# Scraper with Tesseract

February 25, 2022

[1]:
```
import os
import re
import requests
from bs4 import BeautifulSoup as bs
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.desired_capabilities import DesiredCapabilities
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.by import By
import simplejson as json
from datetime import datetime
```

```
/Users/azicon/anaconda3/lib/python3.7/site-packages/requests/__init__.py:91:
RequestsDependencyWarning: urllib3 (1.26.8) or chardet (3.0.4) doesn't match a
supported version!
  RequestsDependencyWarning)
```

### 0.1 Get website response and font data

[2]:
```
browserOptions = Options()
#browserOptions.add_argument("--headless")

capa = DesiredCapabilities.CHROME
capa["pageLoadStrategy"] = "none"
capa["goog:loggingPrefs"] = {"performance": "ALL"}
driver = webdriver.Chrome(desired_capabilities=capa)
wait = WebDriverWait(driver, 20)

#create snapshot of the entire page to prevent it from constantly changing
driver.get("https://piaofang.maoyan.com/dashboard/movie")
test = None
while not test:
    try:
        test = wait.until(EC.presence_of_element_located((By.CLASS_NAME,␣
↪'moviename-td')))
```

```
    except:
        driver.refresh();

now = datetime.now().strftime("%d-%m-%Y_%H:%M:%S") # get exact datetime at the␣
 ↪time of scrape
os.mkdir("logs/" + now)

driver.get_screenshot_as_file("logs/" + now + "/screenshot.png") # save␣
 ↪screenshot to sanity check later

logs_raw = driver.get_log("performance")
logs = [json.loads(lr["message"])["message"] for lr in logs_raw]

def log_filter(log_):
    return (
        # is an actual response
        log_["method"] == "Network.responseReceived"
        # and json
        and "json" in log_["params"]["response"]["mimeType"]
    )

responses = []

for log in filter(log_filter, logs):
    request_id = log["params"]["requestId"]
    resp_url = log["params"]["response"]["url"]
    print(f"Caught {resp_url}")
    response = driver.execute_cdp_cmd("Network.getResponseBody", {"requestId":␣
 ↪request_id})
    responses.append(response)
```

```
Caught https://piaofang.maoyan.com/dashboard-ajax/movie?orderType=0&uuid=8889a5d
f-571f-45aa-9e7c-803e93565323&timeStamp=1645843657776&User-Agent=TW96aWxsYS81LjA
gKE1hY2ludG9zaDsgSW50ZWwgTWFjIE9TIFggMTBfMTVfNykgQXBwbGVXZWJLaXQvNTM3LjM2IChLSFR
NTCwgbGlrZSBHZWNrbykgQ2hyb21lLzk3LjAuNDY5Mi45OSBTYWZhcmkvNTM3LjM2&index=820&chan
nelId=40009&sVersion=2&signKey=6ddead5b4a3f722d46590a745bbc6101
```

[3]:
```
# Get this instance's font file from backend server
body0 = json.loads(responses[0]['body'])
movieList = body0['movieList']['list']
date = body0['calendar']['today']
font_url = body0['fontStyle'].split('"')[-2]

# Get reference fonts from the file tree
from fontTools.ttLib import TTFont
headers = {
```

```
            "User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_4)␣
 ↪AppleWebKit/537.36 (KHTML, like Gecko) "
              "Chrome/66.0.3359.139 Safari/537.36 "
    }

woff_url = 'http:' + font_url
response_woff = requests.get(woff_url, headers=headers).content

print("Woff retrieval succuessful: " + str(len(response_woff) > 0))

with open('temp/fonts.woff', 'wb') as f:
    f.write(response_woff)
```

```
Woff retrieval succuessful: True
```

### 0.2 Getting digits from the font data using pytesseract

[4]:
```
from fontTools.ttLib import TTFont
from PIL import ImageFont, Image, ImageDraw, ImageOps
import pytesseract
import cv2
import numpy as np
import random

def uniToHex(uni):
    return "&#x" + uni[3:].lower()

def uni_2_png_stream(txt: str, font: str, img_size=512, font_size=0.7,␣
 ↪invert=False):
    img = Image.new('1', (img_size, img_size), 255)
    draw = ImageDraw.Draw(img)
    font = ImageFont.truetype(font, int(img_size * font_size))

    txt = chr(txt)
    x, y = draw.textsize(txt, font=font)
    draw.text(((img_size - x) // 2, (img_size - y) // 2), txt, font=font,␣
 ↪fill=0)
    if invert:
        img = img.convert('L')
        img = ImageOps.invert(img)
        img = img.convert('1')
    #img.save(txt + '.png')
    return img

def predict_neural(unicode, fontFile):
    image = uni_2_png_stream(int(unicode[3:], 16), fontFile, img_size=28,␣
 ↪font_size=0.5, invert=True)
```

```
    image.save(str(unicodeToInt[unicode]) + '_neuro.png')
    matrix_form = np.array(image)
    weighted_predictions = np.ndarray.flatten(neural_network.run(matrix_form))
    most_possible = np.argmax(weighted_predictions)
    return most_possible

def predict_tesseract(unicode, fontFile, fontSize=0.5):
    image = uni_2_png_stream(int(unicode[3:], 16), fontFile, img_size=1024,␣
 ↪font_size=fontSize)
    image.save('logs/' + str(now) + '/' + str(unicode) + '.png')
    text = pytesseract.image_to_string(image, lang="eng", config="--psm 10␣
 ↪outputbase digits -c tessedit_char_whitelist=0123456789")
    return text

def predict_tesseract_definite(unicode, fontFile):
    result, size = '', 1
    while not result and size >= 0:
        result = predict_tesseract(x, filename, fontSize=size)
        size -= 0.01
    return result
```

[5]:
```
# Map contours to numbers - the prediction phase may be very slow
filename = 'temp/fonts.woff'
f = TTFont(filename)
hexToInt = {}
for x in f.getGlyphNames()[1:-1]:
    predict = predict_tesseract_definite(x, filename)
    hexToInt[uniToHex(x)] = int(predict)

hexToInt
```

```
2 extra bytes in post.stringData array
```

[5]:
```
{'&#xe1ae': 4,
 '&#xe63f': 1,
 '&#xe673': 5,
 '&#xe809': 3,
 '&#xed85': 6,
 '&#xf452': 2,
 '&#xf697': 7,
 '&#xf6e5': 8,
 '&#xf72e': 0,
 '&#xf820': 9}
```

### 0.3 Parsing the data into pandas dataframe

[6]:
```
import pandas as pd
df = pd.DataFrame.from_records(movieList)
```

[7]:
```
unitLookup = {'': 100, '': 1000, '': 10000, '': 1*10**8}

#converts the weird character to a float
def convertToFloat(string):
    spCharLst = string.split(';')
    result = ''
    for i in spCharLst:
        if len(i) > 7: #has a dot in front
            result += '.' + str(hexToInt[i[1:]])
        elif len(i) == 7: #in case of bad parsing
            result += str(hexToInt[i])
    return float(result)

#helper function for converting the entire block to a single int
def convertDictToInt(dictionary):
    return int(convertToFloat(dictionary['num']) *␣
 ↪unitLookup[dictionary['unit']])
```

[8]:
```
df['boxSplitUnit'] = df['boxSplitUnit'].apply(convertDictToInt)
df['splitBoxSplitUnit'] = df['splitBoxSplitUnit'].apply(convertDictToInt)
df['movieInfo'] = df['movieInfo'].apply(lambda x : x['movieName'])
df
```

