電影迷尋蹤

凌銘陽　張佩瑜　李亞珊　吳依玲

Outline

01 **動機**

專案動機

02 **目標**

專題目標、概念

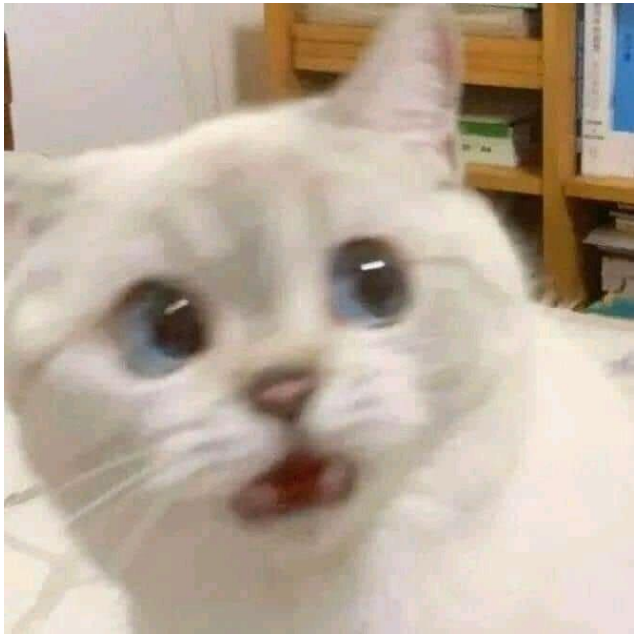
03 **程式撰寫**

程式撰寫

04 **成果及展望**

成果、挑戰、未來展望

這麼多電影
到底要看哪部



這麼多電影院
到底要去哪

我們
都為你準備好了♡

專題目標

專屬你的最佳菜

依使用者偏好的電影類型及分類
爬取各大電影評論網進行資料彙整及篩選
推薦使用者院線電影

專屬你的最佳點

依照使用者的位置
透過全台電影院位置資料集擷取資料
回傳距離使用者最近的電影院

概念

利用爬蟲抓取
Yahoo和IMDb電影資訊

利用爬蟲與Google Maps API建立
電影、電影院資料庫

導入電影DB隨機推薦電影

導入Google Maps API、
電影院DB計算電影院距離

以flask架構建置LINE Bot

程式撰寫

#電影院推薦

戲院資料庫建立

1. 從政府資料平台爬取全台電影院的事業名稱與地址
2. 藉由Google MAPs API 獲取各家電影院的經緯度與在Google MAPs上的網址
3. 以政府為主要資料庫新增新設的電影院以及刪除已不存在的電影院

```
def dataset_webcraw(url):
    response = requests.get(url)
    res = BeautifulSoup(response.text) # 預設html.parser，不打也相同
    latest_data = res.find_all('div', class_ = 'download-item')[-1].a['href']
    return latest_data
url = 'https://data.gov.tw/dataset/22213'
latest_data = dataset_webcraw(url)
```

```
df = pd.read_csv(latest_data)
cinema_data = df[['事業名稱','地址']]
cinema_data.dropna(inplace=True)
# 臺 轉 台
cinema_data['地址'] = cinema_data['地址'].apply(lambda x: '台' + x[1:] if x.startswith('臺') else x)

# 換順序
new_df = cinema_data.reindex(columns = ['事業名稱', '地址'])
```

