Company Name Clustering

-SHUBHAM BHATT

Aim :

To effectively cluster a set of company names, including various formats, acronyms,typing errors and variations, to identify similar entities despite differences in their representations.

Dataset:

1. Dataset used consists of variations of names of popular companies.
2. It includes:

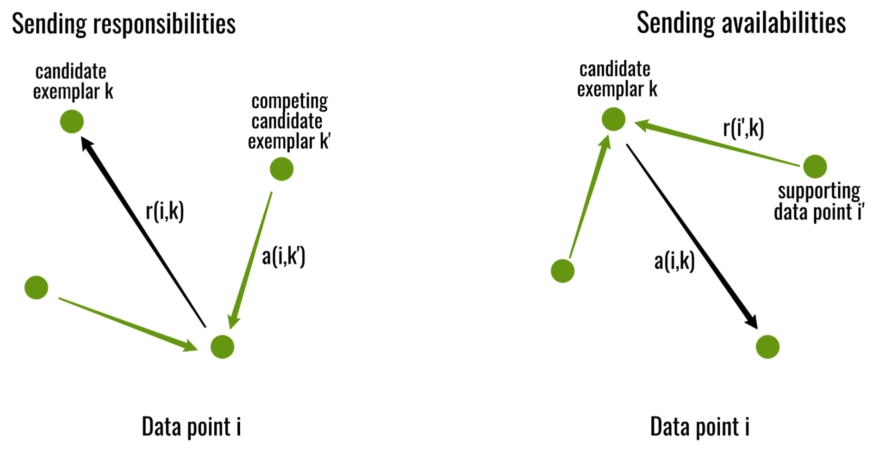
   company_names = ['Apple Inc.', 'Apple', 'AAPL Inc.', 'Apple Corp', 'Apple Computers', 'Appl Inc.', 'Apple Incorporation', 'Applle', 'Aple Inc', 'Microsoft Corporation', 'Microsoft Corp.', 'MSFT Inc.', 'MS Corp', 'Microsoft', 'Microsft', 'Microsof Corporation', 'Microsfot', 'Amazon.com, Inc.', 'Amazon', 'AMZN', 'Amazon Corp', 'Amazon Services', 'Amazn', 'Amzon', 'Amazom Inc', 'Amazom', 'Alphabet Inc.', 'Alphabet', 'Google LLC', 'Google Inc.', 'Google', 'Alphabt', 'Alphabe Inc.', 'Googel', 'Goole Inc', 'Facebook, Inc.', 'Facebook', 'FB Inc.', 'Meta Platforms', 'Meta', 'Facebok', 'Facbook Inc.', 'Metaa', 'Mta Platforms', 'Tesla, Inc.', 'Tesla Motors', 'Tesla', 'TSLA', 'Tesla Energy', 'Tesl Inc', 'Telsa', 'Tesla Incorporation', 'Tesl', 'Berkshire Hathaway Inc.', 'Berkshire Hathaway', 'Berkshire Inc.', 'BRK.A', 'BRK.B', 'Berkshire Hatway', 'Berkshire Hthaway Inc.', 'Berkshre', 'Berkshire Hathway', 'Alibaba Group Holding Limited', 'Alibaba', 'Alibaba Group', 'BABA', 'Ali Baba', 'Alibba', 'Alibab Group', 'AliBaba Holding', 'Alibab', 'Johnson & Johnson', 'Johnson and Johnson', 'J&J', 'JNJ', 'Johnson', 'Johnsn & Johnson', 'Johnson & Johson', 'J & J', 'Johnson Johnson', 'JPMorgan Chase & Co.', 'JP Morgan Chase', 'JPM Chase', 'JPMC', 'J.P. Morgan', 'JPMorgan', 'JP Morgan', 'JP MorganChase', 'JPMogan', 'Visa Inc.', 'Visa', 'V', 'Visa Corporation', 'Visa Corp', 'Vsa', 'Visa Incorporation', 'Visa Inc', 'VisaIncorporation', 'Samsung Electronics', 'Samsung', 'Samsung Corp.', 'Samsung Ltd.', 'Samsung Co.', 'Samsng', 'Samusng', 'Sammsung', 'Samsong', 'Nestlé S.A.', 'Nestle', 'Nestle Inc.', 'Nestle SA', 'Nestle Ltd.', 'Nestl', 'Nestl Inc', 'Nestl S.A', 'Nestlee', 'Procter & Gamble', 'Procter and Gamble', 'P&G', 'Proctor & Gamble', 'Procter & Gambl', 'Proctre & Gamble', 'P and G', 'P&G Inc.', 'Procter', 'Toyota Motor Corporation', 'Toyota', 'Toyota Motors', 'TM', 'Toyota Corp', 'Toyta', 'Toyot', 'Toyota Motor Corp.', 'Toyta Motor', 'Walt Disney Company', 'Disney']

