1. Reading the csv files

- We have scraped the resumes for 3 different job titles such as
  - Data Scientist
  - Senior Software engineer
  - Vice President

Labeling the Data

- We have manually given the labels to each different resume type while scraping.
- Now in below code, we are importing each of the datasets separately.

In [ ]:

```
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
import csv
import numpy as np

with open("indeed_scraped_data_science.csv", "r",encoding="ISO-8859-1") as
f:
    reader = csv.reader(f, delimiter=',')
    text1 = [(row[8]) for row in reader if row[0]=='1']


with open("indeed_scraped_data_science.csv", "r",encoding="ISO-8859-1") as
f:
    reader = csv.reader(f, delimiter=',')
    text2 = [(row[8]) for row in reader if row[0]=='2']

with open("indeed_scraped_data_science.csv", "r",encoding="ISO-8859-1") as
f:
    reader = csv.reader(f, delimiter=',')
    text3 = [(row[8]) for row in reader if row[0]=='3']
```

2. Tokenization

- By using nltk corpus and collocations, we removed punctuations and stop words.
- Also in the above code as you can see, instead of UTF-8 we used ISO-8859 in order to remove non-ASCII characters.
- Also applied regular expressions to clean the data.

In [41]:

```
import string
import nltk
import re
import csv
import ast

from re import compile
```

```
#for sorting dictionary
import operator
from nltk.corpus import stopwords
from nltk.collocations import *

def tokenize(text):
    tokens=[]
    # write your code here
    stop_words = stopwords.words('english')
    text = text.lower()
    #print(text)
    #pattern = r'\w'
    tokens = nltk.word_tokenize(text)
    tokens = [token.strip(string.punctuation) for token in tokens]

    tokens = [token for token in tokens if re.match("^[A-Za-z_-]*$",
token)]
    #print(tokens)
    # to remove punctuations from begging and starting of the tokens

    # now removing extra empty characters from tokens
    tokens = [token.strip() for token in tokens if token.strip()!='']
    tokens = [token for token in tokens if len(token)>1]
    tokens = [token for token in tokens if token not in stop_words]
    return tokens
```

3. tokenization on every dataset

- Using Keras Tokenizer we are tokenizing each of the important columns to get the meaningful data.
- the below example is for Project Lead.

In [42]:

```
# get a Keras tokenizer

MAX_NB_WORDS=8000
# documents are quite long in the dataset
MAX_DOC_LEN=1000

tokenizer1 = Tokenizer(num_words=MAX_NB_WORDS)
tokenizer1.fit_on_texts(text1)

tokenizer2 = Tokenizer(num_words=MAX_NB_WORDS)
tokenizer2.fit_on_texts(text2)

tokenizer3 = Tokenizer(num_words=MAX_NB_WORDS)
tokenizer3.fit_on_texts(text3)

input_tokenizer = Tokenizer(num_words=MAX_NB_WORDS)
input_tokenizer.fit_on_texts(text1[0])

with open("project_lead.csv", "r",encoding="ISO-8859-1") as f:
    reader=csv.reader(f, delimiter=',')
    rows=[(row[8]) for row in reader]

#print(rows[0])
#input_tokens = []
```

```
input_tokens = tokenize(rows[10])


#print(input_tokens)
```

4. top words frequency

- Getting the top frequent words from each of the columns in descending order.

In [43]:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

df=pd.DataFrame.from_dict(tokenizer1.word_counts, orient="index")
df.columns=['freq']
print(df.head())
df=df['freq'].value_counts().reset_index()
df.columns=['word_freq','count']

df=df.sort_values(by='word_freq')
#print(tokenizer1.word_counts)
print(df.head())

df=pd.DataFrame.from_dict(tokenizer2.word_counts, orient="index")
df.columns=['freq']
print(df.head())
df=df['freq'].value_counts().reset_index()
df.columns=['word_freq','count']

df=df.sort_values(by='word_freq')
#print(tokenizer1.word_counts)
print(df.head())


df=pd.DataFrame.from_dict(tokenizer3.word_counts, orient="index")
df.columns=['freq']
print(df.head())
df=df['freq'].value_counts().reset_index()
df.columns=['word_freq','count']

df=df.sort_values(by='word_freq')
#print(tokenizer1.word_counts)
print(df.head())
```

```
             freq
description   323
fis            23
provides      190
financial     425
software      637
   word_freq  count
1          1   3129
0          2   3476
2          3   1239
3          4   1201
5          5    529
```

```
           freq
'product     25
package     122
planner      23
united       32
parcel        4
   word_freq  count
0          1   7634
1          2   4459
2          3   1506
3          4   1253
4          5    642
              freq
'taught          6
students        64
the           2589
introduction     6
to            2587
   word_freq  count
0          1   5065
1          2   1752
2          3    783
3          4    521
4          5    380
```

5. Analysis on Average Work Experience

- The below function will convert the given work duration into number of years.
- And then will count the average number of years.
- For e.g. if in the given column the date is given as January, 2017 to January, 2018 - it will result in [1] year.
- It also deals with the inputs which has values like "January, 207 to Present"

In [44]:

```
import string
import nltk
import re
import csv
import ast
#for initial data filtering
import preprocessing
from re import compile
#for sorting dictionary
import operator
from nltk.corpus import stopwords
from nltk.collocations import *
bigram_measures = nltk.collocations.BigramAssocMeasures()



def dateToSum(filename):
    temp = 0
    with open(filename, "r",encoding="ISO-8859-1") as f:
        reader = csv.reader(f, delimiter=',')

        rows = [row[7] for row in reader]
```

```
        tot = len(rows)
        # print(tot)
        for rro in rows:
            if len(rro) > 0:
                exp_len = ast.literal_eval(rro)
                # print(exp_len)
                for t in exp_len:
                    tokens = nltk.word_tokenize(t)
                    for i in range(len(tokens)):
                        if tokens[i] == "NA":
                            tokens[i] = ""
                        elif tokens[i] == "Present":
                            tokens[i] = "2018"
                    rex = compile('[^0-9]')
                    filteredData = [x for x in tokens if not rex.match(x)]

                    if len(filteredData) == 2:
                        temp = (int(filteredData[1]) - int(filteredData[0]))
+ temp

        print("\n\nAverage Work Experience : ")
        print(round(temp/tot))
```

6.Dataset Analysis of every column

- Previous Job Descriptions.
- Previous Job Titles.
- Skills.
- Previous Education Details.

analysis of the same in the effective and more user friendly graphs can be found in the PPT.

In [52]:

```
def performance_evaluate(input_file):
    trigrams_list =[]
    bigrams_list=[]
    # write your code here
    with open(input_file, "r",encoding="ISO-8859-1") as f:
        reader=csv.reader(f, delimiter=',')
        # row[1] or second column is header of resume which states the curr
ent job position.
        # Although we did data scraping for particular job positions, curre
nt job positions varied a lot
        # in initial analysis phase- so even after scraping resumes for one
particular position, we are making sure the
        # actual job postion for which analysis will be done.
        rows=[(row[1], row[5], row[4], row[10], row[3],row[8]) for row in re
ader]



    #print(row_len)
    for i in rows:
        token_list = tokenize(i[0])
        tmp_list = list(nltk.trigrams(token_list))
```

```
        tmp_big_list = list(nltk.bigrams(token_list))
        #print(tmp_big_list)
        for j in tmp_list:
            trigrams_list.append(j)
        for l in tmp_big_list:
            bigrams_list.append(l)
    #print(bigrams_list)

    #Here for job titles we cannot use NLTK's trigram collocation finder as
it will filter out some important details.
    #so finding it manually.
    trigram_freq={}
    bigram_freq={}

    for k in trigrams_list:
        if k in trigram_freq:
            trigram_freq[k]+=1
        else:
            trigram_freq[k]=1

    for k in bigrams_list:
        if k in bigram_freq:
            bigram_freq[k]+=1
        else:
            bigram_freq[k]=1

    # uncomment below print line to see the frequency for each trigram
    # Sorting the dictionary below to easily see and get the top values whe
n needed. For trigrams
    sorted_freq = sorted(trigram_freq.items(), key=operator.itemgetter(1),r
everse=True)
    #print(sorted_freq)
    #print("Sorted by frequency - trigrams from heading\n\nSorted by Freque
ncy Bigrams from heading")
    #for bigrams
    bi_sorted_freq = sorted(bigram_freq.items(),
key=operator.itemgetter(1),reverse=True)
    #print(bi_sorted_freq)

    # now the top most value is what we are looking for. and all our analys
is will be for that position.
    desired_position = sorted_freq[0][0]
    print("\n\nThis analysis is for ",sorted_freq[0][0])
    print("\n##############################################\n")
    #-----------------------------------------
    # now for 5th column i.e. job titles.
    #-----------------------------------------
    exp_trigrams_list =[]
    exp_bigrams_list=[]
    for i in rows:
        #second param is job exp
        exp_list = tokenize(i[1])
        #print(exp_list[0])
        tmp_list = list(nltk.trigrams(exp_list))
        tmp_big_list = list(nltk.bigrams(exp_list))

        for j in tmp_list:
            exp_trigrams_list.append(j)
        for l in tmp_big_list:
            exp_bigrams_list.append(l)
```

```
    exp_trigram_freq={}
    exp_bigram_freq={}

    for k in exp_trigrams_list:
        if k in exp_trigram_freq:
            exp_trigram_freq[k]+=1
        else:
            exp_trigram_freq[k]=1

    for k in exp_bigrams_list:
        if k in exp_bigram_freq:
            exp_bigram_freq[k]+=1
        else:
            exp_bigram_freq[k]=1


    exp_sorted_freq = sorted(exp_trigram_freq.items(),
key=operator.itemgetter(1),reverse=True)
    #print(exp_sorted_freq)
    print("'input resume' can have more chances to become
",desired_position," if it has following job experience terms\n")

    for i in range(0,50):
        print(exp_sorted_freq[i][0], exp_sorted_freq[i][1])
    print("\n\nOR----####----####----####\n\n")
    #for bigrams
    exp_bi_sorted_freq = sorted(exp_bigram_freq.items(),
key=operator.itemgetter(1),reverse=True)
    #print(exp_bi_sorted_freq)
    for i in range(0,50):
        print(exp_bi_sorted_freq[i][0], exp_bi_sorted_freq[i][1])


    print("\n##############################################\n")
    #-----------------------------------------
    # now for 4th column i.e. skills.
    #-----------------------------------------
    skills_list = []
    for i in rows:
        token_list = tokenize(i[2])

        for j in token_list:
            skills_list.append(j)

    #print(len(skills_list))
    skill_dict={}
    for k in skills_list:
        if k in skill_dict:
            skill_dict[k]+=1
        else:
            skill_dict[k]=1


    # uncomment below print line to see the frequency for each trigram
    # Sorting the dictionary below to easily see and get the top values whe
n needed. For trigrams
    sorted_freq = sorted(skill_dict.items(),
key=operator.itemgetter(1),reverse=True)
    print("\n\nThese are preferred skills for the position.\n")
```

```
    custom = ["years","year","na","less","than","and"]
    for i in range(0,60):
        if(sorted_freq[i][0] not in custom):
            print(sorted_freq[i][0], sorted_freq[i][1])



    print("###############  Description top words  ################3")

    skills_list = []
    for i in rows:
        token_list = tokenize(i[5])

        for j in token_list:
            skills_list.append(j)



    #print(len(skills_list))
    skill_dict={}
    for k in skills_list:
        if k in skill_dict:
            skill_dict[k]+=1
        else:
            skill_dict[k]=1


    # uncomment below print line to see the frequency for each trigram
    # Sorting the dictionary below to easily see and get the top values whe
n needed. For trigrams
    sorted_freq = sorted(skill_dict.items(),
key=operator.itemgetter(1),reverse=True)
    print("\n\nThese are preferred description words for the position.\n")
    custom = ["years","year","na","less","than","and","using"]
    return_list = []
    for i in range(0,len(sorted_freq)-5):
        return_list.append(sorted_freq[i])

    for i in range(0,60):
        if(sorted_freq[i][0] not in custom):
            print(sorted_freq[i][0], sorted_freq[i][1])


    print("\n##############################################\n")
    #-----------------------------------------
    # now for 10th column i.e. skills.
    #-----------------------------------------
    skills_list = []
    trigrams_list = []
    for i in rows:
        token_list = tokenize(i[3])
        tmp_list = list(nltk.trigrams(token_list))
        for j in tmp_list:
            trigrams_list.append(j)
        for k in token_list:
            skills_list.append(k)


    skill_dict={}
    trigram_freq = {}
```

```
    for l in trigrams_list:
        if l in trigram_freq:
            trigram_freq[l]+=1
        else:
            trigram_freq[l]=1
    for k in skills_list:
        if k in skill_dict:
            skill_dict[k]+=1
        else:
            skill_dict[k]=1


    # uncomment below print line to see the frequency for each trigram
    # Sorting the dictionary below to easily see and get the top values whe
n needed. For trigrams
    sorted_freq = sorted(skill_dict.items(),
key=operator.itemgetter(1),reverse=True)
    tri_education = sorted(trigram_freq.items(), key=operator.itemgetter(1)
,reverse=True)
    print("\n\nThese are preferred Education details for the position.\n")
    for j in range(0,8):
        print(tri_education[j])

    custom =
["years","year","na","less","than","and","in","of","data","technology"]
    for i in range(0,20):
        if(sorted_freq[i][0] not in custom):
            print(sorted_freq[i])


    print("\n################################################\n")
    print("\nWork authorization requirement for this position in USA\n")
    #-----------------------------------------
    # now for 9th column i.e. Education Details and Universities.
    #-----------------------------------------
    auth_list=[]
    na = ["NA","na"]
    for i in rows:
        auth_list.append(i[4])
    auth_dict = {}
    for i in auth_list:
        if i in auth_dict:
            auth_dict[i]+=1
        else:
            auth_dict[i]=1
    sort_auth =  sorted(auth_dict.items(),
key=operator.itemgetter(1),reverse=True)
    for j in range(0,3):
        if (sort_auth[j][0] not in na):
            print(sort_auth[j][0])

    return return_list
```

7. Unigrams, Bigrams and Trigrams

Below function will find the unigrams, bigrams and trigrams for the different according fields and columns. The last portion of the code describes the main Driver part of the our code.

In [53]:

```
def description_analysis(input_file):
    trigrams_list =[]
    bigrams_list=[]
    # write your code here
    with open(input_file, "r",encoding="ISO-8859-1") as f:
        reader=csv.reader(f, delimiter=',')
        rows=[(row[8]) for row in reader]
    text=". ".join(rows).lower()

    #print(row_len)
    for i in rows:
        token_list = tokenize(i)
        tmp_list = list(nltk.trigrams(token_list))
        tmp_big_list = list(nltk.bigrams(token_list))
        #print(tmp_big_list)
        for j in tmp_list:
            trigrams_list.append(j)
        for l in tmp_big_list:
            bigrams_list.append(l)
    #print(bigrams_list)
    finder = BigramCollocationFinder.from_words(\
        nltk.word_tokenize(text))
    print(token_list[0:10])
    finder.apply_freq_filter(5)
    print(finder.nbest(bigram_measures.pmi, 10))



if __name__ == "__main__":
    #this is for job title analysis
    file1 = "data_science.csv"
    file2 = "project_lead.csv"
    file3 = "vice_president.csv"
```

Calling these all implementation on dataset of Data Scientist.

In [54]:

```
training_data_science = performance_evaluate(file1)
```

```
This analysis is for  ('data', 'scientist', 'intern')

###############################################

'input resume' can have more chances to become   ('data', 'scientist', 'int
ern')  if it has following job experience terms

('data', 'scientist', 'data') 478
('scientist', 'data', 'scientist') 320
('data', 'scientist', 'intern') 140
('scientist', 'data', 'analyst') 88
('data', 'analyst', 'data') 88
```

('analyst', 'data', 'analyst') 60
('senior', 'data', 'scientist') 44
('data', 'analyst', 'intern') 40
('scientist', 'data', 'modeler') 38
('graduate', 'research', 'assistant') 38
('scientist', 'intern', 'data') 37
('machine', 'scientist', 'machine') 36
('data', 'modeler', 'data') 35
('data', 'scientist', 'research') 35
('analyst', 'data', 'scientist') 34
('scientist', 'machine', 'scientist') 33
('engineer', 'data', 'scientist') 32
('data', 'scientist', 'senior') 30
('intern', 'data', 'analyst') 26
('intern', 'data', 'scientist') 25
('assistant', 'data', 'scientist') 23
('graduate', 'teaching', 'assistant') 23
('associate', 'data', 'scientist') 22
('modeler', 'data', 'analyst') 20
('na', 'na', 'na') 20
('scientist', 'machine', 'learning') 19
('data', 'scientist', 'business') 19
('scientist', 'research', 'assistant') 19
('research', 'assistant', 'data') 18
('lead', 'data', 'scientist') 17
('data', 'scientist', 'graduate') 17
('machine', 'learning', 'data') 16
('analyst', 'data', 'modeler') 16
('data', 'scientist', 'assistant') 16
('intern', 'research', 'assistant') 16
('data', 'engineer', 'data') 16
('data', 'scientist', 'machine') 15
('learning', 'data', 'scientist') 15
('modeler', 'data', 'modeler') 15
('data', 'scientist', 'na') 15
('data', 'scientist', 'co-op') 15
('scientist', 'data', 'engineer') 14
('jr', 'data', 'scientist') 13
('sr', 'data', 'scientist') 13
('scientist', 'intern', 'graduate') 13
('data', 'scientist', 'project') 13
('developer', 'data', 'scientist') 12
('data', 'scientist', 'statistician') 12
('software', 'engineer', 'intern') 12
('assistant', 'data', 'analyst') 11

OR----####----####----####

('data', 'scientist') 1276
('scientist', 'data') 500
('data', 'analyst') 335
('scientist', 'intern') 148
('research', 'assistant') 133
('analyst', 'data') 124
('data', 'modeler') 71
('software', 'engineer') 69
('intern', 'data') 65
('engineer', 'data') 60

('analyst', 'intern') 58
('senior', 'data') 56
('scientist', 'machine') 52
('data', 'science') 50
('graduate', 'research') 49
('scientist', 'research') 49
('machine', 'scientist') 47
('teaching', 'assistant') 43
('research', 'scientist') 42
('machine', 'learning') 41
('assistant', 'data') 39
('scientist', 'senior') 38
('data', 'engineer') 37
('modeler', 'data') 36
('business', 'analyst') 36
('na', 'na') 34
('software', 'developer') 29
('associate', 'data') 27
('decision', 'scientist') 27
('graduate', 'teaching') 27
('research', 'associate') 26
('consultant', 'data') 26
('systems', 'engineer') 25
('scientist', 'graduate') 24
('big', 'data') 24
('project', 'manager') 24
('python', 'developer') 23
('lead', 'data') 22
('data', 'analytics') 22
('sr', 'data') 21
('intern', 'research') 21
('engineer', 'intern') 21
('learning', 'data') 20
('developer', 'data') 20
('intern', 'graduate') 20
('programmer', 'analyst') 19
('scientist', 'business') 19
('clinical', 'data') 19
('business', 'intelligence') 18
('scientist', 'assistant') 17

##

These are preferred skills for the position.

python 398
sql 383
data 365
learning 245
analysis 201
machine 193
hadoop 178
excel 125
apache 117
java 116
sas 101
matlab 83
database 80
tableau 76

tableau 76
aws 71
mining 70
business 69
visualization 68
linux 63
microsoft 61
ms 60
office 57
analytics 55
intelligence 55
spark 52
statistics 52
statistical 51
css 49
git 48
management 48
html 46
language 44
science 44
server 43
mysql 42
deep 42
bi 40
hive 39
javascript 39
modeling 35
spss 35
algorithms 35
databases 33
access 31
unix 29
processing 29
programming 29
clustering 29
natural 27
marketing 27
development 26
mapreduce 24
scala 23
serial 23
attached 23
scsi 23
############### Description top words ################3

These are preferred description words for the position.

data 15654
analysis 3514
business 2842
sql 2418
developed 2277
python 2271
models 2217
used 2155
learning 2116
model 1987
machine 1809
worked 1578
various 1525

various 1525
project 1522
team 1464
performed 1441
design 1381
development 1337
reports 1246
database 1245
requirements 1228
regression 1222
created 1200
system 1185
based 1178
statistical 1146
implemented 1146
environment 1124
management 1105
tableau 1078
analytics 1050
algorithms 1043
hadoop 999
server 984
hive 981
spark 954
designed 944
responsibilities 940
customer 939
new 937
application 929
modeling 926
services 923
different 911
systems 908
process 863
involved 837
quality 807
web 804
performance 803
source 801
testing 791
test 781
like 778
ms 760
support 751
tools 746
time 734
hdfs 729

##

These are preferred Education details for the position.

(('science', 'computer', 'science'), 73)
(('computer', 'science', 'engineering'), 52)
(('science', 'data', 'science'), 47)
(('master', 'science', 'computer'), 44)
(('engineering', 'computer', 'science'), 33)
(('computer', 'science', 'bachelor'), 31)
(('data', 'science', 'bachelor'), 31)

(('data', 'science', 'bachelor'), 31)
(('computer', 'science', 'computer'), 31)
('science', 807)
('engineering', 510)
('bachelor', 464)
('master', 391)
('computer', 328)
('statistics', 171)
('mathematics', 139)
('business', 118)
('information', 105)
('analytics', 92)
('applied', 89)
('management', 88)
('electronics', 81)
('electrical', 73)
('masters', 69)
('ms', 64)
('systems', 62)
('physics', 62)

##

Work authorization requirement for this position in USA

Authorized to work in the US for any employer
Sponsorship required to work in the US

Running the same on file 2 i.e. Senior Software Engineer.

In [55]:

```
    training_software_engineer = performance_evaluate(file2)
```

This analysis is for ('senior', 'software', 'engineer')

##

'input resume' can have more chances to become   ('senior', 'software', 'engineer') if it has following job experience terms

('senior', 'software', 'engineer') 772
('software', 'engineer', 'senior') 321
('engineer', 'senior', 'software') 256
('sr', 'software', 'engineer') 227
('software', 'engineer', 'software') 173
('engineer', 'software', 'engineer') 127
('software', 'engineer', 'project') 107
('software', 'engineer', 'sr') 91
('engineer', 'project', 'lead') 80
('engineer', 'sr', 'software') 65
('lead', 'software', 'engineer') 58
('software', 'engineer', 'lead') 55
('software', 'engineer', 'technical') 45
('consultant', 'senior', 'software') 40
('lead', 'senior', 'software') 39

('lead', 'technical', 'lead') 34
('software', 'engineer', 'team') 30
('engineer', 'team', 'lead') 28
('project', 'lead', 'software') 28
('software', 'engineer', 'consultant') 28
('technical', 'lead', 'technical') 27
('member', 'technical', 'staff') 27
('developer', 'software', 'developer') 26
('project', 'lead', 'sr') 26
('software', 'engineer', 'na') 26
('engineer', 'project', 'manager') 25
('software', 'engineer', 'independent') 25
('engineer', 'independent', 'consultant') 25
('sr', 'programmer', 'analyst') 24
('lead', 'team', 'lead') 23
('senior', 'software', 'developer') 23
('lead', 'project', 'lead') 22
('lead', 'sr', 'software') 22
('software', 'developer', 'software') 22
('engineer', 'technical', 'lead') 20
('project', 'lead', 'senior') 20
('project', 'manager', 'software') 19
('developer', 'senior', 'software') 19
('project', 'manager', 'senior') 19
('independent', 'consultant', 'senior') 19
('project', 'lead', 'project') 18
('software', 'engineer', 'contractor') 18
('senior', 'developer', 'consultant') 17
('software', 'engineer', 'principal') 17
('developer', 'net', 'developer') 17
('engineer', 'lead', 'software') 16
('engineer', 'software', 'developer') 16
('technical', 'project', 'manager') 16
('net', 'developer', 'net') 16
('consultant', 'software', 'engineer') 16

OR----####----####----####

('software', 'engineer') 1368
('senior', 'software') 838
('engineer', 'senior') 352
('sr', 'software') 256
('project', 'lead') 226
('engineer', 'software') 211
('engineer', 'project') 135
('project', 'manager') 134
('software', 'developer') 106
('engineer', 'sr') 100
('team', 'lead') 99
('technical', 'lead') 98
('lead', 'software') 96
('programmer', 'analyst') 81
('engineer', 'lead') 72
('lead', 'senior') 71
('consultant', 'senior') 69
('web', 'developer') 56
('engineer', 'technical') 55
('senior', 'developer') 53
('net', 'developer') 53

('net', 'developer') 53
('lead', 'project') 51
('developer', 'senior') 51
('developer', 'software') 48
('lead', 'developer') 48
('lead', 'sr') 48
('lead', 'technical') 44
('project', 'leader') 43
('tech', 'lead') 40
('software', 'development') 40
('developer', 'consultant') 37
('manager', 'senior') 37
('developer', 'project') 36
('engineer', 'team') 35
('independent', 'consultant') 35
('software', 'consultant') 34
('java', 'developer') 33
('project', 'engineer') 33
('lead', 'engineer') 32
('systems', 'engineer') 32
('lead', 'team') 31
('engineer', 'consultant') 31
('member', 'technical') 30
('systems', 'analyst') 29
('software', 'architect') 29
('engineer', 'na') 29
('sr', 'programmer') 29
('technical', 'staff') 28
('architect', 'senior') 28
('system', 'analyst') 28

##

These are preferred skills for the position.

sql 138
java 78
server 65
oracle 61
net 53
web 51
javascript 42
ms 39
asp 39
linux 34
visual 33
development 33
html 32
services 31
jquery 27
windows 26
xml 24
apache 24
software 24
design 24
unix 22
database 22
perl 21
python 20

python 20
android 19
studio 18
css 18
management 16
spring 14
agile 14
mvc 14
framework 14
systems 14
api 13
git 13
microsoft 13
mysql 12
json 12
eclipse 11
angularjs 11
engineer 11
embedded 11
ajax 10
project 10
js 10
svn 10
rest 9
php 9
data 9
hadoop 9
mapreduce 9
typescript 9
scripting 8
architecture 8
shell 8
azure 8
############### Description top words ################3

These are preferred description words for the position.

system 3100
application 2792
development 2782
data 2682
software 2521
developed 2499
project 2423
design 2298
web 2093
server 1999
team 1829
management 1712
sql 1670
used 1617
business 1435
services 1376
testing 1369
test 1327
applications 1287
support 1267
designed 1259
database 1230
new 1205

new 1205
based 1171
java 1113
code 1107
requirements 1048
windows 1041
implemented 1034
oracle 1014
user 1005
involved 1005
product 966
visual 966
systems 964
created 928
responsible 921
net 914
analysis 899
various 885
process 864
technical 860
client 807
worked 779
reports 774
service 773
implementation 773
interface 766
framework 758
xml 739
lead 724
customer 721
integration 679
information 623
develop 613
access 605
architecture 591
performance 582
technologies 581

##

These are preferred Education details for the position.

(('science', 'computer', 'science'), 53)
(('bachelor', 'science', 'computer'), 32)
(('computer', 'science', 'engineering'), 19)
(('engineering', 'computer', 'science'), 18)
(('computer', 'science', 'computer'), 18)
(('computer', 'science', 'bachelor'), 17)
(('master', 'science', 'computer'), 17)
(('computer', 'information', 'systems'), 16)
('science', 343)
('computer', 318)
('bachelor', 174)
('engineering', 162)
('master', 88)
('information', 58)
('systems', 44)
('business', 41)
('management', 37)

