

Project movie clustering

Machine learning

วัตถุประสงค์

โดยส่วนมาก เว็บไซต์ดูหนัง จะให้ผู้ใช้เลือกประเภทและหมวดหมู่หนัง เช่น แอคชั่น ดrama แต่เราจะจัดทำเว็บไซต์ กีฬา ความรู้ในด้าน machine learning เพื่อ แนะนำหนัง โดยให้ผู้ใช้กรอกเป็น เรื่องย่อหรือ หรืออะไรก็ได้ที่อยากดู และระบบจะเอาข้อความไปเข้า โมเดล ผ่าน
นอัลกอริทึม k mean และวิวัฒนาจะได้ว่าจะได้หนังอยู่กลุ่มไหน ซึ่งจากข้อมูลที่เราสอน เราฝึก
โมเดลให้เรียนรู้จากหนัง 200 เรื่อง



netflix

✓ Netflix Original Series Top 100 (ranked) - IMDb

ดึงค่า ข้อมูลชื่อ, ผู้เขียน, และโหวต เก็บไว้ใน DataFrame มันทีก DataFrame เป็นไฟล์ Excel และ CSV

```
[ ] # สร้างลิสต์เปล่าสำหรับเก็บข้อมูลชื่อ, ผู้เขียน, และโหวต
name_list = []
title_list = []
votes_list = []
rating_list = []

data = requests.get('https://www.imdb.com/list/ls041667789/')
soup = bs4.BeautifulSoup(data.text)

# วน loop ในทุกๆ element ที่มี class เป็น "lister-item-content"
for c in soup.find_all("div", {"class": "lister-item-content"}):
    # หาชื่อของรายการและเพิ่มลงใน name_list
    shirt = c.find("h3", {"class": "lister-item-header"})
    name_list.append(shirt.find('a').text.strip("\n").replace(" ", ""))

    # หาชื่อของรายการและเพิ่มลงใน title_list
    title = c.find("p", {"class": ""}).text.strip("\n")
    title_words = title.split() # แยกคำในชื่อออกเป็นคำโดยใช้ช่องว่างเป็นตัวแบ่ง
    title_list.append(title_words)

    # หาคะแนนและเพิ่มลงใน rating_list
    rating = c.find("span", {"class": "ipl-rating-star__rating"})
    rating_list.append(rating.text.strip() if rating else 'N/A')

    # หาค่า Votes และเพิ่มลงใน votes_list
    votes = c.find("span", {"name": "nv"})
    votes_list.append(votes['data-value'] if votes else 'N/A')

# สร้าง DataFrame จาก name_list, title_list, และ votes_list
table = pd.DataFrame([name_list, title_list, rating_list, votes_list]).transpose()
table.columns = ['name', 'title', 'rating', 'votes']

# กำหนด index เริ่มต้นที่ 1
table.index += 1
```

movie rate by Roten Tomato

```
# สร้างลิสต์เปล่าสำหรับเก็บข้อมูลชื่อ, ผู้เขียน, และโหวต
name_list = []
title_list = []
votes_list = []
rating_list = []

data = requests.get('https://www.imdb.com/list/ls041667789/')
soup = bs4.BeautifulSoup(data.text)

# วน loop ในทุกๆ element ที่มี class เป็น "lister-item-content"
for c in soup.find_all("div", {"class": "lister-item-content"}):
    # หาชื่อของรายการและเพิ่มลงใน name_list
    shirt = c.find("h3", {"class": "lister-item-header"})
    name_list.append(shirt.find('a').text.strip("\n").replace(" ", ""))

    # หาชื่อของรายการและเพิ่มลงใน title_list
    title = c.find("p", {"class": ""}).text.strip("\n")
    title_words = title.split() # แยกคำในชื่อออกเป็นคำโดยใช้ช่องว่างเป็นตัวแบ่ง
    title_list.append(title_words)

    # หาคะแนนและเพิ่มลงใน rating_list
    rating = c.find("span", {"class": "ipl-rating-star__rating"})
    rating_list.append(rating.text.strip() if rating else 'N/A')

    # หาค่า Votes และเพิ่มลงใน votes_list
    votes = c.find("span", {"name": "nv"})
    votes_list.append(votes['data-value'] if votes else 'N/A')

# สร้าง DataFrame จาก name_list, title_list, และ votes_list
tableTomato = pd.DataFrame([name_list, title_list, rating_list, votes_list]).transpose()
tableTomato.columns = ['name', 'title', 'rating', 'votes']

# กำหนด index เริ่มต้นที่ 1
tableTomato.index += 1

# บันทึก DataFrame เป็นไฟล์ Excel และ CSV
tableTomato.to_excel('TOP100Tomato.xlsx', engine='openpyxl') # บันทึกเป็น Excel
tableTomato.to_csv('TOP100Tomato.csv', index=False) # บันทึกเป็น CSV โดยไม่รวม index
```

join table

```
▶ import pandas as pd

# รวม DataFrame โดยเข้ามูลตามแท่ง (เพิ่มข้อมูลใหม่เป็น列ใน DataFrame ตัวแรก) โดยเป็นการ รวมโดยไม่สนใจ index เดิม
table = pd.concat([table, tableTomato], ignore_index=True)

# แสดงผลลัพธ์
table
```

| | | | | |
|-----|--------------------------|---|-----|---------|
| 0 | OurPlanet | [Explores, and, unravels, the, mystery, of, ho... | 9.3 | 50563 |
| 1 | TheLastDance | [Led, by, Michael, Jordan,, the, 1990s, Chicag... | 9.1 | 150847 |
| 2 | WhenTheySeeUs | [Five, teens, from, Harlem, become, trapped, i... | 8.8 | 137207 |
| 3 | BlackMirror | [Featuring, stand-alone, dramas, --, sharp,, s... | 8.7 | 636724 |
| 4 | Narcos | [A, chronicled, look, at, the, criminal, explo... | 8.8 | 468156 |
| ... | ... | ... | ... | ... |
| 195 | Up | [78-year-old, Carl, Fredricksen, travels, to, ... | 8.3 | 1120585 |
| 196 | Vertigo | [A, former, San, Francisco, police, detective,... | 8.3 | 425579 |
| 197 | WarforthePlanetoftheApes | [After, the, apes, suffer, unimaginable, losse... | 7.4 | 280726 |
| 198 | WonderWoman | [When, a, pilot, crashes, and, tells, of, conf... | 7.3 | 698124 |
| 199 | Zootopia | [In, a, city, of, anthropomorphic, animals,, a... | 8 | 543937 |

