redirects-script-send

April 10, 2024

1 Redirect Script

Run the cell below this one
Change the **index_range = slice(0,300000)** line for more precise ranges

[ ]:
```
import aiohttp
import asyncio
import nest_asyncio
import pandas as pd
from urllib.parse import urlparse
import validators
import time

nest_asyncio.apply()

def initial_processing(url):
    if not url or url != url or pd.isna(url):
        return ''

    # Sanitize URL
    corrected_url = sanitize_url(url)
    return corrected_url

# Function to sanitize/correct URLs missing pieces
def sanitize_url(url):
    # Parse URL to correct any issues then reconstruct
    parsed_url = urlparse(url)

    if not parsed_url.scheme:
    # Assume http scheme
        corrected_url = 'http://'+parsed_url.netloc + parsed_url.path +␣
 ↪parsed_url.params + parsed_url.query + parsed_url.fragment
    else:
        corrected_url = parsed_url.geturl()

    return corrected_url

async def check_url(session, url, semaphore):
```

```
    async with semaphore:
        try:
            async with session.head(url, allow_redirects=True, timeout=100) as␣
 ↪response:
                return str(response.url) # Return final URL as string
        # Catch errors
        except asyncio.TimeoutError as te:
            return 'Timeout Error'
        except aiohttp.ClientError as ce:
            return 'Client Error'
        except ValueError as ve:
            return 'Value Error'

async def process_urls(urls, MAX_CONCURRENT_REQUESTS):
    print(f"processing {len(urls)} urls")
    semaphore = asyncio.Semaphore(MAX_CONCURRENT_REQUESTS)
    async with aiohttp.ClientSession() as session:
        tasks = [check_url(session, url, semaphore) for url in urls]
        results = await asyncio.gather(*tasks)
        print('re',results)
        return results

def update_redirect_urls(file_path, index_range, redirect_urls):
    df = pd.read_csv(file_path, low_memory=False)
    df_col = df['Website Redirect'] if 'Website Redirect' in df else␣
 ↪['']*len(df)
    new_list = list(df_col[:index_range.start]) + redirect_urls +␣
 ↪list(df_col[index_range.stop:])
    df['Website Redirect'] = new_list
    df.to_csv(file_path, index=False)

def main():
    start_time = time.time()
    MAX_CONCURRENT_REQUESTS = 1000

    file_path = './Excel_Sheets_Public/Website_Redirects_230919.csv'
    df = pd.read_csv(file_path, low_memory=False)

    # Change this line for more precise ranges
    index_range = slice(16,32)

    if 'Website' not in df.columns:
        print("The CSV file must have a 'Website' column containing the URLs.")
    else:
        raw_urls, redirect_urls = df['Website'][index_range].tolist(), []
        if 'Website Redirect' in df:
```

```
            redirect_urls = df.get('Website Redirect', pd.Series(dtype=str)).
↪tolist()[index_range]

        # Check if 'Website Redirect' column is already populated (with valid␣
↪URL)
        for i, redirect_url in enumerate(redirect_urls):
            if redirect_url and validators.url(redirect_url):
                raw_urls[i] = redirect_url

        print('raw',len(raw_urls))
        # Process the URLs asynchronously
        sanitized_urls = [initial_processing(url) for url in raw_urls]
        valid_urls = [url if validators.url(url) else '' for url in␣
↪sanitized_urls]


        # Run the asynchronous function using asyncio.run()
        loop = asyncio.get_event_loop()
        final_urls = loop.run_until_complete(process_urls(valid_urls,␣
↪MAX_CONCURRENT_REQUESTS))

        # Update 'Website Redirect' column in the CSV file with final URLs
        update_redirect_urls(file_path, index_range, final_urls)

        print(f"'Website Redirect' column updated with {len(final_urls)} new␣
↪urls in {time.time()-start_time} seconds.")

main()
```

[ ]:
```
import pandas as pd

file_path = './Excel_Sheets/Pieces/Website_Redirects_230919_850t2150.csv'
df = pd.read_csv(file_path, low_memory=False)

errno, count = 0,0
nav_list = list(df['Nav'])
for i, rd in enumerate(list(df['Website Redirect'])):
    # if type(rd) == str and 'Timeout Error' in rd:
    if type(rd) == str:
        if 'Error' in rd or ('Access Denied' in nav_list[i] or 'Forbidden' in␣
↪nav_list[i]):
            errno+=1
        count +=1

errno/count
```

[ ]: