jupyter api_dataset Last Checkpoint: 10 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

[1]:
```
from googleapiclient.discovery import build #to interact with YouTube Data API
import pandas as pd
```

[2]:
```
# YouTube API credentials
# generated on  Google Cloud Console for youtube service
api_key = 'AIzaSyBEfx6qOYZIovmdhQZk7mcBYIt8zhMpmtQ'

# Build the YouTube Data API client
# 'youtube' is the name of the API service
# 'v3' is the version of the API
# 'developerKey' is the API key used for authenticating requests
youtube = build('youtube', 'v3', developerKey=api_key)

# Print the youtube object to see if it was created successfully
print(youtube)
```

From Goole Cloud Console - YouTube Service

```
<googleapiclient.discovery.Resource object at 0x00000192123C0980>
```

[3]:
```
# specific channels for several categories
channels = {
    'news': ['BBCNews' ,'NBCNews'],
    'education': ['FITiug'],
    'cooking': ['gordonramsay'],
    'sport': ['SkatingISU'],
    'children': ['Osratouna']
}
channels
```

[3]:
```
{'news': ['BBCNews', 'NBCNews'],
 'education': ['FITiug'],
 'cooking': ['gordonramsay'],
 'sport': ['SkatingISU'],
 'children': ['Osratouna']}
```

[4]:
```
# get the channel ID using the username
def get_channel_id(username):
    # Create a request to YouTube Data API to list channels
    request = youtube.channels().list(
        part='id',  # only retern id  of the channel resource
        forUsername=username  #username -  which fetch there ID
    )
    # request excuction
    response = request.execute()
    # if the response contains any items (channels)
    if response['items']:
        # channel ID of the first item in the response
        return response['items'][0]['id']
    # if no items return None
    return None
```

[5]:
```
#  get the details of a video using video ID
def get_video_details(video_id):
    #  request to the YouTube Data API to list videos
    request = youtube.videos().list(
        part='snippet,statistics',  # retrieve: snippet and statistics
        id=video_id  # which vedion get details
    )
    # request excution
    response = request.execute()
    # if the response contains any items (videos)
    if response['items']:
        #  first video in the response
        video = response['items'][0]
        # dictionary of video details
        published_at = video['snippet']['publishedAt']
        publish_date, publish_time = published_at.split('T')
        publish_time = publish_time.split('Z')[0]
        return {
            'title': video['snippet']['title'],  # title of the video
            'description': video['snippet']['description'],  # description of the video
            'publish_date': publish_date,
            'publish_time': publish_time,
            'likes': video['statistics'].get('likeCount', 0),  # The number of likes the video has received
            'views': video['statistics'].get('viewCount', 0)  # The number of views the video has received
        }
    # if no items return None
    return None
```

[6]:
```
# collect all videos from a channel
def collect_videos(channel_id, max_results=200):
    video_list = []  # list to store video details
    next_page_token = None  # token for the next page of results

    # Loop until max_results or there are no more videos
    while len(video_list) < max_results:
        # request to the YouTube Data API to list videos for a channel
        request = youtube.search().list(
            part='id',  # id part of the video resource
            channelId=channel_id,  # want to collect videos form this channel
            maxResults=min(max_results - len(video_list), 50),  # limit the number of results per request
            pageToken=next_page_token,  # Token for the next page of results (if any)
            type='video'  # want video results
        )
        # request excution
        response = request.execute()

        #for all items in the response
        for item in response['items']:
```

```
            # details of each video using the video ID
            video_details = get_video_details(item['id']['videoId'])
            # If video details ... append them to the video list
            if video_details:
                video_list.append(video_details)

        # token for the next page of results (if any)
        next_page_token = response.get('nextPageToken')
        # if no next page token, break
        if not next_page_token:
            break

    # list of collected video details
    return video_list
```

[7]:
```
all_data = []  # list to store all the collected video data

# Loop through each category
for category, usernames in channels.items():
    for username in usernames: # category can contain many channels/user name
        print(f"Collecting data for : {username} channel in : {category} category")

        # get channel ID using the username
        channel_id = get_channel_id(username)

        # if the channel retrieved
        if channel_id:
            # extract videos from the channel using the channel ID
            videos = collect_videos(channel_id)

            # Loop through all collected videos
            for video in videos:
                # Add additional information to each video (category and channel ID)
                video['category'] = category
                video['channel_id'] = channel_id

                # append the video details to the all_data list
                all_data.append(video)
        else:
            # if the channel ID was not found
            print(f"Channel ID not found for username: {username}")
```

```
Collecting data for : BBCNews channel in : news category
Collecting data for : NBCNews channel in : news category
Collecting data for : FITiug channel in : education category
Collecting data for : gordonramsay channel in : cooking category
Collecting data for : SkatingISU channel in : sport category
Collecting data for : Osratouna channel in : children category
```

[8]:
```
# Save data to a CSV file
df = pd.DataFrame(all_data)
df.index += 1  # index from 1
df.to_csv('youtube_videos.csv', index_label='index')
print("Data saved to youtube_videos.csv")
```

```
Data saved to youtube_videos.csv
```

[9]:
```
data = pd.read_csv('youtube_videos.csv')
data.info() #info about data
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1200 entries, 0 to 1199
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   index         1200 non-null   int64
 1   title         1200 non-null   object
 2   description   1168 non-null   object
 3   publish_date  1200 non-null   object
 4   publish_time  1200 non-null   object
 5   likes         1200 non-null   int64
 6   views         1200 non-null   int64
 7   category      1200 non-null   object
 8   channel_id    1200 non-null   object
dtypes: int64(3), object(6)
memory usage: 84.5+ KB
```