200 rows × 4 columns

cut stop word and specifit word

ตัดคำเฉพาะ

```
[ ] import pandas as pd
import nltk
import re
from nltk.tokenize import word_tokenize
from collections import Counter
nltk.download('averaged_perceptron_tagger')
nltk.download('punkt')

# นับความถี่ของคำทั้งหมดใน DataFrame
all_words = [word for sublist in table['title'] for word in sublist]
word_freq = Counter(all_words)

# เลือกคำที่เป็นคำเฉพาะ
proper_nouns = [word for word, tag in nltk.pos_tag(all_words) if tag.startswith('NNP') or tag.startswith('NNPS')]

# สร้าง pattern จากคำที่ต้องการลบ
pattern = '|'.join(r'\b{}\b'.format(re.escape(word)) for word in proper_nouns)

# ลบคำที่เป็นคำเฉพาะออกจาก column 'title' ใน DataFrame
table['title'] = table['title'].apply(lambda x: [word for word in x if word not in pattern])
```

```
import pandas as pd
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords

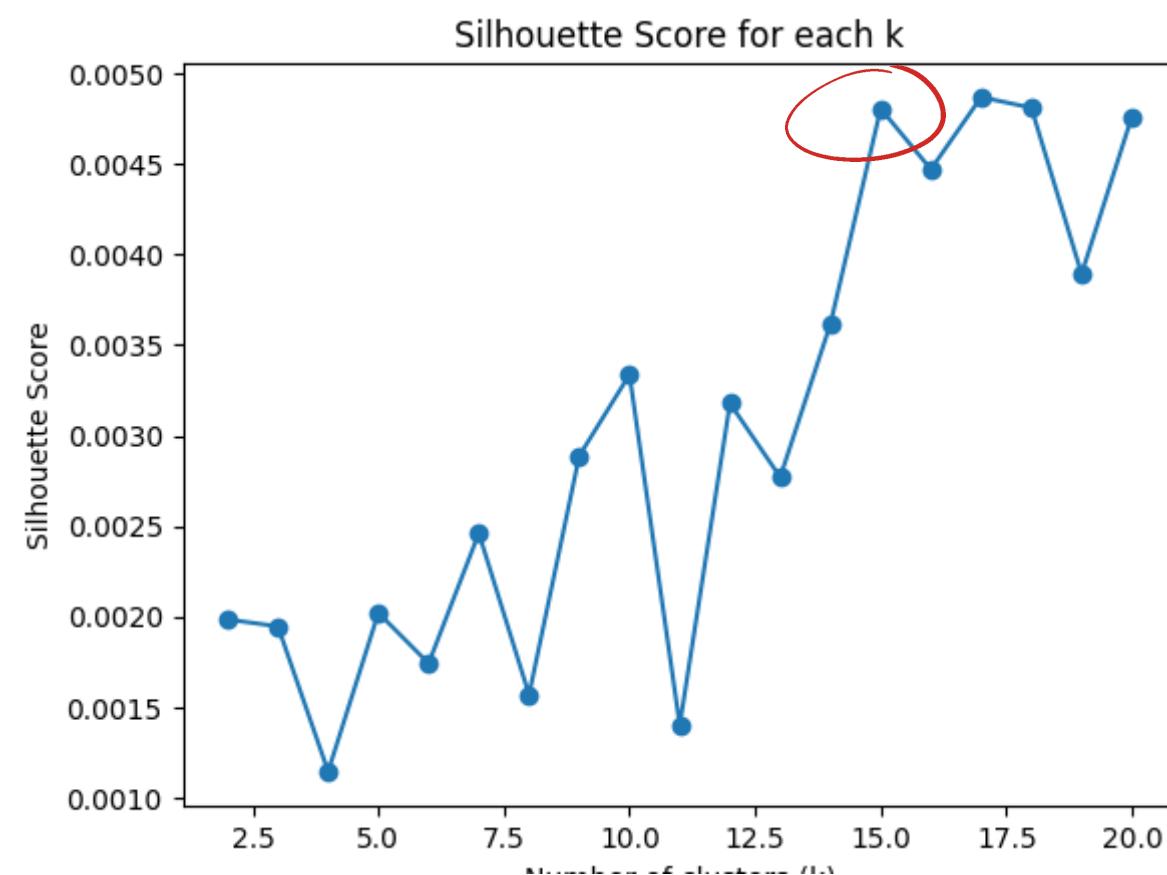
nltk.download('stopwords')

# นำเข้าคำที่ไม่จำเป็น
stop_words = set(stopwords.words('english'))

# ฟังก์ชันสำหรับลบ stop words
def remove_stopwords(title):
    if isinstance(title, list): # ตรวจสอบว่าข้อมูลเป็นลิสต์ของสตริงหรือไม่
        tokens = title
    else:
        tokens = word_tokenize(title)
    filtered_tokens = [word for word in tokens if word.lower() not in stop_words]
    return " ".join(filtered_tokens) # เชื่อมต่อคำที่กรองด้วยช่องว่าง

# ลบ stop words ออกจากคอลัมน์ 'title' ใน DataFrame
table['title'] = table['title'].apply(remove_stopwords)
```

ใช้ KMeans แบ่งกลุ่มข้อมูลโดยใช้จำนวนกลุ่มต่างๆ (k)
ตั้งแต่ 2 ถึง 20 เพื่อหาค่า k ที่ดีที่สุด



KMeans
KMeans(max_iter=100, n_clusters=16, n_init=1)

▼ การทำนายจากข้อความ unseen โดยทำนายว่าตรงกับ Cluster ใด

```
▶ # ลบตัวแปร unseen_text ถ้ามีอยู่เพื่อให้แน่ใจว่าไม่มีค่าเก่าเหลืออยู่  
del unseen_text  
  
# กำหนดข้อความที่ไม่เคยเห็นมาก่อน  
unseen_text = "The history of the United States from the 1950s to the '70s unfolds from the perspective of an Alabama man with an IQ of 75, who yearns to be reunited with his childhood sweetheart."  
# แปลงข้อความที่ไม่เคยเห็นมาก่อนให้อยู่ในรูปแบบเวกเตอร์เมื่ອนกับข้อมูลที่ใช้สร้างโมเดล  
unseen_text_vectorized = vectorizer.transform([unseen_text])  
  
# ทำนาย cluster สำหรับข้อความที่ไม่เคยเห็นมาก่อน  
predicted_cluster = model.predict(unseen_text_vectorized)  
  
# พิมพ์ผลลัพธ์ที่ทำนายได้สำหรับ cluster ของข้อความที่ไม่เคยเห็นมาก่อน  
print("Predicted cluster for the unseen text:", predicted_cluster[0])
```

Predicted cluster for the unseen text: 0



แสดงหนังที่มีค่า cluster เหมือนกับค่าที่ท่านยาได้

```
[ ] # ใช้คำสั่งดูไปมีค่าพานั้นที่มี cluster เมื่อพานั้นที่ท่านยาได้จากข้อมูลที่ไม่เคยเห็นมาก่อน  
findMovie = table[table['cluster'] == predicted_cluster[0]]  
  
# พิมพ์ DataFrame ของหนังที่มี cluster เมื่อพานั้นที่ท่านยาได้  
findMovie
```

| | name | title | rating | votes | cluster |
|-----|---|---|--------|--------|---------|
| 9 | TheCrown | political rivalries romances reign events shap... | 8.6 | 255502 | 0 |
| 12 | Formula1DrivetoSurvive | Docuseries following across multiple seasons. | 8.5 | 50512 | 0 |
| 21 | Love,Death&Robots | collection animated short stories span various... | 8.4 | 195588 | 0 |
| 26 | TheLastKingdom | defends kingdom invaders, raised claim ancestr... | 8.5 | 164762 | 0 |
| 28 | Unbelievable | teenager charged lying raped, two detectives f... | 8.3 | 119004 | 0 |
| 30 | TheMechanism | fiction series loosely inspired investigation ... | 8 | 36668 | 0 |
| 31 | Kingdom | strange rumors ill grip kingdom, crown prince ... | 8.3 | 54068 | 0 |
| 39 | MasterofNone | personal professional life Dev, 30-year-old actor | 8.2 | 76151 | 0 |
| 55 | Lucifer | decided he's enough dutiful servant decides sp... | 8.1 | 354319 | 0 |
| 59 | Lovesick | finding STD, must get back touch every girl ev... | 8 | 22528 | 0 |
| 61 | DotaDragon'sBlood | encounters dragon princess mission, becomes em... | 7.7 | 21622 | 0 |
| 69 | DeadtoMe | series powerful friendship blossoms tightly wo... | 7.9 | 99504 | 0 |
| 71 | GLOW | look personal professional lives group women p... | 8 | 50395 | 0 |
| 92 | ConversationswithaKiller.TheTedBundyTapes | look inside mind serial killer Bundy, featurin... | 7.7 | 32822 | 0 |
| 95 | IThinkYouShouldLeavewithTimRobinson | sketch show guests spend segment driving someo... | 8 | 20983 | 0 |
| 99 | TheSerpent | twisting, real-life story Sobhraj, murderer, t... | 7.6 | 49380 | 0 |
| 100 | 12AngryMen | jury murder trial frustrated single member who... | 9 | 860786 | 0 |
| 101 | 12YearsaSlave | antebellum States, Northup, free man upstate a... | 8.1 | 739520 | 0 |
| 103 | AStreetcarNamedDesire | moves sister tormented brutish brother-in-law ... | 7.9 | 114012 | 0 |
| 109 | ArmyofShadows | account underground resistance fighters France. | 8.1 | 25749 | 0 |

