A3_section2_Friday_Yusuf

April 25, 2021

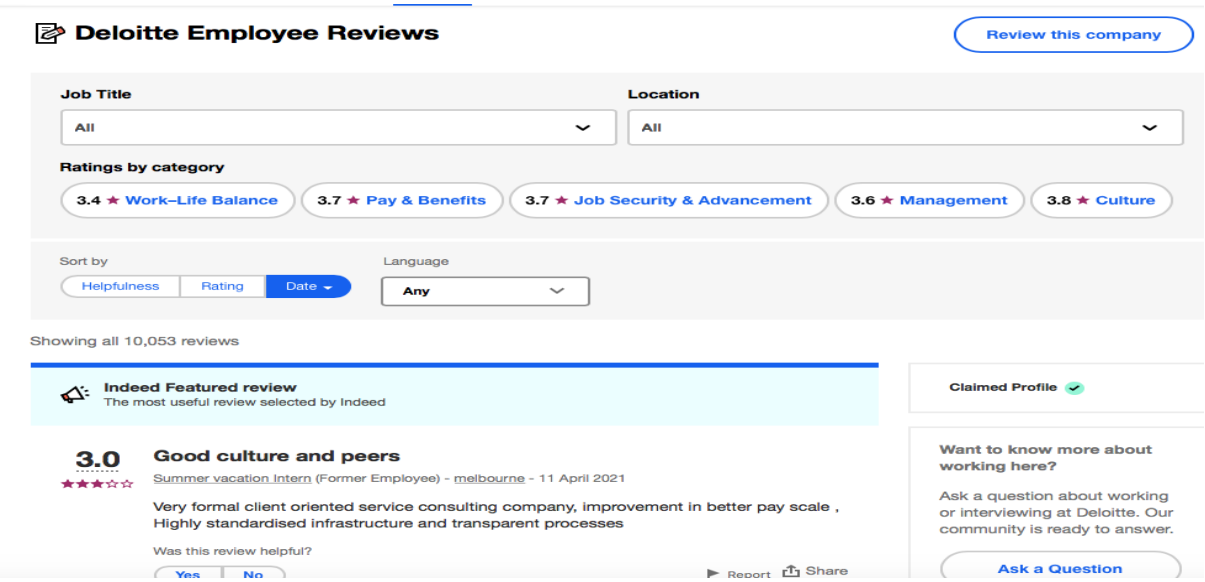
Deloitte's employees' review from past or current employees was web scrapped from au.indeed.com. Indeed is a very popular employment websites with a collection of job listings from thousands of websites, including job boards, staffing firms, associations, and company career websites. It is accessible in over sixty countries and twenty-eight languages. Indeed also provide job reviews where users rank their employers according to a 5-star scale and list pros and cons. Averagely, 10 job are added on indeed per second, it also has a total rating and review of 350 million.

Job reviews are very valuable assets to employees, especially new employees who are trying to join a Company. It helps them decide if it is a company they want to be a part of. Job reviews are important for employees who are trying to understand an employer's culture and brand. It also helps employees know whether their compensation and pay rates are fair and provides knowledge about human resources benefits and company policies. There are few websites to scrap employees' reviews from which, are indeed, seek, glassdoor or google that allowed past and current employees to make a review about the working experience while with the company or organisation.

The employees' review for Deloitte was chosen for this study because it is one of the biggest employers of skilled labour in various field like accounting, audit, environmental science, data science/analysis, accounting. Deloitte is one of the big fours comprising of Deloitte Australia, Ernst & Young (EY), PricewaterhouseCoopers (PwC) and KPMG. It is the dream and career aspiration of many graduates to work for one of the big fours. This is mainly due to their size, reputation, workforce diversity, career progression and work culture. However, no study has particularly analysed these claims from the perspective of previous or current employees. Hence, the reason for the extraction of Deloitte's employees' review to provide more concrete insight into these claims.

Ratings and reviews from April 2011 to June 2020 were collected. This consist of four thousand reviews from employees all over the world. Each review contains the following information: the job title of the reviewer, the date of the review, a rating from 1 to 5 of the company, the location of the job, the text of the review and the pros and cons.

To get familiar with the basic structure of the HTML, an HTML tag was used to identify the information to be scrapped. The website HTML was then accessed via a browser.