#電影院推薦

距離最近電影院推薦

1. 導入GoogleMaps API、戲院DB
2. 先透過使用者座標與戲院座標算出純數學的直線距離，選出距離短的前6間戲院
3. 再從六間內，透過上面Google map方法算出交通時間、交通距離,排序最接近的3間

```python
import googlemaps
import sqlite3
import pandas as pd
import dotenv
from Function_list import RoutePlanning

class CinemaFinder:
    def __init__(self,location):

        self.apikey
        self.gmaps = googlemaps.Client(key=
        self.location = location

    # 取得規劃路線距離、估計時間
    def GM(self, destination, mode='driving'):
        response = self.gmaps.distance_matrix(self.location, destination,
        try:
            distance = response['rows'][0]['elements'][0]['distance']['va
            duration = response['rows'][0]['elements'][0]['duration']['te
            return distance, duration
        except:
            return None, None

    # 取得附近"直線距離"前6間電影院
    def MathJudge_dist(self, df):
        df['緯度'] = df['經緯度'].map(lambda x: float(x.split(',')[0])) - t
        df['經度'] = df['經緯度'].map(lambda x: float(x.split(',')[1])) - t
        df['直線距離'] = (df['緯度'] ** 2 + df['經度'] ** 2) ** 0.5
        df = df.drop(['緯度', '經度'], axis=1)
        df = df.sort_values(['直線距離']).reset_index(drop=True)
        return df.head(6)

    # 取得附近"交通距離"前3間電影院 & 規劃路線頁面
    def order_df_dist_google(self, df, mode='driving'):
        def format_RP_with_link(destination):
            url = RoutePlanning(self.location, destination)
            return f'{url}'

    # main
    df = df.copy()
    df['temp'] = df['經緯度'].map(lambda x: self.GM(x, mode))
    df['距離(公尺)'] = df['temp'].apply(lambda x: x[0])
    df['預計時間'] = df['temp'].apply(lambda x: x[1])
    df['路線規劃'] = df['事業名稱'].map(lambda x: format_RP_with_link(
    df['戲院googlemap'] = df.apply(lambda row: place_id_finder(row['

    df.drop('temp', axis=1, inplace=True)
    #依照距離選出前三家電影院
    df = df.sort_values('距離(公尺)').reset_index(drop=True)
    res = df[['事業名稱', '距離(公尺)', '預計時間', '路線規劃', '戲院googl
    return res

# 生成網頁表格
def generate_html_table(self, df):
    html_table = df.to_html(index=False, escape=False)
    with open('CinemaFinder_table.html', 'w') as file:
        file.write(html_table)

# 連線資料庫、SQL搜尋、轉df
conn = sqlite3.connect('C:/U
data_df = pd.read_sql_query("SELECT * FROM TW_cinema_info_latest", conn
conn.close()
# print(apikey)
# # 建立物件
finder = CinemaFinder( location)
filtered_data = finder.MathJudge_dist(data_df)
result = finder.order_df_dist_google(filtered_data)
result=result.to_dict()
```

#隨機推薦

電影資料庫建立

1. 爬取公開資料網站,建立電影資料庫
2. 清理雜訊、新增&減少特徵欄位

```python
def dataset_webcraw(url):
    response = requests.get(url)
    res = BeautifulSoup(response.text) # 預設html.parser，不打也相同
    latest_data = res.find_all('div', class_ = 'download-item')[-1].a['href']
    return latest_data
url = 'https://data.gov.tw/dataset/59820'
response = requests.get(url)
res = BeautifulSoup(response.text)
dataset_lst = res.find_all('div', class_ = 'download-item')
dataset_eachfile = {}
data_df = {}
for i in dataset_lst:
    title = i.text.split('\n')[-1].strip()
    downlaodLink = i.a['href']
    dataset_eachfile[title] = downlaodLink
    data_df[title] = pd.read_csv(downlaodLink)
```

```python
# 處理104~109 欄位未分割問題
for title,m  in data_df.items():
    try:
        m['年度'] = m['年度'] + 1911
        m['中文片名'] = m['中文片名/外文片名'].apply(lambda row : row.split('\n')[0])
        m['原文片名'] = m['中文片名/外文片名'].apply(lambda row : row.split('\n')[1] if len(row.split('\n')) > 1 else 'No Eng
        m['國別'] = m['國別/語別'].apply(lambda row : row.split('\n')[0])
        m['語言'] = m['國別/語別'].apply(lambda row : row.split('\n')[1] if len(row.split('\n')) > 1 else 'Unknown')
        df = m[['年度', '中文片名', '原文片名', '國別', '語言']]
        data_df[title] = df.drop_duplicates()
    except KeyError:
        df = m[['年度', '中文片名', '原文片名', '國別', '語言']]
        data_df[title] = df.drop_duplicates()

# check
for v in data_df.values():
    print(v.isnull().sum().sum(), len(v.columns), sep =',')

# 清理中文片名()內含有2D、3D、imax(大小寫)的部分
# 刪除現場XXXXX劇院現場的資料
# 去重複
pattern = r'\([^()]*?(2D|3D|IMAX|.*版)[^()]*?\)'
for key, df in data_df.items():
    df['中文片名'] = df['中文片名'].apply(lambda x: re.sub(pattern, '', x, flags=re.IGNORECASE))
    df = df[~df['中文片名'].str.contains('劇院現場')].reset_index(drop=True)
    data_df[key] = df.drop_duplicates()
```

