In [ ]:

```
# 計算詞頻
word_count = Counter(word_list)

# 取出前100個高頻詞
top100_word_freq = word_count.most_common(100)
import jieba.analyse

# 計算TF-IDF權重
tfidf = jieba.analyse.extract_tags(text, topK=100, withWeight=True)
```

In [ ]:

```
word_list =
```

In [2]:

```
import json
from sklearn.feature_extraction.text import TfidfVectorizer

# 读取 movies.json 文件
with open('movies_12.json', 'r', encoding='utf-8') as f:
    movies = json.load(f)

# 获取所有电影的文本内容，组成一个列表
movie_texts = [movie['intro'] for movie in movies]

# 使用 TfidfVectorizer 进行分词和计算 TF-IDF
vectorizer = TfidfVectorizer()
tfidf = vectorizer.fit_transform(movie_texts)
```

In [9]:

```
import json
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

with open("movies.json", "r", encoding="utf-8") as f:
    movies = json.load(f)


texts = [movie["intro"] for movie in movies]
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(texts)


train_data = X[:12396]
train_labels = [movie["moive_label"][0] for movie in movies[:12396]]
test_data = X[12396:]
test_labels = [movie["moive_label"][0] for movie in movies[12396:]]


knn = KNeighborsClassifier(n_neighbors=5, metric="cosine")
knn.fit(train_data, train_labels)

# 预测测试数据集的分类结果
predictions = knn.predict(test_data)

#
accuracy = accuracy_score(test_labels, predictions)
print("KNN: {:.2f}%".format(accuracy * 100))
```

```
KNN: 39.18%

C:\Users\user\anaconda3\lib\site-packages\sklearn\neighbors\_classificati
on.py:228: FutureWarning: Unlike other reduction functions (e.g. `skew`,
`kurtosis`), the default behavior of `mode` typically preserves the axis
it acts along. In SciPy 1.11.0, this behavior will change: the default va
lue of `keepdims` will become False, the `axis` over which the statistic
is taken will be eliminated, and the value None will no longer be accepte
d. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

In [7]:

```
#svm
import json
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score


with open("movies.json", "r", encoding="utf-8") as f:
    data = json.load(f)
texts = [d["intro"] for d in data]
labels = [d["moive_label"] for d in data]


vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(texts)


X_train, X_test, y_train, y_test = train_test_split(X, labels, test_size=0.1, random_sta


svm = SVC(kernel='linear', C=1.0)
svm.fit(X_train, y_train)
y_pred = svm.predict(X_test)


accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: {:.2f}%".format(accuracy * 100))
```

```
Accuracy: 56.94%
```

In [8]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
import json


with open('movies_12.json', 'r', encoding='utf-8') as f:
    movies = json.load(f)

texts = [movie['intro'] for movie in movies]
labels = [movie['moive_label'][0] for movie in movies]

vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(texts)

train_X, train_y = X[:12396], labels[:12396]
test_X, test_y = X[12396:], labels[12396:]

# Train
clf = SVC(kernel='linear')
clf.fit(train_X, train_y)

# Predict
pred_y = clf.predict(test_X)

#accuracy
accuracy = accuracy_score(test_y, pred_y)
print(f'SVM classification accuracy: {accuracy:.4f}')
```

```
SVM classification accuracy: 0.6168
```

In [ ]: