DSCI-6007-01

Team-11

GITHUB Repository

Code:

HeadlineScraper.py

```
import boto3
import os
from datetime import datetime
from bs4 import BeautifulSoup as bs
import requests

def lambda_handler(event, context):
    url = "https://news.google.com/"

    response = requests.get(url) #GET request to URL
    results = []

    #Check if request is successful
    if response.status_code == 200:
        soup = bs(response.content, "html.parser") #Parse HTML content
```

```
        headlines = soup.find_all('a', class_ = 'gPFEn')

        for headline in headlines:
            results.append(headline.text.strip())

        print("Fetched headlines:")
        print(results)

    else:
        return {
            "statusCode": 500,
            "body": f"Failed to fetch the page. Status code: {response.status_code}"
        }

    # Define the S3 bucket and file name
    s3_bucket = "team11headlines"
    file_name =
f"results/result_{datetime.now().strftime('%Y%m%d_%H%M%S')}.txt"

    # Save the result to a file in /tmp (Lambda's temporary storage)
    temp_file_path = f"/tmp/{file_name.split('/')[-1]}"
    with open(temp_file_path, "w") as file:
        file.write("\n".join(results))

    # Upload the file to S3
```

```
    s3_client = boto3.client("s3")
    print(f"Uploading file to bucket: {s3_bucket}, path: {file_name}")
    try:
        s3_client.upload_file(temp_file_path, s3_bucket, file_name)
        print("Upload successful")
    except Exception as e:
        print(f"Upload failed: {e}")

    # Return the S3 file path
    return {
        "statusCode": 200,
        "body": f"File uploaded to S3: {s3_bucket}/{file_name}"
    }
```

Keyword.py

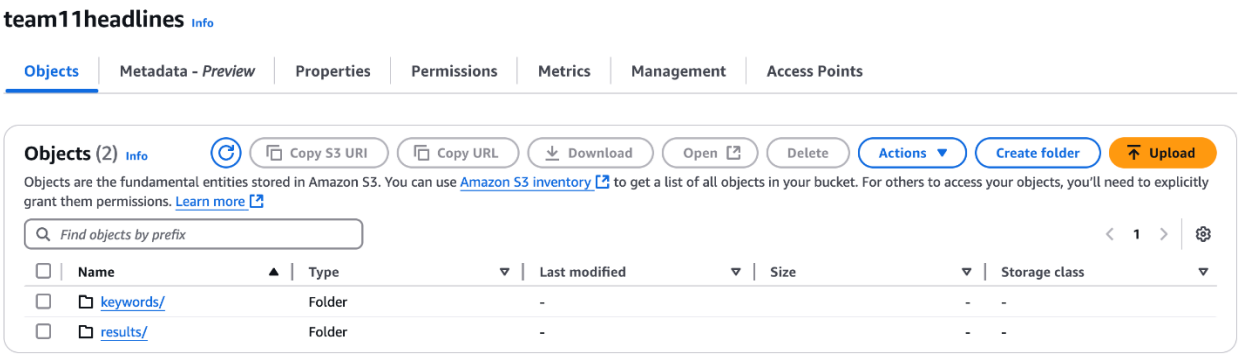
```
import boto3
from collections import Counter
import re
from datetime import datetime

STOPWORDS = {
    "a", "an", "and", "the", "is", "it", "to", "of", "in", "for", "on",
    "with", "as", "this", "by", "at", "from", "that", "or", "be", "are",
    "was", "were", "but", "not", "which", "you", "we", "they", "he",
    "she", "his", "her", "their", "our", "my", "your", "its", "so"
```

```
}

def clean_and_tokenize(text):
    words = re.findall(r'\b\w+\b', text.lower())
    return words

def remove_stopwords(words):
    return [word for word in words if word not in STOPWORDS]

def find_most_common_keywords(text, n=10):
    words = clean_and_tokenize(text)
    filtered_words = remove_stopwords(words)
    word_counts = Counter(filtered_words)
    return word_counts.most_common(n)

def lambda_handler(event, context):
    bucket_name = "team11headlines"
    s3_client = boto3.client("s3")

    # Combine all text from S3 files
    all_text = ""
    response = s3_client.list_objects_v2(Bucket=bucket_name, Prefix="results/")
    if "Contents" in response:
        for obj in response["Contents"]:
            file_key = obj["Key"]
```

```
            print(f"Processing file: {file_key}")
            file_content = s3_client.get_object(Bucket=bucket_name,
Key=file_key)["Body"].read().decode("utf-8")
            all_text += file_content + "\n"

    # Process text and find keywords
    if not all_text.strip():
        print("No text to process.")
        return {
            "statusCode": 400,
            "body": "No text found in bucket files to process."
        }

    print("Processing text to extract keywords...")
    common_keywords = find_most_common_keywords(all_text, n=10)

    # Save keywords to a file in S3
    timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
    output_file_name = f"keywords/keyword_analysis_{timestamp}.txt"
    temp_file_path = f"/tmp/{output_file_name.split('/')[-1]}"
    with open(temp_file_path, "w") as file:
        file.write("\n".join([f"{word}: {count}" for word, count in
common_keywords]))

    s3_client.upload_file(temp_file_path, bucket_name, output_file_name)
```

```
    print(f"Keywords file uploaded to S3: s3://{bucket_name}/{output_file_name}")


    return {
        "statusCode": 200,
        "body": f"Keyword analysis completed. Results saved to s3://{bucket_name}/{output_file_name}"
    }
```

Results:



keyword_analysis_20241207_211059.txt

Close

keyword_analysis_20241207_211059.txt

```
notre: 4
dame: 4
syrian: 4
damascus: 4
rebels: 3
assad: 3
cathedral: 2
reopening: 2
paris: 2
live: 2
```

result_20241207_201634.txt

Close

result_20241207_201634.txt

```
Trump attends Notre Dame Cathedral reopening in Paris
World leaders gather for reopening of Notre-Dame Cathedral in Paris
Live updates: Notre Dame bells ring for the first time since fire
An archbishop's knock formally restores Notre Dame to life as winds howl and heads of state look on
Syrian rebels begin to encircle Damascus amid denials Assad has fled
Syrian Forces Withdraw From Damascus Suburbs, Monitors Say: Live Updates
Syrian rebels threaten Damascus, Assad from north and south
Syrian rebels battle for Homs and advance on Damascus, Assad's rule at stake
```

GITHUB LINK:

https://github.com/tpowell48/News-Article-Data-Pipeline