# Web Mining Lab CAT

# -Devang Mehrotra

# 18BCE0763

# Set-C

# Importing Libraries

In [23]:

```
import numpy as np
import pandas as pd
import nltk
import re
import os
import codecs
```

In [24]:

```
from sklearn import feature_extraction
from nltk.stem.snowball import SnowballStemmer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import ward, dendrogram
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
```

# Importing Files

In [25]:

```
titles = []
data = []
for i in os.listdir('.\Q3'):
    if i.endswith('.txt'):
        titles.append(i)
        with open(".\Q3\\" + str(i), 'r+',encoding='utf-8',errors='ignore') as text:
            data.append(text.read())
```

In [74]:

```
data
```

Out[74]:

```
['Electric automotive maker Tesla Inc. is likely to introduce its products i
n India sometime in the summer of 2017.',
 'Automotive major Mahindra likely to introduce driverless car',
 'BMW plans to introduce its own motorcycles in india',
 'Just drive, a self-drive car rental firm uses smart vehicle technology bas
ed on IoT',
 'Automotive industry going to hire thousands in 2018',
 'Famous cricket player Dhoni brought his priced car Hummer which is an SU
V',
 'Dhoni led india to its second world cup victory',
 'IoT in car will lead to more safety and make driverless vehicle revolution
possible',
 'Sachin recommended Dhoni for the indian skipper post']
```

In [26]:

```
stopwords = nltk.corpus.stopwords.words('english')
stemmer = SnowballStemmer('english')
```

In [27]:

```
def tokenize_and_stem(text):
    tokens = [word for sent in nltk.sent_tokenize(text) for word in nltk.word_tokenize(sent
    filtered_tokens = []
    for token in tokens:
        if re.search('[a-zA-Z]', token):
            filtered_tokens.append(token)
    stems = [stemmer.stem(t) for t in filtered_tokens]
    return stems
```

In [28]:

```
def tokenize(text):
    tokens = [word.lower() for sent in nltk.sent_tokenize(text) for word in nltk.word_token
    filtered_tokens = []
    for token in tokens:
        if re.search('[a-zA-Z]', token):
            filtered_tokens.append(token)
    return filtered_tokens
```

In [29]:

```
totalvocab_stemmed = []
totalvocab_tokenized = []

for i in data:
    allwords_stemmed = tokenize_and_stem(i)
    totalvocab_stemmed.extend(allwords_stemmed)

    allwords_tokenized = tokenize(i)
    totalvocab_tokenized.extend(allwords_tokenized)

vocab = pd.DataFrame({'words': totalvocab_tokenized}, index = totalvocab_stemmed)
```

# Representing the documents in Vector Space Model using the given Keywords

In [30]:

```
vocab
```

Out[30]:

| | words |
|---|---|
| **electr** | electric |
| **automot** | automotive |
| **maker** | maker |
| **tesla** | tesla |
| **inc.** | inc. |
| **...** | ... |
| **for** | for |
| **the** | the |
| **indian** | indian |
| **skipper** | skipper |
| **post** | post |

100 rows × 1 columns

In [31]:

```
tfidf_vectorizer = TfidfVectorizer(max_df=0.8, max_features=200000,min_df=0.2, stop_words='
%time tfidf_matrix = tfidf_vectorizer.fit_transform(data)
```

```
Wall time: 46.9 ms

C:\ProgramData\Anaconda3\envs\dev\lib\site-packages\sklearn\feature_extracti
on\text.py:386: UserWarning: Your stop_words may be inconsistent with your p
reprocessing. Tokenizing the stop words generated tokens ['abov', 'afterwar
d', 'alon', 'alreadi', 'alway', 'ani', 'anoth', 'anyon', 'anyth', 'anywher',
'becam', 'becaus', 'becom', 'befor', 'besid', 'cri', 'describ', 'dure', 'el
s', 'elsewher', 'empti', 'everi', 'everyon', 'everyth', 'everywher', 'fift
i', 'forti', 'henc', 'hereaft', 'herebi', 'howev', 'hundr', 'inde', 'mani',
'meanwhil', 'moreov', 'nobodi', 'noon', 'noth', 'nowher', 'onc', 'onli', 'ot
herwis', 'ourselv', 'perhap', 'pleas', 'sever', 'sinc', 'sincer', 'sixti',
'someon', 'someth', 'sometim', 'somewher', 'themselv', 'thenc', 'thereaft',
'therebi', 'therefor', 'togeth', 'twelv', 'twenti', 'veri', 'whatev', 'when
c', 'whenev', 'wherea', 'whereaft', 'wherebi', 'wherev', 'whi', 'yourselv']
not in stop_words.
  'stop_words.' % sorted(inconsistent))
```

In [32]:

```
terms = tfidf_vectorizer.get_feature_names()
dist = 1 - cosine_similarity(tfidf_matrix)
```

In [66]:

```
X = ward(dist)
```

# Defining the Number of Iterations

In [106]:

```
max_iter = 4
```

# K-Means Clustering(K=4)

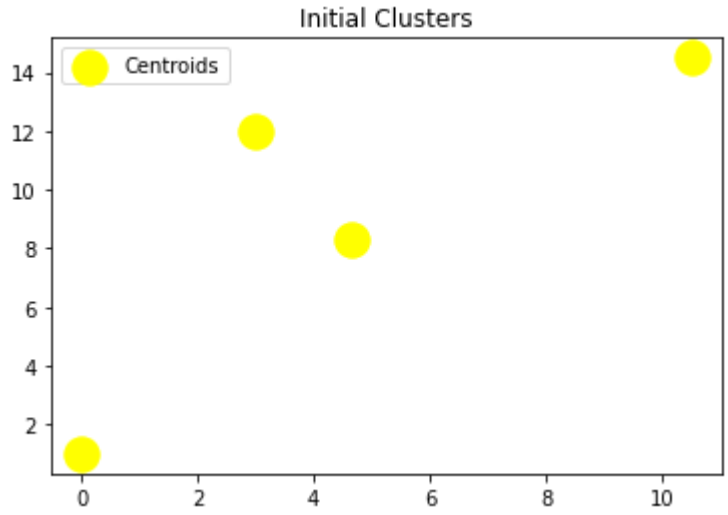
In [107]:

```
kmeans = KMeans(n_clusters = 4, init = 'k-means++',random_state = 42,max_iter=4)
y_kmeans = kmeans.fit_predict(X)
```

# Initial Centroids

In [108]:

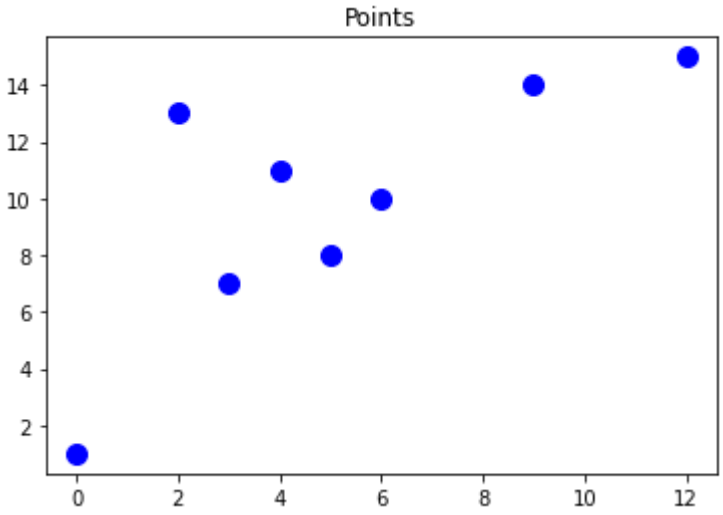
```
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c = 'yel
plt.title('Initial Clusters')
plt.legend()
plt.show()
```