หนังที่มีค่าตรงกันกับ Unseen มากที่สุด

```
from sklearn.metrics.pairwise import cosine_similarity  
  
# แปลงข้อความที่ไม่เคยเห็นมาให้อยู่ในรูปแบบของภาษาเดอร์เท่มีอยู่บันทึกว่าที่ใช้สร้างโมเดล  
unseen_text_vectorized = vectorizer.transform([unseen_text])  
  
# ทำการท่านยา cluster สำหรับข้อความที่ไม่เคยเห็นมาก่อน  
predicted_cluster = model.predict(unseen_text_vectorized)  
  
# กรองแค่ว่า cluster เมื่อกัน predicted_cluster  
cluster_data = table[table['cluster'] == predicted_cluster[0]]  
  
# คำนวณ cosine similarity ระหว่างบทเดอร์ของข้อความที่ไม่เคยเห็นมา กับหนังเดอร์ของหัวขอทั้งหมดใน cluster ที่กรองแล้ว  
similarities = cosine_similarity(unseen_text_vectorized, X[cluster_data.index])  
  
# หาผู้ซึ่งของหัวขอที่มีความคล้ายคลึงที่สุดภายใน cluster  
most_similar_index = similarities.argmax()  
  
# รับชื่อและหัวขอที่เก็บไว้จาก DataFrame  
most_similar_title = cluster_data.iloc[most_similar_index]['title']  
most_similar_name = cluster_data.iloc[most_similar_index]['name']  
  
print("Most similar name in the DataFrame:", most_similar_name)  
print("Most similar title in the DataFrame:", most_similar_title)
```

Most similar name in the DataFrame: 12YearsaSlave
Most similar title in the DataFrame: antebellum States, Northup, free man upstate abducted sold slavery.

▼ แสดงDataFrame ที่มีcluster เท่ากับค่าที่ทำนายได้ ที่มีค่า rating มากไปน้อย 10 อันดับ

```
# กรองแค่ว่า cluster เท่ากับค่า cluster ที่ทำนายได้  
cluster_13_data = table[table['cluster'] == predicted_cluster[0]]  
# เรียงลำดับตามค่าลั่งนี้ 'rating' ในลำดับจากมากไปน้อยและแสดงเฉพาะ 10 แถวแรก  
cluster_13_top_10 = cluster_13_data.sort_values(by='rating', ascending=False).head(10)  
  
cluster_13_top_10
```

| | name | title | rating | votes | cluster |
|-----|--------------------------|---|--------|--------|---------|
| 100 | 12AngryMen | jury murder trial frustrated single member who... | 9 | 860786 | 0 |
| 177 | TheGood,theBadandtheUgly | bounty hunting scam joins two men uneasy allia... | 8.8 | 808477 | 0 |
| 9 | TheCrown | political rivalries romances reign events shap... | 8.6 | 255502 | 0 |
| 26 | TheLastKingdom | defends kingdom invaders, raised claim ancestr... | 8.5 | 164762 | 0 |
| 12 | Formula1:DrivetoSurvive | Docuseries following across multiple seasons. | 8.5 | 50512 | 0 |
| 21 | Love,Death&Robots | collection animated short stories span various... | 8.4 | 195588 | 0 |
| 196 | Vertigo | former police detective juggles wrestling pers... | 8.3 | 425579 | 0 |
| 31 | Kingdom | strange rumors ill grip kingdom, crown prince ... | 8.3 | 54068 | 0 |
| 28 | Unbelievable | teenager charged lying raped, two detectives f... | 8.3 | 119004 | 0 |
| 188 | TheWagesofFear | decrepit village, four men hired transport urg... | 8.2 | 66217 | 0 |

cool!



Thank you
for listening!