min {round((time.time() -
start_time) % 60, 1)} sec")
# 10640/10640 [10:33<00:00, 16.79it/s]
eid_art_tags_df = pd.DataFrame(eid_art_tags).fillna(0) # fill in 0s; converts to float64
eid_art_tags_df.to_excel('eid_art_tags_df.xlsx', engine='openpyxl')

pd.concat(
   [eid_art_tags_df.quantile(axis=0, q=pct/100) for pct in range(0, 110, 10)],
   axis=1
   ).to_excel('eid_art_tags_quantiles.xlsx', engine='openpyxl')
# [116 rows x 11 columns]

#%% Focus on elements in <head>

only_head = SoupStrainer(name='head')
x = BeautifulSoup(eid_art_html[8690], 'lxml') # 10/4/03-0509_article

def eid_soup_head_count(soup):
   """Process selected metadata from HTML <head> element.
   Using SoupStrainer makes this even more efficent."""
   citation_author = len(soup.find_all('meta', attrs={'name': 'citation_author'}))
   citation_doi = len(soup.find_all('meta', attrs={'name': 'citation_doi'}))
   citation_title = len(soup.find_all('meta', attrs={'name': 'citation_title'}))
   description = len(soup.find_all('meta', attrs={'name': 'description'}))
   keywords = len(soup.find_all('meta', attrs={'name': 'keywords'}))
   og_description = len(soup.find_all('meta', attrs={'property': 'og:description'}))
```

```
111     og_title = len(soup.find_all('meta', attrs={'property': 'og:title'}))
112     og_url = len(soup.find_all('meta', attrs={'property': 'og:url'}))
113     canonical_link = len(soup.find_all('link', attrs={'rel': 'canonical'}))
114     return dict(citation_author=citation_author, citation_doi=citation_doi,
115        citation_title=citation_title, description=description, keywords=keywords,
116        og_description=og_description, og_title=og_title, og_url=og_url,
117        canonical_link=canonical_link)
118
119 eid_soup_head_count(x)
120
121 %timeit -r 11 -n 20 BeautifulSoup(eid_art_html[8690], 'lxml', parse_only=only_head)
122 # 29.4 ms ± 2.67 ms per loop (mean ± std. dev. of 11 runs, 20 loops each)
123 %timeit -r 11 -n 20 eid_soup_head_count(BeautifulSoup(eid_art_html[8690], 'lxml',
123 parse_only=only_head))
124 # 30.8 ms ± 950 µs per loop (mean ± std. dev. of 11 runs, 20 loops each)
125 %timeit -r 11 -n 20 BeautifulSoup(eid_art_html[8690], 'lxml')
126 # 47.5 ms ± 4.67 ms per loop (mean ± std. dev. of 11 runs, 20 loops each)
127 %timeit -r 11 -n 20 eid_soup_head_count(BeautifulSoup(eid_art_html[8690], 'lxml'))
128 # 108 ms ± 2.24 ms per loop (mean ± std. dev. of 11 runs, 20 loops each)
129
130 y = pd.DataFrame([eid_soup_head_count(BeautifulSoup(html, 'lxml', parse_only=only_head))
131                 for html in tqdm(eid_art_html)])
132 y.to_excel('eid_soup_head_count.xlsx', engine='openpyxl')
133
134 # citation_author can appear 0, 1, or more times
135 # the others appear 1 time; description and keywords can appear 0 times
136 def eid_soup_head(soup):
137     """Process selected metadata from HTML <head> element.
138     Using SoupStrainer makes this even more efficent."""
139     title = soup.title.string.strip()
140     citation_author = '|'.join([item.get('content')
141        for item in soup.find_all('meta', attrs={'name': 'citation_author'})])
142     citation_doi = soup.find('meta', attrs={'name': 'citation_doi'}).get('content')
143     citation_title = soup.find('meta', attrs={'name':
143     'citation_title'}).get('content').strip()
144     description = soup.find('meta', attrs={'name': 'description'})
145     description = '' if description is None else description.get('content').strip()
146     keywords = soup.find('meta', attrs={'name': 'keywords'})
147     keywords = '' if keywords is None else re.sub(', ', '|', keywords.get('content'))
148     og_description = soup.find('meta', attrs={'property':
148     'og:description'}).get('content').strip()
149     og_title = soup.find('meta', attrs={'property': 'og:title'}).get('content').strip()
150     og_url = soup.find('meta', attrs={'property': 'og:url'}).get('content')
151     canonical_link = soup.find('link', attrs={'rel': 'canonical'}).get('href')
152     # volume and issue number derived from URL; also in title
153     vol_iss = re.search(r'/(?P<vol>\d{1,2})/(?P<iss>\d{1,2})/', canonical_link)
154     volume, issue = int(vol_iss.group('vol')), int(vol_iss.group('iss'))
155     year = volume + 1994
156     return dict(
157        title_head=title, volume_head=volume, issue_head=issue, year_head=year,
158        citation_doi=citation_doi, canonical_link=canonical_link,
159        citation_title=citation_title, citation_author=citation_author,
160        description=description, keywords=keywords,
161        og_title=og_title, og_description=og_description, og_url=og_url)
162
163 eid_soup_head(x)
```