#隨機推薦

電影隨機推薦

1. 串聯電影資料庫查詢(1985~2022, 約34000筆資料)
2. 根據使用者輸入來限制搜尋
3. 隨機推3部符合限制的電影
4. 大部分沒中文名稱、分類皆是英文

```python
import sqlite3
import random

class Recommand_System():
    def __init__(self):
        self.conn = sqlite3.connect('Movies.db')
        self.cursor = self.conn.cursor()

    def cleanmsg(self, msg):
        row_msg = msg.strip().split('\n')
        result = {item.split(':')[0]: item.split(':')[1]
        return result
    def random_recommand(self, dict:dict = {}):

        year = dict.get('年度')
        chinese_title = dict.get('中文片名')
        original_title = dict.get('原文片名')
        country = dict.get('國別')
        language = dict.get('語言')
        genre = dict.get('類型')
        query = "SELECT 年度,中文片名,原文片名,國別,語言,類型 FI
        params = []

        # 年度搜索
        if year:
            if '-' in year:
                # 年份範圍搜索
                start_year, end_year = year.split('-')
                query += " AND 年度 BETWEEN ? AND ?"
                params.extend([start_year, end_year])
            elif year.endswith('>'):
                # 某年之後搜索
                year_offset = year[:-1]
                query += " AND 年度 >= ?"
                params.append(year_offset)
            elif year.endswith('<'):
                # 某年之前搜索
                year_offset = year[:-1]
                query += " AND 年度 <= ?"
                params.append(year_offset)
            else:
                # 指定某年
                query += " AND 年度 = ?"
                params.append(year)

        # 可模糊搜尋
        if chinese_title:
            query += " AND 中文片名 LIKE '%' || ? || '%'"
            params.append(chinese_title)

        # 可模糊搜尋
        if original_title:
            query += " AND 原文片名 LIKE '%' || ? || '%'"
            params.append(original_title)

        # 可模糊搜尋
        if country:
            query += " AND 國別 LIKE '%' || ? || '%'"
            params.append(country)

        # 可模糊搜尋
        if language:
            query += " AND 語言 LIKE '%' || ? || '%'"
            params.append(language)

        # 可多重搜尋
        if genre:
            genre_keywords = genre.split()
            for keyword in genre_keywords:
                query += " AND 類型 LIKE '%' || ? || '%'"
                params.append(keyword)

        self.cursor.execute(query, params)
        results = self.cursor.fetchall()
        self.conn.close()

        if not results: return print("無結果")
        # 隨機從限搜下推3個
        # 有重複名稱就從新隨機(英文片名判斷)
        if len(results) > 3:
            while True:
                random_results = random.sample(results, 3)
                check = set([item[2] for item in random_results])
                if len(check) == 3: break
        else:
            random_results = results
        movie=[]
        #回傳列表式格式
        for row in random_results:
            if row[1]=='unknown':
                aa=f'年度: {row[0]}  \n原文片名: {row[2]}   \n國別: {row
            else:
                aa=f'年度: {row[0]}  \n中文片名: {row[1]} \n原文片名: {ro
            movie.append(aa)
        return movie
```

#電影資訊搜尋

Yahoo電影爬蟲

- 爬取指定Yahoo電影，使用者輸入(中英文片名)後返回該片的資訊，包括
  - 中英文片名
  - 上映日期
  - 滿意度評分
  - IMDb評分
  - 最新5則評論

```python
import requests
import re
from bs4 import BeautifulSoup
from time import sleep
from random import random
from linebot import LineBotApi, WebhookHandler
from linebot.exceptions import InvalidSignatureError
from linebot.models import TextSendMessage# 載入 TextSendMessage 模組
from linebot.models import MessageEvent, TextMessage,TextSendMessage, ImageSendMessage, StickerSendMessag
import json
# YahooMovies 邏輯 : 使用者輸入 > 搜尋電影 > 確認哪一部(沒有就重找or沒有這部) > 找這部資訊 > 返回搜尋結果
# IMDbMovies 邏輯 : 從YahooMovies拿完整確定查找的電影名稱(有不同或多選擇需要使用者去Y/N決定) > 搜尋該電影評論 > 返回資訊
class YahooMovies():
    def __init__(self):
        self.movie_link = None

    def user_input(self, movie_name): # 返回使用者輸入的電影名稱(可能不是完整電影名)
        while True:
            try:
                movie_name = movie_name
                if movie_name.strip():
                    return movie_name
            except:
                print('\nTry Again')
                return None

    def search_movie(self, userquery): # 返回搜尋列表、使用者查的電影名稱(可能不是完整電影名)
        encoded_query = requests.utils.quote(userquery)
        url = f"https://movies.yahoo.com.tw/moviesearch_result.html?keyword={encoded_query}"
        response = requests.get(url)
        soup = BeautifulSoup(response.text, "html.parser")
        movie_titles = soup.find_all("div", class_="release_movie_name")
        movie_urls=[]
        movie_yahoo_titles=[]
        movie_titles_string = ""
        i=0
        # movie_links = soup.select(".release_movie_name a")
        res_count=len(movie_titles)
        if res_count==0:
            return "", 0, [], []
        #超過1筆以上
        elif res_count>=2:


            if res_count>=3:
                res_count=3
            #數量
            count=res_count

            #網址
            for div in soup.find_all('div', class_='release_movie_name'):
                if len(movie_urls)==4:
                    break
                movie_urls.append(div.a['href'])
            #電影標題
            for font in soup.find_all('font', class_='highlight'):
                text = f'{font.string}{font.find_next_sibling(string=True)}'
                if len(movie_yahoo_titles)==4:
                    break
                movie_yahoo_titles.append(text)
```

#電影資訊搜尋

Yahoo電影爬蟲

- 爬取指定Yahoo電影，使用者輸入(中英文片名)後返回該片的資訊，包括
  - 中英文片名
  - 上映日期
  - 滿意度評分
  - IMDb評分
  - 最新5則評論

```
        #linebot能接受之訊息格式
        for i,title in enumerate(movie_titles, 1):
            if i==4:
                break
            astr=title.get_text().rstrip()
            movie_titles_string += f"第{i}筆. {astr}\n"
        movie_titles_string= movie_titles_string.split('\n')
        movie_titles_string = [line.strip() for line in movie_titles_string if line.strip()]
        movie_titles_string= '\n'.join(movie_titles_string)
        pattern = r'第\d+筆\.\n(.+?)\n'
        movie_yahoo_titles = re.findall(pattern, movie_titles_string, re.DOTALL)  # 使用re.findall找到符合條件的所有電影標題


        return movie_titles_string, count, movie_urls, movie_yahoo_titles
    #數量等於1筆
    elif res_count==1:

        #數量
        count=res_count

        #網址
        movie_urls=movie_titles[0].a['href']


        #linebot能接受之訊息格式
        for i,title in enumerate(movie_titles, 1):
            astr=title.get_text().rstrip()
            movie_titles_string += f"第{i}筆. {astr}\n"
        movie_titles_string= movie_titles_string.split('\n')
        movie_titles_string = [line.strip() for line in movie_
        movie_titles_string= '\n'.join(movie_titles_string)
        pattern = r'第\d+筆\.\n(.+?)\n'
        movie_yahoo_titles = re.findall(pattern, movie_titles_


        return movie_titles_string, count, movie_titles[0].a['
    else:
        return "", 0, [], []

def specific_movie_info(self, movie_link): # 返回這部電影所需的資
    def review_latest(review_link):
        if review_link is None:
            return None
        res = []
        response = requests.get(review_link)
        soup = BeautifulSoup(response.text, "html.parser")
        reviews = soup.find_all('div', class_='usercom_inner _

        review_counts=0
        for i in reviews:
            if review_counts == 5:
                break
            content = re.sub(r'[\s]{2,}|[\n\r\t]+', '', i.find
            #排除廣告評論
            pattern = re.compile(r'(微信|line|外約|娛樂城|私聊)',
            if pattern.search(content):
                continue

            res.append(content)
            review_counts += 1
        return res if res else '該電影目前無評論'
```

```
movie_info = {}
if movie_link:
    response = requests.get(movie_link)
    soup = BeautifulSoup(response.text, "html.parse
    movie = soup.find('div', class_='movie_intro_in
    review = soup.find('div', class_='btn_plus_more
    review_link = review.a['href'] if review else N
    starscore = soup.find('div', class_='score_num
    starbox = soup.find('div', class_='starbox2')

    chname = re.sub(r'[\s]{2,}|[\n\r\t]+', '', movi
    engname = re.sub(r'[\s]{2,}|[\n\r\t]+', '', mov

    if starscore and starbox:
        starscore = f'{starscore.text} / 5 {starbox
    else:
        starscore = None

    reviews = review_latest(review_link)

    release_date, IMDB = None, None
    for i in movie.find_all('span'):
        text = i.text.strip()
        if '上映日期' in text:
            release_date = text.split('：')[1]
        elif 'IMDb分數' in text:
            IMDB = f"{text.split('：')[1]} / 10"
    movie_info['中文名稱'] = chname
    movie_info['英文名稱'] = engname
    movie_info['上映日期'] = release_date
    movie_info['滿意度'] = starscore
    movie_info['IMDb'] = IMDB
    movie_info['Yahoo最新評論'] = reviews

    return movie_info
```