[8]:

| | avgSeatView | avgShowView | boxRate | boxSplitUnit | movieInfo | showCount | \ |
|---|---|---|---|---|---|---|---|
| 0 | 1.1% | 2 | 25.8% | 6043400 | | 71278 | |
| 1 | 0.7% | 2 | 12.2% | 2868399 | | 67234 | |
| 2 | 1.2% | 2 | 11.6% | 2719800 | | ů 49189 | |
| 3 | 2.7% | 3 | 10.8% | 2543500 | | 24173 | |
| 4 | 2.2% | 3 | 9.9% | 2320800 | ů | 24790 | |
| 5 | 2.7% | 3 | 7.5% | 1753000 | | 16943 | |
| 6 | 0.8% | 1 | 4.3% | 1012300 | | 27188 | |
| 7 | 1.2% | 2 | 3.8% | 905699 | | 12004 | |
| 8 | 0.6% | 1 | 3.4% | 801800 | | 31338 | |
| 9 | 0.5% | 1 | 2.8% | 667300 | | 27933 | |
| 10 | 1.8% | 2 | 1.5% | 360500 | 5494 | | |
| 11 | 5.3% | 6 | 0.8% | 204400 | | 648 | |
| 12 | 1.0% | 1 | 0.7% | 164500 | | 4941 | |
| 13 | 5.5% | 6 | 0.6% | 163500 | | 303 | |
| 14 | 0.3% | 1 | 0.6% | 150800 | | 10991 | |
| 15 | 0.7% | 1 | 0.5% | 122600 | | 4085 | |
| 16 | 35.0% | 44 | 0.4% | 107500 | | 54 | |
| 17 | 3.3% | 4 | 0.3% | 78900 | | 695 | |
| 18 | 70.1% | 65 | 0.1% | 35600 | | 11 | |
| 19 | 0.5% | 1 | <0.1% | 21500 | | 1156 | |

```
20        0.6%           1   <0.1%         17900              886
21        0.3%           1   <0.1%         12600             1088
22        0.2%           1   <0.1%         10200        1922       1273
```

