Building a Document Collection with a Personalized, Focused Web Crawler

Robin Singh

Northwestern University

Abstract:

The CDC Novel Coronavirus Reports corpus created a corpus of reports form the CDC discussing the most recent findings and information regarding COVID-19, something we know relatively little about due to its sudden rise last year. The corpus is created using Beautiful Soup to scrape relevant information from the reports such as the URL, title and text content to allow potential users to conduct analytical experiments of their own. It could provide the most up to date terminology regarding the coronavirus, which could be beneficial to communities if such information was analyzed as soon as possible and characterized to determine potential risks and elaborate on precautions for further safety of communities.

Introduction:

For this assignment, I will be working with the CDC Novel Coronavirus Reports to create a corpus of words that describe the impact of COVID-19 on the world today. These Novel Coronavirus Reports are one of the most up-to-date ways to get information regarding the status and updates surrounding the coronavirus. Multiple reports are released every week, discussing result of various scientific experiments and population impacts that the COVID-19 virus has had. By using such a corpus, we can analyze the frequency of new findings and use it to determine what the most impactful studies could be. Those studies need to be reported quickly to the public, so that they may better care for themselves and their loved ones. We can also alert specific communities of rises in cases and potential hazards in their area. This could be vital in keeping individuals and families safe, and enacting policies to prevent further spread of the virus.

Literature Review:

A corpus regarding COVID-19 was also created by Erdal Baran and Dimitar Dimitrov. This corpus was called TweetsCOV19. This corpus utilized Tweets from the time period between October 2019 through April 2020. The corpus aimed at capturing online discourse about various aspects of the pandemic and its societal impact. However, this corpus was quite extensive, containing the Username, Tweet ID, Timestamp, the text within the Tweet, and hashtags. All-in-all, the dataset contains 8,151,524 tweets posted by 3,664,518 users.

This corpus itself was a subset from an existing public corpus, TweetsKB, which is an anonymized data for a large collection of annotated tweets. This corpus contains more than 7 billion tweets beginning from February of 2013. The way it was used to create the TweetsCOV19 corpus was by compiling a seed of 268 COVID-19-related keywords and filtering tweets from TweetsKB to get the ones relevant to COVID-19.

Methods:

To create the CDC Novel Coronavirus Reports corpus, I utilized Beautiful Soup, which is Python package for parsing HTML and XML documents. Beautiful Soup parses through websites and scrapes the information you dictate as relevant for your use. I started a Beautiful Soup project within Google Colab, which I used as my coding environment.

Within Beautiful Soup, I created a method which took all of the URLs off of the main CDC Novel Coronavirus Reports page and created a list of the 258 links on that page. I did this by finding all of the href elements on the page, and eliminating the first four, which were not relevant to my corpus as they were not links that went to the pages of the reports. Now the algorithm would know which websites to crawl through to extract the information.

I then created a for loop, which went through each of the links it had found and extracted the information I found relevant for my corpus. To analyze which sections were most relevant to me, I looked for which elements of the website contained both the title and its main text content. I used Google Chrome's "Inspect" tool to find the .CSS elements that made them up. The title was in an 'h1' element and text content being a 'div class "order-4 w-100" element. I added a counter to create an ID tag for each report, as well as extracted the URL to include with each report's extracted text. These were then appended to a blank list.

```
counter=0

finalList=[]
#testlink='https://www.cdc.gov/mmwr/volumes/70/wr/mm7016e1.htm?s_cid=mm7016e1_w'
for link in list2:
  r = requests.get(link,headers=headers)
  soup = BeautifulSoup(r.content,'lxml')
  counter=counter
  link=r.url
  name=soup.find('h1').text

  text=([i.text for i in soup.find_all('div', class_='order-4 w-100')])
  text=''.join(str(v) for v in text)
  docs={
      'count':counter,
      'url':link,
      'name':name,
      'text':text
  }
  counter=counter+1
  finalList.append(docs)
```

This concluded the algorithm to parse through the listed URL's and extract the relevant information. I then converted the list to a data frame, and the data frame to a dictionary, which was then all output into a json file called reports.jl.

Results:

This assignment allowed me to create a corpus regarding up-to-date relevant information surrounding COVID-19. It would include terminology of recent findings and its community

impact to extract meaning from which could be helpful in one's analysis of those topics. The documents are official, and from a reputable source, and only grabs text from the specific Novel Coronavirus Reports section of the CDC website. This could be improved by including a classifier and eliminating very common words with perhaps little meaning. Another aspect worth considering would be following links from the individual report's pages to extract even more terminology relevant to the topic.

Conclusion:

With this generated .json file of the CDC Novel Reports Corpus, the users may implement neural networks to assign variables and vectorize the strings in the corpus. That could then be used to generate importance of recent findings and studies, which may be beneficial in analyzing and getting the word out, so that communities may be alerted, and families could be kept safe. It could also be used to provide people with hard facts about the COVID-19 virus, of which we know so little about, so that we can develop further improvements to public health policies and conduct further medical research. This corpus is limited however, as it has a fixed list of websites to extract data from. Still, it can be run every time an update to the page is made and include the most recent reports added to the main page.

Appendix:

Beginning of code, including code to extract links from main webpage:

```
pip install jsonlines
```

```
Collecting jsonlines
  Downloading https://files.pythonhosted.org/packages/d4/58/06f430ff7607a2929f80f07bfd820acbc508a4e977542fefcc522cde9dff/jsonlines-2.0.0-py3-none-any.whl
Installing collected packages: jsonlines
Successfully installed jsonlines-2.0.0
```

```
[2] import requests
    from bs4 import BeautifulSoup
    import pandas as pd
    import jsonlines
    import json
```

```
[3] baseurl='https://www.cdc.gov/'
```

```
[4] headers = {
        'User-Agent': 'https://developers.whatismybrowser.com/useragents/parse/720201safari-mac-os-x-webkit'
    }
```

```
[5] r = requests.get('https://www.cdc.gov/mmwr/Novel_Coronavirus_Reports.html')
```

```
[6] soup = BeautifulSoup(r.content,'lxml')
    report=soup.find_all('div',class_='col-md-12 splash-col')
```

```
[7] links=[]
    for item in report:
      for link in item.find_all('a', href=True):
        links.append(baseurl + link['href'])
```

List of links with removal of links that were not relevant.

```
print(len(links))
```

```
262
```

```
links[:5]
```

```
['https://www.cdc.gov//mmwr/mmwrpodcasts.html',
 'https://www.cdc.gov/https://tools.cdc.gov/campaignproxyservice/subscriptions.aspx?topic_id=USCDC_921',
 'https://www.cdc.gov///www.youtube-nocookie.com/embed/9pVy8sRC440?autohide=0&enablejsapi=1&playerapiid=56233&modestbranding=1&rel=0&origin=https://www.cdc.(
 'https://www.cdc.gov/https://www.youtube.com/watch?v=9pVy8sRC440',
 'https://www.cdc.gov//mmwr/volumes/70/wr/mm7015a3.htm?s_cid=mm7015a3_w']
```

```
[9] list2=links[4:]
```

```
[10] list2
```

```
['https://www.cdc.gov//mmwr/volumes/70/wr/mm7015a3.htm?s_cid=mm7015a3_w',
 'https://www.cdc.gov//mmwr/volumes/70/wr/mm7015e1.htm?s_cid=mm7015e1_w',
 'https://www.cdc.gov//mmwr/volumes/70/wr/mm7015e2.htm?s_cid=mm7015e2_w',
 'https://www.cdc.gov//mmwr/volumes/70/wr/mm7015e3.htm?s_cid=mm7015e3_w',
 'https://www.cdc.gov//mmwr/volumes/70/wr/mm7015a4.htm?s_cid=mm7015a4_w',
 'https://www.cdc.gov//mmwr/volumes/70/wr/mm7015a7.htm?s_cid=mm7015a7_w',
 'https://www.cdc.gov//mmwr/volumes/70/wr/mm7016e1.htm?s_cid=mm7016e1_w',
 'https://www.cdc.gov//mmwr/volumes/70/wr/mm7014a2.htm?s_cid=mm7014a2_w',
 'https://www.cdc.gov//mmwr/volumes/70/wr/mm7014a3.htm?s_cid=mm7014a3_w',
 'https://www.cdc.gov//mmwr/volumes/70/wr/mm7014e1.htm?s_cid=mm7014e1_w',
 'https://www.cdc.gov//mmwr/volumes/70/wr/mm7014e2.htm?s_cid=mm7014e2_w',
 'https://www.cdc.gov//mmwr/volumes/70/wr/mm7014e3.htm?s_cid=mm7014e3_w',
 'https://www.cdc.gov//mmwr/volumes/70/wr/mm7014a4.htm?s_cid=mm7014a4_w',
 'https://www.cdc.gov//mmwr/volumes/70/wr/mm7014a5.htm?s_cid=mm7014a5_w',
 'https://www.cdc.gov//mmwr/volumes/70/wr/mm7013a2.htm?s_cid=mm7013a2_w',
 'https://www.cdc.gov//mmwr/volumes/70/wr/mm7013a3.htm?s_cid=mm7013a3_w',
 'https://www.cdc.gov//mmwr/volumes/70/wr/mm7013a4.htm?s_cid=mm7013a4_w',
 'https://www.cdc.gov//mmwr/volumes/70/wr/mm7013e1.htm?s_cid=mm7013e1_w',
 'https://www.cdc.gov//mmwr/volumes/70/wr/mm7013e2.htm?s_cid=mm7013e2_w',
 'https://www.cdc.gov//mmwr/volumes/70/wr/mm7013e3.htm?s_cid=mm7013e3_w',
 'https://www.cdc.gov//mmwr/volumes/70/wr/mm7012a3.htm?s_cid=mm7012a3_w',
```

0s completed at 1:43 AM

Code to extract the URL, title, and text from the page, as well as add a counter element for the ID.

```
counter=0

finalList=[]
#testlink='https://www.cdc.gov/mmwr/volumes/70/wr/mm7016e1.htm?s_cid=mm7016e1_w'
for link in list2:
  r = requests.get(link,headers=headers)
  soup = BeautifulSoup(r.content,'lxml')
  counter=counter
  link=r.url
  name=soup.find('h1').text

  text=([i.text for i in soup.find_all('div', class_='order-4 w-100')])
  text=''.join(str(v) for v in text)
  docs={
      'count':counter,
      'url':link,
      'name':name,
      'text':text
  }
  counter=counter+1
  finalList.append(docs)
```

Creating .jl file:

```
dictionary = df.to_dict('reports')
with jsonlines.open('reports.jl','w') as writer:
    writer.write_all(dictionary)
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/frame.py:1490: FutureWarning: Using short name for 'orient' is deprecated. Only the options: ('dict', lis
  FutureWarning,
```

```
[13]
```