Beautiful Soup is a Python library mainly used to parse and extract information from an HTML string. It comes with a variety of HTML parsers that allow the extraction of information even from a badly formatted HTML (Patel 2020). The parsing library *beautiful soup* provides a means to separate HTML and XML to extract the data. Application of the beautifulSoup library is quite easy to perform. The major challenges are carefully examining the HTML structure of the targeted web site and finding the related tags and classes to get the relevant data or information. Retrieving the data and verifying the status were slightly more complex. Beautiful Soup objects are accompanied by a long list of existing methods with very instinctive names, such as FindParents, findPreviousSiblings and findPreviousSibling, for traversing the HTML tags, which may help the user navigate the HTML tree.

Indeed is an open website sources which allows data to be scraped from thier website. Many websites do not appreciate having their data scraped specifically if it contains user identification information for instance scrapping data from LinkedIn, Facebook, twitter, Instagram, e.tc. However, for privacy, the names of the reviewers are not included in the data. Therefore, the data been scraped is not protected under copyright. There was no metadata supplement for the web scrapping, all the scrapping was done from au.indeed.com.

**Content extractor to export the important aspect of the data and/or metadata**

All required libraries were imported into python notebook.

```
In [1]: import pandas as pd
        import numpy as np
        from bs4 import BeautifulSoup4
        import lxml
        import matplotlib.pyplot as plt
        import seaborn as sns
        %matplotlib inline
        import re
        import time
```

```
In [2]: web_url = requests.get('https://au.indeed.com/cmp/Deloitte/reviews?fcountry=ALL&start=', timeout=5)
        print(web_url.text)
```

The request library with a get function was used. This requests the au.indeed.com server for the URL content and stores the web server's response as a variable called web_url. The print function and text were used to view the HTML text files as shown in figure 1.3 below.

```
web_url = requests.get('https://au.indeed.com/cmp/Deloitte/reviews?fcountry=ALL&start=', timeout=5)
print(web_url.text)
```

```
ating" itemscope="" itemType="http://schema.org/Rating"><meta itemProp="ratingValue" content="1"/><button class="cmp-
ReviewRating-text">1.0</button><div class="cmp-RatingStars cmp-RatingStars--sm" aria-label="1.0 out of 5 stars."><div
class="cmp-RatingStars-starsUnfilled"><div class="cmp-RatingStars-starsFilled" style="width:15px"></div></div></div><
div class="cmp-ReviewRating-popup"><div class="cmp-SubRating"><div class="cmp-RatingStars cmp-RatingStars--sm"><div c
lass="cmp-RatingStars-starsUnfilled"><div class="cmp-RatingStars-starsFilled" style="width:15px"></div></div></div><d
iv class="cmp-SubRating-text">Job Work/Life Balance</div></div><div class="cmp-SubRating"><div class="cmp-RatingStars
cmp-RatingStars--sm"><div class="cmp-RatingStars-starsUnfilled"><div class="cmp-RatingStars-starsFilled" style="width
:15px"></div></div></div><div class="cmp-SubRating-text">Salary/Benefits</div></div><div class="cmp-SubRating"><div c
lass="cmp-RatingStars cmp-RatingStars--sm"><div class="cmp-RatingStars-starsUnfilled"><div class="cmp-RatingStars-sta
rsFilled" style="width:15px"></div></div></div><div class="cmp-SubRating-text">Job Security/Advancement</div></div><d
iv class="cmp-SubRating"><div class="cmp-RatingStars cmp-RatingStars--sm"><div class="cmp-RatingStars-starsUnfilled">
<div class="cmp-RatingStars-starsFilled" style="width:15px"></div></div></div><div class="cmp-SubRating-text">Managem
ent</div></div><div class="cmp-SubRating"><div class="cmp-RatingStars cmp-RatingStars--sm"><div class="cmp-RatingStar
s-starsUnfilled"><div class="cmp-RatingStars-starsFilled" style="width:15px"></div></div></div><div class="cmp-SubRat
ing-text">Job Culture</div></div></div></div></div><div class="cmp-Review-content"><div class="cmp-Review-title" data
-testid="title"><a href="/cmp/Deloitte/reviews/toxic?id=b5c6a132881d7357" rel="nofollow" class="cmp-Review-titleLink"
data-tn-element="individualReviewLink" data-tn-link="true" data-testid="titleLink" lang="en-US">Toxic </a></div><div
class="cmp-Review-author"><span class="cmp-ReviewAuthor" itemProp="author" itemscope="" itemType="http://schema.org/P
erson"><meta itemProp="name" content="Consultant"/><a class="cmp-ReviewAuthor-link" rel="nofollow" href="/cmp/Deloitt
e/reviews?fjobtitle=Consultant">Consultant</a> <!-- -->(Former Employee)<!-- --> - <a class="cmp-ReviewAuthor-link" r
```

