AmbitionWebScrapping

December 24, 2024

[ ]:
```python
import pandas as pd
import requests
from bs4 import BeautifulSoup
import random
import time

# List of User Agents
user_agents = [
    'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like␣
 ↪Gecko) Chrome/91.0.4472.124 Safari/537.396',
    'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/605.1.15␣
 ↪(KHTML, like Gecko) Version/16.4 Safari/605.1.15',
    'Mozilla/5.0 (iPhone; CPU iPhone OS 14_6 like Mac OS X) AppleWebKit/605.1.
 ↪15 (KHTML, like Gecko) CriOS/92.0.4515.107 Mobile/15E148 Safari/604.1'
]

def get_random_user_agent():
    return random.choice(user_agents)

def make_request(url, max_retries=3, delay=1):
    for attempt in range(max_retries):
        try:
            headers = {
                'User-Agent': get_random_user_agent(),
                'Accept-Language': 'en-US,en;q=0.9',
                'Accept-Encoding': 'gzip, deflate, br',
                'DNT': '1',
                'Connection': 'keep-alive',
                'Upgrade-Insecure-Requests': '1',
                'Sec-Fetch-Dest': 'document',
                'Sec-Fetch-Mode': 'navigate',
                'Sec-Fetch-Site': 'none',
                'Sec-Fetch-User': '?1',
                'Cache-Control': 'max-age=0'
            }

            response = requests.get(url, headers=headers, timeout=30)
            response.raise_for_status()
            return response
        except requests.RequestException as e:
            if attempt == max_retries - 1:
                raise
            print(f"Attempt {attempt + 1} failed. Retrying in {delay} seconds...
 ↪")
            time.sleep(delay * (2 ** attempt))

    raise Exception("All attempts failed")

# Main execution
url = 'https://www.ambitionbox.com/list-of-companies?page=1'
try:
    webpage = make_request(url)
    soup = BeautifulSoup(webpage.content, 'lxml')
    print(soup.prettify())
except Exception as e:
    print(f"An error occurred: {e}")
```

[102]:
```
company = soup.find_all("div",class_="companyCardWrapper")   #main conatiner
```

[103]:
```
len(company)
```

[103]:
```
20
```

0.1 company name

[108]:
```
count = 0
for i in soup.find_all("h2"):
    print(i.text.strip())
    count += 1
    if count >= 20:
        break
```

```
TCS
Accenture
Wipro
Cognizant
Capgemini
HDFC Bank
ICICI Bank
Infosys
HCLTech
Tech Mahindra
Genpact
Teleperformance
Concentrix Corporation
```

```
Axis Bank
Amazon
Jio
Reliance Retail
IBM
iEnergizer
LTIMindtree
```

0.2 Rating

```
[111]: for i in soup.find_all("div",class_="rating_text rating_text--md"):
           print(i.text.strip())
```

```
3.7
3.9
3.7
3.8
3.8
3.9
4.0
3.7
3.6
3.6
3.9
3.9
3.8
3.8
4.1
3.9
3.9
4.1
4.7
3.9
```

```
[ ]: print(soup.find_all("span",class_= "companyCardWrapper__ActionCount"))
```

```
[ ]:
```

```
[116]: for i in soup.find_all("span",class_= "companyCardWrapper__ActionCount")[0]:
           print(i.text.strip())
```

```
84.4k
```

0.3 Bottom container

```
[119]: for i in soup.find_all("span",class_= "companyCardWrapper__ActionCount")[0]:
           print(i.text.strip())
```

```
84.4k
```

[121]:
```
for i in soup.find_all("span",class_= "companyCardWrapper__ActionCount")[1]:
    print(i.text.strip())
```

```
8.6L
```

[123]:
```
for i in soup.find_all("span",class_= "companyCardWrapper__ActionCount")[2]:
    print(i.text.strip())
```

```
10.1k
```

[125]:
```
for i in soup.find_all("span",class_= "companyCardWrapper__ActionCount")[3]:
    print(i.text.strip())
```

```
88
```

[127]:
```
for i in soup.find_all("span",class_= "companyCardWrapper__ActionCount")[4]:
    print(i.text.strip())
```

```
11.8k
```

[129]:
```
for i in soup.find_all("span",class_= "companyCardWrapper__ActionCount")[5]:
    print(i.text.strip())    #no need
```

```
87
```

0.4 creating an empty data frame

[132]:
```
name =[ ]
ratings=[]
reviews = []
salaries= []
interviews =[]
jobs =[]
benefits =[]
photos=[]
```

