Summer 2022 CS 59000-04I AI Software Engr & Application DIS

Welcome Students from **Group 1**

The dataset to be used at the tasks should be mounted from Google Drive for each student.

The datasets are provided as a shared Google Drive folder, if you have not received the link, please inform us.

The instructions for mount the Google Drive folder at the Google Colab and to access the data is:

https://colab.research.google.com/notebooks/io.ipynb

The dataset available are:

- Job postings
- Resumes

If the share "datasets" folder is not showing in Google Colab, follow these instructions to add a shortcut to your drive:

https://support.google.com/drive/answer/2375057

For any follow-up questions/queries:

- Venkata Inukollu inukollv@pfw.edu
- Leandro de Almeida almel01@pfw.edu

Disclaimer: The provided datasets are restrict to be used only during the academic tasks

References:

- Pandas Caculate Statistics/ Summary/ Columns

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mour
```

Material/References:

- Pandas - Filter row and columns
- Pandas - Drop multiple columns
- Pandas - Check Pandas data type
- Data - Columns Views - Original Data
- Pandas - Convert value in columns
- Time Ranges/ Time Comparision
- Remove columns or Rows in Pandas
- Remove rows with certain citeria in Python Pandas
- AI BOOKS
- https://towardsdatascience.com/gentle-start-to-natural-language-processing-using-python-6e46c07addf3
- https://monkeylearn.com/keyword-extraction/
- https://www.justintodata.com/use-nlp-in-python-practical-step-by-step-example/
- https://mathdatasimplified.com/
- https://www.kdnuggets.com/2019/11/content-based-recommender-using-natural-language-processing-nlp.html (7/28/2022)

```
# Data Pre-Processing - Job listing Dataset
# Import necessary packages
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import sklearn
import json
import os
import gc # For garbage collection when deal with memory
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mour
```

```
os.getcwd()
```

```
'/content'
```

Double-click (or enter) to edit

Job Data Analyst

- Goal: Top 3 Career choices, success factor (example Salaries growth, location, etc as per AI Attributes)

Read Data:

```
# FileNames is a list with the names of the csv files contained in the 'dataset' path
def get_file_names(path):
  filenames = []
  for file in os.listdir(path):
    if file.endswith('.csv'):
      filenames.append(file)
  return filenames

# function that reads the file from the FileNames list and makes it become a dataFrame
def GetFile(fnombre, path):
  location = path + fnombre
  df = pd.read_csv(location)
  return df

file_path_job = './drive/MyDrive/datasets/jobposting/'
# combine all the data frame as one using list complehesion
dfjob = pd.concat([GetFile(file, file_path_job) for file in get_file_names(file_path_job)])
```

```
dfjob.shape
```

```
(300000, 22)
```

Attributes Validation:

```
dfjob['salary_formatted'].value_counts()
```

```
$15 an hour                  2136
From $15 an hour             1441
$15 - $20 an hour            1273
$17 an hour                  1246
$16 an hour                  1188
                             ...
$64,000 - $80,000 a year        1
$15.00 - $17.64 an hour         1
$40 - $75 a day                 1
$3,092 - $3,762 a month         1
$32,176 - $47,316 a year        1
Name: salary_formatted, Length: 20577, dtype: int64
```

```
dfjob['region'].value_counts()
# Remove because there are all missing value here
```

```
EU          3510
AS          2761
SA          1863
AF           900
OC           505
Americas      54
Name: region, dtype: int64
```

```
dfjob['qualifications'].value_counts()
# Can't remove because need this for further basic qualification
# Convert the NAN to 'No requirement'
```

```
["US work authorization (Required)"]
2930
["US work authorization (Preferred)"]
2089
["High school or equivalent (Preferred)"]
2001
["Driver's License (Required)"]
973
["Bachelor's (Preferred)"]
960

...
["HVAC Certification (Required)","US work authorization (Required)","Secret
(Required)","Associate (Preferred)"]                                    1
["Restaurant experience: 4 years (Required)","Day Shift (Preferred)","Night Shift
(Preferred)"]                                                            1
["IT support: 5 years (Required)"]
1
["Microsoft Excel: 4 years (Preferred)","Tableau: 1 year (Preferred)","Data analytics:
4 years (Preferred)","US work authorization (Preferred)"]       1
["UI: 7 years (Preferred)","React: 7 years (Preferred)","Angular: 7 years
(Preferred)","JavaScript: 7 years (Preferred)"]                              1
Name: qualifications, Length: 20165, dtype: int64
```

```
dfjob['benefits'].value_counts()
```

```
["Health insurance"]
5325
["Flexible schedule"]
2717
["401(k)","Dental insurance","Health insurance","Paid time off","Vision insurance"]
1873
["Paid time off"]
908
["Dental insurance","Health insurance","Paid time off","Vision insurance"]
855
```

```
...
["403(b)","403(b) matching","Dental insurance","Flexible spending account","Health
insurance","Paid time off","Parental leave","Vision insurance"]
1
["401(k)","AD&D insurance","Dental insurance","Disability insurance","Employee
assistance program","Employee discount","Flexible schedule","Health insurance","Life
insurance","Paid time off","Referral program","Tuition reimbursement","Vision
insurance","Wellness program"]       1
["401(k)","401(k) matching","Flexible schedule","Health insurance","Health savings
account","Life insurance","Paid time off","Professional development assistance","Safety
equipment provided","Tuition reimbursement"]
1
["On-the-job training","Pet insurance","Tools provided","Tuition reimbursement"]
1
["401(k)","AD&D insurance","Commuter assistance","Dental insurance","Disability
insurance","Employee assistance program","Flexible spending account","Health
insurance","Health savings account","Paid time off","Parental leave","Tuition
reimbursement","Vision insurance"]          1
Name: benefits, Length: 19372, dtype: int64
```