A function called parse was defined and then BeautifulSoup was used to parse the HTML content of the au.indeed.com site. The lxml was used in case the parser of the HTML was not well formed. Most websites often adjust their HTML script by changing the name parent names or child relationships of the data in the containers. This change can affect your item. In such scenario, the function needs to be adjusted to fit the HTML script figure 1.4.

```
In [5]: def parse(all_url):

            content = BeautifulSoup(all_url.content, 'lxml')
            respository = content.findAll('div',{'class':'cmp-Review-container'})
            df = pd.DataFrame(columns = ['rating', 'rating_title',  'rating_description','rating_pros', 'rating_cons'])

            for item in respository:
                try:
                    rating = item.find('', {'class': 'cmp-ReviewRating-text'}).text.replace('\n', '')
                except:
                    rating = None

                try:
                    rating_title = item.find('div',{'class': 'cmp-Review-title'}).text.replace('\n', '')
                except:
                    rating_title = None

                try:
                    rating_description = item.find('span',{'itemprop': 'reviewBody'}).text.replace('\r', '. ')
                except:
                    rating_description = None

                try:
                    rating_pros = item.find('div',{'class': 'cmp-ReviewProsCons-prosText'}).text.replace('\n', '')
                except:
                    rating_pros = None

                try:
                    rating_cons = item.find('div',{'class': 'cmp-ReviewProsCons-consText'}).text.replace('\n', '')
                except:
                    rating_cons = None
```

figure 1.4

The HTML was inspected thoroughly to identify the root container or parent tag that houses the nested tag which contain the information to scrape. Looking at every tag that houses the required data, such as rating, review title, review description, pros, and cons.

In order to scrape the required information which is contained precisely inside one root element. Examining the HTML below it shows that the <div tag contains all the information for rating, review title, pros, cons while review description is found in <span container. It is also seen with all the other class attribute names. The ''cmp-Review-Container'' stores the review data. Thus function called findall() was used to extract all the div container which had a class attribute of ''cmp-Review-Container''.

An empty pandas dataframe called df was created and all the scraped data was appended. Since all the information needed to be scraped was identified in the container. It is shown that the review rating is housed in <div tag with a class attribute ''cmp-Review-Container-text'' which stored the 3.0 star rating. The tag and class attribute that stored this information was carefully noted and passed on for the python script. The same process was repeated for the remaining data to be extracted. As soon as the appropriate tags were identified, the python code was used to extract the information. This was done using 'for- loop' with 'try' and 'except' block, so that the loop searches for the container to identify the tag using find() function in python. In this case, find() was used instead of the findall() because, only the match data was required. The scraped data were then appended to the empty dataframe called df created earlier.  Up till this stage, only the first page was scrapeed by performing the parse () function which only contained twenty records. In order to scrape the remaining data of all the reviews from the remaining web pages.

To scrape all remaining web pages of the reviews, a new empty dataframe was created to collect all the reviews as it iterates over all the web pages. There are twenty web pages of the review to be scraped, twenty

counter variables were initiated then use a while loop to iterate until the number of reviews is equal to twenty or greater than four thousand shown in figure 15 below.

```
        df = df.append({'rating': rating,
             'rating_title': rating_title,
             'rating_description': rating_description,
             'rating_pros': rating_pros,
             'rating_cons': rating_cons}, ignore_index=True)
    return df
```

```
In [6]: web_url = 'https://au.indeed.com/cmp/Deloitte/reviews?fcountry=ALL&start='

all_pages_df = pd.DataFrame(columns = ['rating', 'rating_title', 'rating_description','rating_pros', 'rating_cons'])

num_reviews = 20
# you can adjust this number on how many reviews you which to scrape
while num_reviews < 4000:

    all_url = web_url + str(num_reviews)

    df_url = requests.get(all_url, timeout=5)

    partial_pages_df = parse(df_url)
    all_pages_df = all_pages_df.append(partial_pages_df, ignore_index=True)

    num_reviews += 20
```

```
In [7]: all_pages_df.to_csv('indeed_data.csv')
```

figure 1.5

The reason for the four thousand review is that most of the time the website time out and the time was set for five minutes to avoid blocking the web crawling. Four thousand reviews will be sufficient for the analysis. Once again get.requst function was used to get the entire HTML, parse function was then applied on the page to get the scraped data into the data frame which then saved into csv file called indeed_data on the computer. To view the structure of the scrapped data, head() function was used to read the first five observation in figure 1.6

```
In [8]: all_pages_df.head()
```