```
164
165 # robust version - when file lacks some of these metadata elements
166 def eid_soup_head_(soup):
167    """Process selected metadata from HTML <head> element.
168    Using SoupStrainer makes this even more efficent."""
169    title = soup.title.string.strip()
170    citation_author = '|'.join([item.get('content')
171       for item in soup.find_all('meta', attrs={'name': 'citation_author'})])
172    citation_doi = soup.find('meta', attrs={'name': 'citation_doi'})
173    citation_doi = '' if citation_doi is None else citation_doi.get('content').strip()
174    citation_title = soup.find('meta', attrs={'name': 'citation_title'})
175    citation_title = '' if citation_title is None else
175    citation_title.get('content').strip()
176    description = soup.find('meta', attrs={'name': 'description'})
177    description = '' if description is None else description.get('content').strip()
178    keywords = soup.find('meta', attrs={'name': 'keywords'})
179    keywords = '' if keywords is None else re.sub(', ', '|', keywords.get('content'))
180    og_description = soup.find('meta', attrs={'property': 'og:description'})
181    og_description = '' if og_description is None else
181    og_description.get('content').strip()
182    og_title = soup.find('meta', attrs={'property': 'og:title'})
183    og_title = '' if og_title is None else og_title.get('content').strip()
184    og_url = soup.find('meta', attrs={'property': 'og:url'})
185    og_url = '' if og_url is None else og_url.get('content').strip()
186    canonical_link = soup.find('link', attrs={'rel': 'canonical'})
187    canonical_link = '' if canonical_link is None else canonical_link.get('href').strip()
188    # volume and issue number derived from URL; also in title
189    if canonical_link == '' or canonical_link is None:
190       volume, issue, year = None, None, None
191    else:
192       vol_iss = re.search(r'/(?P<vol>\d{1,2})/(?P<iss>\d{1,2})/', canonical_link)
193       if vol_iss is None:
194          volume, issue, year = None, None, None
195       else:
196          volume, issue = int(vol_iss.group('vol')), int(vol_iss.group('iss'))
197          year = volume + 1994
198    return dict(
199       title_head=title, volume_head=volume, issue_head=issue, year_head=year,
200       citation_doi=citation_doi, canonical_link=canonical_link,
201       citation_title=citation_title, citation_author=citation_author,
202       description=description, keywords=keywords,
203       og_title=og_title, og_description=og_description, og_url=og_url)
204
205 eid_soup_head_(x)
206
207 eid_head_data = [eid_soup_head(BeautifulSoup(html, 'lxml', parse_only=only_head))
208                  for html in tqdm(eid_art_html)]
209 # 11211/11211 [04:47<00:00, 39.00it/s]
210 # pickle.dump(eid_head_data, open("eid_head_data.pkl", "wb"))
211 # eid_head_data = pickle.load(open("eid_head_data.pkl", "rb"))
212 pd.DataFrame(eid_head_data).to_excel('eid_head_data.xlsx', engine='openpyxl')
213
214 #%% Focus on elements in <main>
215
216 only_main = SoupStrainer(name='main') # contains main body of article
217
```

```
218 x = BeautifulSoup(eid_art_html[8690], 'lxml', parse_only=only_main)
219 len(x.main) # 36
220 len(list(x.main.children)) # 36
221
222 def eid_soup_main_count(soup):
223     """Process selected metadata from HTML <main> element.
224     Using SoupStrainer makes this even more efficent."""
225
226     # main metadata
227     vol_type = soup.find_all('h5') # volume and article type
228     vol_type = sum([x.parent.name == 'main' for x in vol_type])
229     title = len(soup.find_all('h3', attrs={'class': 'article-title'}))
230     doi = len(soup.find_all('p', attrs={'id': 'article-doi-footer'}))
231
232     # content
233     abstract = len(soup.find_all('div', attrs={'id': 'abstract'}))
234     mainbody = len(soup.find_all('div', attrs={'id': 'mainbody'}))
235     figures = len(soup.find_all('ul', attrs={'id': 'figures'}))
236     tables = len(soup.find_all('ul', attrs={'id': 'tables'}))
237
238     # detailed metadata
239     authors = len(soup.find_all('div', attrs={'id': 'authors'}))
240     subauthors = len(soup.find_all('div', attrs={'id': 'sub-authors'}))
241     author_affil = len(soup.find_all('div', attrs={'id': 'author-affiliations'}))
242     subauthor_affil = len(soup.find_all('div', attrs={'id': 'sub-author-affiliations'}))
243     acks0 = len(soup.find_all('a', attrs={'id': 'acknowledgements'}))
244     acks1 = soup.find_all('div', attrs={'class': 'blockquote-indent'})
245     acks1 = sum([x.parent.name == 'main' for x in acks1])
246     refs0 = len(soup.find_all('a', attrs={'id': 'references'}))
247     refs1 = len(soup.find_all('div', attrs={'id': 'articlereferences'}))
248     subrefs = len(soup.find_all('div', attrs={'id': 'sub-references'}))
249
250     return dict(
251         vol_type=vol_type, title=title, doi=doi,
252         abstract=abstract, mainbody=mainbody, figures=figures, tables=tables,
253         authors=authors, subauthors=subauthors,
254         author_affil=author_affil, subauthor_affil=subauthor_affil,
255         acks0=acks0, acks1=acks1, refs0=refs0, refs1=refs1, subrefs=subrefs)
256
257 eid_soup_main_count(x)
258
259 y = pd.DataFrame([eid_soup_main_count(BeautifulSoup(html, 'lxml', parse_only=only_main))
260                   for html in tqdm(eid_art_html)])
261 y.to_excel('eid_soup_main_count.xlsx', engine='openpyxl')
262 # 10640/10640 [15:43<00:00, 11.27it/s]
263
264 # conclusions: need to take care with vol_type and acks1 components
265 # remove all markup
266 # rember to combine ack0 and ack1, refs0 and refs1
267
268 # figure out various h5 main.children
269 eid_main_h5 = [h5 for html in tqdm(eid_art_html)
270     for h5 in BeautifulSoup(html, 'lxml', parse_only=only_main).find_all('h5')
271     if h5.parent.name == 'main']
272 with open('eid_main_h5.txt', 'w') as file_out:
273     file_out.write(str(eid_main_h5))
```

```
274 ## extract, recode
275 # <h5 class="header">Volume[^<>]</h5>    volume and month
276 # <h5 class="header"><em>[^<>]</em></h5> issue type
277 # <h5 class="header"><em></em></h5>      theme issue in Sep 2015
278 ## omit
279 # <h5 class="header online-only">Peer Reviewed Report Available Online Only</h5>
280 # <h5>Figure</h5> <h5>Figures</h5>
281 # <h5>Table</h5>  <h5>Tables</h5>
282
283 eid_main_div_xml_section = [div for html in tqdm(eid_art_html)
284     for div in BeautifulSoup(html, 'lxml', parse_only=only_main).find_all('div',
284     class_='xml-section')
285     if div.parent.name == 'main']
286 # <a id="acknowledgements"> is just the section heading
287 # <a id="references"> is just the section heading
288
289 # sub-authors, sub-author-affiliantions, and sib-references are where
290 # a letter or notice is followed by a reply on the same page
291 # there appear to be about 9 of them
292 # we will harvest the main letter but not the 'subbody'
293 # eid_art_dframe.iloc[[5590, 5591, 8497, 8534, 8540, 8541, 8624, 9865],:]
294
295 def eid_soup_main(soup):
296    """Process selected metadata from HTML <main> element.
297    Using SoupStrainer makes this even more efficent."""
298
299    # find_all() where possibly >1; find() where 0 or 1
300    # main metadata
301    # volume and article type, but only for direct children of <main> element
302    vol_type = [h5 for h5 in soup.find_all('h5') if h5.parent.name == 'main']
303    vol_re = re.compile(r'''
304       Volume\ (?P<vol>\d{1,2}),\                          # volume
305       (Number\ (?P<iss>\d{1,2})|(?P<supp>Supplement)).+? # issue (# or Supp)
306       (?P<mon>\w+)\ (?P<year>\d{4})                      # month, year
307       ''', re.VERBOSE)
308    vol_match = vol_re.search(vol_type[0].get_text('|', strip=True))
309    volume, year, month = vol_match.group('vol', 'year', 'mon')
310    issue = vol_match.group('iss') if vol_match.group('iss') is not None \
311       else vol_match.group('supp')
312    type_ = vol_type[1].get_text('|', strip=True)
313    # in rare case (21x) where empty, pull info from vol_type[0]
314    if type_ == '':
315       type_ = '|'.join(list(vol_type[0].stripped_strings)[1:])
316
317    title = soup.find('h3', class_='article-title')
318    title = '' if title is None else title.get_text('|', strip=True)
319    doi = soup.find('p', id='article-doi-footer')
320    doi = '' if doi is None else doi.get_text('|', strip=True)
321
322    # content
323    abstract = soup.find('div', id='abstract')
324    abstract = '' if abstract is None else abstract.get_text('|', strip=True)
325    mainbody = soup.find('div', id='mainbody')
326    mainbody = '' if mainbody is None else mainbody.get_text('|', strip=True)
327    figures = soup.find('ul', id='figures')
328    figures = '' if figures is None else figures.get_text('|', strip=True)
```

```
329    tables = soup.find('ul', id='tables')
330    tables = '' if tables is None else tables.get_text('|', strip=True)
331
332    # detailed metadata
333    authors = soup.find('div', id='authors')
334    authors = '' if authors is None else authors.get_text('|', strip=True)
335    author_affil = soup.find('div', id='author-affiliations')
336    author_affil = '' if author_affil is None else author_affil.get_text('|', strip=True)
337    acknowls = soup.find('div', class_='blockquote-indent')
338    acknowls = '' if acknowls is None else acknowls.get_text('|', strip=True)
339    references = soup.find('div', id='articlereferences')
340    references = '' if references is None else references.get_text('|', strip=True)
341
342    return dict(
343       title_main=title, volume_main=volume, issue_main=issue, year_main=year,
344       month_main=month, type_=type_, doi_main=doi,
345       abstract=abstract, mainbody=mainbody, figures=figures, tables=tables,
346       authors=authors, author_affil=author_affil,
347       acknowls=acknowls, references=references)
348
349 eid_soup_main(x)
350
351 # acknowledgements
352 x = BeautifulSoup(eid_art_html[5590], 'lxml', parse_only=only_main)
353
354 y = [elem for html in tqdm(eid_art_html[8490:9870])
355    for elem in BeautifulSoup(html, 'lxml', parse_only=only_main).find_all('a',
355    id='references')
356    if elem.parent.name == 'main']
357 [5590, 5591, 8497, 8534, 8540, 8541, 8624, 9865]
358
359 eid_main_data = [eid_soup_main(BeautifulSoup(html, 'lxml', parse_only=only_main))
360                  for html in tqdm(eid_art_html)]
361 # 11211/11211 [07:50<00:00, 23.85it/s]
362 # pickle.dump(eid_main_data, open("eid_main_data.pkl", "wb"))
363 # eid_main_data = pickle.load(open("eid_main_data.pkl", "rb"))
364 pd.DataFrame(eid_main_data).to_excel('eid_main_data.xlsx', engine='openpyxl')
365
366 #%% Combine <head> and <main> data
367
368 # dict.update(other_dict) modifies dict in place, so first copy <head>
369 eid_parsed_data = eid_head_data.copy()
370
371 # now append <main> data in place to copy of <head> data
372 # this requires that keys in dict and other_dict be distinct
373 # (else other_dict will overwrite corresponding keys' values)
374 # since dict.update(other_dict) returns None, assign list result to nul
375 nul = [parsed.update(main) for parsed, main in
376        tqdm(zip(eid_parsed_data, eid_main_data))]
377 # takes less than a second
378
379 nul.count(None) # 10640
380 [len(x) for x in eid_parsed_data].count(28) # 10640
381 del nul
382
383 # pickle.dump(eid_parsed_data, open('eid_parsed_data.pkl', 'wb'))
```

```
384
385 import json
386 # write in JSON format using UTF-8 rather than ASCII
387 with open('eid_parsed_data.json', 'w') as json_out:
388    json.dump(eid_parsed_data, json_out, ensure_ascii=False)
389
390 # eid_parsed_data_ = json.load(open('eid_parsed_data.json'))
391 # eid_parsed_data == eid_parsed_data_ # True
392
393 pd.DataFrame(eid_parsed_data).to_excel('eid_parsed_data.xlsx', engine='openpyxl')
394 # 42 mainbody fields and 2 references fields are truncated at 32767 characters
395 # must use pickle or json for full fidelity
396
397 #%% Output objects for learning resources
398 # eid_dframe -> CSV
399 # eid_art_html -> eid_art_html.json
400 # eid_art_dframe + eid_head_data + eid_main_data -> eid_parsed_data.{json,xlsx}
401
402 eid_dframe.to_csv('eid_dframe.csv')
403
404 with open('eid_art_html.json', 'w') as json_out:
405    json.dump(eid_art_html, json_out, ensure_ascii=False)
406 # eid_art_html_ = json.load(open('eid_art_html.json'))
407 # eid_art_html == eid_art_html_ # True
408
409 eid_parsed_dframe = pd.concat([eid_art_dframe, pd.DataFrame(eid_head_data),
410                                pd.DataFrame(eid_main_data)], axis=1)
411 eid_parsed_dframe.sort_values(by="citation_doi", inplace=True)
412 eid_parsed_dframe.to_excel('eid_parsed_dframe.xlsx', engine='openpyxl')
413 eid_parsed_dframe.to_json('eid_parsed_dframe.json', orient='records')
414
415 # x = pd.read_json('eid_parsed_dframe.json')
416 # x.to_excel('x-temp.xlsx', engine='openpyxl')
417 # differences appear to be: (1) index, (2) dtype of volume_main, year_main
418 # differences do not appear to be substantive
```