```
('mathematics', 36)
('electronics', 34)
('bs', 34)
('ms', 31)
('administration', 28)
('applications', 24)
('electrical', 21)
('diploma', 21)
('masters', 21)

##############################################


Work authorization requirement for this position in USA

Authorized to work in the US for any employer
Sponsorship required to work in the US
```

Running the sam on 3rd Dataset i.e. Vice President

In [56]:

```
    training_vice_president = performance_evaluate(file3)
```

```
This analysis is for  ('vice', 'president', 'valley')

##############################################

'input resume' can have more chances to become   ('vice', 'president', 'val
ley')  if it has following job experience terms

('vice', 'president', 'vice') 130
('president', 'vice', 'president') 130
('assistant', 'vice', 'president') 31
('vice', 'president', 'senior') 26
('vice', 'president', 'director') 23
('vice', 'president', 'sales') 23
('vice', 'president', 'assistant') 22
('vice', 'president', 'na') 19
('president', 'assistant', 'vice') 18
('vice', 'president', 'associate') 16
('vice', 'president', 'operations') 12
('director', 'vice', 'president') 11
('senior', 'vice', 'president') 9
('na', 'vice', 'president') 8
('vice', 'president', 'branch') 7
('vice', 'president', 'regional') 7
('vice', 'president', 'intern') 7
('president', 'na', 'vice') 7
('president', 'branch', 'manager') 6
('manager', 'vice', 'president') 6
('president', 'senior', 'vice') 6
('vice', 'president', 'president') 6
('vice', 'president', 'marketing') 6
('president', 'director', 'vice') 5
('vice', 'president', 'commercial') 5
('vice', 'president', 'manager') 5
('vice', 'president', 'executive') 5
```

('vice', 'president', 'executive') 5
('associate', 'vice', 'president') 5
('vice', 'president', 'consultant') 5
('president', 'sales', 'marketing') 5
('consultant', 'vice', 'president') 4
('vice', 'president', 'information') 4
('president', 'regional', 'vice') 4
('regional', 'vice', 'president') 4
('president', 'sales', 'associate') 4
('president', 'senior', 'director') 4
('vice', 'president', 'general') 4
('vice', 'president', 'analyst') 4
('vice', 'president', 'asst') 4
('president', 'asst', 'vice') 4
('asst', 'vice', 'president') 4
('president', 'associate', 'associate') 4
('president', 'director', 'operations') 4
('president', 'president', 'president') 4
('personal', 'trainer', 'personal') 4
('trainer', 'personal', 'trainer') 4
('president', 'senior', 'project') 3
('senior', 'project', 'manager') 3
('president', 'information', 'systems') 3
('vice', 'president', 'asset') 3

OR----####----####----####

('vice', 'president') 665
('president', 'vice') 132
('assistant', 'vice') 31
('president', 'senior') 26
('president', 'director') 25
('president', 'sales') 25
('president', 'assistant') 23
('president', 'na') 19
('president', 'associate') 16
('project', 'manager') 12
('president', 'operations') 12
('president', 'president') 12
('director', 'vice') 11
('account', 'executive') 10
('branch', 'manager') 9
('senior', 'vice') 9
('sales', 'manager') 9
('account', 'manager') 9
('sales', 'associate') 8
('na', 'vice') 8
('president', 'branch') 7
('president', 'regional') 7
('president', 'intern') 7
('associate', 'associate') 7
('customer', 'service') 7
('manager', 'vice') 6
('president', 'marketing') 6
('sales', 'marketing') 6
('sales', 'representative') 6
('financial', 'analyst') 6
('personal', 'trainer') 6
('manager', 'assistant') 5

('president', 'commercial') 5
('senior', 'director') 5
('na', 'na') 5
('president', 'manager') 5
('president', 'executive') 5
('associate', 'vice') 5
('asst', 'vice') 5
('intern', 'intern') 5
('store', 'manager') 5
('president', 'consultant') 5
('business', 'development') 5
('sales', 'director') 5
('director', 'director') 5
('associate', 'analyst') 5
('manager', 'account') 5
('national', 'sales') 5
('consultant', 'vice') 4
('president', 'information') 4

##

These are preferred skills for the position.

management 90
microsoft 68
office 62
excel 54
development 34
sales 33
word 29
powerpoint 25
customer 24
project 23
marketing 23
analysis 22
business 22
operations 22
planning 21
financial 21
ms 19
service 19
leadership 17
training 15
strategic 14
skills 14
budget 13
data 12
problem 11
relationship 11
sql 10
accounting 10
excellent 10
quickbooks 9
solving 9
communication 9
team 8
organizational 8
access 7

human 7
resources 7
autocad 7
contract 7
finance 7
written 7
construction 6
change 6
building 6
oriented 6
lending 6
credit 6
event 6
cpr 6
relations 6
account 5
writing 5
asp 5
python 5
retail 5
series 5
############### Description top words ################3

These are preferred description words for the position.

business 585
management 566
sales 528
new 512
team 413
including 397
development 357
responsible 333
managed 318
clients 314
operations 289
financial 282
company 281
services 273
customer 272
developed 266
client 260
service 245
data 234
process 217
marketing 203
project 202
support 195
analysis 193
market 184
training 177
design 168
million 166
program 166
implemented 164
projects 160
system 159
reporting 157
created 157

```
credit 155
planning 154
product 149
led 146
portfolio 145
revenue 142
within 140
relationships 139
compliance 139
accounts 138
staff 138
manage 131
worked 128
manager 128
various 127
risk 126
lead 126
daily 126
products 126
based 126
strategic 125
president 124
systems 123
across 123

##############################################



These are preferred Education details for the position.

