# Xử lý ngôn ngữ tự nhiên

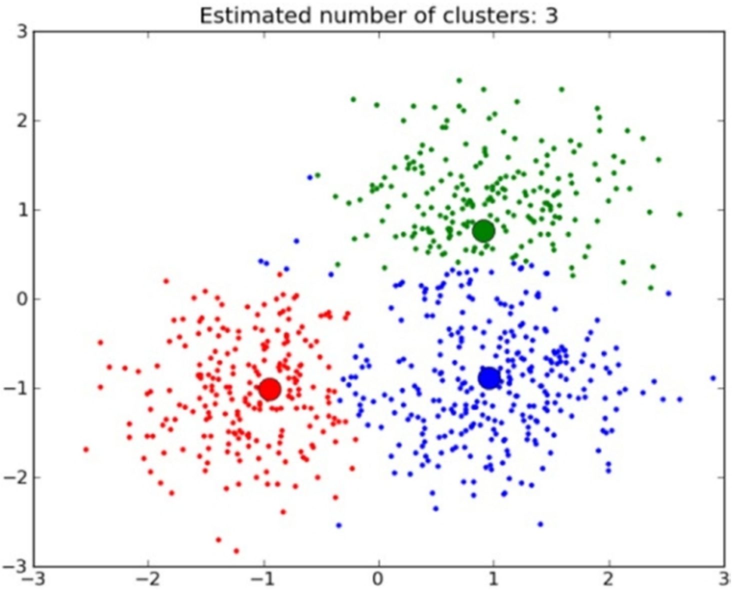
# Chương 8: Text similarity and clustering - part 2

Khoa CNTT, Đại học Kỹ thuật - Công nghệ Cần Thơ

# Document clustering

- Document clustering, cluster analytics là lĩnh vực nghiên cứu của NLP ứng dụng học không giám sát (unsupervised learning)
- Cơ sử của document clustering là sự bắt đầu với toàn bộ corpus, sau đó phân tách (segregation) thành cách nhóm (groups) dựa trên tính chất khác biệt (distinctive properties), thuộc tính (attributes), features của documents

- Không biết trước bao nhiêu group cần phân tách
- Các documents thuộc về 1 group sẽ có sự tương đồng cao hơn so với các documents thuộc group khác
- `http://scikit-learn.org/stable/modules/clustering.html`


Estimated number of clusters: 3
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3

# Một số kỹ thuật và phương pháp tìm cluster

- Hierarchical clustering models
- Centroid-based clustering models
- Distribution-based clustering models
- Density-based clustering models

# Clustering greatest movies of all time

- Chúng ta tiến hành clustering 100 bộ phim phổ biến dựa trên internet movie database[1]
- Mỗi bộ phim gồm tên phim và phần tóm tắt (synopses)
- Top 100 Greatest Movies of All Time (The Ultimate List): `https://www.imdb.com/list/ls055592025/`
- movie_data.csv trên hệ thống LearnWeb