```
# Remove src name
dfjob['srcname'].isnull().sum()
```

```
193418
```

```
dfjob['country'].value_counts()
```

```
US    300000
Name: country, dtype: int64
```

```
dfjob['country_code'].value_counts()
```

```
US    282417
BR       800
GB       681
CO       668
CA       612
       ...
CF         1
BY         1
ZM         1
MD         1
UG         1
Name: country_code, Length: 166, dtype: int64
```

```
dfjob['company_name'].value_counts()
```

```
Deloitte                                  3804
ASSURANCE Independent Agents              1774
Amazon.com Services LLC                   1401
Aya Healthcare                            1224
Soliant                                   1075
```

```
                                                ...
Ardent Counseling Center                          1
Mobile Management llc                             1
Duro Electric                                     1
CareerStaff Unlimited - Nashville, TN             1
Sanel Corp                                        1
Name: company_name, Length: 97715, dtype: int64
```

```
dfjob['company_link'].value_counts()
```

```
https://www.indeed.com/cmp/The-Est%C3%A9e-Lauder-Companies-1?
campaignid=vjcmpinfo&from=vjcmpinfo&tk=1g54cpcg728qo000&fromjk=00009f127a9e34a7
1
https://www.indeed.com/cmp/Holistic-Healing-Collective?
campaignid=vjcmpinfo&from=vjcmpinfo&tk=1g54t7pg6pkej800&fromjk=96522f26f3a8fcba
1
https://www.indeed.com/cmp/United-Premier?
campaignid=vjcmpinfo&from=vjcmpinfo&tk=1g54t62jeq072800&fromjk=96524389f8fbf9ac
1
https://www.indeed.com/cmp/Temp-Experts?
campaignid=vjcmpinfo&from=vjcmpinfo&tk=1g54t5uj7t48o800&fromjk=965241afe500d938
1
https://www.indeed.com/cmp/Kum-&-Go?
campaignid=vjcmpinfo&from=vjcmpinfo&tk=1g54t6r8lq051800&fromjk=96523fc6ef1ed652
1

..
https://www.indeed.com/cmp/Trugreen?
campaignid=vjcmpinfo&from=vjcmpinfo&tk=1g54t4ctsj3vu800&fromjk=94750524588fee11
1
https://www.indeed.com/cmp/Red-Knight-Solutions,-LLC?
campaignid=vjcmpinfo&from=vjcmpinfo&tk=1g54t48h6isah800&fromjk=9475094e35b7bc1f
1
https://www.indeed.com/cmp/Bon-Secours?
campaignid=vjcmpinfo&from=vjcmpinfo&tk=1g54t48k5i7kk800&fromjk=947510b2566d0887
1
https://www.indeed.com/cmp/Beaumont-Health?
campaignid=vjcmpinfo&from=vjcmpinfo&tk=1g54t4aq5h5i1800&fromjk=947512447005a768
1
https://www.indeed.com/cmp/Disney-Media-and-Entertainment-Distribution?
campaignid=vjcmpinfo&from=vjcmpinfo&tk=1g54fb8l5lekk800&fromjk=0710b77c13b3dd2e    1
Name: company_link, Length: 286348, dtype: int64
```

```
dfjob.head()
```

	jobid	apply_link	company_link
0	00009f127a9e34a7	https://www.indeed.com/applystart?jk=00009f127...	https://www.indeed.com/cmp/The-Est%C3%A9e-Laud...
1	0001783849fce183	NaN	https://www.indeed.com/cmp/H-A-Mapes,-Inc?camp...
2	00027f45e5373e13	https://www.indeed.com/applystart?jk=00027f45e...	https://www.indeed.com/cmp/Accenture?campaigni...
3	00028cda307fcffa	NaN	https://www.indeed.com/cmp/Techo--bloc?campaig...

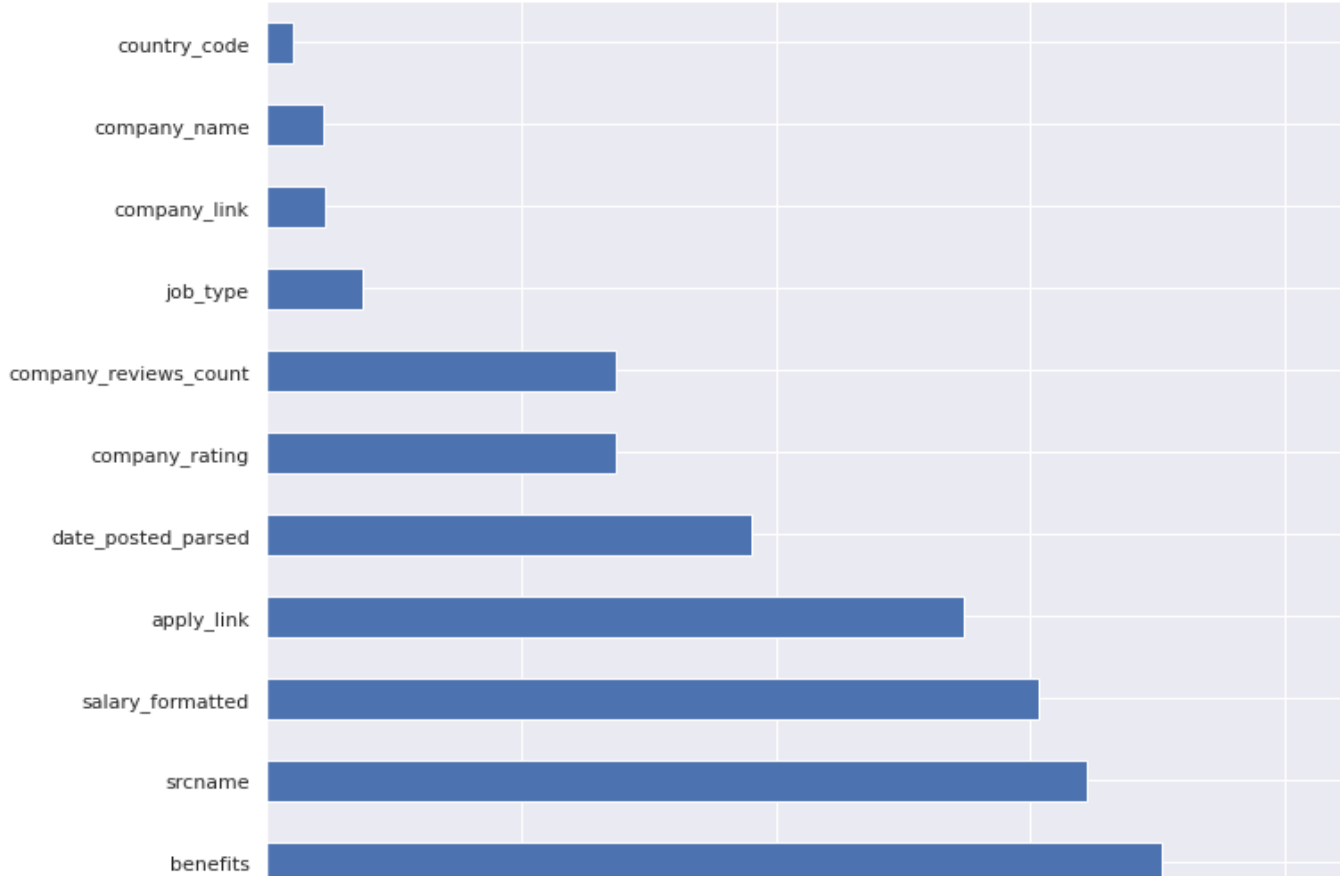
Check Missing Values and Clean Up

Truc Report:

- Data cleaning took me a total of more than 8hrs to looks for the approriate data that need to keep or drop.
- All the attributes need to make sense and support the machine learning model
- Data that consider biased will be drop
- Data that is missing need to fix and transform to meaningful data

