Assignment 1.1: APIs and Web Scraping

Renetta Nelson

May 15, 2023

ADS 509 Section 1

This notebook has two parts. In the first part, you will scrape lyrics from AZLyrics.com. In the second part, you'll run code that verifies the completeness of your data pull.

For this assignment you have chosen two musical artists who have at least 20 songs with lyrics on AZLyrics.com. We start with pulling some information and analyzing them.

Importing Libraries

In [ ]:

```python
import os
import datetime
import re

# for the lyrics scrape section
import requests
import time
from bs4 import BeautifulSoup
from collections import defaultdict, Counter
import random
```

In [ ]:

```python
# Use this cell for any import statements you add

from bs4 import BeautifulSoup
import shutil
import git
```

Lyrics Scrape

This section asks you to pull data by scraping www.AZLyrics.com. In the notebooks where you do that work you are asked to store the data in specific ways.

In [ ]:

```
artists = {'maverickcity':"https://www.azlyrics.com/m/maverickcitymusic.html",
           'cece':"https://www.azlyrics.com/c/cecewinans.html"}
# we'll use this dictionary to hold both the artist name and the link on AZlyrics
```

A Note on Rate Limiting

The lyrics site, www.azlyrics.com, does not have an explicit maximum on number of requests in any one time, but in our testing it appears that too many requests in too short a time will cause the site to stop returning lyrics pages. (Entertainingly, the page that gets returned seems to only have the song title to a Tom Jones song.)

Whenever you call `requests.get` to retrieve a page, put a `time.sleep(5 + 10*random.random())` on the next line. This will help you not to get blocked. If you *do* get blocked, which you can identify if the returned pages are not correct, just request a lyrics page through your browser. You'll be asked to perform a CAPTCHA and then your requests should start working again.

Part 1: Finding Links to Songs Lyrics

That general artist page has a list of all songs for that artist with links to the individual song pages.

Q: Take a look at the `robots.txt` page on www.azlyrics.com. (You can read more about these pages here.) Is the scraping we are about to do allowed or disallowed by this page? How do you know?

A: According to the robots.txt, web crawlers (except 008 which is blocked from everything) are blocked from crawling the following folders: lyricsdb and songs. The scraping that we are about to do is allowed because we are not scraping any pages from the folders specified. We are scraping from the lyrics folder of the artists.

In [ ]:

```
# Let's set up a dictionary of lists to hold our links
lyrics_pages = defaultdict(list)
```

```
for artist, artist_page in artists.items() :
    # request the page and sleep
    r = requests.get(artist_page)
    print(r.status_code)
    time.sleep(5 + 10*random.random())

    # now extract the links to lyrics pages from this page
    rlink = BeautifulSoup(r.text, 'lxml')
    total_links = []

    if artist == "maverickcity":
        for rlinks in rlink.find_all('a', attrs={'href': re.compile("/lyrics/maverickcitymusic/")}):
            lyrics_pages[artist].append(rlinks.get('href'))
    else:
        for rlinks in rlink.find_all('a', attrs={'href': re.compile("/lyrics/cecewinans/")}):
            lyrics_pages[artist].append(rlinks.get('href'))

    # store the links `lyrics_pages` where the key is the artist and the
    # value is a list of links.

    #lyrics_pages[artist].append(total_links)
    #lyrics_pages[artist].append(rlinks.get('href'))

#print(lyrics_pages)
```

```
200
200
```

Let's make sure we have enough lyrics pages to scrape.

In [ ]:

```
for artist, lp in lyrics_pages.items() :
    assert(len(set(lp)) > 20)
```

In [ ]:

```
# Let's see how long it's going to take to pull these lyrics
# if we're waiting `5 + 10*random.random()` seconds
for artist, links in lyrics_pages.items() :
    print(f"For {artist} we have {len(links)}.")
    print(f"The full pull will take for this artist will take {round(len(links)*10/3600,2)} hours.")
```

```
For maverickcity we have 174.
The full pull will take for this artist will take 0.48 hours.
For cece we have 138.
The full pull will take for this artist will take 0.38 hours.
```

Part 2: Pulling Lyrics

Now that we have the links to our lyrics pages, let's go scrape them! Here are the steps for this part.