(('bachelor', 'science', 'business'), 18)
(('high', 'school', 'diploma'), 17)
(('bachelor', 'business', 'administration'), 16)
(('business', 'administration', 'finance'), 13)
(('master', 'business', 'administration'), 12)
(('science', 'business', 'administration'), 10)
(('science', 'mechanical', 'engineering'), 9)
(('high', 'school', 'equivalent'), 9)
('bachelor', 177)
('business', 127)
('science', 120)
('administration', 80)
('arts', 57)
('master', 51)
('finance', 49)
('engineering', 48)
('management', 38)
('diploma', 30)
('high', 29)
('school', 29)
('accounting', 27)
('associate', 26)
('bs', 24)
('economics', 23)
('mba', 21)
('marketing', 21)

##############################################
```

```
Work authorization requirement for this position in USA

Authorized to work in the US for any employer
```

Output/ Result of the analysis

In [50]:

```
    #this is for calculating avg job experience for data science
    print("\n##################################################\n")
    #dateToSum("indeed_scraped_data_science.csv")

    tot = len(training_data_science)
    #print(input_tokens)
    #print(len(training_data_science))
    print("\n##################################################\n")

    #these functions will analyze the job description of all positions from
training dataset.

    print("\nBelow are the preferred job experience terms for the data scie
nce position.\n")
    description_analysis(file1)
    print("\n####################")

    print("\nBelow are the preferred job experience terms for the project l
ead position.\n")
    description_analysis(file2)
    print("####################")

    print("\nBelow are the preferred job experience terms for the vice pres
ident position.\n")
    description_analysis(file3)
```

```
##############################################

14021
Probability of the input resume of being data science is:

##############################################



Below are the preferred job experience terms for the given position.

['utilize', 'lcms', 'analyze', 'target', 'analytes', 'assist', 'diagnosis',
'biochemical', 'disorders', 'utilize']
[('diabetic', 'retinopathy'), ('dow', 'jones'), ('higgs', 'boson'), ('abn',
'amro'), ('learn/', 'scipy/'), ('rac/10g', 'rac/9i'), ('scipy/', 'numpy/'),
('seed/production', 'fermenters'), ('tremendous', 'pride'), ('lin', 'z.')]
##################3
['lmm', 'legacy', 'managed', 'market', 'cm', 'customer', 'master', 'remedia
tion', 'project', 'multiple']
[('ezx', 'xwindows'), ('jd', 'edwards'), ('proving', 'grounds'), ('rhode',
'island'), ('basking', 'ridge'), ('condor', 'arinc-429'), ('td', 'ameritrad
e'), ('daimler', 'chrysler'), ('loosely', 'coupled'), ('checkfree', 'apl')]
```

```
####################
['cmu', 'engineering', 'club', 'formula', 'sae', 'team', 'coordinate', 'col
orado', 'mesa', 'university']
[('hong', 'kong'), ('los', 'angeles'), ('merrill', 'lynch'), ('richard', 'e
llis'), ('subject', 'matter'), ('st.', 'louis'), ('fed', 'ex'), ('cold', 'c
alling'), ('square', 'feet'), ('@', 'gmail.com')]
```

Finding probability of becoming Data scientist, VP and Software Engineer from input resume

In [51]:

```
def compare_prob(input_tokens,training_data_science):
    count = 0
    for itk in training_data_science:
        j = 0
        itk = itk[0].lower()
        #print(itk)
        for tdk in input_tokens:
            j = j + 1
            tdk = tdk.lower()
            #print(tdk)
            if(itk == tdk):
                count = count + (tot-j)

    return count

tmp1 = compare_prob(input_tokens,training_data_science)
tmp2 = compare_prob(input_tokens,training_software_engineer)
tmp3 = compare_prob(input_tokens,training_vice_president)

tot = tmp1+tmp2+tmp3

#input sample is taken from with the skills of Project Lead.

print("Chances of input resume to be a Data Scientist (CNN on job descripti
on) are: ")
print(tmp1/tot * 100)

print("\n\nChances of input resume to be a Project Lead (CNN on job descrip
tion) are: ")
print(tmp2/tot * 100)

print("\n\nChances of input resume to be a Vice President (CNN on job descr
iption) are: ")
print(tmp3/tot * 100)
```

```
Chances of input resume to be a Data Scientist (CNN on job description) are
:
84.10631807763958
Chances of input resume to be a Project Lead (CNN on job description) are:
52.37256176994119
Chances of input resume to be a Vice President (CNN on job description) are
:
23.521120152419222
```