```
# Set figure size
plt.rcParams["figure.figsize"]=13,11
sns.set(style='darkgrid')

missing_percentage = dfjob.isna().sum().sort_values(ascending=False)/len(dfjob)
missing_percentage[missing_percentage!=0].plot(kind='barh')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8979d66e10>
```




```
list(dfjob.columns)
```

```
['jobid',
 'apply_link',
 'company_link',
 'company_name',
 'company_rating',
 'company_reviews_count',
 'country',
 'country_code',
 'current_url',
 'date_posted',
 'date_posted_parsed',
 'description',
 'description_text',
 'domain',
 'job_title',
 'job_type',
 'location',
 'region',
 'salary_formatted',
 'benefits',
 'qualifications',
 'srcname']
```

```
dfjob.drop(['jobid','apply_link','company_link','country','current_url','date_posted','date_p
```

```
# Out put will be company name and job title
# Remove apply_link because it will not be necessary to have it (we want to analyze the suces
# Apply_link can be removed when the job is filled which is a good sign to analyze these job

# Drop the row where the company name or link is blank:
dfjob.dropna(axis=0, how='all',subset=['company_name', 'job_type'], thresh=2, inplace=True)

# Change null in qualification to no requirement
dfjob['qualifications'] = dfjob['qualifications'].fillna('["No requirement"]')

# Change null in benefits to no benefits
dfjob['benefits'] = dfjob['benefits'].fillna('["No benefits"]')

# Assume all the mssing value in salary_formated is negotiable (50% of the dataset)
dfjob['salary_formatted'] = dfjob['salary_formatted'].fillna('Negotiable')

# Assume all the missing country is Others
dfjob['country_code']=dfjob['country_code'].fillna('Other')

# Fill in the rating with 0
dfjob['company_rating']=dfjob['company_rating'].fillna(0.0)
dfjob['company_reviews_count']=dfjob['company_reviews_count'].fillna(0.0)

dfjob.shape

# After clean up and drop, we have a new data set of 265633 row and 12attributes
```

```
(265633, 12)
```

Extended Analyst on the company rating and company review

```
# Create norating subset that hold the company doesn't has rating
norating = dfjob.loc[dfjob['company_rating']==0]

# Validate the result
norating.head()
```

	company_name	company_rating	company_reviews_count	country_code	description	des
1	Harry's Convenience Stores	0.0	0.0	LB	<p>At Harry’s the Store Associate / Foodservic...	At
6	The Michigan Theater Foundation	0.0	0.0	US	<p>The historic Michigan Theater, located in t...	M
11	Facing History and Ourselves, Inc	0.0	0.0	US	\n <p> <b>Position: Director of Developmen...	Po
16	The Eye Care	0.0	0.0	US	<p> <b>Education and	

```
# Use value counts to check the name of the company
norating['company_name'].value_counts

# Turn out all the company in this section only post their job one time
```

```
<bound method IndexOpsMixin.value_counts of 1                    Harry's Convenience
Stores
6               The Michigan Theater Foundation
11            Facing History and Ourselves, Inc
16                      The Eye Care Institute
22          Copper Whiskey Bar & Grill, Bozeman
                          ...
29981                                  SunStop
29982    Ford - Lincoln Veteran Careers Program
29988                   Acro Metal Stamping Co.
29992                         Messick and Gray
29997                               Sanel Corp
Name: company_name, Length: 66460, dtype: object>
```

```
# Validate if there is any rating associate with the number of user
sum(norating['company_rating']==norating['company_reviews_count'])
```

```
66460
```

```
# Check the shape of the dataset
norating.shape
```

```
(66460, 12)
```

*Result: *

- Due to all the above analyst, I can conclude the norating consist of the company who only post their job one time. So the change to promote in these company is small due to the amount of jobs posted and rating. There for I will remove the row associate with these company where the change is small and the review is none.

```
## Delete norating since we do not need it
del norating
```

```
# Drop the row where the company name or link is blank:
dfjob = dfjob[(dfjob.company_rating != 0) & (dfjob.company_reviews_count != 0)]
```

```
# [name][title][rating][count] list or string
# List of the top 50 we found
```

Double-click (or enter) to edit

```
dfjob.head()
```

company_name	company_rating	company_reviews_count	country_code	description	des
The Estée				\n <p>The	

```
dfjob.shape
```

```
(199173, 12)
```

Double-click (or enter) to edit

```
# Open text file resume
file1 = open('./drive/MyDrive/resume.txt', 'r')
resume_data = []

while True:
    # Get next line from file
    line = file1.readline()
    resume_data.append(line)

    # if line is empty or end of file is reached
    if not line:
        break

file1.close()
```

```
resume_data
```

```
['Truc Huynh\t\t\n',
 '[jackyhuynh87@gmail.com] \t\n',
 '[https://www.linkedin.com/in/trucdev/]\n',
 '[https://github.com/jackyhuynh]\n',
 'Experienced full-stack developer, project management & coordination through
different industries with a strong technical background and analyzing skills. \n',
 'EDUCATION\n',
 'Master of Science, Computer Science [Purdue University Fort Wayne, Fort Wayne,
IN]\n',
 'Certificate in Data Scientist & Machine Learning [North Carolina State
University]\n',
 'Bachelor of Science, Computer Science [ECPI University, Newport News, VA]\n',
 'Associate in Cyber Security [De Anza College]\n',
 'Introduce to Self-Driving Car Certificate [Udacity.com]\n',
 'Python, React, Git, Docker, web development: self-learn [Udemy, HackerRank,
Codility, and DataCamp]\n',
 'SKILLS\n',
 'Proficient with Office 365, MS Project, MS Visio, Jira, Git, GitHub, and Slack\n',
 'Experience in web application development with (HTML, CSS, JSON, Bootstrap, R
Shiny, Python web server framework, Flask, RESTful API, and JavaScript, Heroku
(PaaS))\n',
 'Experience in the software development lifecycle, Agile methodologies, data
visualization, dashboard design (BI)\n',
```

```
 'Classroom experience in data mining, machine learning, ML/AI (Python, R), database
technologies (SQL, MySQL), data analytics, data structures & algorithms (Python, C/
C++, Java), framework (Spring MVC, NET., Flask), IDE (Eclipse, Visual Studio, VS
Code, Anaconda, PyCharm, Jupiter Notebook), Servlet (Apache Tomcat), automation,
bot, software testing, script, ethical hacking, cyber security, UI/UX Design\n',
 'Experience in relationship building and explaining technical concepts to non-
technical audiences. Highly collaborative, team-oriented, and pay attention to
detail with the ability to multitask\n',
 'Working knowledge with Virtual machines, Windows, Linux, Mac OSX, AutoCAD, 3D
Modelling\n',
 'Experience with inventory planning and supply chain management (forecasting,
planning, optimization & logistics)\n',
 'EXPERIENCE\n',
 'Co-founder & Program Manager: [7 Figures Trader]\n',
 'Build an online training program to teach people to trade (stock, options, futures
contracts). Work and strategize with others on cross-functional teams and
stakeholders to design the web application and organization’s structure. Design the
application prototype, content, communication methods, process of implementation,
and risk management strategies. Using previous data to predict the next coming
trade, monitor market volatility, and create the trading plans for the coming
weeks.\n',
 'Co-founder & Business Analyst: [Luxe Nails & Spa L.L.C]\n',
 'Lead cross-functional decision-making & implementation to define infrastructure
plan, business support systems, and operation plan. Provided advice and instruction
on work methods, practices, and procedures to improve profits (through marketing,
social media management, and staff training). Financial monitoring, budget
management, internal management, and customer services. The result shows an increase
in the sales records by 250% (second year) compared to the first fiscal year\n',
 'Logistic Specialist & Watercraft Engineer: [US Army • Fort Eustis, VA]\n',
 'Developing an inventory record and tracking database that ensures VSO warehouse
equipment and inventory are 100% accountable. This database was evaluated to benefit
my company in years (Army Achievement Medals & Commander’s Recommendation Letters).
Maintaining inventory records of the Vessel Support Office. Preparing work
schedules, and routine reports to the higher headquarters. Inventory planning and
supply chain management \n'
```

```
# Clean up the resume
import re

# Clean up address, school, name, number, take only character in to the new string list
for i in range(0,len(resume_data)):
    resume_data[i] = re.sub(r'\[.*?\]', '', resume_data[i])
    word1 = " ".join(re.findall("[a-zA-Z]+", resume_data[i]))
    resume_data[i] = word1

# Using the keywords dictionary to hold all the keyword
keyword_dict = []

for line in resume_data:
    li = list(line.split(" "))
    for string_ in li:
        keyword_dict.append(string_.lower()) # Convert the string to lower

# Character that does not necessary to the search can be removed
remove_characters = ['','a','truc','huynh','through','self','classroom','ide','concepts','fou
```

```
                    'an','to','on','and','that','this','the','by','in','with','s','of','non'
                    'may','guided','submit','vietnam','cis','any','unsatisfied','services','
                    'customer','ensure','supply','work','year','plans','customer','developin
                    'ensures','supply','options','learn','master','recommendation','science'
                    'previous','concerns','structures','budget','next','methods','stakeholde
                    'visual','higher','coming','teaching','letters','chain','content','tradi
                    'advice','highly','shows','toward','commander','compare','fiscal','direc
                    'ethical','teach','trade']

soft_skill_remove = ["structure", "experience", "requirements", "worked", "years", "others",
                    "company", "information", "plan", "knowledge", "benefit", "process", "tr
                    "provided", "business", "operation", "systems", "oriented", "level", "ba
                    "reports", "office", "people", "certificate", "pay", "industries", "acco
                    "maintaining", "design", "record", "clients", "bachelor", "projects", "i
                    "meet", "implementation", "sales", "background", "detail", "preparing",
                    "marketing", "result", "weeks", "testing", "financial", "security", "pro
                    "driving", "first", "futures", "instruction", "contracts", "strategies",

for char in remove_characters:
    while(char in keyword_dict) :
        keyword_dict.remove(char)

for char in soft_skill_remove:
    while(char in keyword_dict) :
        keyword_dict.remove(char)

