In [1]:
```
import json
import numpy as np
import pandas as pd

data = pd.read_json('hw2.json')
data.head()
```

Out[1]:

	doc_id	cname	ename	label	released_date	intro	
0	0	一世狂野	Blow	[劇情, 犯罪, 歷史/傳記]	2001-10-12	喬治戎格一生都在追求所謂的美國夢，也就是享受美好富裕的生活，但是他卻不願像他父親那樣一輩子都...	[https://movies.yahoo.com.tw/movie
1	1	玩命關頭	The Fast and the Furious	[動作, 劇情, 犯罪, 懸疑/驚悚]	2001-10-13	唐米尼杜洛托是洛城街頭賽車界的老大哥，他身邊有一群忠心耿耿的手下，他白天忙著組裝高性能跑車，...	[https://movies.yahoo.com.tw/movie
2	2	戰雲密佈	Storm Catcher	[動作, 犯罪, 懸疑/驚悚, 戰爭]	2001-10-13	美國空軍最高機密的隱形戰機驚傳失蹤！祕密訓練的飛行軍官傑克，被誣陷勾結恐怖組織，參與竊取	[https://movies.yahoo.com.tw/movie

						戰機...	
3	3	騎士風雲錄	A Knight's Tale	[動作, 冒險, 喜劇]	2001-10-19	14世紀中古時期的社會階級分明，出身卑微的平民不論如何努力和奮鬥，都無法跨越階級制度而翻身致...	[https://movies.yahoo.com.tw/movie
4	4	金法尤物	Legally Blonde	[喜劇]	2001-10-19	在【歡樂谷】、【危險性遊戲】挑大樑的瑞絲薇斯朋飾演【金法尤物】中飽受眾人歧視的金髮美女，因為...	[https://movies.yahoo.com.tw/movie

In [2]:
```
for i, d in data.iterrows():
    if (d["label"]):
        data.at[i, "label"] = d["label"][0]
    else:
        data = data.drop([i])
data.head()
```

Out[2]:

	doc_id	cname	ename	label	released_date	intro	
0	0	一世狂野	Blow	劇情	2001-10-12	喬治戎格一生都在追求所謂的美國夢，也就是享受美好富裕的生活，但是他卻不願	[https://movies.yahoo.com.tw/movie

						像他父親那樣一輩子都...	
1	1	玩命關頭	The Fast and the Furious	動作	2001-10-13	唐米尼杜洛托是洛城街頭賽車界的老大哥，他身邊有一群忠心耿耿的手下，他白天忙著組裝高性能跑車，...	[https://movies.yahoo.com.tw/movie
2	2	戰雲密佈	Storm Catcher	動作	2001-10-13	美國空軍最高機密的隱形戰機驚傳失蹤！祕密訓練的飛行軍官傑克，被誣陷勾結恐怖組織，參與竊取戰機...	[https://movies.yahoo.com.tw/movie
3	3	騎士風雲錄	A Knight's Tale	動作	2001-10-19	14世紀中古時期的社會階級分明，出身卑微的平民不論如何努力和奮鬥，都無法跨越階級制度而翻身致...	[https://movies.yahoo.com.tw/movie
						在【歡樂谷】、【危險	

4	4	金法尤物	Legally Blonde	喜劇	2001-10-19	性遊戲】挑大樑的瑞絲薇斯朋飾演【金法尤物】中飽受眾人歧視的金髮美女，因為...	[https://movies.yahoo.com.tw/movie

In [3]:
```
label = data["label"]
label.head()
```

Out[3]:
```
0    劇情
1    動作
2    動作
3    動作
4    喜劇
Name: label, dtype: object
```

from ckiptagger import data_utils

data_utils.download_data_gdown("./")

In [4]:
```
from ckiptagger import WS, POS, NER

ws = WS("./data")
pos = POS("./data")
ner = NER("./data")
```

```
/Users/hsiu/opt/anaconda3/envs/tensorflow/lib/python3.9/site-packages/ckiptagger/model_ws.py:106: UserWarning: `tf.nn.rnn_cell.LSTMCell` is deprecated and will be removed in a future version. This class is equivalent as `tf.keras.layers.LSTMCell`, and will be replaced by that in Tensorflow 2.0.
  cell = tf.compat.v1.nn.rnn_cell.LSTMCell(hidden_d, name=name)
2023-04-11 17:22:47.402769: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:357] MLIR V1 optimization pass is not enabled
2023-04-11 17:22:47.468443: W tensorflow/tsl/platform/profile_utils/cpu_utils.cc:128] Failed to get CPU frequency: 0 Hz
/Users/hsiu/opt/anaconda3/envs/tensorflow/lib/python3.9/site-packages/ckiptagger/model_pos.py:56: UserWarning: `tf.nn.rnn_cell.LSTMCell` is deprecated and will be removed in a future version. This class is equivalent as `tf.keras.layers.LSTMCell`, and will be replaced by that in Tensorflow 2.0.
  cell = tf.compat.v1.nn.rnn_cell.LSTMCell(hidden_d, name=name)
/Users/hsiu/opt/anaconda3/envs/tensorflow/lib/python3.9/site-packages/ckiptagger/model_ner.py:57: UserWarning: `tf.nn.rnn_cell.LSTMCell` is deprecated and will be removed in a future version. This class is equivalent as `tf.keras.layers.LSTMCell`, and will be replaced by that in Tensorflow 2.0.
  cell = tf.compat.v1.nn.rnn_cell.LSTMCell(hidden_d, name=name)
```