0.5 Company Name

[135]:
```
name = []

count = 0
for i in soup.find_all("h2",class_="companyCardWrapper__companyName"):
    text = i.text.strip()
    name.append(text)
    print(text)
    count += 1
    if count >= 20:
        break
```

```
TCS
Accenture
Wipro
Cognizant
Capgemini
HDFC Bank
ICICI Bank
Infosys
HCLTech
Tech Mahindra
Genpact
Teleperformance
Concentrix Corporation
Axis Bank
Amazon
Jio
Reliance Retail
IBM
iEnergizer
LTIMindtree
```

0.6 Ratings Count

[138]:
```python
ratings = []

for i in company:
    try:
        div_elements = i.find_all("div", class_="rating_text rating_text--md")
        if div_elements:
            rating_text = div_elements[0].text.strip()
            ratings.append(rating_text)
        else:
            ratings.append("")  # Or some default value
    except AttributeError as e:
        print(f"Error processing {i}: {e}")
        ratings.append("")  # Add a default value

print(ratings)
```

```
['3.7', '3.9', '3.7', '3.8', '3.8', '3.9', '4.0', '3.7', '3.6', '3.6', '3.9',
'3.9', '3.8', '3.8', '4.1', '3.9', '3.9', '4.1', '4.7', '3.9']
```

0.7 Reviews Count

[141]:
```python
from bs4 import BeautifulSoup

reviews = []
```

```
containers = soup.find_all("div", class_="companyCardWrapper")

for container in containers:
    review_count = container.find("span",␣
 ↪class_="companyCardWrapper__ActionCount")
    if review_count:
        reviews.append(review_count.text.strip())

print(reviews)
```

```
['84.4k', '52.7k', '50.3k', '47.2k', '39k', '37.6k', '36.7k', '36.5k', '33.8k',
'33.1k', '29.8k', '27.3k', '25k', '24.2k', '23.9k', '21.7k', '21.3k', '21.1k',
'20.9k', '19.7k']
```

0.8 Salaries Count

[144]:
```
from bs4 import BeautifulSoup

# Assuming 'soup' is already created with the HTML content
salaries= []

containers = soup.find_all("div", class_="companyCardWrapper")  # Replace with␣
 ↪actual container class

for container in containers:
    salaries_count = container.find_all("span",␣
 ↪class_="companyCardWrapper__ActionCount")[1]
    if salaries_count:
        salaries.append(salaries_count.text.strip())

print(salaries)
```

```
['8.6L', '5.7L', '4.4L', '5.6L', '4.3L', '1.4L', '1.5L', '4.6L', '3.2L', '2.6L',
'2L', '89.1k', '1.2L', '98.1k', '1.2L', '62.4k', '66.1k', '2L', '22.1k', '1.7L']
```

0.9 Interviews Count

[147]:
```
interviews =[]

containers = soup.find_all("div", class_="companyCardWrapper")  # Replace with␣
 ↪actual container class

for container in containers:
    interviews_count = container.find_all("span",␣
 ↪class_="companyCardWrapper__ActionCount")[2]
    if interviews_count:
        interviews.append(interviews_count.text.strip())
```

```
print(interviews)
```

```
['10.1k', '7.8k', '5.5k', '5.4k', '4.7k', '2k', '2.4k', '7.4k', '3.6k', '3.7k',
'2.9k', '1.7k', '1.6k', '1.4k', '4.9k', '1.6k', '1.5k', '2.3k', '525', '2.7k']
```

0.10 Jobs Count

[150]:
```
jobs =[]
containers = soup.find_all("div", class_="companyCardWrapper")  # Replace with␣
 ↪actual container class

for container in containers:
    jobs_count = container.find_all("span",␣
 ↪class_="companyCardWrapper__ActionCount")[3]
    if jobs_count:
        jobs.append(jobs_count.text.strip())

print(jobs)
```

```
['88', '21.9k', '567', '1.4k', '1.2k', '164', '--', '989', '155', '361', '2.5k',
'308', '56', '136', '104', '4.1k', '28', '3.1k', '91', '106']
```

0.11 Benefits Count

[153]:
```
benefits =[]

containers = soup.find_all("div", class_="companyCardWrapper")  # Replace with␣
 ↪actual container class

for container in containers:
    benefits_count = container.find_all("span",␣
 ↪class_="companyCardWrapper__ActionCount")[4]
    if benefits_count:
        benefits.append(benefits_count.text.strip())

print(benefits)
```

```
['11.8k', '7.3k', '5.2k', '6k', '4.1k', '3.3k', '3.8k', '5.3k', '4.2k', '3.7k',
'3.8k', '2.2k', '3.4k', '2.2k', '4.4k', '2.7k', '2k', '2.8k', '559', '1.2k']
```

0.12 Photos Count

[156]:
```
photos=[]
containers = soup.find_all("div", class_="companyCardWrapper")  # Replace with␣
 ↪actual container class

for container in containers:
    photos_count = container.find_all("span",␣
↪class_="companyCardWrapper__ActionCount")[5]
    if photos_count:
        photos.append(photos_count.text.strip())

print(photos)
```

```
['87', '39', '90', '69', '41', '29', '55', '108', '33', '63', '46', '31', '55',
'80', '78', '65', '113', '23', '25', '34']
```

[ ]:
```

```

[ ]:
```

```

[160]:
```
webpage = make_request(url)
if webpage.content is None:
    print(f"No content returned for URL: {url}")
else:
    soup = BeautifulSoup(webpage.content, 'lxml')
```

```
Attempt 1 failed. Retrying in 1 seconds…
```

[161]:
```
soup = BeautifulSoup(webpage.content, 'html.parser')
```

[162]:
```
try:
    soup = BeautifulSoup(webpage.content, 'lxml')
except Exception as e:
    print(f"Error parsing HTML: {e}")
```

[163]:
```
if webpage.content.strip():
    soup = BeautifulSoup(webpage.content, 'lxml')
else:
    print("No valid content returned")
```

0.13 Fetching the first page in a DataFrame

[182]:
```
name =[ ]
ratings=[]
reviews = []
salaries= []
interviews =[]
jobs =[]
benefits =[]
photos = []
for i in company:
  name.append(i.find('h2').text.strip())
  ratings.append(i.find('div', class_='rating_text rating_text--md').text.
 ↪strip() if card.find('div', class_='rating_text rating_text--md') else '')
  reviews.append(i.find_all('span',class_='companyCardWrapper__ActionCount')[0].
 ↪text.strip())
  salaries.append(i.
 ↪find_all('span',class_='companyCardWrapper__ActionCount')[1].text.strip())
  interviews.append(i.
 ↪find_all('span',class_='companyCardWrapper__ActionCount')[2].text.strip())
  jobs.append(i.find_all('span',class_='companyCardWrapper__ActionCount')[3].
 ↪text.strip())
  benefits.append(i.
 ↪find_all('span',class_='companyCardWrapper__ActionCount')[4].text.strip())
  photos.append(i.find_all('span',class_='companyCardWrapper__ActionCount')[5].
 ↪text.strip())
```

[ ]:

[184]:
```
df=pd.DataFrame({'name':name,
   'ratings':ratings,
   'reviews':reviews,
   'salaries':salaries,
   'interviews':interviews,
   'jobs':jobs,
   'benefits':benefits,
   'photos':photos,
   })
```

[186]:
```
df.head()
```

[186]:
```
         name ratings reviews salaries interviews   jobs benefits photos
0         TCS     3.7   84.4k     8.6L      10.1k     88    11.8k     87
1   Accenture     3.9   52.7k     5.7L       7.8k  21.9k     7.3k     39
2       Wipro     3.7   50.3k     4.4L       5.5k    567     5.2k     90
3   Cognizant     3.8   47.2k     5.6L       5.4k   1.4k       6k     69
4   Capgemini     3.8     39k     4.3L       4.7k   1.2k     4.1k     41
```

[110]:
```
df.shape
```

[110]:
```
(20, 8)
```

[190]:
```
df.count()
```

[190]:
```
name          20
ratings       20
reviews       20
```

```
salaries      20
interviews    20
jobs          20
benefits      20
photos        20
dtype: int64
```

[198]:
```python
import pandas as pd
import requests
from bs4 import BeautifulSoup
import random
import time

final = pd.DataFrame()

# User agents
user_agents = [
    'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like␣
 ↪Gecko) Chrome/91.0.4472.124 Safari/537.396',
    'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/605.1.15␣
 ↪(KHTML, like Gecko) Version/16.4 Safari/605.1.15',
    'Mozilla/5.0 (iPhone; CPU iPhone OS 14_6 like Mac OS X) AppleWebKit/605.1.
 ↪15 (KHTML, like Gecko) CriOS/92.0.4515.107 Mobile/15E148 Safari/604.1'
]

def get_random_user_agent():
    return random.choice(user_agents)

def make_request(url, max_retries=3, delay=1):
    for attempt in range(max_retries):
        try:
            headers = {
                'User-Agent': get_random_user_agent(),
                'Accept-Language': 'en-US,en;q=0.9',
                'Accept-Encoding': 'gzip, deflate, br',
                'DNT': '1',
                'Connection': 'keep-alive',
                'Upgrade-Insecure-Requests': '1',
                'Sec-Fetch-Dest': 'document',
                'Sec-Fetch-Mode': 'navigate',
                'Sec-Fetch-Site': 'none',
                'Sec-Fetch-User': '?1',
                'Cache-Control': 'max-age=0'
            }

            response = requests.get(url, headers=headers, timeout=30)
            response.raise_for_status()
            return response
        except requests.RequestException as e:
            if attempt == max_retries - 1:
                raise
            print(f"Attempt {attempt + 1} failed. Retrying in {delay} seconds...
↪")
            time.sleep(delay * (2 ** attempt))

    raise Exception("All attempts failed")

# Main execution
for j in range(0, 6):
    url = f'https://www.ambitionbox.com/list-of-companies?page={j}'

    try:
        webpage = make_request(url)
        soup = BeautifulSoup(webpage.content, 'lxml')

        # Find all company cards
        company_cards = soup.find_all('div', class_='companyCardWrapper')

        name = []
        ratings = []
        reviews = []
        salaries = []
        interviews = []
        jobs = []
        benefits = []
        photos = []

        for card in company_cards:
            name.append(card.find('h2').text.strip())
            ratings.append(i.find('div', class_='rating_text rating_text--md').
↪text.strip() if card.find('div', class_='rating_text rating_text--md') else␣
↪'')
            #ratings.append(card.find('div', class_='rating_text␣
↪rating_text--md').text.strip() if card.find('div', class_='rating_text␣
↪rating_text--md') else '')
            reviews.append(card.find_all('span',␣
↪class_='companyCardWrapper__ActionCount')[0].text.strip() if len(card.
↪find_all('span', class_='companyCardWrapper__ActionCount')) > 0 else '')
            salaries.append(card.find_all('span',␣
↪class_='companyCardWrapper__ActionCount')[1].text.strip() if len(card.
↪find_all('span', class_='companyCardWrapper__ActionCount')) > 1 else '')
            interviews.append(card.find_all('span',␣
↪class_='companyCardWrapper__ActionCount')[2].text.strip() if len(card.
↪find_all('span', class_='companyCardWrapper__ActionCount')) > 2 else '')
```

```
            jobs.append(card.find_all('span',␣
↪class_='companyCardWrapper__ActionCount')[3].text.strip() if len(card.
↪find_all('span', class_='companyCardWrapper__ActionCount')) > 3 else '')
            benefits.append(card.find_all('span',␣
↪class_='companyCardWrapper__ActionCount')[4].text.strip() if len(card.
↪find_all('span', class_='companyCardWrapper__ActionCount')) > 4 else '')
            photos.append(card.find_all('span',␣
↪class_='companyCardWrapper__ActionCount')[5].text.strip() if len(card.
↪find_all('span', class_='companyCardWrapper__ActionCount')) > 5 else '')

        df = pd.DataFrame({
            'name': name,
            'ratings': ratings,
            'reviews': reviews,
            'salaries': salaries,
            'interviews': interviews,
            'jobs': jobs,
            'benefits': benefits,
            'photos': photos
        })

        final = pd.concat([final, df], ignore_index=True)

        print(f"Processed page {j}")
    except Exception as e:
        print(f"An error occurred while processing page {j}: {e}")

print("Data collection completed.")
print(final.head())
```

```
Attempt 1 failed. Retrying in 1 seconds…
Attempt 2 failed. Retrying in 1 seconds…
An error occurred while processing page 0: 404 Client Error: Not Found for url:
https://www.ambitionbox.com/list-of-companies?page=0
Processed page 1
Processed page 2
Processed page 3
Processed page 4
Processed page 5
Data collection completed.
        name ratings reviews salaries interviews   jobs benefits photos
0        TCS     3.9   84.4k     8.6L      10.1k     88    11.8k     87
1  Accenture     3.9   52.7k     5.7L       7.8k  21.9k     7.3k     39
2      Wipro     3.9   50.3k     4.4L       5.5k    567     5.2k     90
3  Cognizant     3.9   47.2k     5.6L       5.4k   1.4k       6k     69
4  Capgemini     3.9     39k     4.3L       4.7k   1.2k     4.1k     41
```

```
[200]: final.head()
```

```
[200]:         name ratings reviews salaries interviews   jobs benefits photos
       0        TCS     3.9   84.4k     8.6L      10.1k     88    11.8k     87
       1  Accenture     3.9   52.7k     5.7L       7.8k  21.9k     7.3k     39
       2      Wipro     3.9   50.3k     4.4L       5.5k    567     5.2k     90
       3  Cognizant     3.9   47.2k     5.6L       5.4k   1.4k       6k     69
       4  Capgemini     3.9     39k     4.3L       4.7k   1.2k     4.1k     41
```

```
[202]: final.shape
```

```
[202]: (100, 8)
```

```
[204]: final.to_csv('ambitionbox_data1.csv', index=False)
       print("Data collection and saving completed.")
```

```
Data collection and saving completed.
```

```
[206]: final.count()
```

```
[206]: name          100
       ratings       100
       reviews       100
       salaries      100
       interviews    100
       jobs          100
       benefits      100
       photos        100
       dtype: int64
```

```
[ ]:
```