# remove the repeated word in the dictionary
keyword_dict = list(dict.fromkeys(keyword_dict))

# Figure out the length of the keyword dictionaries
len(keyword_dict)
```

```
102
```

```
def pretty_print(word_list):
  index = 1
  for word in word_list:
    print(word, end=', ')
    if index % 10 == 0:
      print('')
    index += 1


# Display list of dictionary
pretty_print(keyword_dict)
```

```
stack, developer, management, analyzing, data, scientist, machine, cyber, introduce, pyt
react, git, docker, web, development, ms, visio, jira, github, slack,
html, css, json, bootstrap, r, shiny, server, framework, flask, restful,
api, javascript, heroku, paas, lifecycle, agile, methodologies, visualization, dashboard
mining, ml, ai, database, sql, mysql, algorithms, c, java, spring,
```

```
mvc, net, eclipse, studio, vs, code, anaconda, pycharm, jupiter, notebook,
servlet, apache, tomcat, automation, bot, script, hacking, ui, ux, explaining,
multitask, virtual, machines, windows, linux, mac, osx, autocad, d, modelling,
inventory, planning, forecasting, optimization, logistics, teams, prototype, predict, vc
infrastructure, social, media, compared, logistic, vso, army, medals, vessel, lab,
coding, research,
```

Model 1: Nature Language Processing (NLP) with Parts of Speech (POS) tokenization for Keyword Filtering:

Approach will be:

- Sampling
- POS Token

1) Sampling

The amount of words is too large, I will reduce the data to 10,000 row so that we can see how our model works on the smaller scale (sample is 5% of the data)

```
sample_size=10000

# Create a sample of 10,000 rows
dfjob_s_1 = dfjob.sample(n=sample_size)

# Validate the transactions
dfjob_s_1.head()
```

	company_name	company_rating	company_reviews_count	country_code	desc
11452	Famous Footwear	3.6	1944.0	US	\n Overvie At Famous F
71	Deloitte	4.0	10699.0	US	\n In th disruption, orga
					\n \n

```
# Has to import NLTK and download averaged_perceptron_tagger
import nltk
from nltk import pos_tag
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize

nltk.download('averaged_perceptron_tagger')
nltk.download('punkt')
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
True
```

```
tags = pos_tag(keyword_dict)

tags_list = []
for tag in tags:
  tags_list.append(tag[1])

tags_list = list(dict.fromkeys(tags_list))

# Figure out the length of the keyword dictionaries
len(tags_list)
```

```
8
```

```
tags
```

```
('sql', 'NN'),
('mysql', 'NN'),
('algorithms', 'IN'),
('c', 'JJ'),
('java', 'NN'),
('spring', 'NN'),
('mvc', 'JJ'),
```

```
('mvc', 'JJ'),
('net', 'JJ'),
('eclipse', 'NN'),
('studio', 'NN'),
('vs', 'NN'),
('code', 'NN'),
('anaconda', 'IN'),
('pycharm', 'NN'),
('jupiter', 'NN'),
('notebook', 'NN'),
('servlet', 'NN'),
('apache', 'NN'),
('tomcat', 'NN'),
('automation', 'NN'),
('bot', 'IN'),
('script', 'NN'),
('hacking', 'VBG'),
('ui', 'JJ'),
('ux', 'JJ'),
('explaining', 'VBG'),
('multitask', 'JJ'),
('virtual', 'JJ'),
('machines', 'NNS'),
('windows', 'NNS'),
('linux', 'VBP'),
('mac', 'JJ'),
('osx', 'JJ'),
('autocad', 'NN'),
('d', 'NN'),
('modelling', 'VBG'),
('inventory', 'NN'),
('planning', 'VBG'),
('forecasting', 'VBG'),
('optimization', 'NN'),
('logistics', 'NNS'),
('teams', 'VBP'),
('prototype', 'JJ'),
('predict', 'NN'),
('volatility', 'NN'),
('analyst', 'NN'),
('infrastructure', 'NN'),
('social', 'JJ'),
('media', 'NNS'),
('compared', 'VBN'),
('logistic', 'JJ'),
('vso', 'NN'),
('army', 'NN'),
('medals', 'NNS'),
('vessel', 'VBP'),
('lab', 'JJ'),
('coding', 'NN'),
('research', 'NN')]
```

Apply POS tagging

```
ps = PorterStemmer()


# process the job description.
def prepare_job_desc(desc):
    # tokenize description.
    tokens = word_tokenize(desc)

    # Parts of speech (POS) tag tokens.
    token_tag = pos_tag(tokens)

    # Only include some of the POS tags.
    include_tags = ['VBN', 'VBD', 'JJ', 'JJS', 'JJR', 'CD', 'NN', 'NNS', 'NNP', 'NNPS']
    filtered_tokens = [tok for tok, tag in token_tag if tag in include_tags]

    # stem words.
    stemmed_tokens = [ps.stem(tok).lower() for tok in filtered_tokens]
    return set(stemmed_tokens)


dfjob_s_1['job_description_word_set'] = dfjob_s_1['description_text'].map(prepare_job_desc)

# process the keywords
tool_keywords_set = set([ps.stem(tok) for tok in keyword_dict]) # stem the keywords (since th
tool_keywords_dict = {ps.stem(tok):tok for tok in keyword_dict} # use this dictionary to reve


tool_keywords_dict
```

```
      'jira': 'jira',
      'json': 'json',
      'jupit': 'jupiter',
      'lab': 'lab',
      'lifecycl': 'lifecycle',
      'linux': 'linux',
      'logist': 'logistic',
      'mac': 'mac',
      'machin': 'machines',
      'manag': 'management',
      'medal': 'medals',
      'media': 'media',
      'methodolog': 'methodologies',
      'mine': 'mining',
      'ml': 'ml',
      'model': 'modelling',
      'ms': 'ms',
      'multitask': 'multitask',
      'mvc': 'mvc',
      'mysql': 'mysql',
      'net': 'net',
      'notebook': 'notebook',
      'optim': 'optimization',
      'osx': 'osx',
```