1. Create an empty folder in our repo called "lyrics".
2. Iterate over the artists in `lyrics_pages`.
3. Create a subfolder in lyrics with the artist's name. For instance, if the artist was Cher you'd have `lyrics/cher/` in your repo.
4. Iterate over the pages.
5. Request the page and extract the lyrics from the returned HTML file using BeautifulSoup.
6. Use the function below, `generate_filename_from_url`, to create a filename based on the lyrics page, then write the lyrics to a text file with that name.

In [ ]:

```
def generate_filename_from_link(link) :

    if not link :
        return None

    # drop the http or https and the html
    name = link.replace("https","").replace("http","")
    name = link.replace(".html","")

    name = name.replace("/lyrics/","")

    # Replace useless chareacters with UNDERSCORE
    name = name.replace("://","").replace(".","_").replace("/","_")

    # tack on .txt
    name = name + ".txt"

    return(name)
```

In [ ]:

```
# Make the lyrics folder here. If you'd like to practice your programming, add functionality
# that checks to see if the folder exists. If it does, then use shutil.rmtree to remove it and create a new one.

p_dir = "C:/Users/nelso/Desktop/"

new_dir = "lyrics"

lyrics_dir = os.path.join(p_dir, new_dir)
```

```
if os.path.isdir(lyrics_dir) :
    shutil.rmtree(lyrics_dir)

os.mkdir(lyrics_dir)
```

In [ ]:

```
url_stub = "https://www.azlyrics.com"
start = time.time()

total_pages = 0
title = []
lyrics = []



for artist in lyrics_pages :

    # Use this space to carry out the following steps:

    # 1. Build a subfolder for the artist

    path = "C:/Users/nelso/Desktop/lyrics/%s" % artist

    if os.path.isdir(path) :
        shutil.rmtree(path)
    os.mkdir(path)

    os.chdir(path)

    # 2. Iterate over the lyrics pages

    for i in lyrics_pages[artist]:

        # 3. Request the lyrics page.

        art_request = requests.get("https://www.azlyrics.com%s" % i)
        time.sleep(5 + 10*random.random())

        # 4. Extract the title and lyrics from the page.

        extract = BeautifulSoup(art_request.content, "html.parser")
        descript = extract.find('meta', attrs = {'name':'description'}).get("content")

        # 5. Write out the title, two returns ('\n'), and the lyrics. Use `generate_filename_from_url`
        #    to generate the filename.
        file_name = generate_filename_from_link(i)

        f = open(file_name, "x")
        f.close()
        f = open(file_name, "w")

        col_descript = descript.split(':')
        for i in range(0, len(col_descript)):
            f.write(col_descript[i])
            f.write("\n\n")




        f.close()
        f.close()
```

In [ ]:

```
print(f"Total run time was {round((time.time() - start)/3600,2)} hours.")
```

```
Total run time was 2.04 hours.
```

Evaluation

This assignment asks you to pull data by scraping www.AZLyrics.com. After you have finished the above sections , run all the cells in this notebook. Print this to PDF and submit it, per the instructions.

In [ ]:

```
# Simple word extractor from Peter Norvig: https://norvig.com/spell-correct.html
def words(text):
    return re.findall(r'\w+', text.lower())
```

Checking Lyrics

The output from your lyrics scrape should be stored in files located in this path from the directory: `/lyrics/[Artist Name]/[filename from URL]` . This code summarizes the information at a high level to help the instructor evaluate your work.

In [ ]:

```
artist_folders = os.listdir("C:/Users/nelso/Desktop/lyrics/")

#artist_folders = [f for f in artist_folders if os.path.isdir("C:/Users/nelso/Desktop/lyrics" + f)]

for artist in artist_folders :
    artist_files = os.listdir("C:/Users/nelso/Desktop/lyrics/" + artist)
    artist_files = [f for f in artist_files if 'txt' in f or 'csv' in f or 'tsv' in f]

    print(f"For {artist} we have {len(artist_files)} files.")

    artist_words = []

    for f_name in artist_files :
        with open("C:/Users/nelso/Desktop/lyrics/" + artist + "/" + f_name) as infile :
            artist_words.extend(words(infile.read()))


    print(f"For {artist} we have roughly {len(artist_words)} words, {len(set(artist_words))} are unique.")
```

```
For cece we have 138 files.
For cece we have roughly 3641 words, 778 are unique.
For maverickcity we have 174 files.
For maverickcity we have roughly 4812 words, 1099 are unique.
```