1. Telegram Channel Scrapper

Автор задачи: Lantern

Lantern's Discover feature scraps content from a lot of sources so it'll always be available for usage uncensored and to have a backup in case of a takedown.

Make a Telegram channel scraper that either saves the content locally or returns a JSON blob that another service can consume to download the content.

The content in this case are the videos and images for that channel.

Intent: write a linear code for storing the messafes and media of channel in a tar archive and upload this to the AWS S3 bucket, reflect the new archive in a index.html of s3 bucket

```
#!python -m pip install --upgrade telethon
```

```
#!pip install xtarfile boto3
```

```
import pandas as pd , os, json
```

```
from telethon import TelegramClient
```

Get Telegram creds from json

```
with open('creds.json') as json_file:
    creds = json.load(json_file)
```

```
api_id = creds["api_id"]
api_hash = creds['api_hash']
phone = creds['phone']
username = creds["username"]
```

```
# One of telegram chanels for  scraping
```

```
chat = "vatnoeboloto"
chat = "YouTubot"
chat = "Kushnar_media"
chat = "insider_uke"
```

Preparing working environment

```
chat_addr = "https://t.me/{}".format(chat)
print(chat_addr)
download_folder = "download_of_{}".format(chat)
```

```
isExist = os.path.exists(download_folder)
if not isExist:
   os.makedirs(download_folder)
```

https://t.me/insider_uke

scraping messages and media. media files are writing in local folder

```
data = []
n = 0
n_max = 30
async with TelegramClient(username, api_id, api_hash) as client:
    async for message in client.iter_messages(chat_addr,
reverse=False):
        n = n + 1
        if (n > n_max):
            break
        data.append([message, message.sender_id, message.text,
message.date, message.id, message.post_author, message.views,
message.peer_id.channel_id , message.photo ])
        if (message.photo or message.video) :
            await message.download_media("{}\msg-{}-
{}".format(download_folder,message.peer_id.channel_id,message.id))
```

creating dataframe for observing and postproduction

```
columns=["message","message.sender_id", "message.text","
message.date", "message.id",  "message.post_author", "message.views",
"message.peer_id.channel_id", "message.photo" ]
df = pd.DataFrame(data, columns=columns) # creates a new dataframe
#df['message.text.removed_emojis'] = df.apply(lambda row :
remove_emojis(row["message.text"]), axis = 1)
df = df.reset_index()
```

saving messages In json file

```
filename = "messages_of_channel-{}".format(chat)
df.to_csv('{}.csv'.format(filename), encoding='utf-8')
df.drop(columns=['message',"message.photo"]).to_json('{}.json'.format(
filename))
```

```
from os import walk
media_list = []
for (dirpath, dirnames, filenames) in walk(download_folder):
    media_list.extend(filenames)
    break
#f
```

creating tarfile with messages (csv, json) and subfolder with media

```
import xtarfile as tarfile
with tarfile.open('{}.tar'.format(filename), 'w') as archive:
    archive.add('{}.csv'.format(filename))
    archive.add('{}.json'.format(filename))
    for i in media_list:
        archive.add('{}/{}'.format(download_folder,i))
```

uploading tar to AWS s3 bucket with public access

```
import boto3
s3 = boto3.resource("s3")

s3.meta.client.upload_file(
    Filename='{}.tar'.format(filename),
    Bucket="internet-without-borders",
    Key='{}.tar'.format(filename),
)
```

updating index.html in s3 bucket with public access

```
my_bucket = s3.Bucket('internet-without-borders')
s3_tars_list = []
for my_bucket_object in my_bucket.objects.all():
    if ".tar" in my_bucket_object.key:
        s3_tars_list.append("<br><a
href={}>{}</a>".format(my_bucket_object.key, my_bucket_object.key))
s3_tars_list
```

```
['<br><a href=messages_of_channel-
Kushnar_media.tar>messages_of_channel-Kushnar_media.tar</a>',
 '<br><a href=messages_of_channel-YouTubot.tar>messages_of_channel-
YouTubot.tar</a>',
 '<br><a href=messages_of_channel-insider_uke.tar>messages_of_channel-
insider_uke.tar</a>',
 '<br><a href=messages_of_channel-
vatnoeboloto.tar>messages_of_channel-vatnoeboloto.tar</a>']
```

```
index_text =
"<html><body>{}</body></html>".format("".join(s3_tars_list))
index_text
```

```
'<html><body><br><a href=messages_of_channel-
Kushnar_media.tar>messages_of_channel-Kushnar_media.tar</a><br><a
href=messages_of_channel-YouTubot.tar>messages_of_channel-
YouTubot.tar</a><br><a href=messages_of_channel-
insider_uke.tar>messages_of_channel-insider_uke.tar</a><br><a
href=messages_of_channel-vatnoeboloto.tar>messages_of_channel-
vatnoeboloto.tar</a></body></html>'
```

```
with open('index.html', 'w') as f:
    f.write(index_text)

s3.meta.client.upload_file(
    Filename="index.html",
    Bucket="internet-without-borders",
    Key="index.html",
)
```

The public URL of s3-based web-site with archives:

https://internet-without-borders.s3.eu-central-1.amazonaws.com/index.html

TODO:

UTF-8 in messages.text

media download with multithreading

docker image

deploy in AWS EKS or AWS Lambda

public API