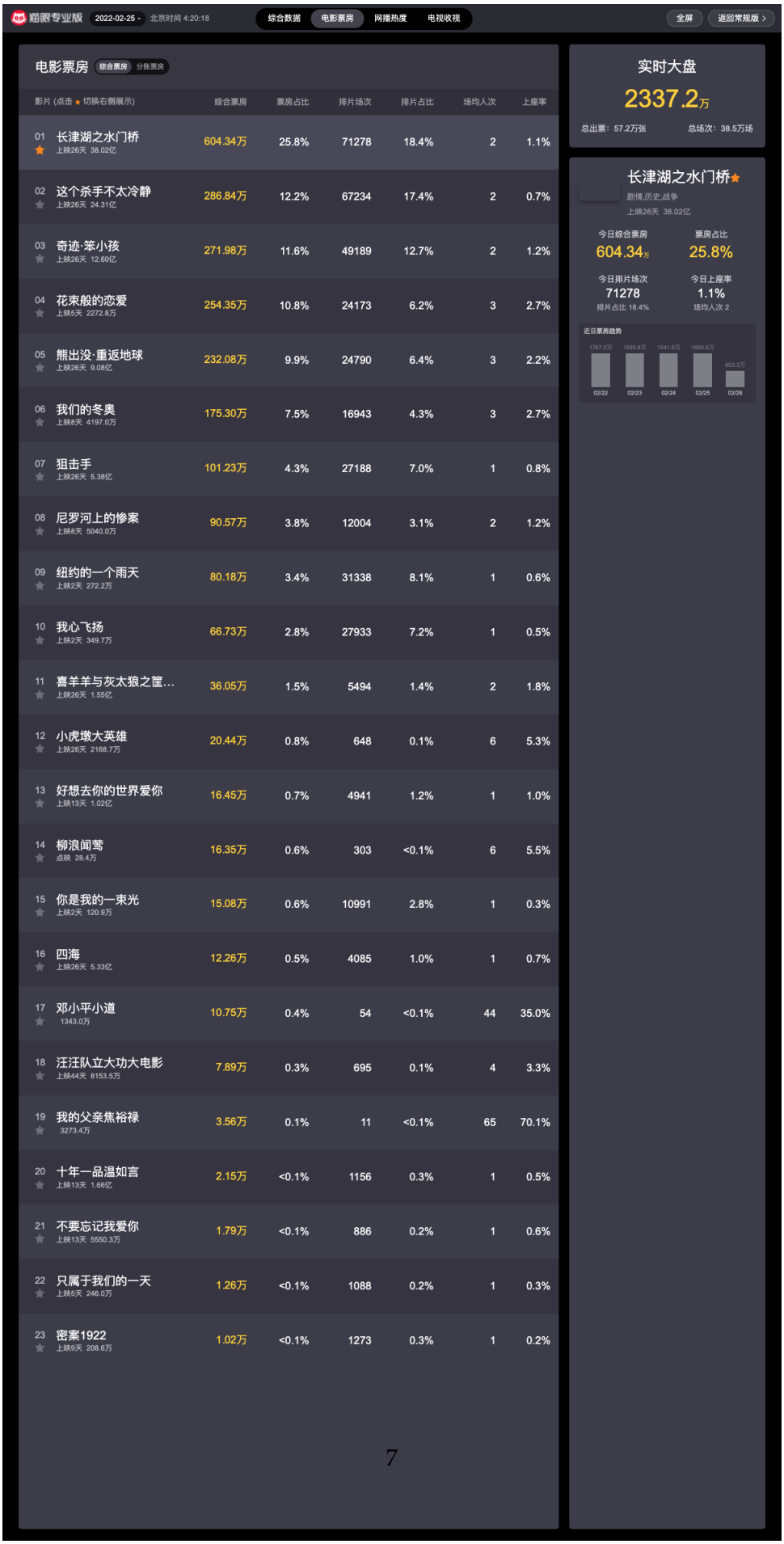
| | showCountRate | splitBoxRate | splitBoxSplitUnit | sumBoxDesc | sumSplitBoxDesc |
|---|---|---|---|---|---|
| 0 | 18.4% | 26.7% | 5758099 | 38.02 | 35.08 |
| 1 | 17.4% | 12.2% | 2635899 | 24.31 | 22.23 |
| 2 | 12.7% | 11.5% | 2479200 | 12.60 | 11.52 |
| 3 | 6.2% | 10.5% | 2259200 | 2272.8 | 2002.5 |
| 4 | 6.4% | 9.9% | 2137100 | 9.08 | 8.33 |
| 5 | 4.3% | 7.4% | 1591300 | 4197.0 | 3847.0 |
| 6 | 7.0% | 4.2% | 923100 | 5.38 | 4.94 |
| 7 | 3.1% | 3.8% | 818600 | 5040.0 | 4540.9 |
| 8 | 8.1% | 3.3% | 721500 | 272.2 | 245.2 |
| 9 | 7.2% | 2.8% | 608400 | 349.7 | 316.6 |
| 10 | 1.4% | 1.5% | 328800 | 1.55 | 1.41 |
| 11 | 0.1% | 0.7% | 163600 | 2168.7 | 2000.6 |
| 12 | 1.2% | 0.6% | 149700 | 1.02 | 9283.5 |
| 13 | <0.1% | 0.7% | 157300 | 28.4 | 26.9 |
| 14 | 2.8% | 0.6% | 135700 | 120.9 | 109.9 |
| 15 | 1.0% | 0.5% | 112899 | 5.33 | 4.82 |
| 16 | <0.1% | 0.4% | 99700 | 1343.0 | 1336.3 |
| 17 | 0.1% | 0.3% | 70400 | 8153.5 | 7218.9 |
| 18 | <0.1% | 0.1% | 32900 | 3273.4 | 3255.1 |
| 19 | 0.3% | <0.1% | 19700 | 1.66 | 1.51 |
| 20 | 0.2% | <0.1% | 16100 | 5550.3 | 5021.3 |
| 21 | 0.2% | <0.1% | 11399 | 246.0 | 224.4 |
| 22 | 0.3% | <0.1% | 9200 | 208.6 | 188.1 |