Procedure:

1. Preprocess the data set
    a. Remove punctuations.
    b. Convert into lower case.
    c. Remove common suffixes and their abbreviations.
2. Generate Vectorized Data Matrix
    a. Two approaches were followed:
        i. TF - IDF Vectorization
        ii. Levenshtein Distance Similarity Matrix
    b. Levenshtein Distance Similarity Matrix provided better results as the values were calculated by comparison between each pair of data elements.
3. Clustering Algorithms
    a. Two algorithms were decided for clustering.
    b. DBSCAN -
        i. Clusters are dense regions in the data space, separated by regions of the lower density of points. The DBSCAN algorithm is based on this intuitive notion of “clusters” and “noise”.The key idea is that for each point of a cluster, the neighborhood of a given radius has to contain at least a minimum number of points.
        ii. Suitable only for compact and well-separated clusters.
        iii. Parameters Required For DBSCAN Algorithm
            1. **eps**: It defines the neighborhood around a data point i.e. if the distance between two points is lower or equal to ‘eps’ then they are considered neighbors.
            2. **MinPts**: Minimum number of neighbors (data points) within eps radius. The larger the dataset, the larger value of MinPts must be chosen.
        iv. Affinity Propagation -
            1. Affinity Propagation creates clusters by sending messages between data points until convergence.
            2. It does not require the number of clusters to be determined or estimated before running the algorithm, for this purpose the two important parameters are the **preference**, which controls how many exemplars (or prototypes) are used, and the **damping factor** which damps the responsibility and availability of messages to avoid numerical oscillations when updating these messages.
            3. The algorithm proceeds by alternating two message passing steps, to update two matrices:
                a. The “responsibility” matrix R has values r(i, k) that quantify how well-suited Xk is to serve as the

exemplar for Xi, relative to other candidate exemplars for Xi.

b. The “availability” matrix A contains values a(i, k) that represent how “appropriate” it would be for xi to pick Xk as its exemplar, taking into account other points’ preference for Xk as an exemplar.



Updating Responsibility r(i,k):

$$r(i,k) \leftarrow s(i,k) - \max_{k' \neq k} \{a(i,k') + s(i,k')\}$$

Updating Availability a(i,k):

$$a(i,k) \leftarrow \min\left(0, r(k,k) + \sum_{i' \notin \{i,k\}} \max(0, r(i',k))\right) \text{ for } i \neq k \text{ and}$$
$$a(k,k) \leftarrow \sum_{i' \neq k} \max(0, r(i',k)).$$

Where, the negative squared distance of two data points was used as s i.e. for points Xi and Xk,

s(i, j)= - || Xi -Xk ||^2

Output:

- **Affinity Propagation accuracy = 3 x accuracy DBSCAN.**

Levenshtein Similarity Matrix along with DBSCAN.

```
Clusters:

Cluster 0: ['Apple Inc.', 'Apple', 'Apple Corp', 'Appl Inc.', 'Apple Incorporation', 'Applle',
'Aple Inc']

Cluster -1: ['AAPL Inc.', 'Apple Computers', 'MSFT Inc.', 'MS Corp', 'Amazon.com, Inc.',
'AMZN', 'Amazon Services', 'Google LLC', 'Googel', 'FB Inc.', 'Meta', 'Metaa', 'Tesla
Motors', 'TSLA', 'Tesla Energy', 'Telsa', 'Tesl', 'Berkshire Inc.', 'BRK.A', 'BRK.B',
'Berkshre', 'Alibaba Group Holding Limited', 'BABA', 'AliBaba Holding', 'J&J', 'JNJ',
'Johnson', 'J & J', 'JPMorgan Chase & Co.', 'JPM Chase', 'JPMC', 'Visa', 'V', 'Vsa',
'VisaIncorporation', 'Samsung Electronics', 'Samsung Co.', 'P&G', 'P and G', 'P&G Inc.',
'Procter', 'TM', 'Walt Disney Company', 'Disney']

Cluster 1: ['Microsoft Corporation', 'Microsoft Corp.', 'Microsoft', 'Microsft', 'Microsof
Corporation', 'Microsfot']

Cluster 2: ['Amazon', 'Amazon Corp', 'Amazn', 'Amzon']

Cluster 3: ['Amazom Inc', 'Amazom']

Cluster 4: ['Alphabet Inc.', 'Alphabet', 'Alphabt', 'Alphabe Inc.']

Cluster 5: ['Google Inc.', 'Google', 'Goole Inc']

Cluster 6: ['Facebook, Inc.', 'Facebook', 'Facebok', 'Facbook Inc.']

Cluster 7: ['Meta Platforms', 'Mta Platforms']

Cluster 8: ['Tesla, Inc.', 'Tesla', 'Tesl Inc', 'Tesla Incorporation']

Cluster 9: ['Berkshire Hathaway Inc.', 'Berkshire Hathaway', 'Berkshire Hatway',
'Berkshire Hthaway Inc.', 'Berkshire Hathway']

Cluster 10: ['Alibaba', 'Ali Baba', 'Alibba', 'Alibab']

Cluster 11: ['Alibaba Group', 'Alibab Group']
```