#電影資訊搜尋

IMDb電影爬蟲

爬取指定IMDb電影後
返回該片前5則評論

```python
class IMDbMovies(YahooMovies):
    def __init__(self, name):
        self.headers = {
            "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/
        }
        self.yahoosearchname = name        # Yahoo找到完整電影名稱(不是使用者輸入的)

    def specific_movie_reviews(self, movie_link): # 返回電影評論list
        response = requests.get(movie_link, headers=self.headers)
        soup = BeautifulSoup(response.text, "html.parser")
        reviews = soup.find_all('div', class_='review-container')

        movie_reviews = []
        for review in reviews[:5]:
            review_title = review.find('a', class_='title').text.strip()
            movie_reviews.append(review_title)
        return movie_reviews if movie_reviews else None

    def search_movie(self, query): # 返回電影評論網址
        # search
        encoded_query = requests.utils.quote(query)
        url = f"https://www.imdb.com/find/?q={encoded_query}&ref_=nv_sr_sm"
        response = requests.get(url, headers=self.headers)
        soup = BeautifulSoup(response.text, "html.parser")
        search_results = soup.select('#__next > main > div.ipc-page-content-conta
        # check search list
        if search_results:
            search_count = len(search_results)
            movie = search_results[0]
            k = movie.select_one('div > div > a')
            movie_number = k.get('href').split('/')[2]
            review_link = f'https://www.imdb.com/title/{movie_number}/reviews?ref
            if query == k.text:
                return 1,k.text,review_link
            if search_count > 1:
                name=[]
                link=[]
                for m in search_results:
                    k = m.select_one('div > div > a')
                    movie_number = k.get('href').split('/')[2]
                    review_link = f'https://www.imdb.com/title/{movie_number}/rev
                    name.append(k.text)
                    link.append(review_link)
                    print(k.text,'m',review_link)
                # 該次搜尋都沒找到，重新搜尋
                return search_count,name,link
        else:
            return 0,None,None
```

LINE Bot選單建置

```python
import requests
import json
def menu():
# 輸入 Access Token，記得前方要加上「Bearer 」( 有一個空白 )
    headers = {'Authorization':'Bearer

    # 需補上'action'設定
    body = {
        'size': {'width': 1250, 'height': 843},   # 設定尺寸
        'selected': 'true',                        # 預設是否顯示
        'name': 'movie bot menu',                  # 選單名稱
        'chatBarText': 'Movie Bot Menu',           # 選單在 LINE 顯示的標題
        'areas':[                                  # 選單內容
            {
            'bounds': {'x': 5, 'y': 0, 'width': 406, 'height': 843}, # 選單位置與大小
            'action': { "type": "uri","label": "Location",'uri':'https://line.me/R/nv/location'} # 點擊地標圖示後的動作
            },
            {
            'bounds': {'x': 421, 'y': 0, 'width': 406, 'height': 843},
            'action': {  "type": "postback","label":"我想看更多","data":"@M1"} # 點擊隨機圖示後的動作
            },
            {
            'bounds': {'x': 837, 'y': 0, 'width':406, 'height': 843},
            'action': { 'type': 'uri', 'label': '點我推薦', 'uri':'https://line.me/R/oaMessage/@708tiikx/?%E9%9B%BB%E5%BD%B1'}  # 點擊放大鏡圖示後的動作
            }
        ]
    }
    # 向指定網址發送 request
    req = requests.request('POST', 'https://api.line.me/v2/bot/richmenu',headers=headers,data=json.dumps(body).encode('utf-8'))
    return req.text
print(menu())
```

LINE Bot建置

使用TemplateSendMessage
呈現文字訊息

```
if moviecount>=3:
    reply_msg=TextSendMessage(YahooMovies_info.search_movie(userquery)[0]),
              TextSendMessage(text=f'找到{moviecount}部電影'),
              TemplateSendMessage(
                  alt_text='Confirm Choice',
                  template=ButtonsTemplate(
                      title='Choose one!',
                      text='選你要的吧！',
                      actions=[
                          PostbackTemplateAction(
                              label=f'{moviename[0]}',
                              display_text=f'{moviename[0]}',
                              data=f'Y{movielink[0]}&{moviename[0]}'
                          ),
                          PostbackTemplateAction(
                              label=f'{moviename[1]}',
                              display_text=f'{moviename[1]}',
                              data=f'Y{movielink[1]}&{moviename[1]}'
                          ),
                          PostbackTemplateAction(
                              label=f'{moviename[2]}',
                              display_text=f'{moviename[2]}',
                              data=f'Y{movielink[2]}&{moviename[2]}'
                          ),
                      ]
```

使用ButtonsTemplate
確認使用者選擇選項

LINE Bot建置

使用QuicklyReplyButton呈現
快速回覆 （沒有照規則輸入時出現）

```
else:
    try:
        reply_msg = TextSendMessage(
            text='抱歉我不懂您的意思，來看看電影吧！',
            quick_reply=QuickReply(
                items=[
                    QuickReplyButton(
                        action=URIAction(label='你要的都在這',uri='https://line.me/R/oa
                    ),
                    QuickReplyButton(
                        action=URIAction(label="點我看說明", uri="https://line.me/R/ho
                    ),
                    QuickReplyButton(
                        action=MessageAction(label="戲劇類", text="我要看戲劇類")
                    ),
                    QuickReplyButton(
                        action=MessageAction(label="喜劇類", text="我要看喜劇類")
                    ),
                    QuickReplyButton(
                        action=MessageAction(label="動作類", text="我要看動作類")
                    ),
                    QuickReplyButton(
                        action=MessageAction(label="驚悚類", text="我要看驚悚類")
                    ),
                    QuickReplyButton(
                        action=MessageAction(label="浪漫類", text="我要看浪漫類")
```

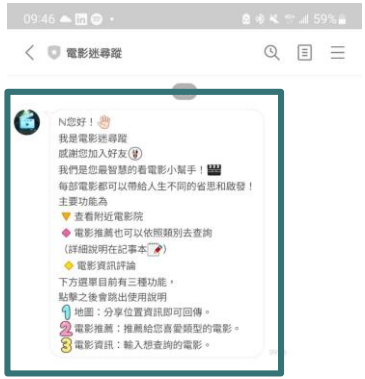
LINE Bot建置

使用carousel呈現
三個推薦最近電影院左右滑的功能


```python
def handle_message1(event):
    tk=event.reply_token
    if event.message.type=='location':
        try:
            #擷取經緯度event.message.latitude,event.message.longitude
            location = '{},{}'.format(event.message.latitude,event.message.longitude)
            #連接資料庫
            conn = sqlite3.connect('C:/Users/zxc62/Downloads/Movie-20230703T224159Z-001/AMovie/TW_cinema_info.db')
            data_df = pd.read_sql_query("SELECT * FROM TW_cinema_info_latest", conn)
            conn.close()
            # 建立物件
            finder = CinemaFinder(location)
            filtered_data = finder.MathJudge_dist(data_df)
            result = finder.order_df_dist_google(filtered_data)
            #將回傳數值轉成字串
            result=result.to_dict()
            flex_message = FlexSendMessage(alt_text='您有新訊息',
                                           contents={"type": "carousel",
                                                     "contents": [{
                                                     "type": "bubble",
                                                     "size": "micro",
                                                     "hero": {
                                                     "type": "image",
                                                     "url": "https://images.pexels.com/photos/9433910/pexels-photo-9433910.
                                                     "size": "full",
```

