205229118

Mahalakshmi S

Lab 10. Clustering the job titles of LinkedIn Connections using Greedy Heuristic Algorithm

In [2]:

```
import requests
import string
import random
CLIENT_ID = "78icgiayp0mu5z"
CLIENT_SECRET = "Pa8inm0abOmALXwb"
REDIRECT_URI = "http://localhost"

# Generate a random string to protect against cross-site request forgery
letters = string.ascii_lowercase
CSRF_TOKEN = ''.join(random.choice(letters) for i in range(24))

auth_params = {'response_type': 'code',
               'client_id': CLIENT_ID,
               'redirect_uri': REDIRECT_URI,
               'state': CSRF_TOKEN,
               'scope': 'r_liteprofile,r_emailaddress,w_member_social'}

html = requests.get("https://www.linkedin.com/oauth/v2/authorization",
                    params = auth_params)

# Print the link to the approval page
print(html.url)
```

```
https://www.linkedin.com/uas/login?session_redirect=%2Foauth%2Fv2%2Flogin-su
ccess%3Fapp_id%3D113884024%26auth_type%3DAC%26flow%3D%257B%2522authorization
Type%2522%253A%2522OAUTH2_AUTHORIZATION_CODE%2522%252C%2522redirectUri%2522%
253A%2522http%253A%252F%252Flocalhost%2522%252C%2522currentStage%2522%253A%2
522LOGIN_SUCCESS%2522%252C%2522currentSubStage%2522%253A0%252C%2522authFlowN
ame%2522%253A%2522generic-permission-list%2522%252C%2522appId%2522%253A11388
4024%252C%2522creationTime%2522%253A1632502847090%252C%2522state%2522%253A%2
522wzlikfryjqbdcvbpouslkxkz%2522%252C%2522scope%2522%253A%2522r_liteprofile%
252Cr_emailaddress%252Cw_member_social%2522%257D&fromSignIn=1&trk=oauth&canc
el_redirect=%2Foauth%2Fv2%2Flogin-cancel%3Fapp_id%3D113884024%26auth_type%3D
AC%26flow%3D%257B%2522authorizationType%2522%253A%2522OAUTH2_AUTHORIZATION_C
ODE%2522%252C%2522redirectUri%2522%253A%2522http%253A%252F%252Flocalhost%252
2%252C%2522currentStage%2522%253A%2522LOGIN_SUCCESS%2522%252C%2522currentSub
Stage%2522%253A0%252C%2522authFlowName%2522%253A%2522generic-permission-lis
t%2522%252C%2522appId%2522%253A113884024%252C%2522creationTime%2522%253A1632
502847090%252C%2522state%2522%253A%2522wzlikfryjqbdcvbpouslkxkz%2522%252C%25
22scope%2522%253A%2522r_liteprofile%252Cr_emailaddress%252Cw_member_social%2
522%257D (https://www.linkedin.com/uas/login?session_redirect=%2Foauth%2Fv2%
2Flogin-success%3Fapp_id%3D113884024%26auth_type%3DAC%26flow%3D%257B%2522aut
horizationType%2522%253A%2522OAUTH2_AUTHORIZATION_CODE%2522%252C%2522redirec
tUri%2522%253A%2522http%253A%252F%252Flocalhost%2522%252C%2522currentStage%2
522%253A%2522LOGIN_SUCCESS%2522%252C%2522currentSubStage%2522%253A0%252C%252
2authFlowName%2522%253A%2522generic-permission-list%2522%252C%2522appId%252
2%253A113884024%252C%2522creationTime%2522%253A1632502847090%252C%2522state%
2522%253A%2522wzlikfryjqbdcvbpouslkxkz%2522%252C%2522scope%2522%253A%2522r_l
iteprofile%252Cr_emailaddress%252Cw_member_social%2522%257D&fromSignIn=1&trk
=oauth&cancel_redirect=%2Foauth%2Fv2%2Flogin-cancel%3Fapp_id%3D113884024%26a
uth_type%3DAC%26flow%3D%257B%2522authorizationType%2522%253A%2522OAUTH2_AUTH
ORIZATION_CODE%2522%252C%2522redirectUri%2522%253A%2522http%253A%252F%252Flo
calhost%2522%252C%2522currentStage%2522%253A%2522LOGIN_SUCCESS%2522%252C%252
2currentSubStage%2522%253A0%252C%2522authFlowName%2522%253A%2522generic-perm
ission-list%2522%252C%2522appId%2522%253A113884024%252C%2522creationTime%252
2%253A1632502847090%252C%2522state%2522%253A%2522wzlikfryjqbdcvbpouslkxkz%25
22%252C%2522scope%2522%253A%2522r_liteprofile%252Cr_emailaddress%252Cw_membe
r_social%2522%257D)
```