Cluster 12: ['Johnson & Johnson', 'Johnson and Johnson', 'Johnsn & Johnson', 'Johnson & Johson', 'Johnson Johnson']

Cluster 13: ['JP Morgan Chase', 'JP MorganChase']

Cluster 14: ['J.P. Morgan', 'JPMorgan', 'JP Morgan', 'JPMogan']

Cluster 15: ['Visa Inc.', 'Visa Corporation', 'Visa Corp', 'Visa Incorporation', 'Visa Inc']

Cluster 16: ['Samsung', 'Samsung Corp.', 'Samsung Ltd.', 'Samsng', 'Samusng', 'Sammsung', 'Samsong']

Cluster 17: ['Nestlé S.A.', 'Nestle SA', 'Nestl S.A']

Cluster 18: ['Nestle', 'Nestle Inc.', 'Nestle Ltd.', 'Nestl', 'Nestl Inc', 'Nestlee']

Cluster 19: ['Procter & Gamble', 'Procter and Gamble', 'Proctor & Gamble', 'Procter & Gambl', 'Proctre & Gamble']

Cluster 20: ['Toyota Motor Corporation', 'Toyota Motors', 'Toyota Motor Corp.', 'Toyta Motor']

Cluster 21: ['Toyota', 'Toyota Corp', 'Toyta', 'Toyot']

- Here all elements that are considered to be **'noise' are a port of Cluster -1**.

Levenshtein Similarity Matrix along with Affinity Propagation.

Clusters:

Cluster 0: ['Apple Inc.', 'Apple', 'AAPL Inc.', 'Apple Corp', 'Appl Inc.', 'Apple Incorporation', 'Opplle', 'Aple Inc']

Cluster 20: ['Apple Computers', 'Alibaba Group Holding Limited', 'VisaIncorporation', 'P and G', 'Walt Disney Company', 'Disney']

Cluster 1: ['Microsoft Corporation', 'Microsoft Corp.', 'MSFT Inc.', 'Microsoft', 'Microsft', 'Microsof Corporation', 'Microsfot']

Cluster 14: ['MS Corp', 'Visa Inc.', 'Visa', 'Visa Corporation', 'Visa Corp', 'Vsa', 'Visa Incorporation', 'Visa Inc']

Cluster 2: ['Amazon.com, Inc.', 'Amazon', 'AMZN', 'Amazon Corp', 'Amazon Services', 'Amazn', 'Amzon', 'Amazom Inc', 'Amazom']

Cluster 3: ['Alphabet Inc.', 'Alphabet', 'Alphabt', 'Alphabe Inc.']

Cluster 4: ['Google LLC', 'Google Inc.', 'Google', 'Googel', 'Goole Inc']

Cluster 5: ['Facebook, Inc.', 'Facebook', 'FB Inc.', 'Facebok', 'Facbook Inc.']

Cluster 6: ['Meta Platforms', 'Meta', 'Metaa', 'TM']

Cluster 19: ['Mta Platforms', 'Tesla Motors', 'Toyota Motor Corporation', 'Toyota', 'Toyota Motors', 'Toyota Corp', 'Toyta', 'Toyot', 'Toyota Motor Corp.', 'Toyta Motor']

Cluster 7: ['Tesla, Inc.', 'Tesla', 'TSLA', 'Tesla Energy', 'Tesl Inc', 'Telsa', 'Tesla Incorporation', 'Tesl']

Cluster 9: ['Berkshire Hathaway Inc.', 'Berkshire Hathaway', 'Berkshire Inc.', 'Berkshire Hatway', 'Berkshire Hthaway Inc.', 'Berkshre', 'Berkshire Hathway']

Cluster 8: ['BRK.A', 'BRK.B']

Cluster 10: ['Alibaba', 'Alibaba Group', 'BABA', 'Ali Baba', 'Alibba', 'Alibab Group', 'AliBaba Holding', 'Alibab']

Cluster 12: ['Johnson & Johnson', 'Johnson and Johnson', 'Johnson', 'Johnsn & Johnson', 'Johnson & Johson', 'Johnson Johnson']

Cluster 11: ['J&J', 'JNJ', 'J & J', 'V']

Cluster 13: ['JPMorgan Chase & Co.', 'JP Morgan Chase', 'JPM Chase', 'J.P. Morgan', 'JPMorgan', 'JP Morgan', 'JP MorganChase', 'JPMogan']

Cluster 18: ['JPMC', 'P&G', 'P&G Inc.']

Cluster 15: ['Samsung Electronics', 'Samsung', 'Samsung Corp.', 'Samsung Ltd.', 'Samsung Co.', 'Samsng', 'Samusng', 'Sammsung', 'Samsong']

Cluster 16: ['Nestlé S.A.', 'Nestle', 'Nestle Inc.', 'Nestle SA', 'Nestle Ltd.', 'Nestl', 'Nestl Inc', 'Nestl S.A', 'Nestlee']

Cluster 17: ['Procter & Gamble', 'Procter and Gamble', 'Proctor & Gamble', 'Procter & Gambl', 'Proctre & Gamble', 'Procter']