[9]:
```
#
body0['movieList']['nationBoxInfo']
```

[9]:
```
{'nationBoxSplitUnit': {'num': '&#xf452;&#xe809;&#xe809;&#xf697;.&#xf452;',
  'unit': ''},
 'nationSplitBoxSplitUnit': {'num': '&#xf452;&#xe63f;&#xe673;&#xf72e;.&#xe809;',
  'unit': ''},
 'showCountDesc': '38.5',
 'title': '',
 'viewCountDesc': '57.2'}
```

[10]:
```
# Comparing with the screenshot earlier
from IPython.display import Image as displayImage
displayImage(filename="logs/" + now + "/screenshot.png")
```

[10]:


猫眼专业版
2022-02-25
北京时间 4:20:18
综合数据
电影票房
网播热度
电视收视
全屏
返回常规版
电影票房
综合票房
分账票房
影片 (点击 ★ 切换右侧展示)
综合票房
票房占比
排片场次
排片占比
场均人次
上座率
01 长津湖之水门桥 上映26天 38.02亿 604.34万 25.8% 71278 18.4% 2 1.1%
02 这个杀手不太冷静 上映26天 24.31亿 286.84万 12.2% 67234 17.4% 2 0.7%
03 奇迹·笨小孩 上映26天 12.60亿 271.98万 11.6% 49189 12.7% 2 1.2%
04 花束般的恋爱 上映5天 2272.8万 254.35万 10.8% 24173 6.2% 3 2.7%
05 熊出没·重返地球 上映26天 9.08亿 232.08万 9.9% 24790 6.4% 3 2.2%
06 我们的冬奥 上映8天 4197.0万 175.30万 7.5% 16943 4.3% 3 2.7%
07 狙击手 上映26天 5.38亿 101.23万 4.3% 27188 7.0% 1 0.8%
08 尼罗河上的惨案 上映8天 5040.0万 90.57万 3.8% 12004 3.1% 2 1.2%
09 纽约的一个雨天 上映2天 272.2万 80.18万 3.4% 31338 8.1% 1 0.6%
10 我心飞扬 上映2天 349.7万 66.73万 2.8% 27933 7.2% 1 0.5%
11 喜羊羊与灰太狼之筐... 上映26天 1.55亿 36.05万 1.5% 5494 1.4% 2 1.8%
12 小虎墩大英雄 上映26天 2168.7万 20.44万 0.8% 648 0.1% 6 5.3%
13 好想去你的世界爱你 上映13天 1.02亿 16.45万 0.7% 4941 1.2% 1 1.0%
14 柳浪闻莺 点映 28.4万 16.35万 0.6% 303 <0.1% 6 5.5%
15 你是我的一束光 上映2天 120.9万 15.08万 0.6% 10991 2.8% 1 0.3%
16 四海 上映26天 5.33亿 12.26万 0.5% 4085 1.0% 1 0.7%
17 邓小平小道 1343.0万 10.75万 0.4% 54 <0.1% 44 35.0%
18 汪汪队立大功大电影 上映44天 8153.5万 7.89万 0.3% 695 0.1% 4 3.3%
19 我的父亲焦裕禄 3273.4万 3.56万 0.1% 11 <0.1% 65 70.1%
20 十年一品温如言 上映13天 1.66亿 2.15万 <0.1% 1156 0.3% 1 0.5%
21 不要忘记我爱你 上映13天 5550.3万 1.79万 <0.1% 886 0.2% 1 0.6%
22 只属于我们的一天 上映5天 246.0万 1.26万 <0.1% 1088 0.2% 1 0.3%
23 密案1922 上映9天 208.6万 1.02万 <0.1% 1273 0.3% 1 0.2%
实时大盘
2337.2万
总出票：57.2万张
总场次：38.5万场
长津湖之水门桥
剧情,历史,战争
上映26天 38.02亿
今日综合票房
604.34万
票房占比
25.8%
今日排片场次
71278
排片占比 18.4%
今日上座率
1.1%
场均人次 2
近日票房趋势
1767.0万
1595.8万
1541.8万
1956.6万
603.3万
02/22
02/23
02/24
02/25
02/26

[ ]: