Last Updated: June 1, 2021

data_formatter.py

1.0 Introduction

Data_formatter.py which contains the DataFormatter class is the part of the stack where all the data/key-word extraction on the job posts happen. It's in between the web scraper and the database upload portion in the pipeline: it takes the web scraped job posts (which are passed in batches) to preprocess it to extract the keywords it searches for, then pass all of that as a dictionary to dbupload.py to be uploaded to the database.

2.0 Input Description

The Web Scraper part of the tech stack will be calling data formatter for every page it's scraping i.e. it will pass job post descriptions to DataFormatter in batches (a linkedin page).

Each batch will be in the form of a .txt file (right now is named "results.txt") and will be passed into DataFormatter.preprocessing() then to DataFormatter.data_extraction():

```
        # print to a results file
        print("BREAK{}".format(job_id), file = self.output_file)
        print(result, file = self.output_file)
    self.output_file.seek(0)
    self.formatter.preprocessing(self.output_file)
    output = self.formatter.data_extraction(job_name, do_upload)
```

Selenium_Navigation.py

The text file:
The text file input should be a file containing job post(s) descriptions with each separated by a line containing "BREAK[job

id]”, the job id is necessary for the output for the database upload part. There’s an example file called sample_jobs.txt

3.0 Output Description

The code will return a long dictionary of the result of data extraction.

```
template = {'4385545':{'skills':['javascript, python, angular'],

'industries':['healthcare','telecommunication'],
                        'degree_level':['bachelor','master'],
                  'languages':['javascript','python']
                        'seniority':'Senior',
                        'yoe':5,
                        'degree_title':['ce','cs']
                       },
            '5345465':{
                ....
            }
           }
```

'skills': used a list() data structure (internally it’s on top of a set() instead to deal with repeats)

'industries': used a list() data structure (internally it’s on top of a set() instead to deal with repeats) The standardized industries can be found in linkedin_industries.txt (146 industries)

'degree_level': used a list() data structure (internally it’s on top of a set() instead to deal with repeats). 'a' - associates 'b'-bachelors 'm' - masters 'p' - phd

'seniority': just a string from one of the standardized LinkedIn terms: ['Internship','Entry level','Associate','Mid-Senior level','Director','Executive']
And -1 when don't have this seniority attribute or have something not one of the above terms (shouldn't be possible, unless LinkedIn is inconsistent which does happen surprisingly)

yoe:
Just an int. Returns -1 when can't find this attribute.

degree_title:
Return list. So far the standardized terms are computer science, computer engineering, electrical engineering, applied math, physics, statistics, bioinformatics/comp-bio, etc. haven't expanded to anything else, since I need to add more error handling for these new ones as these terms might not be used in context of education level.

4.0 Component level Designs

Preprocessing -
1. Abbreviation - B.A. B.S. - after my punctuation remover, it becomes B A B S. Making it miss stuff, handled this by just adding 'b a' 'b s' to the list
2. Didn’t use Tokenization (every word is a separate string) - Miss key words with two or more words like "React Native", so just stopped using tokenization and used "if 'react native' in text'
3. Double spaces - not completely sure yet but I think it's from the punctuation remover, but the regex that removes multiple consecutive spaces can deal with that

5.0 Testing (test_formatter.py)

The unit testing and integration testing for DataFormatter is test_formatter.py. It has unit testing for every single function in DataFormatter, usually 4-5 cases each. There’s also integration testing for DataFormatter as a whole but only has one test case.

When running test_formatter.py you should see it print the assumptions and known bugs for each function in DataFormatter.

6.0 Known Bugs and Assumptions

Buggy keywords - There are keywords that are just prone to creating misinformation and they will be added to bug_keywords.txt in lists/ folder. For example ‘less’ is a web dev framework but each job post often has a ‘show less’ button and now ‘less’ is one of the most in demand technologies.

There are also keywords that are just not that helpful to users (this is subjective and needs more user research on), for example ‘user’, ‘communication’, ‘product’ may not be that helpful yet they will appear on the top of the most in demand list due to how frequently they appear.

Technical bugs:

Entire Test ASSUMES keywords (tech terms, keywords, degree title/level) won’t be the first or last word as it doesn't really happen, (there's space before or after words)

extract_industry() assumes there’s a next line after the ‘industry’ keyword.

extract_languages() assumes GO not golang, so all the golang will be missed.

extract_tech_terms() - Some margin of error dealing with tech words in tech words like react in react native, can't be detected specifically
Some margin of error, dealing with plurals and different ways of saying same terms like Vue vs Vue.js

extract_yoe() - We are finding years of INDUSTRY/WORK experience NOT 'relevant' experience
assumes theres no + after number

7.0 Running data_formatter.py

Everything in data_formatter.py is wrapped up in the class DataFormatter. Therefore you can run each individual function independently like in test_formatter.py and also run it as a whole by calling preprocess() and data_extraction() like in the screenshot of the Web crawler above in Selenium_Navigation.py.

8.0 Future scaling ideas

1. Skim job description to just requirements, preferred req, and education. Makes things faster and less error prone (keywords won't get detected in the description part instead of requirements).
This would require a very comprehensive list of variations of 'education','qualification' , "we would like to see", "what u should have" etc.
2. Handle more types of degree titles, might need to search through just the education section