# Points

In [109]:

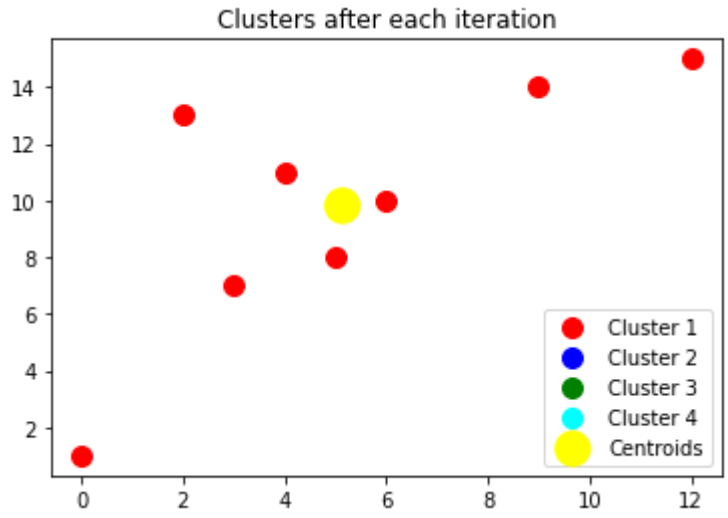
```
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 100, c = 'blue')
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 100, c = 'blue')
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 100, c = 'blue')
plt.scatter(X[y_kmeans == 3, 0], X[y_kmeans == 3, 1], s = 100, c = 'blue')
plt.title('Points')
plt.show()
```

