```
import pandas as pd
import requests
import regex as re
```

```
--NORMAL--
```

```
import os
os.environ['NotebookApp.iopub_data_rate_limit']
```

```
!pip install transformers
```

```
Looking in indexes: https://pypi.org/simple, 
Collecting transformers
  Downloading transformers-4.28.1-py3-none-an
Requirement already satisfied: requests in /u
Collecting huggingface-hub<1.0,>=0.11.0
  Downloading huggingface_hub-0.14.0-py3-none
Requirement already satisfied: filelock in /u
Requirement already satisfied: numpy>=1.17 in 
Requirement already satisfied: pyyaml>=5.1 in 
Requirement already satisfied: packaging>=20.
Collecting tokenizers!=0.11.3,<0.14,>=0.11.1
  Downloading tokenizers-0.13.3-cp39-cp39-man
Requirement already satisfied: tqdm>=4.27 in 
Requirement already satisfied: regex!=2019.12
Requirement already satisfied: fsspec in /usr
Requirement already satisfied: typing-extensi
Requirement already satisfied: idna<4,>=2.5 i
Requirement already satisfied: certifi>=2017.
Requirement already satisfied: urllib3<1.27,>
Requirement already satisfied: charset-normal
Installing collected packages: tokenizers, hu
Successfully installed huggingface-hub-0.14.0
```

xt 1931.txt 1930.txt •••

```
27.64137828350067101
```

```
from transformers import pipeline
sentiment_analysis = pipeline("sentiment-analysi
print(sentiment_analysis("I love this!"))
```

```
Downloading (…)lve/main/config.json: 100%|687/687 [00:00<00:00, 25.9kB/s]
Downloading pytorch_model.bin: 100%|1.42G/1.42G [00:14<00:00, 150MB/s]
Downloading 256/256
```

Want to filter the CVS and only keep the entry/rows that have the year in First Issue Date < 1940 and the year in Last Issue Date > 1920

```
!pip install symspellpy
```

```
Looking in indexes: https://pypi.org/simple, 
Collecting symspellpy
  Downloading symspellpy-6.7.7-py3-none-any.wl
```

```
Collecting editdistpy>=0.1.3
  Downloading editdistpy-0.1.3.tar.gz (57 kB)
```

```
  Installing build dependencies ... done
  Getting requirements to build wheel ... don
  Preparing metadata (pyproject.toml) ... don
Building wheels for collected packages: editd
  Building wheel for editdistpy (pyproject.to
  Created wheel for editdistpy: filename=edit
  Stored in directory: /root/.cache/pip/wheel
Successfully built editdistpy
Installing collected packages: editdistpy, sy
Successfully installed editdistpy-0.1.3 symsp
```

```
from symspellpy import SymSpell
import pkg_resources
```

```
symspell = SymSpell()
dict_path = pkg_resources.resource_filename("sym
symspell.load_dictionary(dict_path, 0, 1) # Load
eng_dictionary = symspell.words # Get keyed list
```

```
from bs4 import BeautifulSoup
```

```
# read in the CSV file
df = pd.read_csv('/content/4_20_CA_newspapers -

print(df.columns)
df.columns = [c.strip() for c in df.columns]

# convert the 'First Issue Date' and 'Last Issue
df['First Issue Date'] = pd.to_datetime(df['Firs
df['Last Issue Date'] = pd.to_datetime(df['Last

# extract the year from the 'First Issue Date' a
df['First Issue Year'] = df['First Issue Date'].
df['Last Issue Year'] = df['Last Issue Date'].dt

# filter the dataframe to only keep entries with
df_filtered = df[(df['First Issue Year'] < 1940)

# print out the filtered dataframe
print(df_filtered)
```

```
Index(['Persistent Link', 'State', 'Title',
       'No. of Issues', 'First Issue Date',
      dtype='object')
                                        Pers
2     https://chroniclingamerica.loc.gov/lc
22    https://chroniclingamerica.loc.gov/lc
26    https://chroniclingamerica.loc.gov/lc
27    https://chroniclingamerica.loc.gov/lc
34    https://chroniclingamerica.loc.gov/lc
...
3992  https://chroniclingamerica.loc.gov/lc
3999  https://chroniclingamerica.loc.gov/lc
4000  https://chroniclingamerica.loc.gov/lc
4001  https://chroniclingamerica.loc.gov/lc
```

```
4004  https://chroniclingamerica.loc.gov/lc


2      The Birmingham age-herald. [volume] (
22     The &quot;J&quot; bird. [volume] (Jur
26     The Alaska daily empire. [volume] (Ju
27     The Alaska fisherman. [volume] (Junea
34         The Alaskan. (Petersburg, Alaska)
...
3992       The Cody enterprise. (Cody, Wyo.)
3999   The Northern Wyoming herald. (Cody, W
4000        Park County herald. (Cody, Wyo.)
4001     Rawlins Republican. (Rawlins, Wyo.)
4004   The Saratoga sun. (Saratoga, Carbon (

             OCLC       ISSN  No. of Issues
2      12607279.0  2692-6318         8237.0
22    767526793.0        NaN           88.0
26      3039521.0  2576-9227         4340.0
27     29593944.0        NaN           97.0
34     31214772.0        NaN          320.0
...           ...        ...            ...
3992   25224927.0        NaN          184.0
3999   25224904.0        NaN          406.0
4000   25224920.0        NaN          117.0
4001   27184809.0        NaN         1617.0
4004    9965363.0  0740-4948         1748.0


2     https://chroniclingamerica.loc.gov/lc
22
26    https://chroniclingamerica.loc.gov/lc
27
34
...
3992  https://chroniclingamerica.loc.gov/lc
3999  https://chroniclingamerica.loc.gov/lc
4000  https://chroniclingamerica.loc.gov/lc
4001  https://chroniclingamerica.loc.gov/lc
4004  https://chroniclingamerica.loc.gov/lc

      Last Issue Year
2              1924.0
22             1936.0
26             1926.0
```

