Lab 6: Clustering the job titles of LinkedIn Connections using Greedy Heuristic Algorithm

NAME:R.RAHINI ROLLNO:215229128

#Clustering job titles using a greedy heuristic

In [ ]:

```
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
```

###Jaccard distance calculation

```python
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', 'Chief', 'Technology', 'Officer'}

Similarity: 0.5
Distance: 0.5
```

In [ ]:

```
job_title_1 = 'Vice President, Sales'.split()
job_title_2 = 'Vice President, Customer Relations'.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)))
```

```
['Vice', 'President,', 'Sales']
['Vice', 'President,', 'Customer', 'Relations']

Intersection:
{'Vice', 'President,'}

Union:
{'Sales', 'Relations', 'President,', 'Vice', 'Customer'}

Similarity: 0.4
Distance: 0.6
```

In [ ]:

```
contacts
```

Out[14]:

```
[OrderedDict([('First Name', 'Sathish'),
              ('Last Name', 'Kumar'),
              ('Email Address', ''),
              ('Company', 'Navabrind IT Solutions Pvt Ltd'),
              ('Position', 'Software Developer'),
              ('Connected On', '07-Aug-22'),
              ('Location', None)]),
 OrderedDict([('First Name', 'Padmanaban '),
              ('Last Name', 'T  Yadava'),
              ('Email Address', 'ctr.padmanaban@gmail.com'),
              ('Company', 'Jio ( Veremax Technologies)'),
              ('Position', 'Executive'),
              ('Connected On', '07-Aug-22'),
              ('Location', None)]),
 OrderedDict([('First Name', 'Rahul'),
              ('Last Name', 'Venkat S'),
              ('Email Address', ''),
              ('Company', ''),
```

```python
import json
# 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']]

        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[title2]:
                continue
            try:
                distance = DISTANCE(set(title1.split()), set(title2.split()))
            except:
                print(title1.split())
                print(title2.split())
                continue

            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()
```

```
Common Titles: Customer Facing Data Scientist, Data Scientist

Descriptive Terms: Data, Scientist
----------------------------------------------------------------------



Common Titles: Student, Masters Student, Student Researcher

Descriptive Terms: Student
----------------------------------------------------------------------



Common Titles: Operation Executive, Senior Operation Executive , Executive

Descriptive Terms: Executive
----------------------------------------------------------------------



Common Titles: Digital Specialist Engineer, Digital Marketing Specialist
```

###How to export data to power a dendogram and node-link tree visualization

```
pip install cluster
```

```
Looking in indexes: https://pypi.org/simple, (https://pypi.org/simple,) http
s://us-python.pkg.dev/colab-wheels/public/simple/ (https://us-python.pkg.de
v/colab-wheels/public/simple/)
Requirement already satisfied: cluster in /usr/local/lib/python3.7/dist-pack
ages (1.4.1.post3)
```