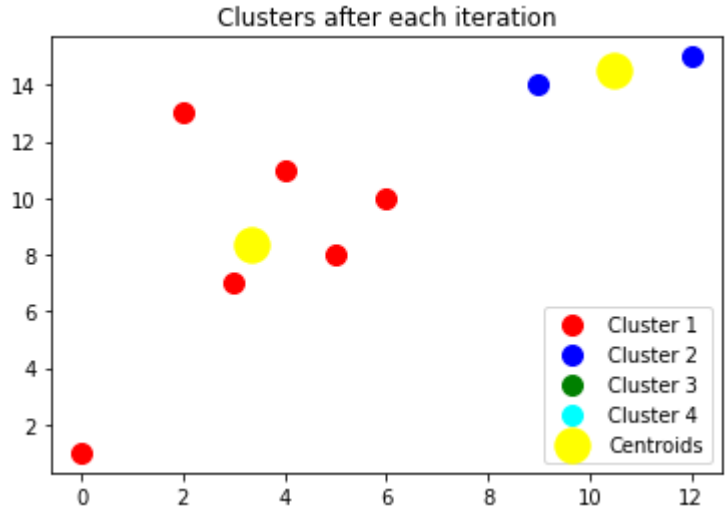


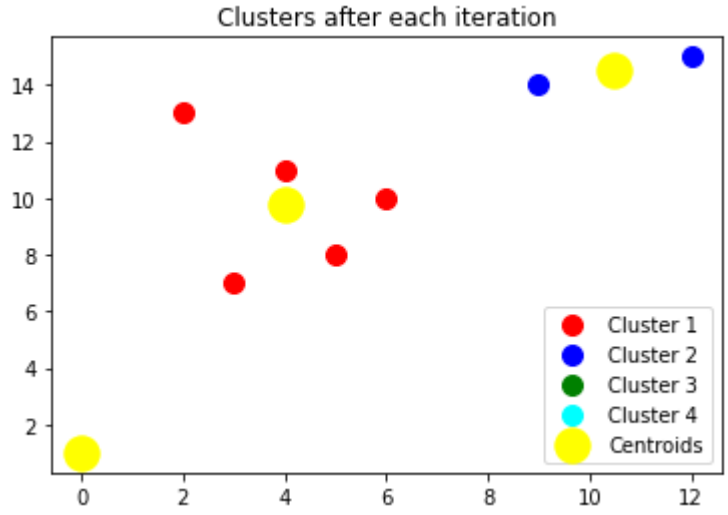

# Clusters after each Iteration and revised Centroids after each Iteration

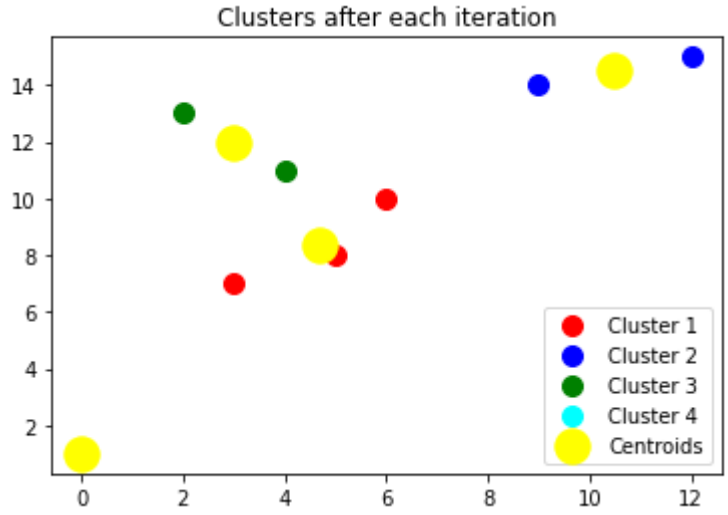
In [117]:

```
for i in range(1,5):
    kmeans = KMeans(n_clusters = i, init = 'k-means++',random_state = 42,max_iter=4)
    y_kmeans = kmeans.fit_predict(X)
    plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 100, c = 'red', label = 'Clus
    plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 100, c = 'blue', label = 'Clu
    plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 100, c = 'green', label = 'Cl
    plt.scatter(X[y_kmeans == 3, 0], X[y_kmeans == 3, 1], s = 100, c = 'cyan', label = 'Clu
    plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c =
    plt.title('Clusters after each iteration')
    plt.legend()
    plt.show()
```



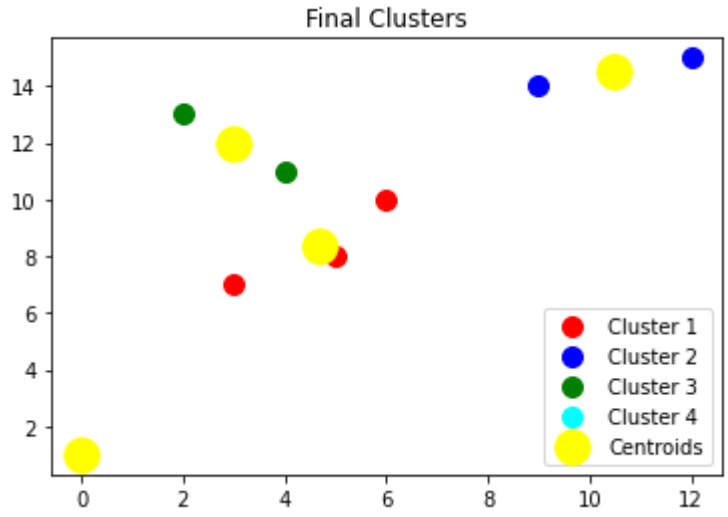



# Final Clustering Results after 4 iterations

In [111]:

```
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 100, c = 'red', label = 'Cluster
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 100, c = 'blue', label = 'Cluster
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 100, c = 'green', label = 'Cluste
plt.scatter(X[y_kmeans == 3, 0], X[y_kmeans == 3, 1], s = 100, c = 'cyan', label = 'Cluster
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c = 'yel
plt.title('Final Clusters')
plt.legend()
plt.show()
```




# Number of iterations required=4 (Specified Earlier)