Carlos Iván Canto Varela 315649888

Proyecto 1

Descripción

Se requiere un programa que informe del clima actual en la ciudad de partida y el clima esperado en la ciudad de llegada para 3 mil vuelos que se realizarán. Para esto, se implementó un web service para el clima de llegada y salida para el aeropuerto de la Ciudad de México.

El procedimiento para tratar los datos conseguidos fue crear un diccionario como caché. Luego, buscar las ciudades según su código IATA dentro del cache. Si no han sido buscadas dichas ciudades, se creará una entrada nueva en el diccionario con la clave IATA como su identificador y sus características climatológicas en una lista como el contenido de la entrada. Luego, cada ciudad buscada será añadida junto con sus características a una lista ordenada de origen o destino. Una vez hecho esto, se repetirá el proceso para cada ciudad de los 3 mil viajes. Al finalizar el proceso, las listas de orígenes y llegadas serán convertidas a dataframes y se mezclarán como columnas para una visualización intuitiva.

Motivación

A primera instancia, mi primera idea fue realizar un scraper que visitara el sitio de Yahoo Weather, buscara las localizaciones según su código IATA y regresara el clima buscado. Después de hacer una pequeña prueba, descubrí que cada localización tardaba un promedio de 2 segundos de procesamiento, por lo que desistí en esa idea y decidí hacerlo con la API de OpenWeatherMap. Además se implementó un caché rústico por medio de diccionarios de Python para evitar llamadas innecesarias a la API. Se escogió Python por la rapidez de escritura que presenta al no ser necesario hacer clases porque mi equipo es de una persona.

Prerrequisitos

Se necesita tener Python (versión 3 en adelante) y haber instalado las paqueterías requests, json, thread6, pandas y numpy. Requests se utiliza para conseguir la información de OpenWeatherMap mediante la función get que utiliza la URL de la API como entrada y regresa la información como interfaz response. Luego, el método json de la paquetería de mismo nombre se encarga de leer la salida de requests. Thread6 se utiliza para aplicar paralelismo con la creación de Threads para cada proceso inquisitivo: una para el clima del lugar de origen y otra para el del lugar destino. Pandas se usa para leer el archivo de datos a buscar (entrada del programa) con extensión csv mediante el método read_csv. Finalmente, numpy se usó para crear índices aleatorios y probar el funcionamiento del programa con vuelos distintos.

Ejecución

El primer paso es descargar la carpeta en un directorio elegido. Luego, en la terminal (Windows), se utilizará el comando

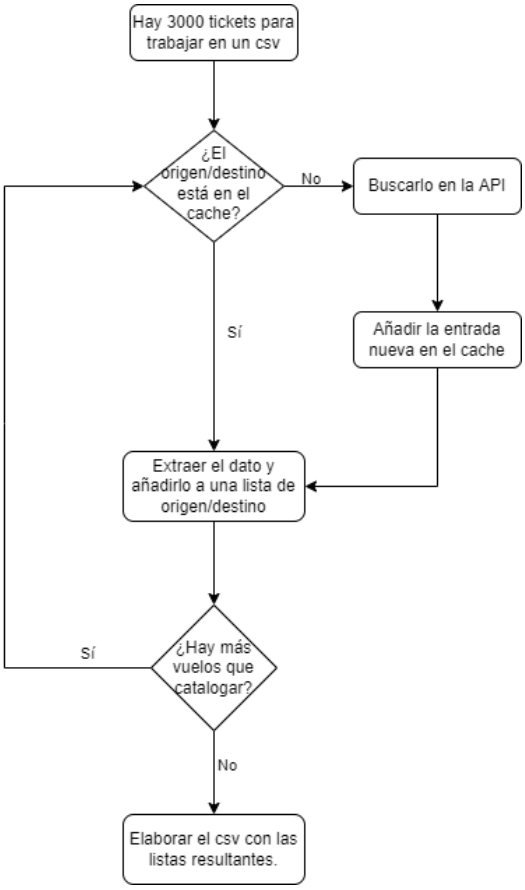
cd path="directorio\de\la\carpeta"

Es crucial asegurarse que el directorio contiene los archivos src.py y test.py. El siguiente y último paso es ejecutar el programa test.py con el comando

python __main__.py

El resultado final será una tabla con la información climática del origen y destino de cinco vuelos; deberá aparecer en la misma terminal.

Funcionamiento



El programa utiliza la paquetería básica de python os para generar un directorio relativo a la localización del programa y encontrar el csv con los vuelos para que no haya necesidad de introducirlo el directorio de la persona que utilice el programa. Así es como el método data() genera la lectura y conversión de dichos datos en un dataframe para poder acceder a las coordenadas de los lugares. Con los métodos datum_origin y datum_destination, cuyas entradas son un índice, el dataframe con los vuelos, un cache de diccionario y una lista, los códigos IATA son buscados en el diccionario para evitar repetir búsquedas. Si no se encuentran dichas llaves se procede a crear una nueva entrada para el caché con la función new_data, la cual llama y genera un índice diferenciador dependiente de si el lugar es un origen o un destino. La entrada al caché de la función mencionada usa el diferenciador proporcionado para buscar las coordenadas de la localización y accede a la API de OpenWeatherMap le proporciona el formato json para acceder a el clima de dicha localización y extrae la información climática requerida, la cual es almacenada en una lista junto con la clave IATA para mostrarlo en el resultado. Posteriormente, se crea una nueva entrada con el código IATA como llave y una lista con el clima del lugar. Finalmente, la información de las listas de origen y destino es juntada en un dataframe ordenado según los índices de los vuelos buscados que se despliega en la terminal una vez acabado el proceso.

Mantenimiento a futuro

Si se decide expandir el servicio a más de 3 mil tickets con una mayor frecuencia, la implementación podría verse insuficiente pues la licencia de estudiante sólo cubre 3 mil búsquedas por minuto. En este caso, el mantenimiento que deberá aplicarse dependerá del presupuesto disponible. Podría simplemente comprarse una licencia de mayor rango y cambiarse la llave de

acceso, lo que tomaría unos minutos, cuando mucho. Mi costo sería el costo de la licencia y unas horas de salario mensualmente. En otro caso, lo que podría hacerse sería adaptar el código para correr 3 mil búsquedas cada minuto hasta cubrir las que son necesarias, lo que me podría llevar desde una hora hasta unos días. Por este servicio cobraría el doble. En un semestre, la llave utilizada para acceder a la API caducará y será necesario conseguir una nueva. Gracias a la implementación del código, el resultado que se visualizará será un error al encontrar los datos. Cuando esto pase y sea contactado (si no deciden utilizar a un tercero) cobraría por la licencia y unas horas de salario. El salario a considerar sería el promedio de un Junior Developer por hora.