In [5]:
```
import collections
from collections import *

#https://ithelp.ithome.com.tw/articles/10295882
def clean(sentence_ws, sentence_pos):
    short_with_pos = []
    short_sentence = []
    stop_pos = set(['Nep', 'Nh', 'Nb']) # 這 3 種詞性不保留
    for word_ws, word_pos in zip(sentence_ws, sentence_pos):
        # 只留名詞和動詞
        is_N_or_V = word_pos.startswith("V") or word_pos.startswith("N")
        # 去掉名詞裡的某些詞性
        is_not_stop_pos = word_pos not in stop_pos
        # 只剩一個字的詞也不留
        is_not_one_charactor = not (len(word_ws) == 1)
        # 組成串列
        if is_N_or_V and is_not_stop_pos and is_not_one_charactor:
            short_sentence.append(word_ws)

    return (short_sentence)



tokenized = []

rec = collections.defaultdict(int)
for _, d in data.iterrows():
    ws_results = ws([d["intro"]])
    pos_results = pos(ws_results)
    short = clean(ws_results[0], pos_results[0])

#     tokenized = np.concatenate((tokenized, short), axis = 0)
    tokenized.append(short)

    for w in set(short):
        rec[w] += 1
print(tokenized[:2])
```

```
[['一生', '追求', '所謂', '美國', '享受', '美好', '富裕', '生活', '父親', '那樣
', '輩子', '出賣', '勞力', '建築', '工人', '搬到', '陽光明媚', '加州', '販賣',
'大麻', '賺錢', '販毒', '享受', '自由自在', '生活', '野心', '勢力', '坐大', '此
時', '入獄', '認識', '能言善道', '自稱', '熟識', '哥倫比亞', '販毒', '集團', '牢
友', '出獄', '當時', '勢力', '毒梟', '介紹給', '認識', '計畫', '古柯鹼', '大量',
'引進', '美國', '迪斯可', '舞廳', '希望', '引領', '吸毒', '狂歡', '風潮', '毒品'
, '供應商', '之外', '介紹', '美艷', '狂野', '女人', '瘋狂', '相愛', '之後', '生
下', '可愛', '女兒', '一生', '最愛', '販毒', '房子', '專門', '存放', '賺進來',
'鈔票', '斗金', '提心吊膽', '生活', '開始', '省思', '繼續', '揮霍', '富裕', '生
活', '女兒', '轉性', '投資', '正當', '事業', '這時', '聯邦', '調查局', '探員',
'開始', '盯上', '毒源禍首'], ['洛城', '街頭', '賽車界', '老大哥', '身邊', '忠心
耿耿', '手下', '白天', '忙著', '組裝', '跑車', '晚上', '愛車', '一萬', '美元',
'賭注', '軋車', '渴望', '接受', '極速', '挑戰', '駕駛', '技術', '信心', '旁觀者'
, '菜鳥', '超炫', '跑車', '老大', '一較高下', '希望', '得到', '青睞', '比賽', '
結束', '一塌塗地', '之後', '警方', '接獲', '風聲', '前來', '取締', '心狠手辣', '
幫派', '份子', '老大', '納入', '老大', '權力', '核心', '老大', '妹妹', '產生',
'好感', '知道', '臥底', '警探', '滲入', '賽車圈', '目的', '調查', '卡車', '搶案'
, '嫌犯', '跑車', '蒙面人', '警方', '聯邦', '調查局', '希望', '逮到', '搶匪', '
卡車', '司機', '採取', '激烈', '手段', '這些', '搶匪', '進行', '報復', '行動',
'嫌疑', '老大', '老大', '形成', '水火不', '相容', '情勢', '老大', '兄妹', '關係'
, '老大', '結為', '好友', '忍不住', '產生', '好感', '同時', '承受', '來自', '警
方', '壓力', '查出', '搶匪', '天人交戰', '法律', '友情', '之間', '做出', '困難',
'決定']]
```

In [6]:

```
from collections import Counter
import math

def calculate_tfidf(doc):
    count = Counter(doc)
    temp = {}
    for w, n in count.items():
        tf = n / len(doc)
        idf = len(tokenized) / rec[w]
        temp[w] = tf * math.log(idf, 10)

    return temp



tfidf = pd.DataFrame([calculate_tfidf(doc) for doc in tokenized])

tfidf = tfidf.fillna(0)
tfidf.head()
```

Out[6]:

	一生	追求	所謂	美國	享受	美好	富裕	生活	
0	0.027098	0.014038	0.019367	0.016947	0.031342	0.0139	0.044995	0.022676	0.00
1	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000	0.000000	0.000000	0.00
2	0.000000	0.000000	0.000000	0.032564	0.000000	0.0000	0.000000	0.000000	0.00
3	0.000000	0.015285	0.000000	0.000000	0.000000	0.0000	0.000000	0.000000	0.01
4	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000	0.000000	0.000000	0.00

5 rows × 83467 columns

In [7]:
```
xtrain, xtest = tfidf[:-500], tfidf[-500: ]
ytrain, ytest = label[:-500], label[-500:]
```

In [11]:
```
xtrain.shape
```

Out[11]:
```
(11518, 83467)
```

In [8]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC

knn = KNeighborsClassifier()
knn.fit(xtrain, ytrain)
pred1 = knn.predict(xtest)
```

In [9]:
```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(n_estimators=100, criterion = 'gini')
rf.fit(xtrain, ytrain)
pred2 = rf.predict(xtest)
```

In [10]:
```
c1 = c2 = 0

for i, v in enumerate(ytest):
    if v == pred1[i]:
        c1 += 1
    if v == pred2[i]:
        c2 += 1
print(f'accuracy of knn: {c1 / 500}')
print(f'accuracy of svm: {c2 / 500}')
```

```
accuracy of knn: 0.268
accuracy of svm: 0.49
```

In [ ]:
```

```