Files created in the project:

1. extract\_skills.py: This file is where the I scrape Indeed for over 10000 job postings. I also extract the skills from the job posting and save it in 2 files. Those files are:
2. data\_analyst.json: This file contains the skills from over 5000 data analyst jobs stored in the list format.
3. dat.json: This file contains the skills from the job title “web developer”. Scraped over 5000 job postings.
4. analyze\_skills.py: This is the file where the skills that are sored in the 2 json files are read, preprocessed to get cleaner data. Also the top skills in for the each job title is calculated here and the similarity between the job titles is also calculated and displayed in this file.

Original List of tasks to be completed for this project:

1. Get a web response from indeed.com website for a given location and job title.
2. pick the URL of one job posting out of several results
3. Extract the list of skills in that job posting.
4. Use word tokenization on the skills from nltk library.
5. Remove the stop words from the list of skills (done with nltk).
6. Use lemmatization to get the core word
7. Use part-of-speech tagging on the remaining list of words.
8. Isolate the words that are ‘tagged’ as nouns, adjectives (or maybe more tags can be considered or removed after further testing) and label them as ‘skills’
9. Create a dictionary where each ‘skill’ will be the key with the frequency with which that skill appears as the ‘value’.
10. Update the dictionary for over 10,000 different job postings times to get the result.

Modified list of tasks that were completed in this project:

1. Get a web response from indeed.com website for a given location and job title.
2. pick the URL of one job posting out of several results
3. Extract the list of skills in that job posting.
4. Use word tokenization on the skills from nltk library.
5. Use lemmatization to get the core word
6. Use part-of-speech tagging on the remaining list of words.
7. Isolate the words that are ‘tagged’ as nouns (nn) and (nnp) and also attach the word that comes prior to these words to make a bi-gram. Eg: “NodeJs framework”
8. Store all the skills for a particular job title in a json file.
9. Read the json files and create 2 dictionaries (1 for each job title) where each ‘skill’ will be the key with the frequency with which that skill appears as the ‘value’.
10. Calculate the similarity between the dictionaries (2 job titles) with cosine similarity.

Steps to run the project:

1. Run extract\_skills.py, make sure you specify the name of the file where you want to store the data on line 116. Also, this code needs to be run several times to get large amounts of data. Running this code once will give you 100-150 job postings at once because indeed.com bans IP address. To make sure you cycle through unique pages, after each run of the program, update line 119 of the program to get unique pages.

Eg: run 1

Graphical user interface, text

Description automatically generated

In this example, lets assume that in the first run of the program we were only able to reach page 50, after which our ip address was banned by indeed. The data collected till page 50 is already stored in the json file, so no worries there, however, in order to get new data, we have to manually update the range function, so when we run the program a second time, either by using a VPN or by waiting for 1 hr, we can update the range function like so:

Graphical user interface, text

Description automatically generated

1. Once you collect all the data, run analyse\_skills.py to get the skill frequency dictionary and the cosine similarity score.