```
 'paa': 'paas',
 'plan': 'planning',
 'predict': 'predict',
 'prototyp': 'prototype',
 'pycharm': 'pycharm',
 'python': 'python',
 'r': 'r',
 'react': 'react',
 'research': 'research',
 'rest': 'restful',
 'scientist': 'scientist',
 'script': 'script',
 'server': 'server',
 'servlet': 'servlet',
 'shini': 'shiny',
 'slack': 'slack',
 'social': 'social',
 'spring': 'spring',
 'sql': 'sql',
 'stack': 'stack',
 'studio': 'studio',
 'team': 'teams',
 'tomcat': 'tomcat',
 'ui': 'ui',
 'ux': 'ux',
 'vessel': 'vessel',
 'virtual': 'virtual',
 'visio': 'visio',
 'visual': 'visualization',
 'volatil': 'volatility',
 'vs': 'vs',
 'vso': 'vso',
 'web': 'web',
 'window': 'windows'}
```

```
pretty_print(tool_keywords_set)
```

```
prototyp, linux, paa, machin, web, autocad, net, mine, notebook, visio,
heroku, slack, databas, analyz, social, scientist, c, data, mysql, algorithm,
manag, visual, github, ui, python, r, mac, media, lab, code,
bot, volatil, autom, vso, script, model, dashboard, javascript, infrastructur, pycharm,
ml, forecast, studio, shini, server, ms, logist, mvc, vs, json,
ai, predict, flask, eclips, inventori, java, agil, docker, explain, lifecycl,
anaconda, virtual, plan, window, framework, vessel, bootstrap, css, d, research,
apach, osx, hack, sql, optim, armi, analyst, team, develop, react,
html, api, introduc, spring, tomcat, servlet, stack, cyber, jira, rest,
git, medal, methodolog, ux, jupit, compar, multitask, bi,
```

```
tool_list = []
```

```
msk = dfjob_s_1['country_code'] != '' # just in case you want to filter the data.
```

```
num_postings = len(dfjob_s_1[msk].index)
for i in range(num_postings):
    job_desc = dfjob_s_1[msk].iloc[i]['description_text'].lower()
    job_desc_set = dfjob_s_1[msk].iloc[i]['job_description_word_set']

    # check if the keywords are in the job description. Look for exact match by token.
    tool_words = tool_keywords_set.intersection(job_desc_set)

    # label the job descriptions without any tool keywords.
    if len(tool_words) == 0:
        tool_list.append('nothing specified')

    tool_list += list(tool_words)

# Sample print ot of job description at index 12 after transformation

pretty_print(dfjob_s_1['job_description_word_set'].iloc[12])
```

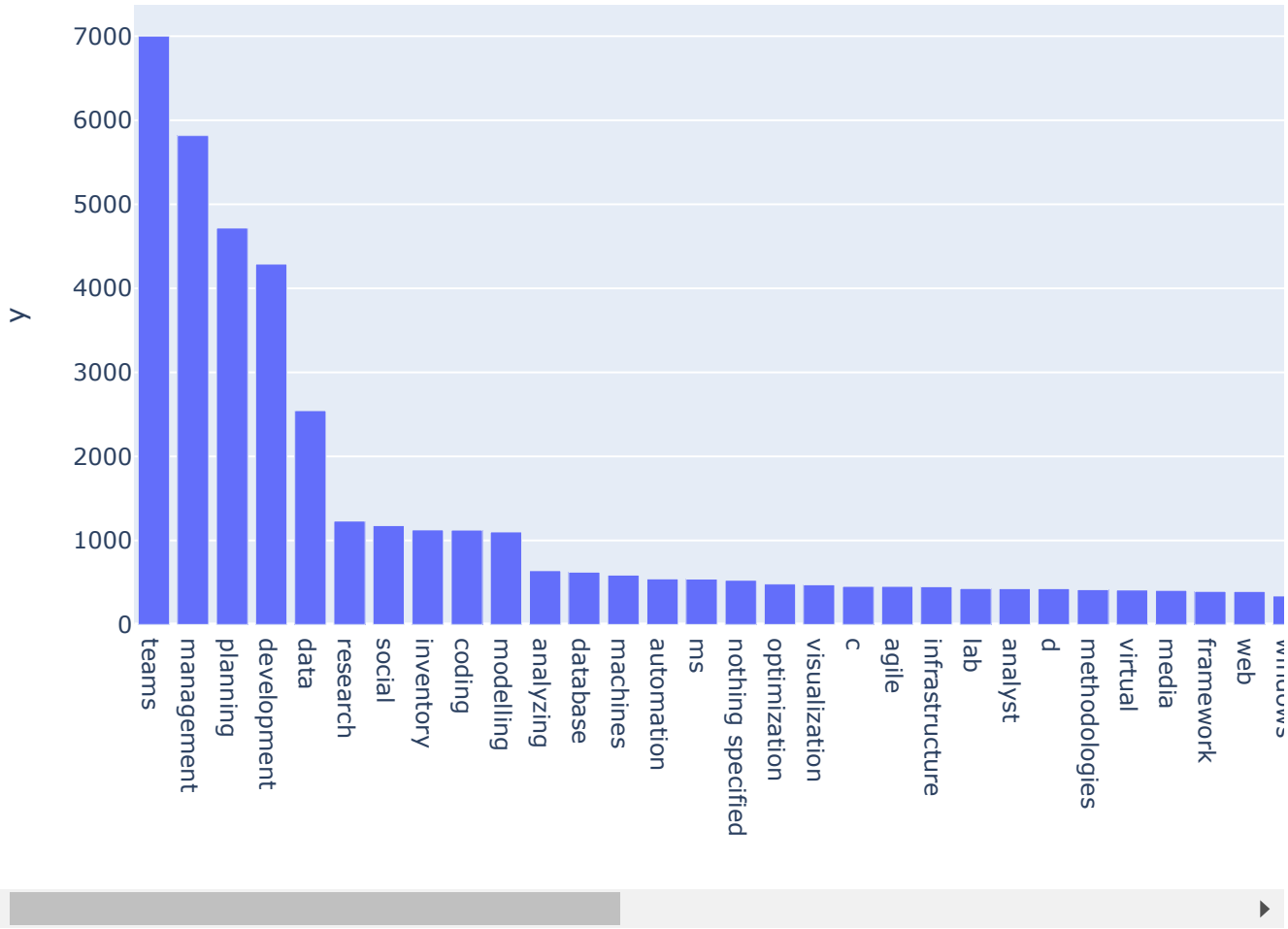
```
experi, happi, 5:00pm, nation, written, patern, travel, equival, record, verbal,
prefer, next, transport, color, real, servic, maintain, test, except, strong,
statement, blood, skill, manner, environ, opportun, center, flexibl, vital, eoe/aa,
live, certif, data, reliabl, log, race, patient, cleric, entri, accredit,
superior, prepar, employ, abl, career, age, payment, profession, great, day,
duties/respons, 8:30a-, duti, requir, time, minim, healthi, diploma, veteran, religion,
techniqu, equal, necessari, track, work, statu, practic, need, standard, comfort,
organ, differ, inform, specimen, due, monday-friday, school, blind, person, perform,
role, peopl, phlebotomist, abil, offic, sexual, proud, face, drug, addit,
venipunctur, proven, site, commun, schedul, continu, client, chanc, phlebotomi, custom,
screen, job, alcohol, pleas, applic, privaci, team, gender, accur, develop,
high, provid, ident, capillari, clean, drive, challeng, group, bill, agenc,
health, more, previou, process, labcorp, collect, passion, step, growth, sex,
administr, origin, orient, disabl, compani, analysi, today, supervis,
```