成果展示

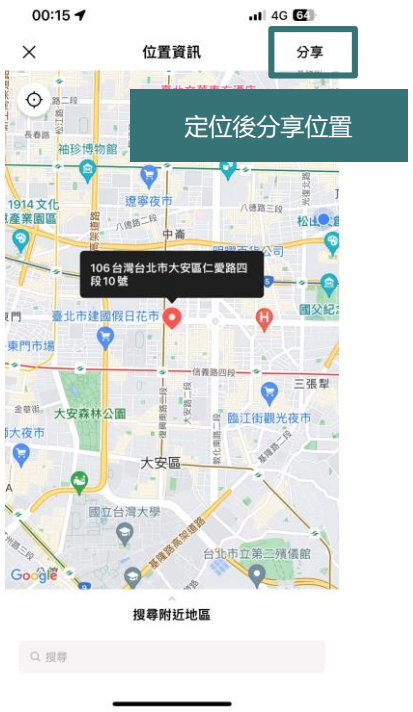
歡迎文字及主畫面選單



歡迎文字

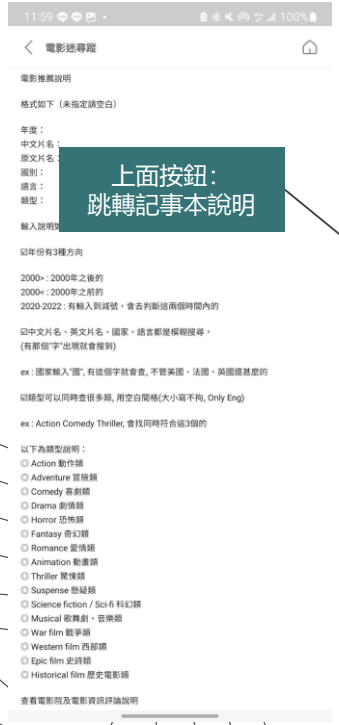
主畫面選單，由左至右分邊為
地圖、隨機推薦、電影資訊和評論

選單左側icon: 地圖


00:15
4G
位置資訊
分享
定位後分享位置
106 台灣台北市大安區仁愛路四段10號
搜尋附近地區


回傳三個
離使用者最近電影院
點我規劃
你想要的在這
可選擇直接規劃路線
或跳轉google map頁面

選單中間icon: 隨機推薦功能



上面按鈕:
跳轉記事本說明

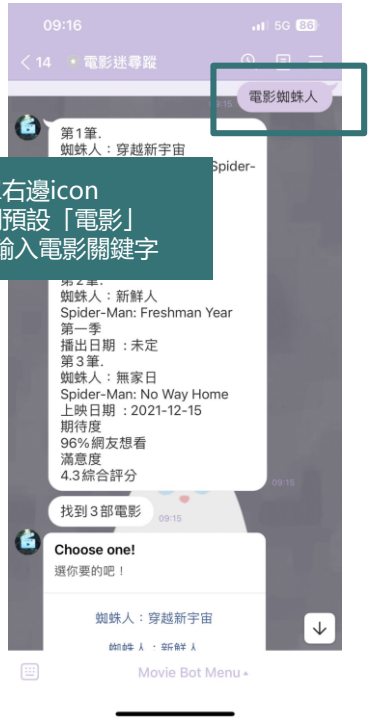
下面按鈕:
跳出篩選清單

使用者填入
偏好的條件查詢

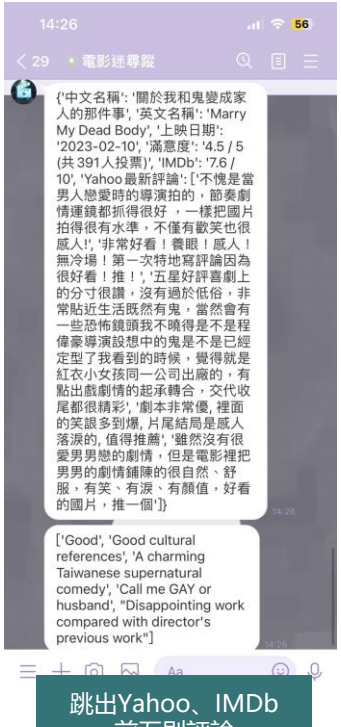
回傳三部
推薦電影

選單右側icon: 電影資訊和評論搜尋功能

點選單右邊icon
跳轉訊息欄預設「電影」
使用者直接輸入電影關鍵字



選擇想要的電影



跳出Yahoo、IMDb
前五則評論

若使用者無依格式傳送訊息：快捷鍵

若使用者沒有依照格式輸入訊息，則出現七個快捷鍵，包含

1. 你要的都在這：跳轉訊息欄出現有條件的篩選清單
2. 點我看說明：跳轉記事本詳細說明
3. 戲劇類、喜劇類、動作類、驚悚類、浪漫類：直接隨機回傳推薦三部該類型電影

