

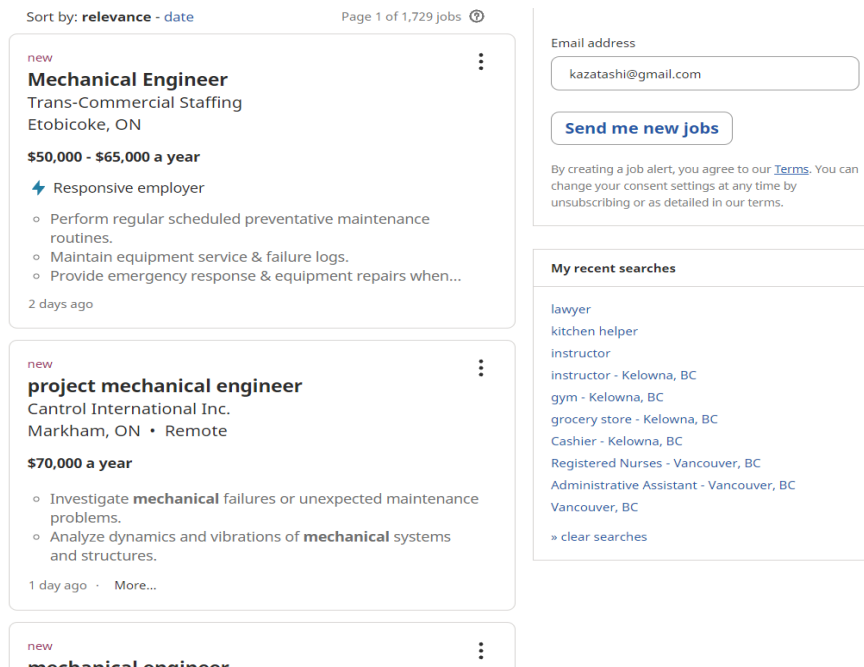
To Run The Project:

In indeed-scrappers.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 title. **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:

```
indeed-scrappers.py
173
174 if __name__ == '__main__':
175     # job search settings
176     # title = 'Avionics Engineer The Structures Company'
177     # loc = 'Vietnam, PR'
178     title = 'Data Scientist'
179     location = ''
180     main(title, location)
181
182
183
184
```

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:




As 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

Mechanical Engineer

Trans-Commercial Staffing
Etobicoke, ON
\$50,000 - \$65,000 a year - Full-time, Permanent

⚡ Responded to 75% or more applications in the past 30 days, typically within 10 days.

Apply now



Coordinate service vendors as required

- Comply with all company Health and Safety directives
- Professional representation of the company at all times
- Interact effectively with other members of the operation teams
- Shift work is required

Qualifications:

- Electromechanical Engineering
- Basic knowledge of electronic circuits and computers is required
- Working knowledge of electrical controls & system interfaces
- Working Knowledge of Both Mechanical and pneumatic operated conveyor Systems
- Ability to read both Electrical and Mechanical Drawings
- Demonstrated ability to handle multiple tasks
- Demonstrated mechanical ability
- Excellent troubleshooting skills

Job Types: Full-time, Permanent

Salary: \$50,000.00-\$65,000.00 per year

Schedule:

- 8 hour shift
- Monday to Friday

Education:

By selecting Follow, you agree to get updated information and new jobs for this company by email. You can cancel alerts at anytime.

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.

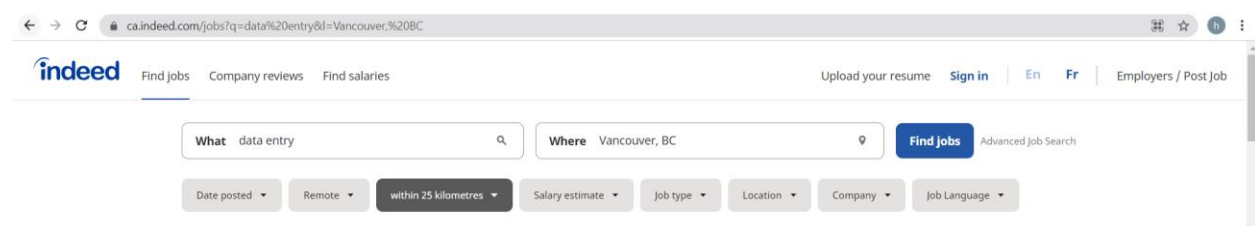
To preprocess the text, first I replace all the contractions into full words, then I remove punctuations and boundary markers. Finally, I remove all numbers and special characters in the targeting text. For the next step, I decided that I will try to tokenize the targeting text into unigram and bigram 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

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):



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 terms left.

The final results are stored in the folder called `data_collection` where you will have 4 files:

- `jobtitle--skills` is the file that contains all the skills with their occurrences in unigram for that job title.
- `jobtitle--skills-bigram` is the file that contains all the skills with their occurrences in bigram for that job title.
- `jobtitle--chosen-skills` is the file that contains the top skills with their occurrences in unigram for that job title.

- `jobtitle--chosen-skills-bigram` is the file that contains the top skills with their occurrences in bigram for that job title.

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.

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 compute the Bhattacharyya coefficient.

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

```
Result for Bhattacharyya coefficient between Programmer and Programmer Entry Level : 2601.440935576227
```

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

```
Result for Bhattacharyya coefficient between Accountant and Programmer Entry Level : 1148.8889474649525
```