```
# create the list of tools.
df_tool = pd.DataFrame(data={'cnt': tool_list})
df_tool = df_tool.replace(tool_keywords_dict)

df_tool_top = df_tool['cnt'].value_counts().reset_index().rename(columns={'index': 'tool'}).i



df_tool_top.head()
```

	tool	cnt	grow_percentage
0	teams	7001	0.7001

```
from plotly import __version__
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
import plotly.graph_objs as go

# visualize the tools.
layout = dict(
    title='Top Skill base on Resume',
    yaxis=dict(
        title='% of job postings',
        tickformat=',.0%',
    )
)

fig = go.Figure(layout=layout)
fig.add_trace(go.Bar(
    x=df_tool_top['tool'],
    y=df_tool_top['cnt']/num_postings
))

iplot(fig)
```

```
import plotly.express as px

fig = px.bar( x=df_tool_top['tool'],
    y=df_tool_top['cnt'])
fig.show()
```



Find most align

Using the list just found and find the most

```
pretty_print(df_tool_top['tool'])
```

```
teams, management, planning, development, data, research, social, inventory, coding, mod
analyzing, database, machines, automation, ms, nothing specified, optimization, visualiz
infrastructure, lab, analyst, d, methodologies, virtual, media, framework, web, windows,
server, logistic, sql, lifecycle, python, script, forecasting, java, restful, r,
compared, stack, spring, api, scientist, predict, javascript, linux, dashboard, studio,
net, ai, react, bi, algorithms, explaining, jira, prototype, autocad, cyber,
```

```
df_tool_top['grow_percentage'] = df_tool_top['cnt']/sample_size
```

```
keyword_list_after_clean = list(df_tool_top['tool'])
```

```
pretty_print(keyword_list_after_clean)
```

```
teams, management, planning, development, data, research, social, inventory, coding, moc
analyzing, database, machines, automation, ms, nothing specified, optimization, visualiz
infrastructure, lab, analyst, d, methodologies, virtual, media, framework, web, windows,
server, logistic, sql, lifecycle, python, script, forecasting, java, restful, r,
compared, stack, spring, api, scientist, predict, javascript, linux, dashboard, studio,
net, ai, react, bi, algorithms, explaining, jira, prototype, autocad, cyber,
```

```
df_tool_top.head()
```

	tool	cnt	grow_percentage
0	teams	7001	0.7001
1	management	5818	0.5818
2	planning	4718	0.4718
3	development	4289	0.4289
4	data	2545	0.2545

Result:

According to the key word list and the percentage, we can extract many information such as:

- Softskill: teams, management, planning, development
- Hardskill: data, modelling, research, inventory, social, coding, analyzing, machines, optimization, database, visualization, methodologies, automation, virtual, infrastructure, media, lab, web, framework, script, agile, forecasting, server, logistic, lifecycle
- Pogramming languages: Python, r, spring, javascript, scientist, compared, studio, linux, dashboard, bi, jira, prototype, algorithms, ai, react, net, cyber, explaining, visio

These are the soft skill, hard skill and programming langages are most in demand.

Model 2: Bags of Words and Cosine Similarity

```
sample_size=10000
```

```
# Create a sample of 10000 rows
dfjob_s_2 = dfjob.sample(n=sample_size)

# Validate the transactions
dfjob_s_2.head()
```

	company_name	company_rating	company_reviews_count	country_code	description
8681	Orthofix	3.5	54.0	US	\n \n Why Orthofix?\n \n <br...
26121	Touchpoints at Farmington	3.7	70.0	US	\n <p>iCare Health Network - Touchpoints ...
15881	Allied Universal	3.1	23770.0	US	\n \n Allied Universal®, North Amer...
13636	Lowe's	3.5	48523.0	US	\n \n What You Will Do\n <b...
19684	First Communities	3.3	174.0	US	\n <p> General Job Description Purpose:...