Code:

```
import re
import sklearn
import gensim
from gensim.models import Word2Vec
import numpy as np

def preprocess_company_names(company_names):
    processed_names = []
    for name in company_names:
        # Remove punctuation
        name = re.sub(r'[^\w\s]', '', name)
        # Convert to lowercase
        name = name.lower()
        # Remove suffixes
        name = re.sub(r'\bltd\b', '', name)
        name = re.sub(r'\bpvt\b', '', name)
        name = re.sub(r'\bcorp\b', '', name)
        name = re.sub(r'\binc\b', '', name)
        name = re.sub(r'\blimited\b', '', name)
        name = re.sub(r'\bprivate\b', '', name)
        name = re.sub(r'\bcorporation\b', '', name)
        name = re.sub(r'\bincorporation\b', '', name)

        processed_names.append(name)

    return processed_names


company_names = [
    'Apple Inc.', 'Apple', 'AAPL Inc.', 'Apple Corp', 'Apple Computers',
'Appl Inc.', 'Apple Incorporation', 'Applle', 'Aple Inc',
    'Microsoft Corporation', 'Microsoft Corp.', 'MSFT Inc.', 'MS Corp',
'Microsoft', 'Microsft', 'Microsof Corporation', 'Microsfot',
    'Amazon.com, Inc.', 'Amazon', 'AMZN', 'Amazon Corp', 'Amazon
Services', 'Amazn', 'Amzon', 'Amazom Inc', 'Amazom',
    'Alphabet Inc.', 'Alphabet', 'Google LLC', 'Google Inc.', 'Google',
'Alphabt', 'Alphabe Inc.', 'Googel', 'Goole Inc',
    'Facebook, Inc.', 'Facebook', 'FB Inc.', 'Meta Platforms', 'Meta',
'Facebok', 'Facbook Inc.', 'Metaa', 'Mta Platforms',
    'Tesla, Inc.', 'Tesla Motors', 'Tesla', 'TSLA', 'Tesla Energy', 'Tesl
Inc', 'Telsa', 'Tesla Incorporation', 'Tesl',
    'Berkshire Hathaway Inc.', 'Berkshire Hathaway', 'Berkshire Inc.',
'BRK.A', 'BRK.B', 'Berkshire Hatway', 'Berkshire Hthaway Inc.',
'Berkshre', 'Berkshire Hathway',
    'Alibaba Group Holding Limited', 'Alibaba', 'Alibaba Group', 'BABA',
'Ali Baba', 'Alibba', 'Alibab Group', 'AliBaba Holding', 'Alibab',
    'Johnson & Johnson', 'Johnson and Johnson', 'J&J', 'JNJ', 'Johnson',
'Johnsn & Johnson', 'Johnson & Johson', 'J & J', 'Johnson Johnson',
    'JPMorgan Chase & Co.', 'JP Morgan Chase', 'JPM Chase', 'JPMC', 'J.P.
Morgan', 'JPMorgan', 'JP Morgan', 'JP MorganChase', 'JPMogan',
    'Visa Inc.', 'Visa', 'V', 'Visa Corporation', 'Visa Corp', 'Vsa',
'Visa Incorporation', 'Visa Inc', 'VisaIncorporation',
    'Samsung Electronics', 'Samsung', 'Samsung Corp.', 'Samsung Ltd.',
'Samsung Co.', 'Samsng', 'Samusng', 'Sammsung', 'Samsong',
    'Nestlé S.A.', 'Nestle', 'Nestle Inc.', 'Nestle SA', 'Nestle Ltd.',
'Nestl', 'Nestl Inc', 'Nestl S.A', 'Nestlee',
    'Procter & Gamble', 'Procter and Gamble', 'P&G', 'Proctor & Gamble',
'Procter & Gambl', 'Proctre & Gamble', 'P and G', 'P&G Inc.', 'Procter',
    'Toyota Motor Corporation', 'Toyota', 'Toyota Motors', 'TM', 'Toyota
Corp', 'Toyta', 'Toyot', 'Toyota Motor Corp.', 'Toyta Motor',
    'Walt Disney Company', 'Disney']
print(len(company_names))



# Preprocess company names
processed_company_names = preprocess_company_names(company_names)

print(processed_company_names)
```

```
from fuzzywuzzy import fuzz
import numpy as np

# Number of companies
n = len(company_names)

# Initialize the similarity matrix
similarity_matrix = np.zeros((n, n))

# Compute similarity scores
for i in range(n):
    for j in range(n):
        if i != j:
            similarity_matrix[i, j] =
fuzz.ratio(processed_company_names[i], processed_company_names[j])
        else:
            similarity_matrix[i, j] = 100  # Maximum similarity with
itself

# Display the similarity matrix
print(similarity_matrix)
```

```
from sklearn.cluster import DBSCAN
from sklearn.metrics import pairwise_distances

# Compute the distance matrix
distance_matrix = 100 - similarity_matrix

from sklearn.metrics.pairwise import cosine_distances

#word_cosine = cosine_distances(distance_matrix)

dbscan = DBSCAN(metric='precomputed', eps=10, min_samples=2)
labels = dbscan.fit_predict(distance_matrix)

print(labels)

clusters = {}
for idx, label in enumerate(labels):
```

```python
    if label not in clusters:
        clusters[label] = []
    clusters[label].append(company_names[idx])

print("\nClusters:")
for cluster_id, members in clusters.items():
    print(f"Cluster {cluster_id}: {members}")
```

```python
for cluster_id, members in clusters.items():
    print(f"Cluster {cluster_id}:" , len(members))
```

```python

from sklearn.metrics import pairwise_distances

distance_matrix = 100 - similarity_matrix

preferences = np.linspace(np.min(similarity_matrix),
np.max(similarity_matrix), 10)


affinity_propagation = AffinityPropagation(random_state=0,
affinity="euclidean", verbose=2, max_iter=500)
labels = affinity_propagation.fit_predict(distance_matrix)

print(labels)

clusters = {}
for idx, label in enumerate(labels):
    if label not in clusters:
        clusters[label] = []
    clusters[label].append(company_names[idx])

print("\nClusters:")
for cluster_id, members in clusters.items():
    print(f"Cluster {cluster_id}: {members}")
```

```python
for cluster_id, members in clusters.items():
    print(f"Cluster {cluster_id}:" , len(members))
```

SUGGESTIONS and IDEAS:

1. Since most company names and their variations have a common first name, we can tokenize the words to use only the first name.
    a. E.g. 'Samsung Electronics', 'Samsung', 'Samsung Corp.', 'Samsung Ltd.', 'Samsung Co.' all share a common first name.

2. Create and map the short forms to the company names and use them to cluster.
    a. E.g. ‘Johnson & Johnson' and its variations were not clustered with their abbreviation 'J&J', 'JNJ', 'J & J'. So creating an acronym for the parent name can allow us to cluster the abbreviations efficiently.