Data Wrangling Submission

Nick Sun

Process

Blink-182 was one of my favorite bands growing up and certainly falls into the category of bands that have changed their sound over the years. Their new style is noticeably more pop and less juvenile than their earlier work, but some elements, such as the use of repetitive melodic recitatives (example: "na na na na na na") remain the same.

I thought it might be interesting to gather a corpus of Blink-182 lyrics to analyze. While there are APIs that have lyrical content for certain artists available, I decided to use web scraping instead since many fan sourced sites have transcriptions of live songs that are not found on commercial APIs. The site I used is azlyrics.com since it contains not only the song lyrics but album information and year of release. I used Python for this assignment, specifically the `BeautifulSoup` and `requests` libraries.

The first step to getting a corpus of lyrics is getting a list of all the songs available for that artist on azlyrics. This can be done by getting the html for Blink-182 artists page and then using a basic CSS selector to grab both the names of all the songs and the corresponding href attribute for the URL to the individual lyrics page and storing this information as dictionaries. Overall there are 176 songs on azlyrics.com for Blink-182, so 176 dictionaries will be created and stored together in a list.

Afterwards, I converted this list of dictionaries into a pandas dataframe to iterate through the list of song URLs. These URLs led to individual song pages that contained both the lyrics and other information such as the album and year of release. The lyrics were contained in a div that did not have any identifying attributes, so the only way to get to this text in the DOM is to use a CSS selector. Thankfully, the individual song pages all have about the same format with only some variation. The text did need to be cleaned slightly since it contained newline and carriage return whitespace characters.

Scraping the other information off the page was slightly more complicated since the div which contains the album name and release date needs to be cleaned using regular expressions. However, some trial and error eventually yielded the correct expression to extract both the album name and the year. There are likely some edge cases that are being missed with this particular regex, such as Unicode characters, but for the Blink-182 discography this approach seems to work.

Some other considerations that I had were using an inner join to connect the dataframe containing the song information with the original dataframe containing all the song titles and URLs. Finally, I had to make sure to put a sleep in between scrapes of the individual song pages, since otherwise azlyrics.com would think I am malicious traffic and block me from their server. This took a few tries to get right, since it was difficult for me to estimate how quick I could make the sleeps without being blocked. I did implement a print statement to stdout so I could tell if and where the script stopped working. The final corpus of lyrics was exported using pandas as both a csv and json file.

Some fun ideas I have for this dataset include analyzing the sentiment of Blink-182 songs over time, charting the length of songs between different albums, or possibly training a Markov Chain to write my own Blink-182 songs!

Initial Data

The initial data of this project were HTML webpages. Trying to `curl` the contents of the webpage produces a request for access page, so unfortunately using this utility to provide an example of the HTML will not work.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="robots" content="noindex,nofollow">
    <title>AZLyrics - request for access</title>

    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
```

Example Data: CSV

```
title,url,lyrics,album_name,year
Reebok Commercial,https://www.azlyrics.com/lyrics/blink182/reebokcommercial.html,"You are better than m
Time,https://www.azlyrics.com/lyrics/blink182/time.html,When the clock strikes two There's so much to d
Red Skies,https://www.azlyrics.com/lyrics/blink182/redskies.html,"Why can't people just understand  mon
Alone,https://www.azlyrics.com/lyrics/blink182/alone.html,"what were doing here, now  no one knows  the
```

Example Data: JSON

This data is output using the `to_json` method in pandas DataFrames. By default, it outputs the data as `orient='index'`, but can instead be output in other formats.

```
{"title":{"0":"Reebok Commercial","4":"Time","8":"Red Skies","9":"Alone","10":"Point Of View","14":"Mar
```

Example of the Loaded Data

A screenshot of the final csv loaded into `R` is provided as a .png file with this submission.

Scripts

The Python module used to scrape the data had the following code:

```
#!/usr/bin/env python

import pandas as pd
import requests
import time
import re
from bs4 import BeautifulSoup
```

```
def get_songs(url):
    """
    Function to get list of songs and corresponding URLs under an artist page
    Input: url to an artist's page on azlyrics.com
    Output: Returns a list of dicts ({'title' : [TITLE], 'url' : [URL]})
    """
    songlist = requests.get(url)
    soup = BeautifulSoup(songlist.text)

    songinfo = []

    for row in soup.findAll('div', attrs = {'class' : 'listalbum-item'}):
        song = {}
        song['title'] = row.a.text
        song['url'] = row.a['href'].replace("..", "https://www.azlyrics.com")
        songinfo.append(song)

    return(songinfo)

def get_song_info(url):
    """
    Function to get lyrics, album, and year information for individual song
    Input: url to a song's page on azlyrics.com
    Output: Dict ({'lyrics' : [string LYRICS],
                   'url' : [string URL],
                   'album_name' : [string ALBUM_NAME],
                   'year': [int YEAR]})
    """
    tmp = requests.get(url)
    tmpsoup = BeautifulSoup(tmp.text)

    try:
        tmpalbum = tmpsoup.find('div', attrs = {'class' : 'songinalbum_title'}).text.replace("album: ",
        albumpattern = r'"([A-Za-z0-9]*)"'
        album_name = re.search(albumpattern, tmpalbum).group(0).replace('\"', '')

        yearpattern = r'\([0-9]*\)'
        album_year = re.search(yearpattern, tmpalbum).group(0).replace('(', '').replace(')', '')
    except:
        album_name = "None"
        album_year = 0

    try:
        tmplyrics = tmpsoup.select('body > div.container.main-page > div > div.col-xs-12.col-lg-8.text-
        lyrics = tmplyrics.replace('\n', ' ').replace('\r', ' ').strip()
    except:
        tmplyrics = tmpsoup.select('body > div.container.main-page > div > div.col-xs-12.col-lg-8.text-
        lyrics = tmplyrics.replace('\n', ' ').replace('\r', ' ').strip()

    output = {}
    output['lyrics'] = lyrics
    output['album_name'] = album_name
    output['year'] = album_year
```

```
    output['url'] = url

    return(output)

def json_to_csv(json_file):
    '''
    Converts json to csv files.

    Args:
        json_file (str) : PATH to json file

    Returns:
        .csv file in current working directory
    '''

    try:
        df = pd.read_json(json_file)
        output = df.to_csv(encoding='utf-8',
                           index=False)
        return(output)
    except:
        print("Error in converting to json")
```

The `main.py` file I used was

```
from lyrics_scraper import get_song_info, get_songs

import pandas as pd
import requests
import time
import re
from bs4 import BeautifulSoup

url = 'https://www.azlyrics.com/b/blink.html'

songinfo = get_songs(url)

blink = []

for song in songinfo:
    print("Working on: {}".format(song['title']))
    song_output = get_song_info(song['url'])
    blink.append(song_output)
    time.sleep(10)

song_info = pd.DataFrame(songinfo)
song_lyrics = pd.DataFrame(blink)

blink_songs = pd.merge(left = song_info,
                       right = song_lyrics,
                       left_on = 'url',
                       right_on = 'url')

blink_songs = blink_songs.drop_duplicates()
```

```
blink_songs.to_csv(r'blink_songs.csv',
                   index = None,
                   header = True)

blink_songs.to_json(r'blink_songs.json')
```