```
# drop for SVM, Bag of words and cosine similarity
dfjob_s_2.drop(['country_code', 'description', 'job_type', 'salary_formatted', 'benefits'],ax

dfjob_s_2.describe(include='all').T
```

	count	unique	top	freq	mean	std	min
company_name	10000	6004	Deloitte	195	NaN	NaN	NaN
company_rating	10000.0	NaN	NaN	NaN	3.5684	0.508383	1.0
company_reviews_count	10000.0	NaN	NaN	NaN	3948.7454	13143.506281	2.0
			Assurance believes that				

```
ps = PorterStemmer()

# process the job description.
def prepare_job_desc(desc):
    # tokenize description.
    tokens = word_tokenize(desc)

    # Parts of speech (POS) tag tokens.
    token_tag = pos_tag(tokens)

    # Only include some of the POS tags.
    include_tags·=·['VBN',·'VBD',·'JJ',·'JJS',·'JJR',·'CD',·'NN',·'NNS',·'NNP',·'NNPS']
    filtered_tokens = [tok for tok, tag in token_tag if tag in include_tags]

    # stem words.
    stemmed_tokens = [ps.stem(tok).lower() for tok in filtered_tokens]
    return set(stemmed_tokens)

dfjob_s_2['keywords'] = dfjob_s_2['description_text'].map(prepare_job_desc)


# Convert the data to list
dfjob_s_2['keywords'] = dfjob_s_2['keywords'].apply(list)

# Drop the unecessary_columns
dfjob_s_2.drop(['description_text'], axis=1, inplace=True)

# Merge the first 5 columns to create the company portfolio
dfjob_s_2['jobs_all_information'] = dfjob_s_2[dfjob_s_2.columns[0:4]].apply(
    lambda x: '|'.join(x.dropna().astype(str)),
    axis=1)

# remove unecessary attributes
dfjob_s_2.drop(['company_name', 'company_rating', 'company_reviews_count', 'job_title', 'loca

# Now merge the bag of words together
dfjob_s_2['bag_of_words'] = dfjob_s_2[dfjob_s_2.columns[1:2]].apply(
    lambda x: ','.join(x.dropna().astype(str)),
    axis=1)

# Drop the unecessary columns
dfjob_s_2.drop(['qualifications','keywords'], axis=1, inplace=True)
```

```
# convert to string
dfjob_s_2['bag_of_words'] = dfjob_s_2['bag_of_words'].apply(str)

# filter out unecessary element
dfjob_s_2['bag_of_words'] = dfjob_s_2['bag_of_words'].apply(lambda x: x.replace("'",''))
dfjob_s_2['bag_of_words'] = dfjob_s_2['bag_of_words'].apply(lambda x: x.replace(",",''))

# reset index
dfjob_s_2 = dfjob_s_2.reset_index(drop=True)
```

```
dfjob_s_2.head()
```

	jobs_all_information	bag_of_words
0	Orthofix|3.5|54.0|Regional Sales Manager - Bio...	[experi computer-bas nondiscrimin healthcar de...
1	Touchpoints at Farmington|3.7|70.0|Maintenance...	[experi exterior ged equip week healthcar rela...
2	Allied Universal|3.1|23770.0|Security Officer ...	[experi action visit vision benefit deter nati...
3	Lowe's|3.5|48523.0|FT-Head Cashier-Flexible	[experi benefit nation import like label coach...
4	First Communities|3.3|174.0|Maintenance Techni...	[experi made hous gutter equip sewer outdoor r...

```
# Only run 1 to add the value to the
# Create the whole string keyword
keyword_dict_str = ' '.join([str(elem) for elem in keyword_dict])

# Add to the end of the resume
dfjob_s_2.loc[len(dfjob_s_2.index)] = ['new_candidates_resume',keyword_dict_str]


# drop if the data is not match
# dfjob_s_2 = dfjob_s_2. drop (index=[1000, 1001, 1002, 1003])


from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import CountVectorizer

# Transform the
count = CountVectorizer()
count_matrix = count.fit_transform(dfjob_s_2['bag_of_words'])

# Create the matrix to compare
cosine_sim = cosine_similarity(count_matrix, count_matrix)

# create a Series of job titles, so that the series index can match the row and column index
indices = pd.Series(dfjob_s_2['jobs_all_information'])
```

```
# Validate similarity matrix
print(cosine_sim)
```

```
[[1.         0.1404583  0.30683112 ... 0.2930527  0.30705185 0.00553255]
 [0.1404583  1.         0.15644203 ... 0.1400346  0.15746301 0.        ]
 [0.30683112 0.15644203 1.         ... 0.31737699 0.24806099 0.        ]
 ...
 [0.2930527  0.1400346  0.31737699 ... 1.         0.25076246 0.00551586]
 [0.30705185 0.15746301 0.24806099 ... 0.25076246 1.         0.04771042]
 [0.00553255 0.         0.         ... 0.00551586 0.04771042 1.        ]]
```

```
def recommend(resume_title, cosine_sim = cosine_sim):
    recommended_jobs = []

    idx = indices[indices == resume_title].index[0]
    score_series = pd.Series(cosine_sim[idx]).sort_values(ascending = False)
    top_10_indices = list(score_series.iloc[1:11].index)

    for i in top_10_indices:
        recommended_jobs.append(list(dfjob_s_2['jobs_all_information'])[i])

    return recommended_jobs
```

```
recommend('new_candidates_resume', cosine_sim = cosine_sim)
```

```
['Saxony Partners|3.0|2.0|Front-End Developer',
 'Bank of America|3.8|31272.0|Software Engineer III',
 'Fiserv, Inc.|3.4|6251.0|Senior Software Developer',
 'KROS-WISE|4.1|10.0|Web Application Developer',
 'CyberCoders|3.6|57.0|Senior Full Stack .NET Developer',
 'Virbela|2.5|2.0|Senior Software Engineer. - 100% REMOTE',
 'WEX Inc.|3.3|196.0|Software Engineer (Applications) 3',
 'Deloitte|4.0|10699.0|Full Stack Senior Developer',
 'Deloitte|4.0|10699.0|Lead Application Developer - DAS Project Omnia',
 'WELLS FARGO BANK|3.7|42353.0|Lead Financial and Risk Modelling Business Analyst']
```

Support Functions

```
# Garage collections
gc.collect()
```

```
408
```

0s completed at 11:51 AM