LIS 875: Topics in Information Processing and Retrieval

Web Scraping Indeed Job Positions

Group members:

Satvik Shukla, Jai Potnuri, Shreyas Vastrad

Introduction

In view of the existing complexity faced by an individual applying for job positions in the market, we have built a data science project that aims at providing job market statistics. As a part of this project, we are initially considering the Indeed jobs site as a data resource. This data science project aims to provide the details of specific job roles in particular areas. Further to narrow down the purpose of job market statistics we have narrowed down the deliverables of this project to address the objectives below in this draft.

1. Collect and compare job market statistics for at least three categories of roles
   - UX Research
   - Software Engineer
   - Data Analyst
2. Collect and compare job market statistics for at least two geographic areas
   - Geographics Areas
   - New York
   - Chicago

Methodology and Approach Implemented

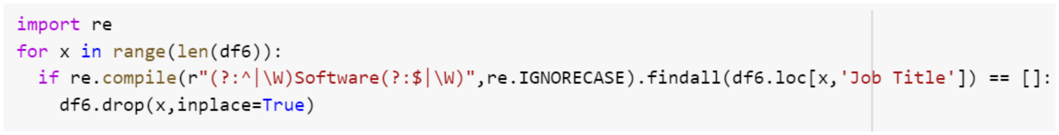
Data Collection: The collection of data is done through web scraping techniques using python with Indeed, as a source of data for analysis. Typical libraries in python such as Request and BeautifulSoup are used for hitting the browser and parsing the data. A total of 1805 results were collected for data analysis using this technique

Data Cleaning: Below are the few steps that we followed for cleaning the data

1. We had made sure that there were no missing values in the data

2. We had dropped the Null values existing in the data

3. We applied a filter for each job title to ensure that we were providing the analysis on the same roles using Regular Expression in python (RegEx). Ex –We had created a filter in regex for Software Engineer positions such that every job title has the word ‘Software’ in its job. Similarly, we made sure that the words ‘Data’ and ‘UX’ exist in the job titles for Data Analyst and UX Research positions. Below is the sample code for the Software Engineer positions we used in our implementation



```
import re
for x in range(len(df6)):
  if re.compile(r"(?:^|\W)Software(?:$|\W)",re.IGNORECASE).findall(df6.loc[x,'Job Title']) == []:
    df6.drop(x,inplace=True)
```

Data Analysis: Analysis is done on top of the data using python and excel. The process we had followed in this project is detailed clearly below

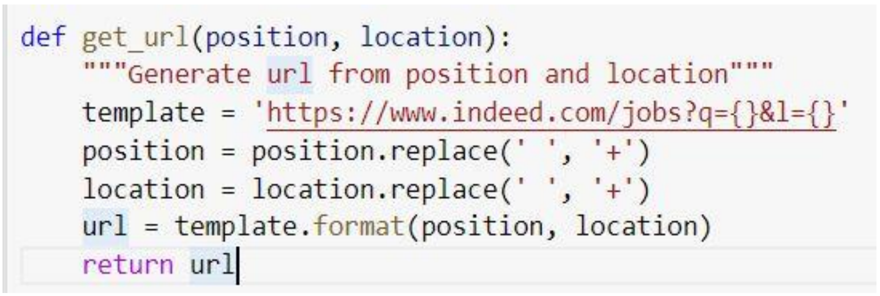
Approach

The complete process involved the following tasks:

1. Creating the URL for required jobs and locations
2. Getting the page response and checking the response status
3. Crawling over the response page to find attributes like Job title, Job Exact Location, Organization, Salary, Job Description, and Job postdate.
4. Cleaning and filtering from Job titles
5. Performing analysis to answer required questions
6. Performing in Depth comparison analysis

Creating the URL for required jobs and locations

Creating a complete URL on top of basic by adding parameters like job role and location. We created a function that takes in parameters (job role and location) and returns the complete URL.



```
def get_url(position, location):
    """Generate url from position and location"""
    template = 'https://www.indeed.com/jobs?q={}&l={}'
    position = position.replace(' ', '+')
    location = location.replace(' ', '+')
    url = template.format(position, location)
    return url
```

Getting the page response

Simply use the get function from the requests package to get the page response. And using the status function on the response tells us if we correctly got the response.

response = requests.get(url)