In [6]:

```
AUTH_CODE ='AQSiVHBNtCf4dbOhaMVLB8d3YzK9h49mL2-dsIzAapQDPtLYMDu2e5DUGlDSDQz0BaAgsYwZqFoecbE
ACCESS_TOKEN_URL = 'https://www.linkedin.com/oauth/v2/accessToken'

qd = {'grant_type': 'authorization_code',
      'code': AUTH_CODE,
      'redirect_uri': REDIRECT_URI,
      'client_id': CLIENT_ID,
      'client_secret': CLIENT_SECRET}

response = requests.post(ACCESS_TOKEN_URL, data=qd, timeout=60)

response = response.json()

access_token = response['access_token']

print ("Access Token:", access_token)
print ("Expires in (seconds):", response['expires_in'])
```

```
Access Token: AQVr1RhTe-by9LWejc0Ec5r8ohnuDKCGlEXsHNgAOXr8kbcgMAAG_OBCkun1qw
MXWg9XU71HATH693VgqfbT8xRFZg0A9uRK1ZsXsj4csDlYvkl-jm6_cNJDpfuy2dZUNqvWJLPCYD
NoZTnWEakNlR9_n-z1BoAkMgKfaHr6d4vfIj4_bwIGUWty8SvNa2uXuAxgtx-TCzdKB-pTSqHLV_
8df4tUyXRIXPtFQsGdwCarQGd6LbbejxFJIeI8Ye6I275JewSNQF5tkCZoRbjXG9JuTMF5ZEB7V9
t8rhe2poVOaOWW_p6nF7X200Uc7WMGioym1AJDt4KpMef99z9RURLfssNAbQ
Expires in (seconds): 5183999
```

In [7]:

```
import json
params = {'oauth2_access_token': access_token,
          'fields': ["localizedFirstName,localizedLastName,id"]}
response = requests.get('https://api.linkedin.com/v2/me', params = params)
print(json.dumps(response.json(), indent=1))
```

```
{
 "localizedLastName": "S",
 "id": "VEYlKD4E-7",
 "localizedFirstName": "Mahalakshmi"
}
```

In [8]:

```
import json

params = {'oauth2_access_token': access_token,
          'fields': ['lastName:(preferredLocale:(country,language))']}
response = requests.get('https://api.linkedin.com/v2/me', params = params)

print(json.dumps(response.json(), indent=1))
```

```
{
 "lastName": {
  "preferredLocale": {
   "country": "US",
   "language": "en"
  }
 }
}
```

In [9]:

```
import os
import csv

# Point this to your 'Connections.csv' file.
CSV_FILE = ('Connections.csv')

csvReader = csv.DictReader(open(CSV_FILE), delimiter=',', quotechar='"')
contacts = [row for row in csvReader]
```

In [10]:

```
contacts
```

Out[10]:

```
[{'First Name': 'Sethuraman',
  'Last Name': 'Periannan',
  'Email Address': '',
  'Company': '',
  'Position': '',
  'Connected On': '16-Sep-21'},
 {'First Name': 'SIVARAJ',
  'Last Name': 'D',
  'Email Address': '',
  'Company': '',
  'Position': '',
  'Connected On': '16-Sep-21'},
 {'First Name': 'rene',
  'Last Name': 'irakoze ',
  'Email Address': '',
  'Company': 'The Sparks Foundation',
  'Position': 'Data Science and business analytics intern',
  'Connected On': '16-Sep-21'},
```