In [ ]:

```
import cluster
```

In [ ]:

```
contacts
```

Out[23]:

```
[OrderedDict([('First Name', 'Sathish'),
              ('Last Name', 'Kumar'),
              ('Email Address', ''),
              ('Company', 'Navabrind IT Solutions Pvt Ltd'),
              ('Position', ['Software Developer']),
              ('Connected On', '07-Aug-22'),
              ('Location', None)]),
 OrderedDict([('First Name', 'Padmanaban '),
              ('Last Name', 'T  Yadava'),
              ('Email Address', 'ctr.padmanaban@gmail.com'),
              ('Company', 'Jio ( Veremax Technologies)'),
              ('Position', ['Executive']),
              ('Connected On', '07-Aug-22'),
              ('Location', None)]),
 OrderedDict([('First Name', 'Rahul'),
              ('Last Name', 'Venkat S'),
              ('Email Address', ''),
              ('Company', ''),
import nltk
nltk.download('stopwords')
from nltk.metrics.distance import jaccard_distance
from nltk.corpus import stopwords # nltk.download('stopwords')
from cluster import HierarchicalClustering # pip install cluster
import os
CSV_FILE = os.path.join('Connections.csv')

# Tweak this distance threshold and try different distance calculations
# during experimentation
DISTANCE_THRESHOLD = 0.5
DISTANCE = jaccard_distance

# Adjust sample size as needed to reduce the runtime of the
# nested loop that invokes the DISTANCE function
SAMPLE_SIZE = 500

def cluster_contacts_by_title(csv_file):

    csvReader = csv.DictReader(open(csv_file), delimiter=',', quotechar='"')
    contacts = [row for row in csvReader]
    contacts = contacts[:SAMPLE_SIZE]

    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']]
        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 tit

        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))
    # Define a scoring function
    def score(title1, title2):
        return DISTANCE(set(title1.split()), set(title2.split()))

    # Feed the class your data and the scoring function
    hc = HierarchicalClustering(all_titles, score)

    # Cluster the data according to a distance threshold
    clusters = hc.getlevel(DISTANCE_THRESHOLD)

    # Remove singleton clusters
    clusters = [c for c in clusters if len(c) > 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['First Name'], contact['Last Name'][0]))

    return clustered_contacts, clusters

def get_descriptive_terms(titles):
    flatten = lambda l: [item for sublist in l for item in sublist]
    title_words = flatten([title.split() for title in titles])
    filtered_words = [word for word in title_words \
                      if word not in stopwords.words('english')]
    counter = Counter(filtered_words)
    descriptive_terms = counter.most_common(2)
    # Get the most common title words from a cluster, ignoring singletons
    descriptive_terms = [t[0] for t in descriptive_terms if t[1] > 1]
    return descriptive_terms


def display_output(clustered_contacts, clusters):
    for title_cluster in clusters:
        descriptive_terms = get_descriptive_terms(title_cluster)
        common_titles_heading = 'Common Titles: ' + ', '.join((t for t in title_cluster))
        descriptive_terms_heading =  'Descriptive Terms: ' + ', '.join((t for t in descript

        print(common_titles_heading)
        print(descriptive_terms_heading)
        print('-' * 70)
        #print(title_cluster)
        #print(clustered_contacts)
        print('\n'.join(clustered_contacts[tuple(title_cluster)]))
        print()
```

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

```
out_file
```

Out[89]:

```
<_io.TextIOWrapper name='sample.json' mode='w' encoding='UTF-8'>
```

In [108]:

```
clustered_contacts
```

Out[108]:

```
{('Associate Director - Data Science',
  'Director - Data Analytics'): ['Ritesh A.', 'Ashwini J.'],
 ('Associate Professor',
  'Associate',
  'Associate Consultant'): ['Dr. Selva Rani B.',
  'Sai Ramcharan T.',
  'Swati T.',
  'Jayaprakash R.'],
 ('Chief Technology Officer',
  'Chief Data Officer',
  'Chief Executive Officer'): ['Atul B.',
  'Yiqun H.',
  'Archna W.',
  'Sharala A.'],
 ('Consultant', 'Principal Consultant'): ['Mario C.', 'Dazil F.'],
 ('Customer Facing Data Scientist',
  'Data Scientist',
  'Senior Data Scientist'): ['Austin C.',
```

```python
def write_d3_json_output(clustered_contacts):

    json_output = {'name' : 'My LinkedIn', 'children' : []}

    for titles in clustered_contacts:

        descriptive_terms = get_descriptive_terms(titles)

        json_output['children'].append({'name' : ', '.join(descriptive_terms)[:30], 'childr
        with open("sample.json", "w") as out_file:
            json.dump(json_output,out_file,indent=1)
            out_file.close()

clustered_contacts, clusters = cluster_contacts_by_title(CSV_FILE)
display_output(clustered_contacts, clusters)
write_d3_json_output(clustered_contacts)
```

```
Common Titles: Group Project Manager, Infrastructure Project Manager
Descriptive Terms: Project, Manager
----------------------------------------------------------------------------------------------------
Enayat Haider (PRINCE2,Cybersecurity, A.
Bhupinder Singh C.

Common Titles: Senior Talent Development Consultant, Senior Immigration Co
nsultant, Senior Consultant, Senior Consultant QA
Descriptive Terms: Senior, Consultant
----------------------------------------------------------------------------------------------------
Roshini S.
Suyash R.
Shivani S.
Bhavna B.

Common Titles: Consultant, Principal Consultant
Descriptive Terms: Consultant
----------------------------------------------------------------------------------------------------
Mario C.
```