[1] www.imdb.com

```python
import numpy as np
import pandas as pd

movie_data = pd.read_csv('movie_data.csv')

print(movie_data.head())

movie_titles = movie_data['Title'].tolist()
movie_synopses = movie_data['Synopsis'].tolist()

print('Movie:', movie_titles[0])
# print first 1000 characters
print('Movie Synopsis:', movie_synopses[0][:1000])
```

```
#  Title                                                            Synopsis
#0              The Godfather  In late summer 1945, guests are gathered for t...
#1  The Shawshank Redemption  In 1947, Andy Dufresne (Tim Robbins), a banker...
#2           Schindler's List  The relocation of Polish Jews from surrounding...
#3                 Raging Bull  The film opens in 1964, where an older and fat...
#4                  Casablanca  In the early years of World War II, December 1...
#Movie: The Godfather
#Movie Synopsis: In late summer 1945, guests are gathered for the wedding reception of Don Vito Corleone's daughter Connie (Talia Shire) and Carlo Rizzi (Gianni Russo). Vito (Marlon Brando), the head of the Corleone Mafia family, is known to friends and associates as "Godfather." He and Tom Hagen (Robert Duvall), the Corleone family lawyer, are hearing requests for favors because, according to Italian tradition, "no Sicilian can refuse a request
```

```python

def remove_characters_after_tokenization(tokens):
    import re
    import string
    pattern = re.compile("[{}]".format(re.escape(string.punctuation)))
    filtered_tokens = list(filter(None, [pattern.sub("",token) for token in tokens]))
    return filtered_tokens

def remove_stopwords(tokens, language="english"):
    stopword_list = nltk.corpus.stopwords.words(language)
    filter_tokens = [token for token in tokens if token not in stopword_list]
    return filter_tokens
```

```python

def remove_repeated_characters(tokens):
    from nltk.corpus import wordnet
    import re

    repeat_pattern = re.compile(r'(\w*)(\w)\2(\w*)')
    match_substitution = r'\1\2\3'
    def replace(old_word):
        if wordnet.synsets(old_word):
            return old_word
        new_word = repeat_pattern.sub(match_substitution, old_word)
        return replace(new_word) if new_word != old_word else new_word

    correct_tokens = [replace(word) for word in tokens]
    return correct_tokens
```

```python

def text_stemming(tokens):
    from nltk.stem.snowball import SnowballStemmer
    stemmer = SnowballStemmer("english")
    def replace(old_word):
        return stemmer.stem(old_word)

    filter_tokens = [replace(token) for token in tokens]
    return filter_tokens

def text_normalization(tokens):
    tokens = remove_characters_after_tokenization(tokens)
    tokens = text_stemming(tokens)
    tokens = remove_stopwords(tokens)
    tokens = remove_repeated_characters(tokens)
    return tokens

def tokens_to_string(tokens):
    return " ".join(tokens)
```

```python

import nltk
default_wt = nltk.word_tokenize

tokens_synoses = []
for syn in movie_synopses:
    tokens_synoses.append(default_wt(text=syn))

tokens_synoses_norm = []
for tokens in tokens_synoses:
    tokens_synoses_norm.append(text_normalization(tokens))

tokens_synoses_norm_string = []
for tokens_norm in tokens_synoses_norm:
    tokens_synoses_norm_string.append(tokens_to_string(tokens_norm))
```

```
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer

def build_feature_matrix(documents, feature_type='frequency', ngram_range=(1, 1), min_df=0.0, max_df=1.0):
    feature_type = feature_type.lower().strip()

    if feature_type == 'binary':
        vectorizer = CountVectorizer(binary=True, min_df=min_df, max_df=max_df, ngram_range=ngram_range)
    elif feature_type == 'frequency':
        vectorizer = CountVectorizer(binary=False, min_df=min_df, max_df=max_df, ngram_range=ngram_range)
    elif feature_type == 'tfidf':
        vectorizer = TfidfVectorizer(min_df=min_df, max_df=max_df, ngram_range=ngram_range)
```

```
    else:
        raise Exception("Wrong feature type entered. Possible values: 'binary', 'frequency', 'tfidf'")

    feature_matrix = vectorizer.fit_transform(documents).astype(float)
    return vectorizer, feature_matrix

vectorizer, feature_matrix = build_feature_matrix(
    tokens_synoses_norm_string, feature_type="tfidf")
print(feature_matrix.shape)
#(100, 10797)

#get feature names
feature_names = vectorizer.get_feature_names()
print(feature_names[:20])
#['10', '100', '1000', '10000', '1000000', '101st', '108', '11', '110',
    '11th', '12', '120pound', '1212', '1280', #'1297', '1298', '12fot
    ', '12yearold', '13', '130']
```

# K-means clustering

- k-means clustering algorithm thuộc nhóm centroid-based clustering
- Nhóm data vào các clusters dựa trên sự khác biệt / giống nhau
- Tiêu chuẩn (criteria) hoặc phép đo (measure) mà giải thuật sử dụng là tối thiểu hóa *inertia*, within-cluster sum-of-squares
- Xác định giá trị $k$ trước khi thực thi giải thuật