Out[8]:

| | rating | rating_title | rating_description | rating_pros | rating_cons |
|---|---|---|---|---|---|
| 0 | 3.0 | Good culture and peers | Very formal client oriented service consulting... | None | None |
| 1 | 5.0 | Productive and competitive workplace with many... | During a typical day at work we attend any cus... | innovation, leading skills, be part of Deloitt... | Not at all, I love working. |
| 2 | 2.0 | Very stressful | Company is very focused on their profits rathe... | Great benefits | Extremely demanding |
| 3 | 5.0 | Excellent culture | The project I worked in is relaxed. The overal... | None | None |
| 4 | 5.0 | I love Deloitte! | Honestly if you feel like applying to work at ... | Treated equally | N/a |

```
In [9]: all_pages_df.shape
```

```
Out[9]: (4179, 5)
```

figure 1.6


Dineen, B. R., Van Hoye, G., Lievens, F., & Rosokha, L. M. (2019). Third party employment branding: What are its signaling dimensions, mechanisms, and sources?. In *Research in personnel and human resources management*. Emerald Publishing Limited.

Mitchell, R. (2018). *Web scraping with Python: Collecting more data from the modern web*. " O'Reilly Media, Inc.".

Mobley, W. H., Griffeth, R. W., Hand, H. H., & Meglino, B. M. (1979). Review and conceptual analysis of the employee turnover process. *Psychological bulletin*, *86*(3), 493.

Patel, J. M. (2020). Web Scraping in Python Using Beautiful Soup Library. In *Getting Structured Data from the Internet* (pp. 31-84). Apress, Berkeley, CA.

Reference

Stamolampros, P., Korfiatis, N., Chalvatzis, K., & Buhalis, D. (2019). Job satisfaction and employee turnover determinants in high contact services: Insights from Employees' Online reviews. *Tourism Management*, *75*, 130-147.

Appendix

```
    importing Libraries
import pandas as pd
 import numpy as np
import matplotlib.pyplot as plt
 import seaborn as sns
 %matplotlib inline
 import re
import time from datetime import datetime
im-port matplotlib.dates as mdates
 import matplotlib.ticker as ticker from urllib.request
import urlopen from bs4
 import BeautifulSoup
import requests
import lxml # web scrapping
web_url = requests.get('https://au.indeed.com/cmp/Deloitte/reviews?fcountry=ALL&start=',
timeout=5) print(web_url.text) # creating a parse function def parse(all_url):

content = BeautifulSoup(all_url.content, 'lxml')

respository = content.findAll('div',{'class':'cmp-Review-container'})

df = pd.DataFrame(columns = ['rating', 'rating_title',          'rating_description','rating_pros', 'r

for item in respository:
```

```
try:
    rating = item.find('', {'class': 'cmp-ReviewRating-text'}).text.replace('\n', '')
except:
    rating = None
try:
```

```
        rating_title = item.find('div',{'class': 'cmp-Review-title'}).text.replace('\n', '')
    except:
        rating_title = None
    try:
        rating_description = item.find('span',{'itemprop': 'reviewBody'}).text.replace('\r', ' except:
        rating_description = None
    try:
        rating_pros = item.find('div',{'class': 'cmp-ReviewProsCons-prosText'}).text.replace(' except:
        rating_pros = None
    try:
        rating_cons = item.find('div',{'class': 'cmp-ReviewProsCons-consText'}).text.replace(' except:
        rating_cons = None
    df = df.append({'rating': rating,
                    'rating_title': rating_title,
                    'rating_description': rating_description,
                    'rating_pros': rating_pros,
```

```
            'rating_cons': rating_cons}, ignore_index=True)

return df

web_url = 'https://au.indeed.com/cmp/Deloitte/reviews?fcountry=ALL&start='

all_pages_df = pd.DataFrame(columns = [‘rating’, ‘rating_title’, ‘rat-
ing_description’,‘rating_pros’, ‘rating_cons’])

num_reviews = 20 # you can adjust this number on how many reviews you which to scrape
while num_reviews < 4000:

all_url = web_url + str(num_reviews)

df_url = requests.get(all_url, timeout=5)

partial_pages_df = parse(df_url)

all_pages_df = all_pages_df.append(partial_pages_df, ignore_index=True)

num_reviews += 20

all_pages_df.to_csv(‘indeed_data.csv’) all_pages_df.head()
```