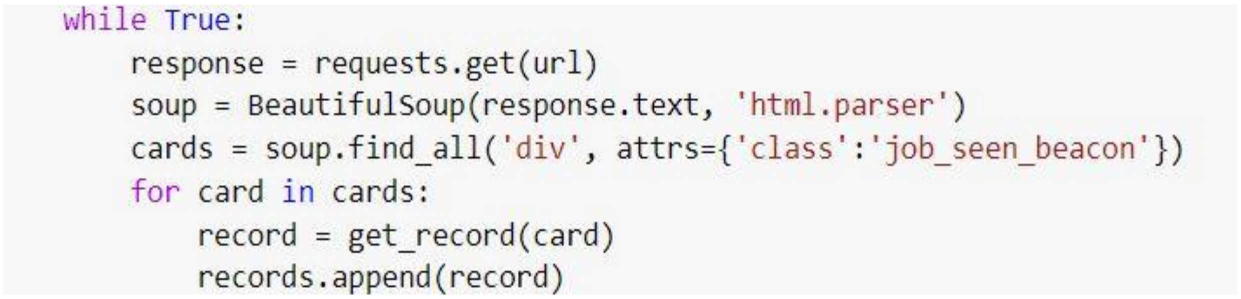
response.status

The important thing to note is that we need to wait for at least 15 seconds before we pass our next get request, if not our further responses would be banned. This can be done by using

time.sleep(15)

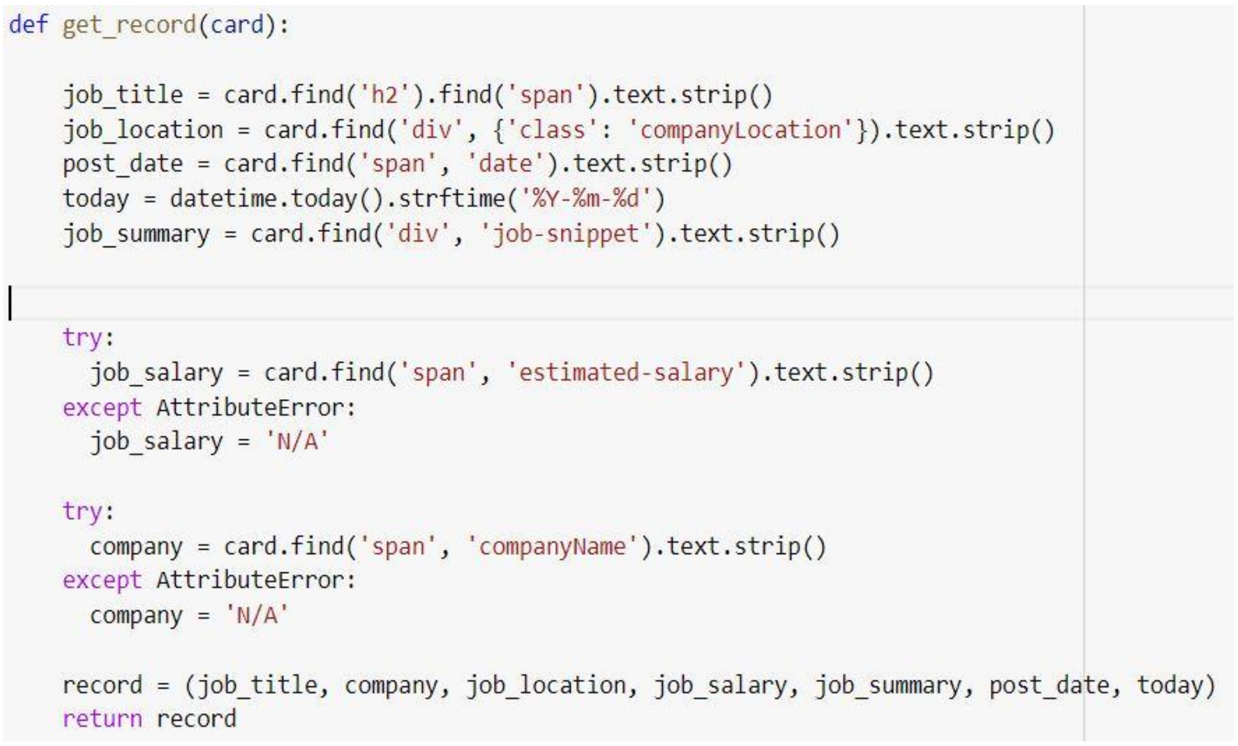
Crawling over the response page

We have used beautiful soap to crawl over the html page to get the exact html tags and extract attributes.



```python
    while True:
        response = requests.get(url)
        soup = BeautifulSoup(response.text, 'html.parser')
        cards = soup.find_all('div', attrs={'class':'job_seen_beacon'})
        for card in cards:
            record = get_record(card)
            records.append(record)
```

As we pass our get request to main indeed page, its response contains a list of jobs. Here each card represents a job posting. So, for getting record or each attribute from each card/job posting we created a get record method.



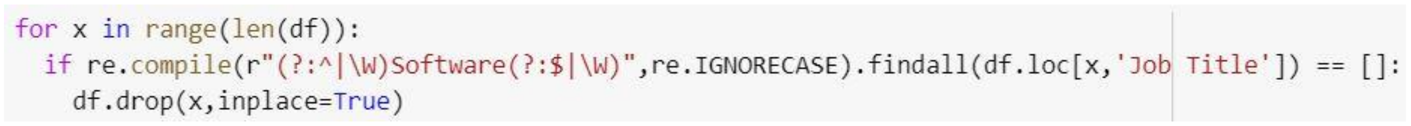
```python
def get_record(card):

    job_title = card.find('h2').find('span').text.strip()
    job_location = card.find('div', {'class': 'companyLocation'}).text.strip()
    post_date = card.find('span', 'date').text.strip()
    today = datetime.today().strftime('%Y-%m-%d')
    job_summary = card.find('div', 'job-snippet').text.strip()


    try:
      job_salary = card.find('span', 'estimated-salary').text.strip()
    except AttributeError:
      job_salary = 'N/A'

    try:
      company = card.find('span', 'companyName').text.strip()
    except AttributeError:
      company = 'N/A'

    record = (job_title, company, job_location, job_salary, job_summary, post_date, today)
    return record
```

From each record, the required attributes are parsed and returned. This creates our data set with features like Job Title, Company, Location, Salary, Job Description, and Posting Date.

Cleaning and filtering

Cleaning the dataset, a major step we took as some of the data we obtained was not clean. So, we came across cleaning the dataset according to the Job Titles described in the job postings.

We tried to filter using regex to find out if all the postings with Job Titles had "Software" (for Software Engineering), "Analyst" (for Data Analyst), and "UX" (for UX) words. Example:



```
for x in range(len(df)):
  if re.compile(r"(?:^|\W)Software(?:$|\W)",re.IGNORECASE).findall(df.loc[x,'Job Title']) == []:
    df.drop(x,inplace=True)
```

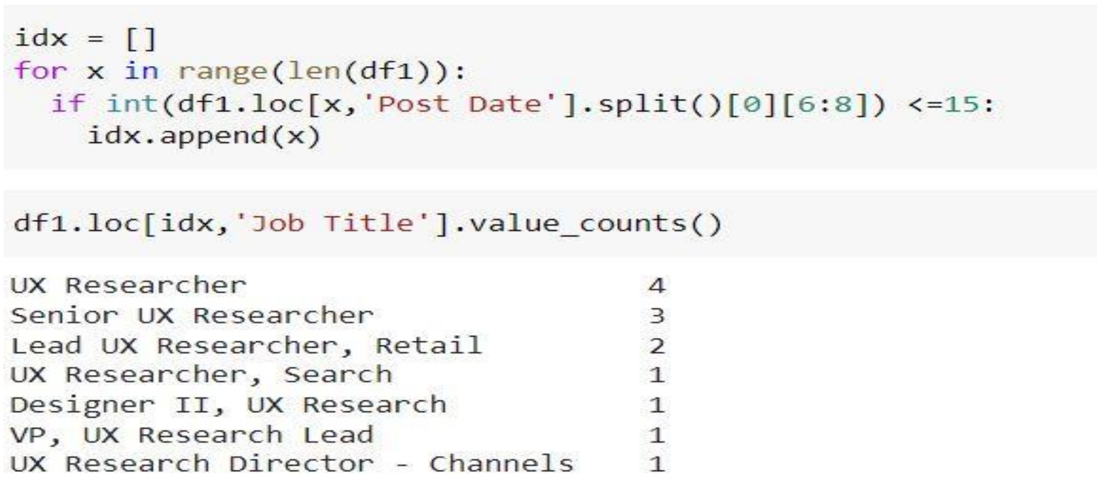
Then we had our final dataset.

Questions to be answered (For each role and location)

1. Top 10 companies
2. Top frequent Job titles used with job postings within the last 15 days
3. % Of Senior level and Junior level roles
4. Major skills needed

How to answer?

1. Approach to the first question was simple, we just had to find the top 10 with respect to different job roles and location
2. Applying a filter (posting <15 days) was the major task here. We used the split python string function to split the posting word (into "posting" + no. Days) and apply a filter. And then we just had to sort according to their frequency.

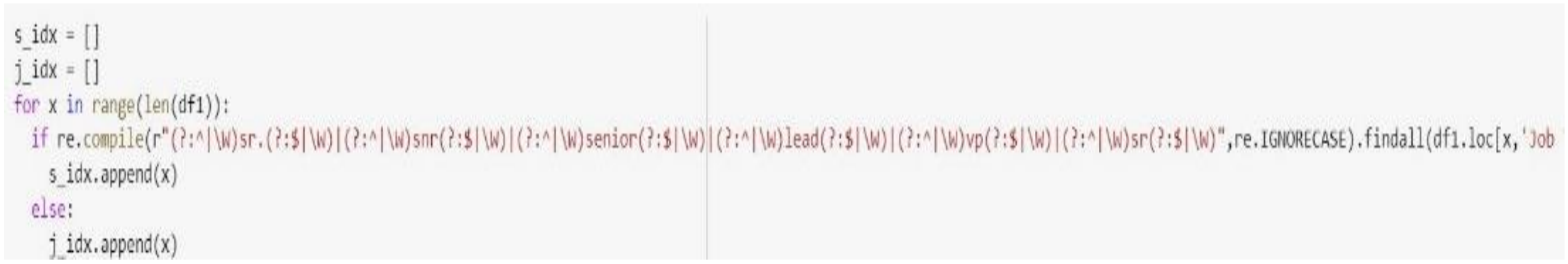


```
idx = []
for x in range(len(df1)):
  if int(df1.loc[x,'Post Date'].split()[0][6:8]) <=15:
    idx.append(x)
```

```
df1.loc[idx,'Job Title'].value_counts()
```

```
UX Researcher                        4
Senior UX Researcher                 3
Lead UX Researcher, Retail           2
UX Researcher, Search                1
Designer II, UX Research             1
VP, UX Research Lead                 1
UX Research Director - Channels      1
```

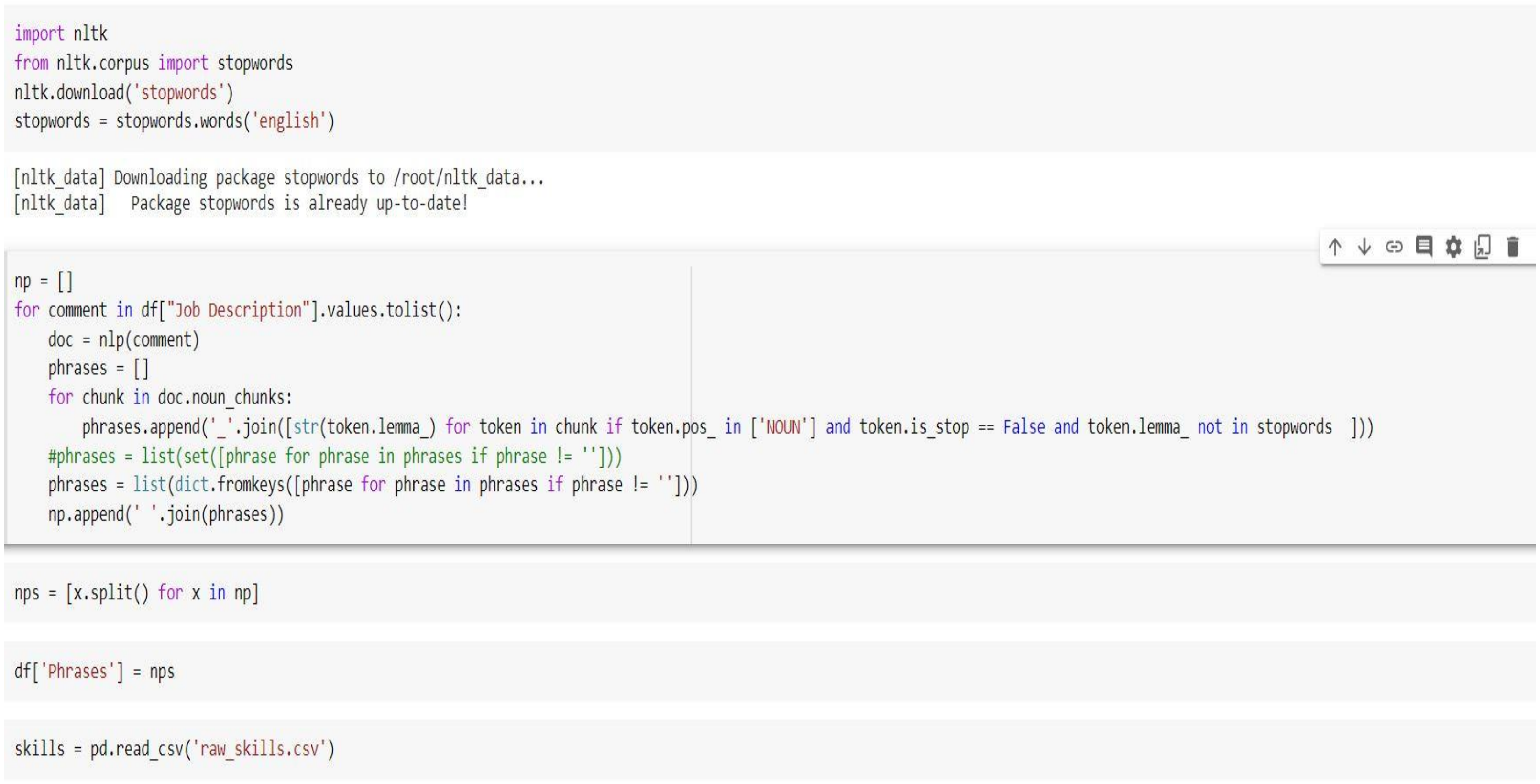
3. For the third question we had to filter job titles that had words like "Sr.", "Senior", "Lead", and "VP" using regex to extract senior roles from the whole data set.



```
s_idx = []
j_idx = []
for x in range(len(df1)):
  if re.compile(r"(?:^|\W)sr.(?:$|\W)|(?:^|\W)snr(?:$|\W)|(?:^|\W)senior(?:$|\W)|(?:^|\W)lead(?:$|\W)|(?:^|\W)vp(?:$|\W)|(?:^|\W)sr(?:$|\W)",re.IGNORECASE).findall(df1.loc[x,'Job
    s_idx.append(x)
  else:
    j_idx.append(x)
```

4. For answering the fourth question we had to download the list of most important technical skills. As skills mentioned in the downloaded list were not single words, we had to perform Noun phrase chunking (using spacy) to extract noun phrase terms and then match phrase terms with the skills and extract the matched skills.

   Code snippet:



```
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
stopwords = stopwords.words('english')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
np = []
for comment in df["Job Description"].values.tolist():
    doc = nlp(comment)
    phrases = []
    for chunk in doc.noun_chunks:
        phrases.append('_'.join([str(token.lemma_) for token in chunk if token.pos_ in ['NOUN'] and token.is_stop == False and token.lemma_ not in stopwords  ]))
    #phrases = list(set([phrase for phrase in phrases if phrase != '']))
    phrases = list(dict.fromkeys([phrase for phrase in phrases if phrase != '']))
    np.append(' '.join(phrases))
```

```
nps = [x.split() for x in np]
```

```
df['Phrases'] = nps
```

```
skills = pd.read_csv('raw_skills.csv')
```

Comparison:

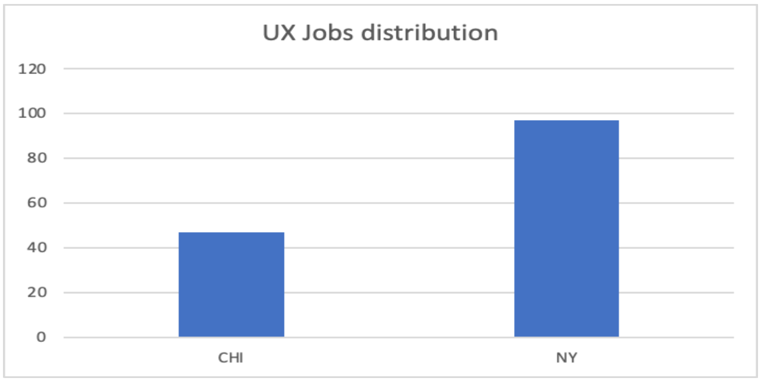
For each job and city, we created 2-3 points of comparison (Metrics). These metrics would help a particular job seeker to choose a position and city which has the maximum probability of selection for that job.

Job: UX, City: Chicago VS New York

- Percentage of Senior Level Positions in UX Researcher roles in the New York Area is 49.48 %
- Percentage of Junior Level Positions in UX Researcher roles in the New York Area is 50.51 %
- Percentage of Senior Level Positions in UX Researcher roles in the Chicago Area 46.80 %

- Percentage of Junior Level Positions in UX Researcher roles in the Chicago Area 53.19 %
- We see that big shots are open for UX jobs In New York, whereas comparatively lower-tier companies tend to hire in Chicago for UX. High % of chances of getting shortlisted at Chicago. For both the locations, we see similar stats W.R.T Exp level.
- Job Distribution graph: for UX in Chicago and New York.

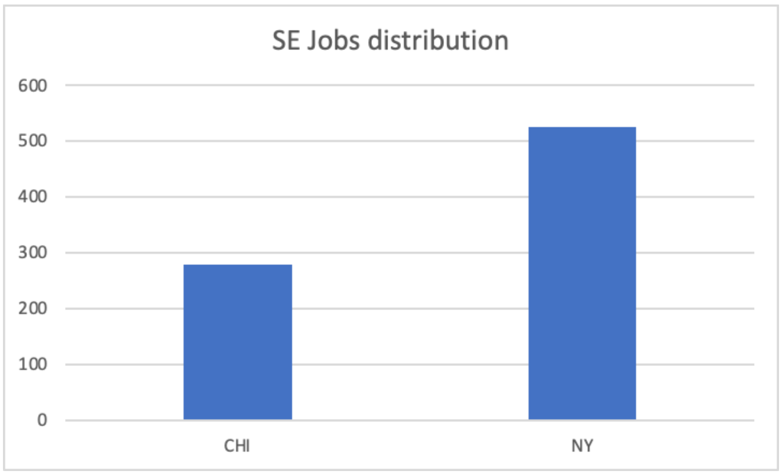
UX job distribution



Job: SE, City: Chicago VS New York

- Percentage of Senior Level Positions in SE roles in the Chicago Area
  25.17%
- Percentage of Junior Level Positions in SE roles in the Chicago Area
  74.82%
- Percentage of Senior Level Positions in SE roles in the New York Area 19.04%
- Percentage of Junior Level Positions in SE roles in the New York Area
  80.95 %
- The company-level tier looks similar in both locations.
- For both locations, we have a greater number of junior-level jobs. But if you are looking for senior-level SDE jobs go to NY

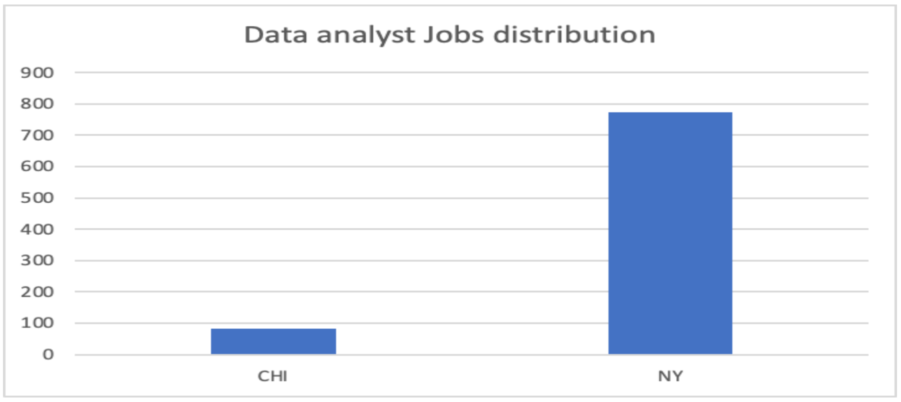
Software Engineering job distribution



Job Data Analyst: Chicago VS New York

- Percentage of Senior Level Positions in DA roles in the Chicago Area 2.38 %
- Percentage of Junior Level Positions in DA roles in the Chicago Area 97.61 %
- Percentage of Senior Level Positions in DA roles in the New York Area 23.41 %
- Percentage of Junior Level Positions in DA roles in the New York Area 76.58 %
- Company level tier looks similar in both locations.
- Comparatively Chicago has all the entry-level jobs for DA (less senior-level jobs). NY has a 76/24 split, 76% Entry, and 24% senior.

Data Analyst job distribution



The Recommendations:

- UX: Both types of candidates can apply anywhere, however, if you have preferences like top companies go for New York, if you want to stay near Madison go for Chicago. Not many differences in the UX market.
- SE: For entry-level positions, we should prefer NY and for senior-level positions, we can go to Chicago.
- DA: For entry-level positions we should go to Chicago and for senior level positions we can go to New York.