CODE DOCUMENTATION

Applied Research Project

INDEED.CO.UK

ZAID ABDEEN

TABLE OF CONTENTS

PHASE I
DATA COLLECTION

The main objective of phase I is to systematically extract update data on job postings in the London area. Essentially, a database of job posts will be continuously updated. The source of the data is chosen to be www.indeed.co.uk.

Web scraping is implemented to achieve the objective using **Python 3.6.7** and **Scrapy 1.7**

https://docs.python.org/3.6/

https://docs.scrapy.org/en/latest/

SEARCH PARAMETERS

The first obstacle was to construct a set of search parameters to capture as much data as possible on a weekly basis. The target website limits searches to a maximum of 1000 results per query, therefore, constructing multiple searches would be the only way to solve the issue. Using a manually constructed list of key words and salary ranges in increments of £1000. Using salary proves to be effective as the website estimates a salary range for job posts without wage information.

*refer to the **url-creator.ipynb** file for the full code.*

To construct the URLs using Python:

'https://www.indeed.co.uk/jobs?q=consulting&l=London,+Greater+London&fromage=7&sort=date&limit=50&filter=0&radius=25&start=0'

the main components of the URL are as follows.

1. *- The raw search query is inputted here after 'q='*
2. *- The location is inputted here after 'l='*
3. *- The maximum for how many days ago the job was posted is placed here as a raw integer. We have chosen 7 days to avoid losing data.*

The code below is a user-defined function to create a list of URLs:

1. *Initiate an empty list*
2. *Splitting the URL string into to two components*
3. *Iterating over the list using a for loop*
4. *Creating a new string variable 'x' which is a string that contains the two original components with the new name in between.*
5. *Append into target list*
6. *Returning a list of URLs in the form of strings.*

```
def createUrl(lst):
    target=[]
    a = 'https://www.indeed.co.uk/jobs?q='
    b = '&l=London%2C+Greater+London&radius=25&fromage=any&limit=50&sort=date'
    for i in lst:
        x = str(a + str(i) + b)
        target.append(x)
    return target
```

The next step is to save the URL list into a format readable by the scraper. The pickle module is utilized to create a file with the desired format as follows:

```
import pickle

with open(PData + "all_urls.txt", "wb") as fp:   #Pickling
    pickle.dump(x, fp)
```

IMPLEMENTING SCRAPY

The first step to building the scraper is to obtain the XPATHs for our required datapoints. This is done using Google Chromes developer console.

To run the scraper, open a terminal window in the ‘indeed-scraper-master’ folder and run the following command:

scrapy crawl jobhunt -o /raw/filename.csv

<u>Main components of the scraper</u>:

I. *jobhunt.py - the CrawlSpider (main file)*
II. *items.py – pipeline to store the data*
III. *settings.py – tweak scraper settings*

1. jobhunt.py

This script is the main piece of the scraper. It utilizes two-way crawling to obtain data from the target website. In other words, it crawls through the search pages of job postings, extracts each individual URLs for each post and continues to scrape the extracted links.

The imports from scrapy are used to build the scraper

LinkExtractor – used for two-way crawling
CrawlSpider – main class of spider used
Rule – used to set parameters for two-way crawling
ItemLoader – append scraped items through items.py file
IndeedItem – refers to a class in the items.py file
MapCompose, Join – basic cleaning

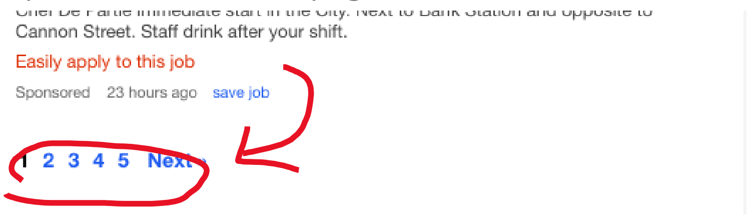
The file of URLs is loaded using pickle.

```
import datetime
from scrapy.linkextractors import LinkExtractor
from scrapy.spiders import CrawlSpider, Rule
from scrapy.loader import ItemLoader
from scrapy.loader.processors import MapCompose, Join
import pickle
from indeedv1.items import IndeedItem

with open("full_urls.txt", "rb") as fp:   # Unpickling
    all_urls = pickle.load(fp)
```

JobhuntSpider is the main class which implements the scraper. The Rules class limits the search for links by restricting the XPATHs.

Link extractor is first initiated as “"//div[@class='pagination']/..” which represents the ‘next’ pages on the website.


Chef De Partie immediate start in the City. Next to Bank Station and opposite to
Cannon Street. Staff drink after your shift.

Easily apply to this job

Sponsored 23 hours ago save job

1 2 3 4 5 Next »

The next XPATH restricted would be =("//*[@class='title']/a" - represents the URL of each job listing.

The final part of the code, `callback='parse_item'` is set to the second Rule as to tell the scraper when to initiate the full crawl.

```
class JobhuntSpider(CrawlSpider):
    name = 'jobhunt'
    allowed_domains = ['indeed.co.uk']

    start_urls = all_urls


    rules = (Rule(LinkExtractor(restrict_xpaths=("//div[@class='pagination']/a[1]",
…                                              "//div[@class='pagination']/a[21]"))
),
             # I use all possible links for pagination  (1-21)
             Rule(LinkExtractor(restrict_xpaths=("//*[@class='title']/a",)),
                  callback='parse_item')
             )
```

This part of the code initiates the ItemLoader class, which uses xpaths to extract data from the response page.

For each ‘z.add_xpath’, the first input is the Field name which should be referenced from items.py, and the second input is the xpath of the relevant datapoint extracted from the website.

Some basic string cleaning is implemented, as with replacing irrelevant punctuation in some fields and slightly cleaning the textual data from the job summaries.

The URL and date scraped are also logged for housekeeping.

```
def parse_item(self, response):
    z = ItemLoader(item=IndeedItem(), response=response)
    z.add_xpath('title', '//div/h3/text()')
    z.add_xpath('company', '//div[@class="icl-u-lg-mr--sm icl-u-xs-mr--
xs"]/text()[1]',
                MapCompose(lambda i: i.replace('-', '')))
```

```python
    z.add_xpath('company', '//div[@class="icl-u-lg-mr--sm icl-u-xs-mr--xs"]/a/text()',
                MapCompose(lambda i: i.replace(',', '')))
    z.add_xpath('salary', '//div/span[@class="jobsearch-JobMetadataHeader-iconLabel"]/text()')
    z.add_xpath('summary', '//li/text()|//p/text()',
                MapCompose(str.strip), Join())
    z.add_xpath('summary', '//*[@id="jobDescriptionText"]/text()')
    z.add_xpath('posted', '//div[@class="jobsearch-JobMetadataFooter"]/text()',
                MapCompose(lambda i: i.replace('-', '')))

    z.add_value('url', response.url)
    z.add_value('date', str(datetime.datetime.now()))

    yield z.load_item()
```

2. items.py

The following variables are ‘Field’ values which represent the pipeline for storing the scraped data.

```python
import scrapy

class IndeedItem(scrapy.Item):
    title = scrapy.Field()
    company = scrapy.Field()
    salary = scrapy.Field()
    summary = scrapy.Field()
    posted = scrapy.Field()
    link = scrapy.Field()
    # Housekeeping fields
    url = scrapy.Field()
    date = scrapy.Field()

    pass
```

3. settings.py

There is not much to change from the default settings template, other than changing the USER AGENT to a desired name, setting ROBOTSTXT_OBEY to FALSE and tweaking the number of seconds between requests if needed with the DOWNLOAD_DELAY option. To change any settings, remove the hashtag prior to the relevant piece of code.

PHASE II DATA PROCESSING

MERGING

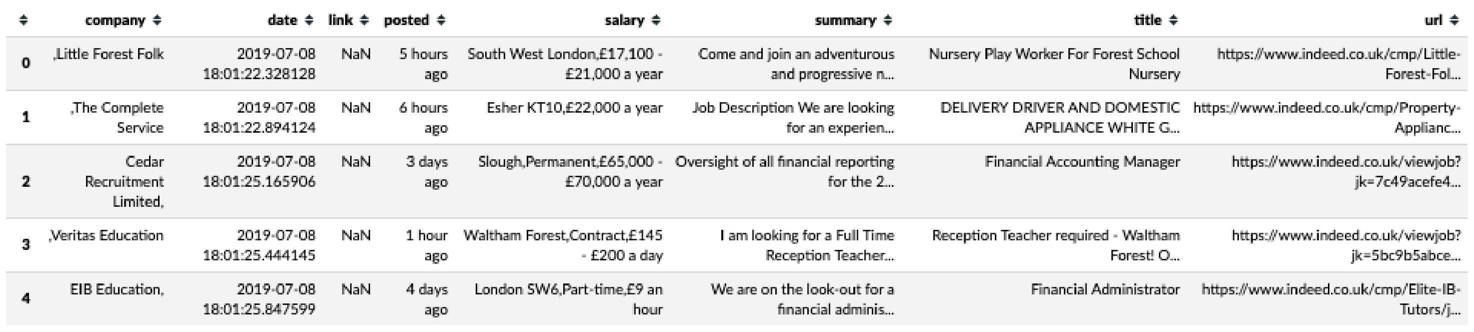
refer to 'merge-notebook.ipynb' for the full code.

The goal of this part of phase II is to merge the weekly scraped datafiles into monthly datasets. In addition, duplicates from the dataset are removed.

Pandas is used to merge the files with the 'concat' function. Following this, duplicates are dropped using the 'drop_duplicates' function in two stages:

i. by URL only
ii. by Title, Date posted and Summary

Below is a snippet of the dataset following the merge. It is apparent that more cleaning is necessary, especially in the 'salary' column.



	company	date	link	posted	salary	summary	title	url
0	,Little Forest Folk	2019-07-08 18:01:22.328128	NaN	5 hours ago	South West London,£17,100 - £21,000 a year	Come and join an adventurous and progressive n...	Nursery Play Worker For Forest School Nursery	https://www.indeed.co.uk/cmp/Little-Forest-Fol...
1	,The Complete Service	2019-07-08 18:01:22.894124	NaN	6 hours ago	Esher KT10,£22,000 a year	Job Description We are looking for an experien...	DELIVERY DRIVER AND DOMESTIC APPLIANCE WHITE G...	https://www.indeed.co.uk/cmp/Property-Applianc...
2	Cedar Recruitment Limited,	2019-07-08 18:01:25.165906	NaN	3 days ago	Slough,Permanent,£65,000 - £70,000 a year	Oversight of all financial reporting for the 2...	Financial Accounting Manager	https://www.indeed.co.uk/viewjob?jk=7c49acefe4...
3	,Veritas Education	2019-07-08 18:01:25.444145	NaN	1 hour ago	Waltham Forest,Contract,£145 - £200 a day	I am looking for a Full Time Reception Teacher...	Reception Teacher required - Waltham Forest! O...	https://www.indeed.co.uk/viewjob?jk=5bc9b5abce...
4	EIB Education,	2019-07-08 18:01:25.847599	NaN	4 days ago	London SW6,Part-time,£9 an hour	We are on the look-out for a financial adminis...	Financial Administrator	https://www.indeed.co.uk/cmp/Elite-IB-Tutors/j...

CLEANING

refer to 'fullclean-notebook.ipynb' for full code.

The second round of cleaning has multiple objectives:

a. *splitting and cleaning any merged columns*
b. *cleaning and formatting all textual data for analysis*
c. *cleaning and extracting salary data*
d. *extracting a 'days_ago' value in order to date the job listings accurately*

The below code:

1. *split the salary column to create location, contract type and salary*
2. *remove ',' from the company column, as it appears frequently using str.replace()*
3. *split the title into title and subtitle and apply the 'title' method for string formatting*

```
## cleaning the salary columns first
new = df["salary"].str.split(",£", n = 1, expand = True)
# making separate location column from new data frame
df["location"]= new[0]
# making separate salary column from new data frame
df.drop(columns =["salary"], inplace = True)
df["Salary"]= new[1]
new = df["location"].str.split(",", n=1, expand = True)
df["location"] = new[0]
df["contract"] = new[1]
company=df['company'].str.replace(',', '')
df['company']=company
title = df['title'].str.split('-')
df['title']=title.str[0]
df['sub_title'] = title.str[1]
```

The next part of the cleaning does, using self defined functions:

a. *extract a number for 'days_ago' using an if statement, it returns NAN for anything that does not have 'days ago' in the value which will later be appended to 0. This is due to values such as '4 hours ago'.*
b. *extract a min and max range for salary. The 'min_max_yearly_wage' function splits the salary column based on the '-' value which is consistent*

across the dataset. It returns a tuple indicating both salaries.

c. *extract a salary value when only one, similar to the previous functions, when only one salary is present.*

d. *the median salary is obtained from the min-max range as a benchmark.*

```python
def date_ago(val):
    if pd.isnull(val):
        return np.nan
    elif ' days ago' in val:
        i=val.split(' days ago')[0]
        if ' ' in i:
            mn=i.split()[0]
            return mn

def min_max_yearly_wage(val):
    if pd.isnull(val):
        return np.nan
    elif ' a year' in val:
        i=val.split(' a year')[0].replace('£',' ').strip()
        if '-' in i:
            mn=i.split('-')[0]
            mx=i.split('-')[1]

            try:
                mn = float(mn.replace(",", "").strip())
                mx = float(mx.replace(",", "").strip())
            except:
                return np.nan
            return mn, mx

def salary_yearly_wage(val):
    if pd.isnull(val):
        return np.nan
    elif ' a year' in val:
        i=val.split(' a year')[0].replace('£', ' ').strip()
        if '-' in i:
            return np.nan
        else:
            mn = float(i.replace(",","").strip())
        return mn

df = df.assign(yearly_range=df['Salary'].apply(min_max_yearly_wage),
               an_salary = df['Salary'].apply(salary_yearly_wage),
              days_ago = df['posted'].apply(date_ago))

df = df.assign(
    median_yearly_salary = df['yearly_range'].apply(
        lambda r: (r[0] + r[1]) / 2 if pd.notnull(r) else r
    ),
)
```

DATE MANIPULATION

refer to datetime-notebook-p3.ipynb for full code.

This section of the workflow refers to getting a more accurate measure for the exact day the job was posted. In the previous cleaning steps, a ‘days_ago’ column was created as a raw integer from the ‘posted’. To extract the date, a new column created called ‘date_posted’ by subtracting a timedelta version of the datetime from the days ago number.

```
df['date'] = pd.to_datetime(df['date'])
df['days_ago'] = df['days_ago'].map(dt.timedelta)
df['date_posted'] = df['date'] - df['days_ago']
```

The exported dataframe is saved to ‘pipeline3’.

EXTRACTING SKILLS

refer to skill-notebook.ipynb for full code.

The chosen strategy to extract skills from the summaries is to superimpose an external list of skills, obtained from:
https://github.com/varadchoudhari/LinkedIn-Skills-Crawler/blob/master/output/all_skills.txt

A package called FlashText, in particular the KeywordProcessor() module, as it results in being much quicker than using traditional python or pandas methods.

https://github.com/vi3k6i5/flashtext

The first part of the workflow is to create a list of strings from the imported list of strings. A user define function uses ‘map’ to change the data type.

```

skills = skills[0].tolist()

def change_list(l, dtype=str):
    return list(map(dtype, l))

skills = change_list(skills, str)
```

Secondly, a list comprehension is imposed onto the skills list to manually remove any unwanted words.

```python
skills = [e for e in skills if e not in ('ONE', 'Range', 'Scheme')]
```

Defining KeywordProcessor() and adding the keywords from the list, in addition to any extra words not present.

```python
kw = KeywordProcessor()
kw.add_keywords_from_list(skills)
kw.add_keyword('TensorFlow')
```

user defined function implementing ‘extract_keywords’ function on the dataframes ‘summary’ column, returning a unique list of skills for every row.

```python
def get_skills_list(df):
    some_text = df['summary']
    lowered = some_text.lower()
    keywords = kw.extract_keywords(df['summary'])
    keywords = list(dict.fromkeys(keywords))
    keywords_string = keywords
    return keywords_string
```

It is recommended to implement the skill extraction directly prior to any analysis in the same workflow to avoid losing the list data structure, therefore, an export is not included in this final section.

FINAL OUTPUT

Below is a sample screenshot of the final dataset following the processing workflow.

The variables are described as follows:

a. *Title – the main job title*
b. *Sub-title – if a subtitle exists, it is indicated by a ‘-‘, which is now in this column, else the value is NA*
c. *Company – company name*
d. *Summary – summary of the job post*

e. Salary_merged – the final output for salary, using the median of the min_max range or the single value present when applicable