Loop through all the sn# & years

```
def get_url(lccn, year):
    # Example url:
    url = f'https://chroniclingamerica.loc.gov/l
    # url = 'https://chroniclingamerica.loc.gov/
    lccn_label = lccn.strip()

    session = requests.Session()
    response = session.get(url).text
    # print(response)

    # Parse out issue dates from the response us
    # pattern = rf'<a href="/lccn/{lccn}/{year}.
    # pattern = rf'<a href="/lccn/{lccn}/{year}-

    pattern = r'<a href="/lccn/'+ lccn.strip() +
    matches = re.findall(pattern, response)

    # Return a dictionary indexed by lccn and ye
    return {'lccn': lccn, 'year': year, 'matches
```

```
import os

def save_matches_to_file(lccn, year, matches):
  # Helper function to save matches to file
  directory = f'/content/urls/{lccn.strip()}/'
  print(f'{directory}{year}.txt')
  if not os.path.exists(directory):
    os.makedirs(directory)
  with open(f'{directory}/{year}.txt', 'w') as o
    outfile.writelines(matches)

def scrape_urls(df_filtered):
  # Main function to scrape URLs and save matche
  # for index, row in df_filtered.iterrows():
  #   print(row)
  for index, row in df_filtered.iterrows():
    lccn = row['LCCN']
    # first_year = row['First Issue Year']
    # last_year = row['Last Issue Year']
    first_year = int(row['First Issue Year'])
    last_year = int(row['Last Issue Year'])
    for year in range(first_year, last_year+1):
      # result = get_url(lccn, year)
      # matches = result['matches'
      if int(year) >= 1924 and int(year) <= 1940
        print("LCCN",lccn,"year:",year)
        texts_score, chunk_number = get_text(lcc
        if (chunk_number > 0):
            score = texts_score/chunk_number
        print("LCCN",lccn,"year:",year, 'score:'
        # save_matches_to_file(lccn, year, match
        save_matches_to_file(lccn, year, score)
      else:
        continue
```

```
import re

# html_text = '<tr><td class="single sun"><a hre
html_text =  ' <a href="/lccn/sn86072192/1897-08
# Use regular expression to find all matches
# pattern = r'<a href="/lccn/sn86072192/1897-.*?
lccn = ' sn86072192 '
year = 1897
pattern = r'<a href="/lccn/'+ lccn.strip() + '/'
matches = re.findall(pattern, html_text)

# Print all matches
for match in matches:
    print(match)
```

```
<a href="/lccn/sn86072192/1897-08-01/ed-1/seq-
```

```
def get_word_rate(text, dictionary): # Function
  num_words = 0
  for word in text.split():
    if word.lower() in dictionary.keys():
      num_words += 1

  return num_words / len(text.split())
```

```
def chunk_text(text, chunk_size):
  chunks = []
  words = text.split()
  i = 0
  while i + chunk_size < len(words):
    chunks.append(' '.join(words[i: i + chunk_si
    i += chunk_size
  chunks.append(' '.join(words[i:]))
  return chunks
```

```
chunk_size = 100
WORD_RATE_THRESHOLD = .6 # Set this to whateve
```

```
def get_text(lccn, year):
    # Fetch the URLs of all issues in the given
    url_data = get_url(lccn, year)
```

```
    issue_urls = url_data['matches']
    head = 'https://chroniclingamerica.loc.gov/
    tail = 'ocr/'
    text = ''
    score = 0
    chunk_number = 0
    old_url = head
    # Loop through all issue URLs and extract th
    for issue_url in issue_urls:
        # Extract the URL from the issues_url va
        url = issue_url.strip('<a href="').strip

        # Combine the URL with the head variable
        full_url = head + url
        response = requests.get(full_url)
        html_content = response.text
        print(full_url)
        # Use regular expression to find all mat
        pattern = r'<a href="/lccn/'+ lccn.strip
        text_matches = re.findall(pattern, html_

        for text_url in text_matches:
            t_url = text_url.strip('<a href="').
            ocr_url = head+ t_url + tail
            print(ocr_url)
            if (ocr_url == old_url):
                continue
            else:
                response = requests.get(ocr_url)
                ocr_content = response.text

                # Parse the HTML content using B
                soup = BeautifulSoup(ocr_content

                # Extract the text content of th
                for elem in soup.find_all('div')
                    if elem.find('p') is not Non
                        text = elem.find('p').ge
                        chunks = chunk_text(text
                        for chunk in chunks:
                            rate = get_word_rate
                            if (rate > WORD_RATE
                                chunk_number +=
                                if chunk_number
                                    break
```

```
                                # score += senti
                                data =sentiment_
                                print(data)

                                if data[0]['labe
                                   score += -1 *
                                elif data[0]['la
                                   score += data
                        if chunk_number > 100:
                            break
                        print('current score', s
                        old_url = ocr_url
                        text = ''

                    if chunk_number > 100:
                        break

            if chunk_number > 100:
                break

    return str(score), str(chunk_number)