隨機回傳推薦
三部該類型電影

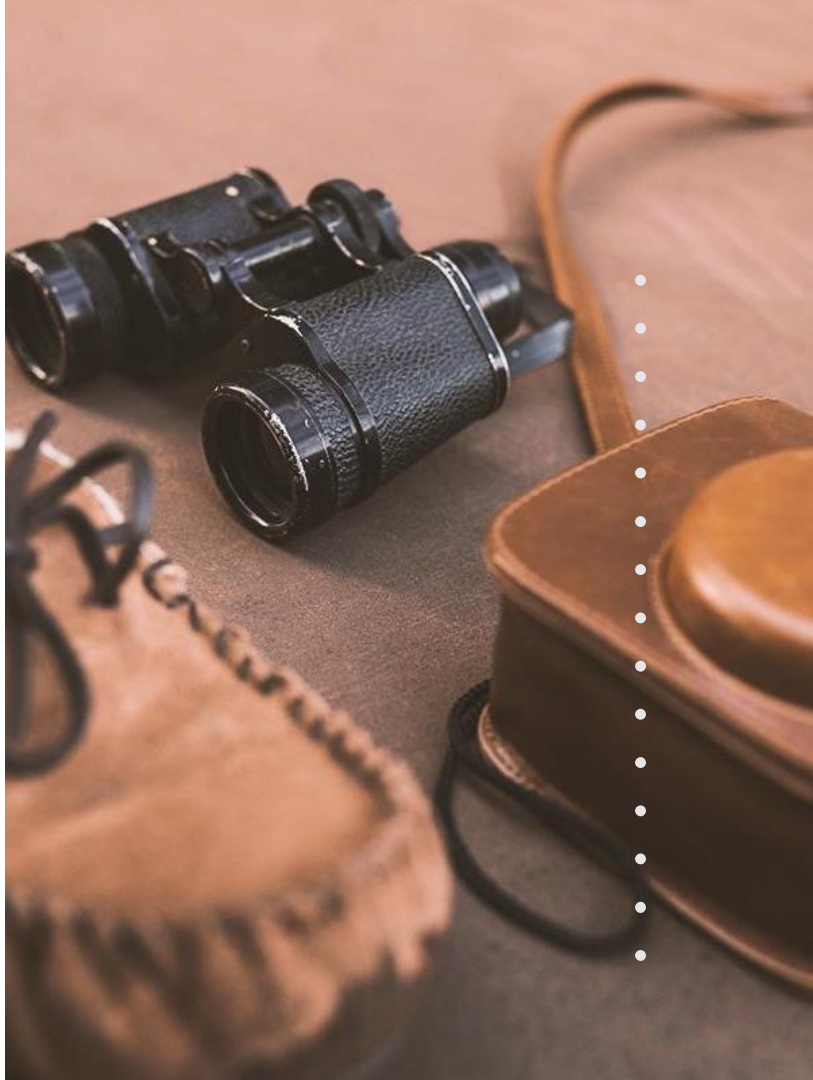
專題展望

本次的挑戰

- 因為政府電影資料庫無標記分類，需要再爬IMDb每一部電影分類、每爬一部電影都要跑一次code，爬起來相當費時
- 原本想串ChatGPT，但研究發現其不受控，因故無串接

展望

- 爬入更多評論，如PTT movie板
- 加入更多功能，如聲量文字雲、票價比較、不同電影版本(3D、IMAX等)



Resources

1. Messaging API Overview. (n.d.). LINE Developers. https://developers.line.biz/en/docs/messaging-api/overview/
2. Janjanjanice. (2024, September 28). Line Bot 完結篇: 體驗確診小幫手~ “鐵人屁桃30日挑戰” & 補充說明~. IT邦幫忙. https://ithelp.ithome.com.tw/articles/10299831
3. [Python+LINE Bot教學]提升使用者體驗的按鈕樣板訊息(Buttons Template Message)實用技巧. (2020, July 12). LEARN CODE WITH MIKE. https://www.learncodewithmike.com/2020/07/line-bot-buttons-template-message.html
4. 快速產生 LINE 網址的小工具. (n.d.). 筆記國度. https://taichunmin.idv.tw/blog/2023-03-11-line-url-generator.html?fbclid=IwAR0GgaUsizHJz1IjtrB9yJy2dYVsPyPmyZURar3ihANt68Xox5jNWnmpI2g
5. [Python+LINE Bot教學]6步驟快速上手LINE Bot機器人. (2020, June 28). LEARN CODE WITH MIKE. https://www.learncodewithmike.com/2020/06/python-line-bot.html
6. Enoxs . (2021, November 27). 【Python SQL : 數據持久化 攻略】SQLite x MySQL x SQLAlchemy. iT邦幫忙. https://ithelp.ithome.com.tw/articles/10282830
7. Pei cheng. (2018, August 12). 透過網址開啟地圖 APP 並顯示特定地點資訊. Medium. https://medium.com/@peicheng_88746/透過網址開啟地圖-app-並顯示特定地點資訊-6cc73ee2a871

Thanks