Step 1 Solution

April 4, 2020

Step 1: Extracting Raw Text from HTML

Overview of project goals: In this project, we aim to take online job postings and understand the data science job market by looking at the major themes (groups) of job postings using their skill requirements. We also want to optimize our resume to better fit the market by finding skills that are in job postings but not in our resume.

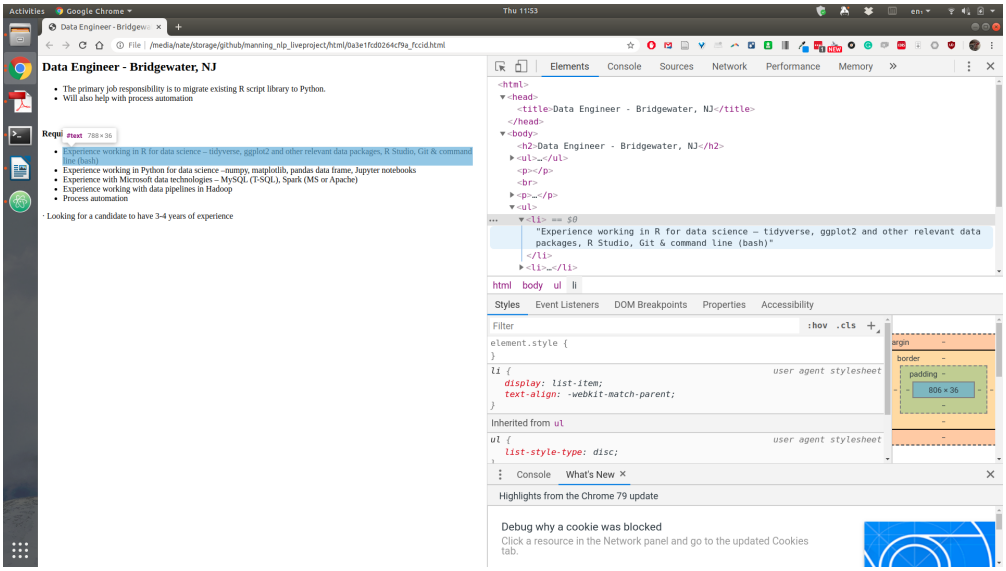
A high-level overview of the project goals and steps is:

- Collect HTML job postings and extract relevant sections for the next steps (primarily the HTML body and skill requirements – you should find where the skill requirements are from looking at some HTML pages).
- Find which jobs are most similar to our resume based on the full text of our resume and full texts of the job posts. Future steps in the project will use this group of most similar job postings.
- Understand the major themes of job postings via the skill requirements of groups of jobs.
- Find which skills our resume is missing compared with what organizations are looking for (e.g. the skill requirements from job postings).

Objective: Extract relevant sections of the job posting webpages for similarity comparison with our resume and analysis of skills missing from our resume.

Workflow:

1. Obtain the HTML job posting pages from the GitHub repository. Examine some HTML pages manually by opening them in a browser and inspecting the HTML elements (e.g. ctrl+shift+i in many browsers or right click on the page and choose "inspect"). Determine which sections should be extracted and stored.
    - Hint: Where do most of the required skills end up showing up? Which HTML tags typically contain the skills requirements for jobs?
2. Load all webpages into Python as strings using any method. Be sure to check for exact duplicate HTML pages and remove them. It may be easier to remove duplicate entries once you have the data in a pandas DataFrame.
3. Store the webpage sections identified in step 1 (e.g. the title, body, and bullet points) into a pandas DataFrame with sensible column names.
    - Examine some of the data in the DataFrame to make sure the data extraction worked as we expect.



inspecting a job posting

4. Filter the jobs to only include data science jobs. Since we already have our data in a pandas DataFrame, this can be done using pandas.
5. Save the DataFrame to disk so we can load it at a later time for future parts of the project.

1. Inspecting HTML pages

After opening some job postings in a browser and inspecting the HTML of the page (with ctrl+shift+i or right-clicking the page and choosing 'inspect'), we can see that typically, required skills are contained in bullet points. Sometimes other information is contained in bullet points as well. With more advanced webpage parsing (e.g. searching for the 'Required Skills' string or something similar) we could potentially limit our bullet points to only those with required skills. However, we won't take those extra steps here since that level of data cleaning can be an entire project on its own.

The bullet points with skill requirements are 'ul' HTML tags with 'li' tags inside them.

We can also see the title of the posting is contained within a 'title' html tag. The entire text of the post is contained within the 'body' tag. So our plan is to separately store the 'body', 'title', and 'li' tags in separate pandas DataFrame columns.

The HTML file used here was 0a3e1fcd0264cf9a_fccid.html, which came up first in the file browser when the files were sorted by name.

2. Load webpages into Python

Next, we get the list of files to be opened and see how many there are. Then we read in the files into a list. We will look at some of the data we read in to make sure it looks ok.

Usually it's best to group your library imports in a cell or two at the beginning of your notebook, to keep it organized. Imports should be grouped as per PEP8: https://www.python.org/dev/peps/pep-0008/#imports. We'll import our libraries needed for each section (step) at the top of that section.

```
In [1]: import glob

        import pandas as pd
        from bs4 import BeautifulSoup as bs

In [2]: # get a list of the files in the html directory
        files = glob.glob('html_job_postings//*.html')

In [3]: len(files)

Out[3]: 1337

In [4]: # load all HTML pages as text into a list -- one entry per HTML page
        html_content = []
        for file in files:
            with open(file, 'r') as f:
                html_content.append(f.read())

In [5]: # inspect the first entry to make sure it looks ok
        html_content[0]

Out[5]: '<html><head><title>Operations Insights and Impact Analyst, YouTube - San Bruno, CA</ti
```

3. Store webpage sections into a pandas DataFrame

Now we are going to parse the HTML into sections – Title, Body, and bullets points ('li' tags which typically hold required skills) and store these in a DataFrame. These will be stored in a dictionary with lists as values, and then converted to a DataFrame.

First, we prototype the code to extract the content from the page. Then we'll incorporate it in a function which takes in all html pages and returns a DataFrame

```
In [6]: # We use a dictionary to store data, which can easily be converted to a pandas DataFra
        html_sections = []
        html_dict = {}
        for key in ['title', 'body', 'bullets']:
            html_dict[key] = []

        # use the first page for prototyping the code
        first_page = html_content[0]
        # converting the text to a beautifulsoup object makes it easily searchable
        soup = bs(first_page, 'lxml')
        title = soup.find('title').text
        body = soup.find('body').text
        bullets = soup.find_all('li')
        html_dict['title'].append(title)
        html_dict['body'].append(body)
        # remove extra leading and trailing whitespace with strip()
        html_dict['bullets'].append([b.text.strip() for b in bullets])

        df = pd.DataFrame(data=html_dict)
```

```
In [7]: df.head()
```

```
Out[7]:                                                  title  \
        0  Operations Insights and Impact Analyst, YouTub...

                                                          body  \
        0  Operations Insights and Impact Analyst, YouTub...

                                                       bullets
        0  [Bachelor's degree in Data Science, Economics,...
```

It's best to always write some documentation for your functions as shown here in the triple quotes.

```
In [8]: def get_html_content(html_pages):
            """
            Extracts title, body, and list items (bullets) from HTML job postigs.
            Returns a pandas dataframe with separate columns for title, body, and bullet items
            """
            html_sections = []
            html_dict = {}
            for key in ['title', 'body', 'bullets']:
                html_dict[key] = []

            for html in html_content:
                soup = bs(html, 'lxml')
                title = soup.find('title').text
                body = soup.find('body').text
                bullets = soup.find_all('li')
                html_dict['title'].append(title)
                html_dict['body'].append(body)
                # remove extra leading and trailing whitespace with strip()
                html_dict['bullets'].append([b.text.strip() for b in bullets])

            df = pd.DataFrame(html_dict)

            return df
```

```
In [9]: df = get_html_content(html_content)
```

```
In [10]: df.head()
```

```
Out[10]:                                                  title  \
         0  Operations Insights and Impact Analyst, YouTub...
         1  Mathematical and Statistical Scientist* - Char...
         2                    Data Science Lead - Chicago, IL
         3      Software Developer 3 - Redwood City, CA 94065
         4                         Data Scientist - Orange, CA
```

```
                                                  body  \
0  Operations Insights and Impact Analyst, YouTub...
1  Mathematical and Statistical Scientist* - Char...
2  Data Science Lead - Chicago, IL\nWho we are:\n...
3  Software Developer 3 - Redwood City, CA 94065\...
4  Data Scientist - Orange, CA\nRole Summary / Pu...

                                             bullets
0  [Bachelor's degree in Data Science, Economics,...
1  [BS in Statistics, Mathematics, or a related f...
2  [Lead the development, implementation and opti...
3  [3+ years of Java, Go, or Python frontend and/...
4  [PhD in Industrial Engineering, computer scien...
```

```
In [11]: df.shape
```

```
Out[11]: (1337, 3)
```

At this point, you may want to do some more data exploration and cleaning. Let's see how many titles actually have 'data scientist' or 'data science' in them. The str.contains method uses regex.

```
In [12]: df[df['title'].str.contains('(data scientist)|(data science)', case=False)].shape
```

```
/home/nate/anaconda3/envs/job-posting-nlp/lib/python3.7/site-packages/pandas/core/strings.py:18
  return func(self, *args, **kwargs)
```

```
Out[12]: (497, 3)
```

Not that many! However sometimes data scientist jobs might have different titles besides data scientist, and there are many similar job titles, like data engineer, machine learning engineer, decision scientist, etc.

4. Filter the jobs to only include data science jobs

Since we are searching for data science jobs, we will take only those jobs with 'data science' or 'data scientist' in the title. We use .copy() to make a copy of our filtered dataframe, sicnce we will get a 'SettingWithCopyWarning' otherwise.

```
In [13]: ds_df = df[df['title'].str.contains('(data scientist)|(data science)', case=False)].co
```

```
In [14]: ds_df.head()
```

```
Out[14]:                                                title  \
2                     Data Science Lead - Chicago, IL
4                         Data Scientist - Orange, CA
6  Manager / Senior Manager - NLP Data Scientist ...
8                   Data Scientist I - Bend, OR 97701
```

```
9                      Data Scientist - Columbus, OH

                                                  body  \
2  Data Science Lead - Chicago, IL\nWho we are:\n...
4  Data Scientist - Orange, CA\nRole Summary / Pu...
6  Manager / Senior Manager - NLP Data Scientist ...
8  Data Scientist I - Bend, OR 97701\nPhenomenal ...
9  Data Scientist - Columbus, OH\nDescription\nAs...

                                                bullets
2  [Lead the development, implementation and opti...
4  [PhD in Industrial Engineering, computer scien...
6  [Build NLP models to deeply understand custome...
8  [Execute discovery processes of low to average...
9  [Applying appropriate techniques, e.g. explora...
```

Using pandas DataFrames enables several convenience functions, such as drop_duplicates. However, we can't drop dupes with columns of lists, so we have to convert the bullets column (which are lists) to tuples first.

```
In [15]: ds_df['bullets'] = ds_df['bullets'].apply(tuple, 1)

In [16]: ds_df.drop_duplicates(inplace=True)
         ds_df.shape

Out[16]: (492, 3)
```

As we can see, there are a few duplicates in there which we have now removed. Now we have our parsed text data ready for more analysis!

5. Save the DataFrame to disk so we can load it at a later time for future parts of the project.

The last step is to save the DataFrame to disk for future steps. There are several methods for storing data, but we will keep it simple with a pickle file (even though it may not have the best performance for many use cases). Using a CSV file would normally work, but since our 'bullets' column is a list, we need to do some extra steps in order to get a list back when reading a CSV.

```
In [17]: ds_df.to_pickle('step1_df.pk')
```