In [11]:

```
import pandas as pd
```

In [12]:

```
con = pd.read_csv("Connections.csv")
```

In [13]:

```
con.head(10)
```

Out[13]:

	First Name	Last Name	Email Address	Company	Position	Connected On
0	Sethuraman	Periannan	NaN	NaN	NaN	16-Sep-21
1	SIVARAJ	D	NaN	NaN	NaN	16-Sep-21
2	rene	irakoze	NaN	The Sparks Foundation	Data Science and business analytics intern	16-Sep-21
3	KARAN	R	NaN	NaN	NaN	16-Sep-21
4	Vimal Doss	R	NaN	NaN	NaN	16-Sep-21
5	Eugene	Kingsley	NaN	Kissflow	Technical Solution Specialist	15-Sep-21
6	Arzoo	Sah	NaN	NaN	NaN	11-Sep-21
7	Janani	Selvaraj PhD	NaN	Bishop Heber College, Tiruchirappalli - 620 017.	Guest Lecturer	08-Sep-21
8	Rajkumar	Kannan	NaN	Bishop Heber College, Tiruchirappalli - 620 017.	Dean - Deanery International Relations	08-Sep-21
9	Jayasurya	V	NaN	Actify Data Labs	Data Analyst	08-Sep-21

In [14]:

```
from prettytable import PrettyTable # pip install prettytable
from collections import Counter
from operator import itemgetter

# Define a set of transforms that converts the first item
# to the second item. Here, we're simply handling some
# commonly known abbreviations, stripping off common suffixes,
# etc.

transforms = [(', Inc.', ''), (', Inc', ''), (', LLC', ''), (', LLP', ''),
              (' LLC', ''), (' Inc.', ''), (' Inc', '')]

companies = [c['Company'].strip() for c in contacts if c['Company'].strip() != '']

for i, _ in enumerate(companies):
    for transform in transforms:
        companies[i] = companies[i].replace(*transform)

pt = PrettyTable(field_names=['Company', 'Freq'])
pt.align = 'l'
c = Counter(companies)

[pt.add_row([company, freq]) for (company, freq) in sorted(c.items(), key=itemgetter(1), re

print(pt)
```

```
+-------------------------------------------------+------+
| Company                                         | Freq |
+-------------------------------------------------+------+
| The Sparks Foundation                           | 12   |
| Bishop Heber College, Tiruchirappalli - 620 017. | 3    |
+-------------------------------------------------+------+
```

In [15]:

```
transforms = [
('Sr.', 'Senior'),
('Sr', 'Senior'),
('Jr.', 'Junior'),
('Jr', 'Junior'),
('CEO', 'Chief Executive Officer'),
('COO', 'Chief Operating Officer'),
('CTO', 'Chief Technology Officer'),
('CFO', 'Chief Finance Officer'),
('VP', 'Vice President'),
]
# Read in a list of titles and split apart
# any combined titles like "President/CEO."
# Other variations could be handled as well, such
# as "President & CEO", "President and CEO", etc.
titles = []
for contact in contacts:
 titles.extend([t.strip() for t in contact['Position'].split('/')
if contact['Position'].strip() != ''])
# Replace common/known abbreviations
for i, _ in enumerate(titles):
  for transform in transforms:
    titles[i] = titles[i].replace(*transform)
# Print out a table of titles sorted by frequency
pt = PrettyTable(field_names=['Job Title', 'Freq'])
pt.align = 'l'
c = Counter(titles)
[pt.add_row([title, freq])
for (title, freq) in sorted(c.items(), key=itemgetter(1), reverse=True)
if freq > 1]
print(pt)
# Print out a table of tokens sorted by frequency
tokens = []
for title in titles:
  tokens.extend([t.strip(',') for t in title.split()])
pt = PrettyTable(field_names=['Token', 'Freq'])
pt.align = 'l'
c = Counter(tokens)
[pt.add_row([token, freq])
for (token, freq) in sorted(c.items(), key=itemgetter(1), reverse=True)
  if freq > 1 and len(token) > 2]
print(pt)
```

```
+-----------------------------------------+------+
| Job Title                               | Freq |
+-----------------------------------------+------+
| Data Scientist                          | 2    |
| Data Science and Business Analystics Intern | 2    |
+-----------------------------------------+------+
+------------+------+
| Token      | Freq |
+------------+------+
| Data       | 16   |
| Science    | 11   |
| and        | 10   |
| Business   | 10   |
| Intern     | 6    |
| Analytics  | 6    |
```

```
| intern     | 4    |
| analytics  | 3    |
| business   | 2    |
| Analyst    | 2    |
| Student    | 2    |
| Scientist  | 2    |
| Analystics | 2    |
+------------+------+
```

