學號:B0928031

姓名:鄭茹云

學系:人工智慧三甲

連結:

https://colab.research.google.com/drive/1Txpr9Q4dgG8u4bKC9a7NGZ2OKRpYPneg?usp=share_link

按兩下 (或按 Enter 鍵) 即可編輯

```
import requests
from bs4 import BeautifulSoup
import networkx as nx
import json

base_url = "https://movies.yahoo.com.tw/movieinfo_main/"
start_page = 1
end_page = 7600
i = 0
G = nx.DiGraph()

for page in range(start_page, end_page+1):
    url = base_url + str(page)
    response = requests.get(url)
    soup = BeautifulSoup(response.content, "html.parser")
    if not soup.find("div", class_="movie_intro_info"):
        continue
    # add the URL
    G.add_node(url)

    doc_id = page
    cname = soup.find("div", class_="movie_intro_info").find("h1").text.strip()
    ename = soup.find("div", class_="movie_intro_info_r").find("h3").text.strip()
    label = soup.find("div", class_="level_name").text.strip()
    intro = soup.find("span", id="story").text.strip().replace('\n\n', '')
    released_date = soup.find("div", class_="movie_intro_info_r").find_all("span")[0].text.strip().repla

    links = []
    for link in soup.find_all("a"):
        href = link.get("href")
        text = link.text.strip()
        if href and text:
            links.append(href)
            G.add_edge(url, href)

    # calculate
    pagerank = nx.pagerank(G)

    # create a dictionary
    movie_info = {
        "doc_id": doc_id,
        "cname": cname,
        "ename": ename,
        "pagerank": pagerank[url]  # use the URL as the key to get the PageRank
```

```
            "pagerank": pagerank[url],  # use the URL as the key to get the PageRank
            "label": label,
            "intro": intro,
            "released_date": released_date,
            "links": links  # store only the hrefs of the links
    }
    i += 1
    print(i)

    #JSON file
    with open("hw2.json", "a", encoding="utf-8") as f:
            json.dump(movie_info, f, ensure_ascii=False)
            f.write("\n")  # add a new line at the end of each JSON object to separate them
```

串流輸出內容已截斷至最後 5000 行。

```
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
```

```
955
956
957
958
959
960
961
```

```
import json
import jieba
from collections import defaultdict

# load data
with open('hw2.json', 'r', encoding='utf-8') as f:
    movies = [json.loads(line) for line in f]


def tokenize(text):
    return list(jieba.cut(text))

inverted_index = defaultdict(list)
for movie in movies:
    tokens = tokenize(movie['intro'] + movie['cname'] + movie['ename'] + movie['label'])
    for token in tokens:
        inverted_index[token].append(movie['doc_id'])

# save to a JSON file
with open('invert.json', 'w', encoding='utf-8') as f:
    json.dump(inverted_index, f, ensure_ascii=False)
    f.write("\n") # add a new line at the end of each JSON object to separate them
```

```
import json

movies = []
with open("hw2.json", "r", encoding="utf-8") as f:
    for line in f:
        movie_info = json.loads(line)
        movies.append(movie_info)

# sort PageRank scores
movies_sorted = sorted(movies, key=lambda x: x["pagerank"], reverse=True)

for i, movie in enumerate(movies_sorted[:]):
    print(f"{i+1}. {movie['cname']} ({movie['released_date']}) - PageRank score: {movie['pagerank']:.
```

```
### 輸出搜尋結果呈現
import json
import re

def search_movies(query):
    with open("hw2.json", "r", encoding="utf-8") as f:
        movies = [json.loads(line) for line in f]

    pageranks = {movie["doc_id"]: movie["pagerank"] for movie in movies}

    matches = [movie for movie in movies if any([query.lower() in str(value).lower() for key,

    sorted_matches = sorted(matches, key=lambda x: pageranks.get(x["doc_id"], 0), reverse=True)
```

```
        relevant_docs = set([movie['doc_id'] for movie in movies if query.lower() in str(movie).lower
        retrieved_docs = set([movie['doc_id'] for movie in matches])
        true_positives = relevant_docs.intersection(retrieved_docs)
        precision = len(true_positives) / len(matches) if len(matches) > 0 else 0
        recall = len(true_positives) / len(relevant_docs) if len(relevant_docs) > 0 else 0

        # print the search results and evaluation metrics
        print("您的搜尋結果 (Sorting by PageRank Value)：")
        print(f"共 {len(sorted_matches)} 筆，符合'{query}' - - - 共 indexing {len(movies)} 筆電影資料
        for movie in sorted_matches:
            intro = movie.get("intro", "")
            print(f"{movie['doc_id']} ({pageranks.get(movie['doc_id'], 0)}): {movie['cname']} ({movie['

        print(f"\nPrecision: {precision:.0%}")
        print(f"Recall: {recall:.0%}")


search_movies("黑暗")
```