- Giải sử ta có dataset $X$ với $N$ data points
- Ta muốn nhóm thành $K$ clusters
- Giải thuật k-means sẽ tuần tự phân chia $N$ data points vào $K$ clusters riêng biệt $C_k$
- Mỗi cluster $C_k$ được xem như trung bình của các cluster samples (data point thuộc cluster được gọi là cluster samples)
- Giá trị trung bình là cluster centroid $\mu_k$
- Chọn các giá trị $\mu_k$ sao cho inertia cực tiểu

$$min \sum_{i=1}^{K} \sum_{x_n \in C_i} \|x_n - \mu_i\|^2 \tag{1}$$

- Chọn $k$ centroids $\mu_k$ ban đầu bằng ngẫu nhiên
- Cập nhật clusters bằng cách gán data points vào centroid point gần nhất

$$C_k = \{x_n : \|x_n - \mu_k\| \leq all\|x_n - \mu_l\|\}$$

- Cập nhật giá trị centroids

$$\mu_k = \frac{1}{C_k} \sum_{x_n \in C_k} x_n$$

```
from sklearn.cluster import KMeans

def k_means(feature_matrix, num_clusters=5):
    km = KMeans(n_clusters=num_clusters, max_iter=10000)
    km.fit(feature_matrix)
    clusters = km.labels_
    return km, clusters

num_clusters = 5
km_obj, clusters = k_means(feature_matrix=feature_matrix,num_clusters=
    num_clusters)
movie_data['Cluster'] = clusters
```

```
from collections import Counter
# get the total number of movies per cluster
c = Counter(clusters)
print (c.items())
#dict_items([(0, 6), (1, 53), (2, 8), (3, 29), (4, 4)])
```

- Kết quả trả về là danh sách các clusters từ 0 đến 4 cùng với số data points thuộc về clusters đó

```python

def get_cluster_data(clustering_obj, movie_data,
    feature_names, num_clusters,
    topn_features=10):

    cluster_details = {}
   # get cluster centroids
    ordered_centroids = clustering_obj.cluster_centers_.argsort()[:,
    ::-1]
   # get key features for each cluster
   # get movies belonging to each cluster
    for cluster_num in range(num_clusters):
        cluster_details[cluster_num] = {}
        cluster_details[cluster_num]['cluster_num'] = cluster_num
        key_features = [feature_names[index]
                for index
                in ordered_centroids[cluster_num, :topn_features]]
        cluster_details[cluster_num]['key_features'] = key_features
```

```python
        movies = movie_data[movie_data['Cluster'] == cluster_num]['Title'].values.tolist()
        cluster_details[cluster_num]['movies'] = movies

    return cluster_details
```

- Kết quả trả về là thông tin cluster:
  - key features có tầm quan trọng xác định cluster từ centroids
  - movie titles thuộc về từng cluster

```python

def print_cluster_data(cluster_data):
    # print cluster details
    for cluster_num, cluster_details in cluster_data.items():
    print('Cluster {} details:'.format(cluster_num))
    print('-'*20)
    print('Key features:', cluster_details['key_features'])
    print('Movies in this cluster:')
    print(', '.join(cluster_details['movies']))
    print('='*40)

import matplotlib.pyplot as plt
from sklearn.manifold import MDS
from sklearn.metrics.pairwise import cosine_similarity
import random
from matplotlib.font_manager import FontProperties
```

```python

def plot_clusters(num_clusters, feature_matrix, cluster_data, movie_data, plot_size=(16,8)):
    # generate random color for clusters
    def generate_random_color():
        color = '#%06x' % random.randint(0, 0xFFFFFF)
        return color

    # define markers for clusters
    markers = ['o', 'v', '^', '<', '>', '8', 's', 'p', '*', 'h', 'H', 'D', 'd']
    # build cosine distance matrix
    cosine_distance = 1 - cosine_similarity(feature_matrix)
    # dimensionality reduction using MDS
    mds = MDS(n_components=2, dissimilarity="precomputed", random_state=1)
    # get coordinates of clusters in new low-dimensional space
    plot_positions = mds.fit_transform(cosine_distance)
    x_pos, y_pos = plot_positions[:, 0], plot_positions[:, 1]
```

```python
# build cluster plotting data
cluster_color_map = {}
cluster_name_map = {}

for cluster_num, cluster_details in cluster_data.items():
    # assign cluster features to unique label
    cluster_color_map[cluster_num] = generate_random_color()
    cluster_name_map[cluster_num] = ', '.join(cluster_details['key_features'][:5]).strip()
# map each unique cluster label with its coordinates and movies
cluster_plot_frame = pd.DataFrame({'x': x_pos,
        'y': y_pos,
        'label': movie_data['Cluster'].values.tolist(),
        'title': movie_data['Title'].values.tolist()
        })
grouped_plot_frame = cluster_plot_frame.groupby('label')
```

```
    # set plot figure size and axes
    fig, ax = plt.subplots(figsize=plot_size)
    ax.margins(0.05)
    # plot each cluster using co-ordinates and movie titles
    for cluster_num, cluster_frame in grouped_plot_frame:
        marker = markers[cluster_num] if cluster_num < len(markers) \
                 else np.random.choice(markers, size=1)[0]
        ax.plot(cluster_frame['x'], cluster_frame['y'],
                marker=marker, linestyle='', ms=12,
                label=cluster_name_map[cluster_num],
                color=cluster_color_map[cluster_num], mec='none')
        ax.set_aspect('auto')
        ax.tick_params(axis= 'x', which='both', bottom='off', top='off',
    labelbottom='off')
        ax.tick_params(axis= 'y', which='both', left='off', top='off',
    labelleft='off')
```

```
fontP = FontProperties()
fontP.set_size('small')
ax.legend(loc='upper center', bbox_to_anchor=(0.5, -0.01), fancybox=
 True, shadow=True, ncol=5, numpoints=1, prop=fontP)
#add labels as the film titles
for index in range(len(cluster_plot_frame)):
    ax.text(cluster_plot_frame.ix[index]['x'],
            cluster_plot_frame.ix[index]['y'],
            cluster_plot_frame.ix[index]['title'], size=8)
# show the plot
plt.show()
```

```
cluster_data = get_cluster_data(clustering_obj=km_obj,
        movie_data=movie_data,
        feature_names=feature_names,
        num_clusters=num_clusters,
        topn_features=5)

print_cluster_data(cluster_data)

plot_clusters(num_clusters=num_clusters,
        feature_matrix=feature_matrix,
        cluster_data=cluster_data,
        movie_data=movie_data,
        plot_size=(16,8))
```

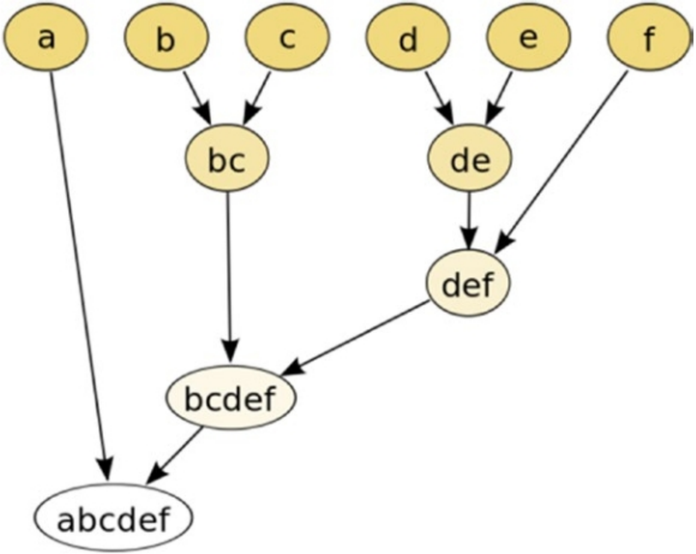
```
#Cluster 0 details:
#--------------------
#Key features: ['michael', 'tramp', 'forest', 'eliot', 'corleon']
#Movies in this cluster:
#The Godfather, The Godfather: Part II, Forrest Gump, E.T. the Extra-Terrestrial, City Lights, Tootsie
#========================================
```

```

#Cluster 1 details:
#————————————————————————
#Key features: ['joe', 'tell', 'teri', 'jeri', 'get']
#Movies in this cluster:
#The Shawshank Redemption, Schindler's List, Casablanca, Citizen Kane,
    The Wizard of Oz, Psycho, Sunset Blvd., #Vertigo, On the Waterfront
    , Star Wars, 2001: A Space Odyssey, Chinatown, Singin' in the Rain,
     Some Like It Hot, #12 Angry Men, Amadeus, Gandhi, The Lord of the
    Rings: The Return of the King, Unforgiven, Raiders of the Lost #Ark
    , Rocky, To Kill a Mockingbird, The Best Years of Our Lives, My
    Fair Lady, Ben-Hur, Doctor Zhivago, Butch #Cassidy and the Sundance
     Kid, The Treasure of the Sierra Madre, The Apartment, The Pianist,
     Goodfellas, The #Exorcist, The French Connection, It Happened One
    Night, Midnight Cowboy, Annie Hall, Good Will Hunting, Terms of #
    Endearment, Fargo, The Grapes of Wrath, The Green Mile, Close
    Encounters of the Third Kind, Network, Nashville, #The Graduate,
    American Graffiti, The Maltese Falcon, Taxi Driver, Wuthering
    Heights, Rebel Without a Cause, Rear #Window, North by Northwest,
```


A Clockwork Orange
The Maltese Falcon
The Exorcist
One Flew Over the Cuckoo's Nest
Amadeus
Giant
Patton
Casablanca
City Lights
Terms of Endearment
Schindler's List
Annie Hall
Yankee Doodle Dandy
The Philadelphia Story
The Deer Hunter
Vertigo
Rocky
The Green Mile
Rear Window
Citizen Kane
To Kill a Mockingbird
Unforgiven
Wuthering Heights
Sunset Blvd.
Double Indemnity
The Third Man
Saving Private Ryan
Nashville
American Graffiti
An American in Paris
The Apartment
The Graduate
E.T. the Extra-Terrestrial
On the Waterfront
The Godfather
The Grapes of Wrath
Fargo
Midnight Cowboy
My Fair Lady
Out of Africa
Raging Bull
Gandhi
The Best Years of Our Lives
The Godfather: Part II
A Streetcar Named Desire
From Here to Eternity
Lawrence of Arabia
Platoon
The Pianist
All Quiet on the Western Front
Apocalypse Now
North by Northwest
Shane
A Place in the Sun
Good Will Hunting
Dances with Wolves
It's a Wonderful Life
Gone with the Wind
Forrest Gump
Braveheart
Butch Cassidy and the Sundance Kid
2001: A Space Odyssey
It Happened One Night
The Shawshank Redemption
Star Wars
Rebel Without a Cause
Close Encounters of the Third Kind
The Treasure of the Sierra Madre
Pulp Fiction
Psycho
Doctor Zhivago
Rain Man
Stagecoach
Tootsie
Ben-Hur
Taxi Driver
The Bridge on the River Kwai
Goodfellas
Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb
West Side Story
Titanic
The Wizard of Oz
Chinatown
Jaws
The Lord of the Rings: The Return of the King
The French Connection
Raiders of the Lost Ark
The African Queen
Network
The Good, the Bad and the Ugly
Gladiator
Mr. Smith Goes to Washington
Mutiny on the Bounty
Singin' in the Rain
The King's Speech
12 Angry Men
The Sound of Music
The Silence of the Lambs
michael, tramp, forest, eliot, corleon
joe, tell, teri, jeri, get
rose, charli, bligh, georg, jake
miller, alex, walac, jet, taylor
traci, mike, georg, nick, alic

# Ward's agglomerative hierarchical clustering

- Cluster thứ bậc (hierarchical clustering) xây dựng cluster lồng vào nhau (nested hierarchy) bởi kết hợp (merging) hoặc chia cách (splitting) cluster kế tiếp (succession)
  - Agglomerative: giải thuật sử dụng tiếp cận bottom-up, bắt đầu từ các data points, kết hợp dần lên cluster lớn
  - Divisive: giải thuật sử dụng tiếp cận top-down, bắt đầu với tất cả data points thuộc về 1 cluster lớn, sau đó chia nhỏ dần đến từng data points


a
b
c
d
e
f
bc
de
def
bcdef
abcdef

# Ta xét 2 công cụ

- distance metric: dùng để đo độ giống (similarity) nhau hoặc khác nhau giữa các data points; Sử dụng Cosine similarity
- linkage criterion: xác định các mà metric sử dụng để kết hợp clusters; Sử dụng phương pháp Ward's

# The Ward's linkage criterion

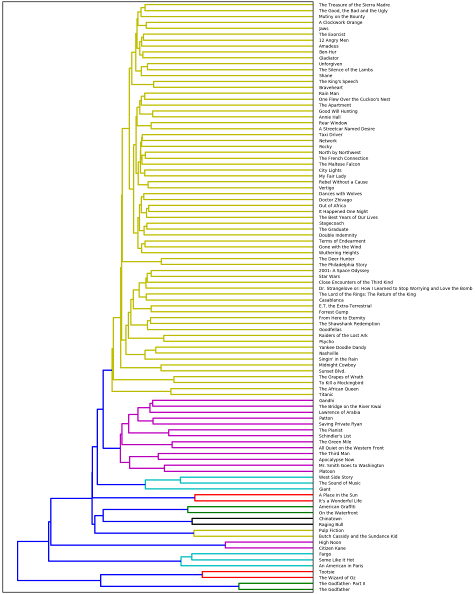
- Tối thiểu hóa tổng của bình phương (sum of squared differences) sự khác nhau trong tất cả các clusters

$$d_{ij} = d\left(\{C_i, C_j\}\right) = \|C_i - C_j\|^2 \tag{2}$$

```python

from scipy.cluster.hierarchy import ward, dendrogram

def ward_hierarchical_clustering(feature_matrix):
    cosine_distance = 1 - cosine_similarity(feature_matrix)
    linkage_matrix = ward(cosine_distance)
    return linkage_matrix

def plot_hierarchical_clusters(linkage_matrix, movie_data, figure_size
    =(8,12)):
    # set size
    fig, ax = plt.subplots(figsize=figure_size)
    movie_titles = movie_data['Title'].values.tolist()
    # plot dendrogram
    ax = dendrogram(linkage_matrix, orientation="left", labels=
    movie_titles)
```

```
plt.tick_params(axis= 'x',
        which='both',
        bottom='off',
        top='off',
        labelbottom='off')
plt.tight_layout()
plt.savefig('ward_hierachical_clusters.png', dpi=200)
plt.show()
```

```python

# build ward's linkage matrix
linkage_matrix = ward_hierarchical_clustering(feature_matrix)
# plot the dendrogram
plot_hierarchical_clusters(linkage_matrix=linkage_matrix,movie_data=
    movie_data,figure_size=(8,10))
```


The Treasure of the Sierra Madre
The Good, the Bad and the Ugly
Mutiny on the Bounty
A Clockwork Orange
Jaws
The Exorcist
12 Angry Men
Amadeus
Ben-Hur
Gladiator
Unforgiven
The Silence of the Lambs
Shane
The King's Speech
Braveheart
Rain Man
One Flew Over the Cuckoo's Nest
The Apartment
Good Will Hunting
Annie Hall
Rear Window
A Streetcar Named Desire
Taxi Driver
Network
Rocky
North by Northwest
The French Connection
The Maltese Falcon
City Lights
My Fair Lady
Rebel Without a Cause
Vertigo
Dances with Wolves
Doctor Zhivago
Out of Africa
It Happened One Night
The Best Years of Our Lives
Stagecoach
The Graduate
Double Indemnity
Terms of Endearment
Gone with the Wind
Wuthering Heights
The Deer Hunter
The Philadelphia Story
2001: A Space Odyssey
Star Wars
Close Encounters of the Third Kind
Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb
The Lord of the Rings: The Return of the King
Casablanca
E.T. the Extra-Terrestrial
Forrest Gump
From Here to Eternity
The Shawshank Redemption
Goodfellas
Raiders of the Lost Ark
Psycho
Yankee Doodle Dandy
Nashville
Singin' in the Rain
Midnight Cowboy
Sunset Blvd.
The Grapes of Wrath
To Kill a Mockingbird
The African Queen
Titanic
Gandhi
The Bridge on the River Kwai
Lawrence of Arabia
Patton
Saving Private Ryan
The Pianist
Schindler's List
The Green Mile
All Quiet on the Western Front
The Third Man
Apocalypse Now
Mr. Smith Goes to Washington
Platoon
West Side Story
The Sound of Music
Giant
A Place in the Sun
It's a Wonderful Life
American Graffiti
On the Waterfront
Chinatown
Raging Bull
Pulp Fiction
Butch Cassidy and the Sundance Kid
High Noon
Citizen Kane
Fargo
Some Like It Hot
An American in Paris
Tootsie
The Wizard of Oz
The Godfather: Part II
The Godfather