In [16]:

```
import os
import csv
from collections import Counter
from operator import itemgetter
from prettytable import PrettyTable

# XXX: Place your "Outlook CSV" formatted file of connections from
# http://www.linkedin.com/people/export-settings at the following
# location: resources/ch03-linkedin/my_connections.csv

CSV_FILE = ('Connections.csv')

# Define a set of transforms that converts the first item
# to the second item. Here, we're simply handling some
# commonly known abbreviations, stripping off common suffixes,
# etc.

transforms = [(', Inc.', ''), (', Inc', ''), (', LLC', ''), (', LLP', ''),
              (' LLC', ''), (' Inc.', ''), (' Inc', '')]

csvReader = csv.DictReader(open(CSV_FILE), delimiter=',', quotechar='"')
contacts = [row for row in csvReader]
companies = [c['Company'].strip() for c in contacts if c['Company'].strip() != '']

for i, _ in enumerate(companies):
    for transform in transforms:
        companies[i] = companies[i].replace(*transform)

pt = PrettyTable(field_names=['Company', 'Freq'])
pt.align = 'l'
c = Counter(companies)
[pt.add_row([company, freq])
 for (company, freq) in sorted(c.items(), key=itemgetter(1), reverse=True)
     if freq > 1]
print(pt)
```

```
+-------------------------------------------------+------+
| Company                                         | Freq |
+-------------------------------------------------+------+
| The Sparks Foundation                           | 12   |
| Bishop Heber College, Tiruchirappalli - 620 017. | 3    |
+-------------------------------------------------+------+
```

In [17]:

```
A = "CEO is a expansion of Chief Executive Officer"
B = "CTO is a expansion of Chief Technology Officer"
```

In [18]:

```
words_doc1 = {'CEO', 'is', 'a', 'expansion', 'of', 'Chief', 'Executive', 'Officer'}
words_doc2 = {'CTO', 'is', 'a', 'expansion', 'of', 'Chief', 'Technology', 'Officer'}
```

In [19]:

```
def Jaccard_Similarity(doc1, doc2):

    # List the unique words in a document
    words_doc1 = set(doc1.lower().split())
    words_doc2 = set(doc2.lower().split())

    # Find the intersection of words list of doc1 & doc2
    intersection = words_doc1.intersection(A)

    # Find the union of words list of doc1 & doc2
    union = words_doc1.union(B)

    # Calculate Jaccard similarity score
    # using length of intersection set divided by length of union set
    return float(len(intersection)) / len(union)
```

In [20]:

```
A = "CEO is a expansion of Chief Executive Officer"
B = "CTO is a expansion of Chief Technology Officer"
Jaccard_Similarity(A,B)
```

Out[20]:

```
0.038461538461538464
```

In [21]:

```
from nltk.util import bigrams
ceo_bigrams = list(bigrams("Chief Executive Officer".split(), pad_left=True, pad_right=True
cto_bigrams = list(bigrams("Chief Technology Officer".split(), pad_left=True, pad_right=Tru
print(ceo_bigrams)
print(cto_bigrams)
print(len(set(ceo_bigrams).intersection(set(cto_bigrams))))
```

```
[(None, 'Chief'), ('Chief', 'Executive'), ('Executive', 'Officer'), ('Office
r', None)]
[(None, 'Chief'), ('Chief', 'Technology'), ('Technology', 'Officer'), ('Offi
cer', None)]
2
```