```

```
def get_text(lccn, year):
    # Fetch the URLs of all issues in the given
    url_data = get_url(lccn, year)
    issue_urls = url_data['matches']
    head = 'https://chroniclingamerica.loc.gov/'
    tail = 'ocr/'
    text = ''
    score = 0
    chunk_number = 0
    old_url = head
    # Loop through all issue URLs and extract th
    for issue_url in issue_urls:
        # Extract the URL from the issues_url va
        url = issue_url.strip('<a href="').strip

        # Combine the URL with the head variable
        full_url = head + url
        response = requests.get(full_url)
        html_content = response.text
        print(full_url)
        # Use regular expression to find all mat
```

```
        # pattern = r'<a href="/lccn/'+ lccn.str
        pattern = r'<a href="/lccn/'+ lccn.strip
        # r'<a href="/lccn/sn86072192/1897-08-22
        text_matches = re.findall(pattern, html_

        for text_url in text_matches:
            t_url = text_url.strip('<a href="').
            ocr_url = head+ t_url + tail
            print(ocr_url)
            if (ocr_url == old_url):
                continue
            else:
        # Fetch the HTML content of the page wit
        # ocr_url = issue_url.replace('/ed-', '/
                response = requests.get(ocr_url)
                ocr_content = response.text

                # Parse the HTML content using B
                soup = BeautifulSoup(ocr_content

                # Extract the text content of th

                for elem in soup.find_all('div')
                    if elem.find('p') is not Non
                        # text += elem.find('p')
                        text = elem.find('p').ge
                        # print(len(text))
                        chunks = chunk_text(text
                        for chunk in chunks:
                          rate = get_word_rate(c
                          #  print(rate)
                          if (rate > WORD_RATE_T
                              # print('I am here
                              # data =sentiment_
                              # # print(data)
                              chunk_number += 1
                              # if data[0]['labe
                              #     score += -1
                              # elif data[0]['la
                              #     score += dat
                        print('current score', s
                        # print('ocr url:', ocr_
                        old_url = ocr_url
                        text = ''
            if chunk_number > 100:
```

```
                break



                    # print(elem.find('p').get_
            # Print the extracted text

    return str(score), str(chunk_number)

```

```
scrape_urls(df_filtered)
```

```
Streaming output truncated to the last 5000
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
```

```
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
https://chroniclingamerica.loc.gov//lccn/sn8
```

```
xxx = "I am here OESEEi THE AGE—HERALD.. J1 , _
print(len(xxx))
```

```
xx = [23.2369414567947410, 39.52144598960876510
years = [year for year in range(1920, 1937)]


```

```
58.36730605363846
```

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.plot(years, xx)
ax.set_xlabel('Year')
ax.set_ylabel('Sentiment')
ax.set_title('Sentiment over Time')
plt.show()
```

Colab paid products - Cancel contracts here

--VISUAL--

1h 33m 49s completed at 11:06 PM