f. Contract – contract type if present, else NA

g. Location – location value

h. Yearly_range – tuple of min and max of salary range

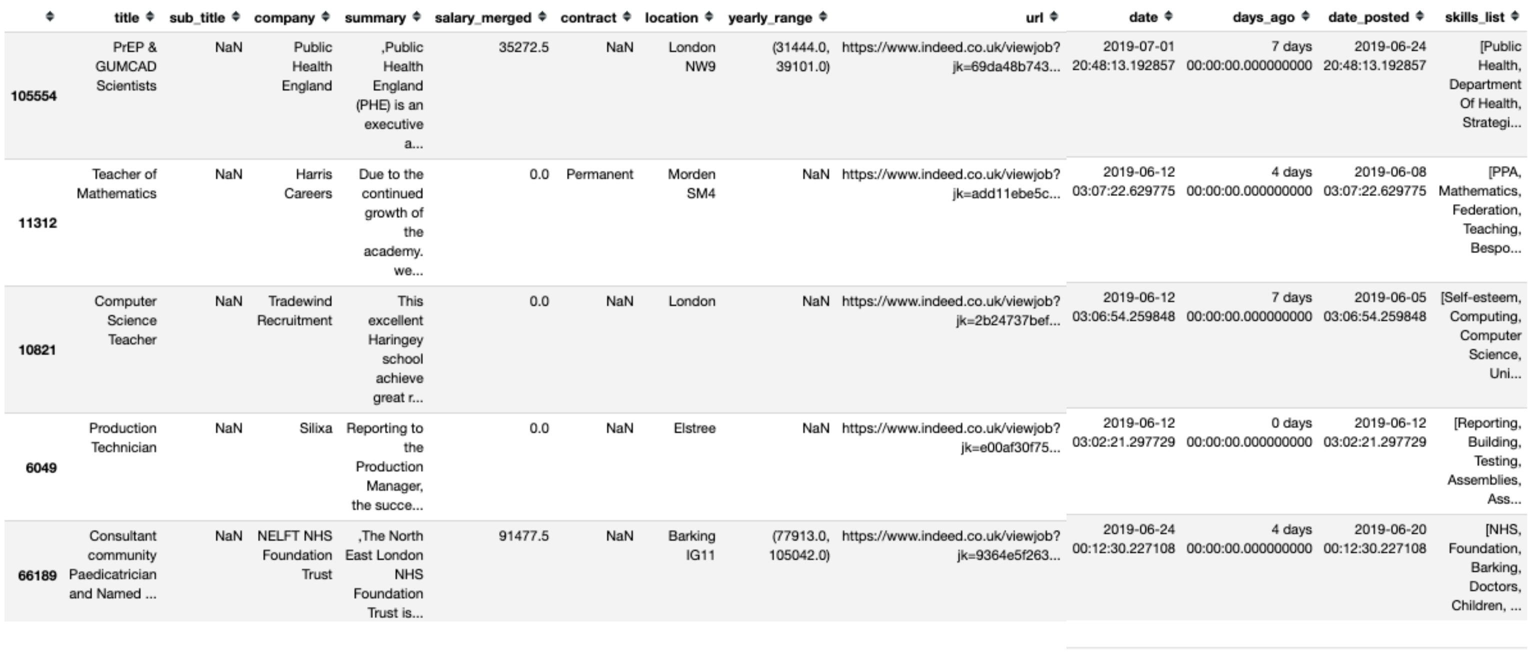
i. url – original url of job listing scraped

j. date – date scraped

k. days_ago – days ago posted from day scraped

l. date_posted – date job posted, following manipulation of date and days ago

m. skills_list – list of skills present in each job post summary based on the workflow in phase II



	title	sub_title	company	summary	salary_merged	contract	location	yearly_range	url	date	days_ago	date_posted	skills_list
105554	PrEP & GUMCAD Scientists	NaN	Public Health England	,Public Health England (PHE) is an executive a...	35272.5	NaN	London NW9	(31444.0, 39101.0)	https://www.indeed.co.uk/viewjob?jk=69da48b743...	2019-07-01 20:48:13.192857	7 days 00:00:00.000000000	2019-06-24 20:48:13.192857	[Public Health, Department Of Health, Strategi...
11312	Teacher of Mathematics	NaN	Harris Careers	Due to the continued growth of the academy. we...	0.0	Permanent	Morden SM4	NaN	https://www.indeed.co.uk/viewjob?jk=add11ebe5c...	2019-06-12 03:07:22.629775	4 days 00:00:00.000000000	2019-06-08 03:07:22.629775	[PPA, Mathematics, Federation, Teaching, Bespo...
10821	Computer Science Teacher	NaN	Tradewind Recruitment	This excellent Haringey school achieve great r...	0.0	NaN	London	NaN	https://www.indeed.co.uk/viewjob?jk=2b24737bef...	2019-06-12 03:06:54.259848	7 days 00:00:00.000000000	2019-06-05 03:06:54.259848	[Self-esteem, Computing, Computer Science, Uni...
6049	Production Technician	NaN	Silixa	Reporting to the Production Manager, the succe...	0.0	NaN	Elstree	NaN	https://www.indeed.co.uk/viewjob?jk=e00af30f75...	2019-06-12 03:02:21.297729	0 days 00:00:00.000000000	2019-06-12 03:02:21.297729	[Reporting, Building, Testing, Assemblies, Ass...
66189	Consultant community Paedicatrician and Named ...	NaN	NELFT NHS Foundation Trust	,The North East London NHS Foundation Trust is...	91477.5	NaN	Barking IG11	(77913.0, 105042.0)	https://www.indeed.co.uk/viewjob?jk=9364e5f263...	2019-06-24 00:12:30.227108	4 days 00:00:00.000000000	2019-06-20 00:12:30.227108	[NHS, Foundation, Barking, Doctors, Children, ...