In [22]:

```
from nltk.metrics.distance import jaccard_distance # pip install nltk

job_title_1 = 'Chief Executive Officer'.split()
job_title_2 = 'Chief Technology Officer'.split()
print(job_title_1)
print(job_title_2)

print()
print('Intersection:')
intersection = set(job_title_1).intersection(set(job_title_2))
print(intersection)

print()
print('Union:')
union = set(job_title_1).union(set(job_title_2))
print(union)
print()
print('Similarity:', len(intersection) / len(union))
print('Distance:', jaccard_distance(set(job_title_1), set(job_title_2)))
```

```
['Chief', 'Executive', 'Officer']
['Chief', 'Technology', 'Officer']

Intersection:
{'Officer', 'Chief'}

Union:
{'Executive', 'Technology', 'Officer', 'Chief'}

Similarity: 0.5
Distance: 0.5
```

In [23]:

```
!pip install cluster
```

```
Collecting cluster
  Downloading cluster-1.4.1.post3-py2.py3-none-any.whl (28 kB)
Installing collected packages: cluster
Successfully installed cluster-1.4.1.post3
```

In [ ]:

```
import os
import csv
from nltk.metrics.distance import jaccard_distance

# Point this to your 'Connections.csv' file
CSV_FILE = os.path.join('Connections.csv')

# Tweak this distance threshold and try different distance calculations
# during experimentation
DISTANCE_THRESHOLD = 0.6
DISTANCE = jaccard_distance

def cluster_contacts_by_title():

    transforms = [
        ('Sr.', 'Senior'),
        ('Sr', 'Senior'),
        ('Jr.', 'Junior'),
        ('Jr', 'Junior'),
        ('CEO', 'Chief Executive Officer'),
        ('COO', 'Chief Operating Officer'),
        ('CTO', 'Chief Technology Officer'),
        ('CFO', 'Chief Finance Officer'),
        ('VP', 'Vice President'),
        ]

    separators = ['/', ' and ', ' & ', '|', ',']

    # Normalize and/or replace known abbreviations
    # and build up a list of common titles.

    all_titles = []
    for i, _ in enumerate(contacts):
        if contacts[i]['Position'] == '':
            contacts[i]['Position'] = ['']
            continue
        titles = [contacts[i]['Position']]
        # flatten list
        titles = [item for sublist in titles for item in sublist]
        for separator in separators:
            for title in titles:
                if title.find(separator) >= 0:
                    titles.remove(title)
                    titles.extend([title.strip() for title in
                    title.split(separator) if title.strip() != ''])

        for transform in transforms:
            titles = [title.replace(*transform) for title in titles]

        contacts[i]['Position'] = titles
        all_titles.extend(titles)

    all_titles = list(set(all_titles))

    clusters = {}
    for title1 in all_titles:
        clusters[title1] = []
        for title2 in all_titles:
            if title2 in clusters[title1] or title2 in clusters and title1 in clusters[titl
```

```
                continue
            distance = DISTANCE(set(title1.split()), set(title2.split()))

            if distance < DISTANCE_THRESHOLD:
                clusters[title1].append(title2)

    # Flatten out clusters

    clusters = [clusters[title] for title in clusters if len(clusters[title]) > 1]

    # Round up contacts who are in these clusters and group them together

    clustered_contacts = {}
    for cluster in clusters:
        clustered_contacts[tuple(cluster)] = []
        for contact in contacts:
            for title in contact['Position']:
                if title in cluster:
                    clustered_contacts[tuple(cluster)].append('{0} {1}.'.format(
                        contact['FirstName'], contact['LastName'][0]))

    return clustered_contacts


clustered_contacts = cluster_contacts_by_title()

for titles in clustered_contacts:
    common_titles_heading = 'Common Titles: ' + ', '.join(titles)

    descriptive_terms = set(titles[0].split())
    for title in titles:
        descriptive_terms.intersection_update(set(title.split()))
    if len(descriptive_terms) == 0: descriptive_terms = ['***No words in common***']
    descriptive_terms_heading = 'Descriptive Terms: ' + ', '.join(descriptive_terms)
    print(common_titles_heading)
    print('\n'+descriptive_terms_heading)
    print('-' * 70)
    print('\n'.join(clustered_contacts[titles]))
    print()
```