***Project Overview:***

This project idea is to scrape the skills from the job postings on Indeed webpage. Then it will use the data to analyze what skills are most chosen for a job title (with location if the user wants). The analysis can be used for users who are looking for a job to know what skills are usually required for their targeted job, then they can prepare those skills and resumes accordingly. Another important feature of the program is that it will calculate the overlap of two specific jobs using Bhattacharyya coefficient. This is useful for the researchers who want to collect data and perform data analysis on the job market.

***To Run The Project:***

In indeed-scrapers.py scroll down to the bottom of the file. You will see two variables called ‘title’ and ‘location’. You can change them to try to scrape different job titles. **Warning: Indeed only allow you to scrape around 150-200 job postings so after each run you have to change your IP using VPN to run another session.** Below is the image of where you should change:

Text

Description automatically generated

***Steps Of The Project:***

1. **Indeed Web Scraping:**

To scrape Indeed, first I try to learn the structure of the job search page of Indeed. From there, I notice that there is a repetitive pattern that we can make use to make the scraping easier. For example, indeed has a search page that looks like this:

Graphical user interface, text, application, email

Description automatically generatedAs you can see there are many ‘Cards’ that contains job posting. And those cards are placed in the div with the same class’ name. Therefore, we just need to extract any div with that class name to retrieve the job’s basic information. For website scraping, I use BeautifulSoup as it is a powerful Python package that specializes in scraping. When we get the div that contains the job’s basic information, we need to get the href that will direct us to the main page the contains detailed information of the targeting job. Fortunately, there is an attribute called href that is stored for every ‘Card’, therefore I use that to get to the targeting job’s website which looks like this: Graphical user interface, text, application

Description automatically generated

As we all know, the recruiters from companies will try to specify the qualifications/requirements/skills for the job inside a section – usually using bullet points like in the example. This section can vary with different companies, but as I have gone through many job postings, I notice that they usually use bullet point that is placed inside an unordered list (‘ul’ tag in HTML). Therefore, I try to come up with a list of words that the companies will most likely use to call the qualifications section and whenever my program detects words in that list, it will get the next closest unordered list and save it in a list called target\_list. This function is not 100% accurate, but its result is fairly high: 90% of the time it gives the correct qualification text that we want.

One more important thing is that to generate the list of words that companies will use to call their qualification section, I have the python file called generate\_target\_words.py. The reason for this file is that I can add more words to the list as we browse through more job postings and it will help the future growth of this application.

1. **Preprocessing:**

To preprocess the text, first I replace all the contractions into full words, then I remove punctuations and boundary markers. Finally, I remove all numbers and special characters in the targeting text. For the next step, I decided that I will try to tokenize the targeting text into unigram and bigram for skills extraction. I use word\_tokenize from NLTK package to tokenize the targeting text into a unigram list and use nltk.bigrams to a bigram list of the targeting text. Then I apply part-of-speech tagging to both lists using nltk.pos\_tag and get the taggings so that I can get to the next step – skill extractions

1. **Getting skills:**

After getting both lists with their pos tagging, I try to get the words that are classified as Noun (‘NN’) or a Noun plural (‘NNS’), The reason for this is that the skills are most likely to be in noun form. There is a limitation for this as sometimes the skills can be a statement such as “know how to drive”, however, this is not common and I want to focus on the noun to get a better and more accurate skill extraction.

After getting all the words that are classified as Noun or Noun plural, I tried to calculate how many times they have been used throughout for one job title. First I will explain how my application runs: It takes in two variables that are job title and location and then creates an indeed url of the corresponding search result which looks like this (job title is data entry and location is Vancouver):

Graphical user interface, text, application

Description automatically generated

Then it will do the web scraping and text and skills extraction as I have mentioned above. So for a job title that the user input, the application will record how many times a skill has appeared throughout all of the jobs that it has scraped, in unigram and bigram. The final result will be two dictionaries of skills in unigram and bigram with key is the skill and value is the occurrences of that skill and is sorted from the largest number of occurrences to the smallest. As I notice that many words have a few occurrences (1 to 10), I tried to limit the chosen skills by setting the threshold for them. Another important idea that I implement is that as I notice many unigram skills has been picked by the bigram skills, I tried to remove all the unigram skills that have been picked by the bigram skills (For example the bigram is ‘python experience’, I will remove the ‘python’ and ‘experience’ from the unigram if it has those skills). Therefore, the bigram skills list is the **most important,** and the unigram skills list is just to catch any single term left.

The final results are stored in the folder called data\_collection where you will have 4 files:

* jobtitle-location-skills is the file that contains all the skills with their occurrences in unigram for that job title (location can be omitted if not specified).
* jobtitle-location-skills-bigram is the file that contains all the skills with their occurrences in bigram for that job title (location can be omitted if not specified).
* jobtitle-location-chosen-skills is the file that contains the top skills with their occurrences in unigram for that job title (location can be omitted if not specified).
* jobtitle-location-chosen-skills-bigram is the file that contains the top skills with their occurrences in bigram for that job title (location can be omitted if not specified).

One last thing that I want to mention is that Indeed doesn’t allow people to use their API unless you become a recruiter (which means you need to submit a website for your company), I only use the normal request method to access Indeed website. Therefore, Indeed always stop me at around 200 job postings because it recognizes me as a bot and as me to resolve a captcha, which will stop my program. Therefore, to meet 10000 postings requirements, I have to run the program for around 50 different jobs and you can access them in the data\_collection folder.

1. **Computing Bhattacharyya coefficient:**

To calculate the Bhattacharyya coefficient, I use the Python package call dictances. The function is located inside the Python file called Bhattacharyya\_calculate that requires the user to put in two job titles (at first\_job\_name and second\_job\_name variables) that are in data\_collection folder to computer the Bhattacharyya coefficient.

I decided to use the bigram list because it gives more accurate skills than the unigram list. For example, for two job titles that are similar to each other such as ‘Programmer’ and ‘Programmer Entry Level’, the function gives the high result:



But if we input two not related skills the result is reduced significantly:



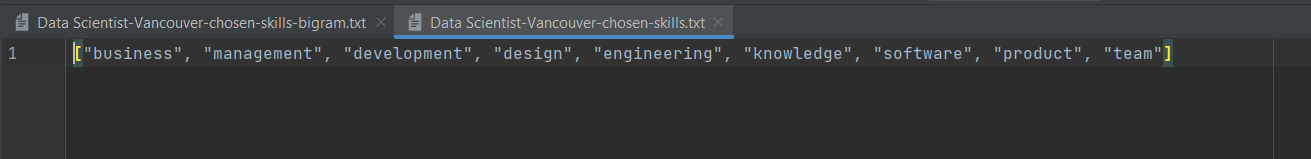
***Test Cases for skills extraction:***

This is the list of bigram skills for Data Scientist in Vancouver, the skills listed make sense:

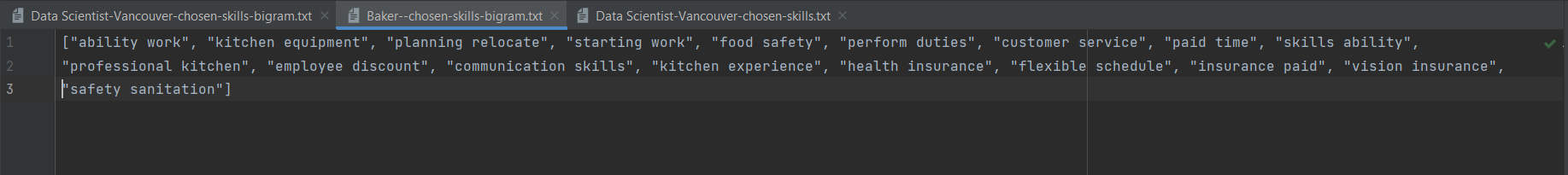
Text

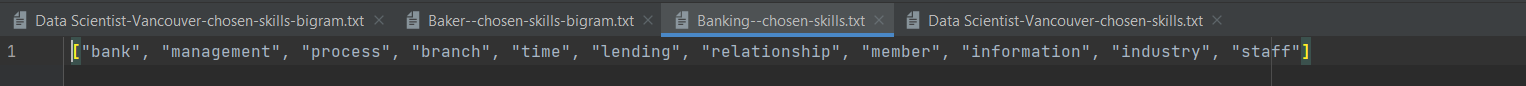
Description automatically generated with medium confidence

This is the list of unigram skills for Data Scientist in Vancouver, the skills listed are not comprehensible, but we can deduct from some of them. For example, design means skills related to designing. This is the limitation of my program that it cannot do the deduction for the user. In addition, as I have mentioned above, the purpose of the unigram list skills is to only catch the words that have the highest frequency and do not appear in bigram lists:



I will demonstrate some more test cases here. This is the list of bigram skills and unigram skills for Baker without a location specified:





***Test Cases for Bhattacharyya coefficient:***

Below are some test cases that demonstrate the Bhattacharyya coefficient of some couples of jobs. You can notice the pattern that the higher the two jobs resemble each other, the higher the score. For example, Python Developer and Programmer or UI designer have high scores whereas Accountant and Labourer have low scores:







