Universidad Autónoma de Madrid

Escuela Politécnica Superior

21 | 22

Trabajo fin de grado

Recomendación de puntos de interés con restricciones temporales

Javier Sanmiguel Treviño

Escuela Politécnica Superior Universidad Autónoma de Madrid C\Francisco Tomás y Valiente nº 11

Campus Internacional excelencia UAM CSIC

www.uam.es

UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería de Textos

TRABAJO FIN DE GRADO

Recomendación de puntos de interés con restricciones temporales

Autor: Javier Sanmiguel Treviño
Tutor: Alejandro Bellogín Kouki

junio 2022

Javier Sanmiguel Treviño
Recomendación de puntos de interés con restricciones temporales

Javier Sanmiguel Treviño
C\ Francisco Tomás y Valiente Nº 11

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

A mi familia y amigos

Debemos abrazar el dolor y quemarlo como gasolina para nuestro viaje.

-Kenji Miyazawa

Agradecimientos

La primera persona a agradecer es mi tutor Alejandro Bellogín Kouki, por tener una dedicación absoluta durante este trabajo de fin de grado; donde el resto sencillamente habrían seguido la corriente, el se ha involucrado como si fuese su propio trabajo.

A continuación, pero no menos importante, quiero agradecer a mi familia por la inspiración de estudiar este ámbito y sobre todo el apoyo recibido durante estos 6 arduos años de estudios universitarios. En especial a mi padre, Javier Sanmiguel Riesgo, por dar los toques de realidad, a mi madre, Natalia Treviño Lafarga, por darme la mano durante los malos tragos que son las bases de datos y a mi hermana, Ana Sanmiguel Treviño, por darme ánimos durante todo este tiempo.

También agradecer a mis compañeros de carrera con los que he formado una pequeña familia durante estos años y que espero que durante más, en especial a Patricia Matos Meza, quien me ha ayudado innumerables veces y con su humor me ha mantenido a flote. Y a mis amigos de fuera de la universidad, por ayudarme a mantener la cordura con dragones y mazmorras durante los estudios.

Resumen

En el mundo globalizado actual, viajar nunca ha sido más fácil y a la vez más difícil por la gran cantidad de lugares e información por escoger. Por ello, millones de personas utilizan aplicaciones con las que programar sus viajes y descubrir lugares que no habrían encontrado sin estas.

Como culmen están las aplicaciones de rutas turísticas o roadtrippers, en las que seleccionas un conjunto de puntos de interés (POIs) o lo dejas al azar y te recrean visualmente la ruta a seguir para visitar todos los lugares incluidos. A esto hay que añadir que el usuario tendrá que comer a lo largo de la ruta para continuar y esto significa un POI adicional que habría que añadir a la ruta.

Todas estas elecciones pueden realizarse con sistemas de recomendación que filtrarán la información para dar al usuario aquellos POIs que coincidan con sus gustos.

Tanto el sistema de recomendación como el roadtripper necesitan de una representación gráfica para seleccionar las opciones y visualizar el resultado de estas. Para ello, en este proyecto se ha desarrollado una plataforma web que permite seleccionar los POIs y gustos culinarios del usuario para después mostrar la ruta que debe seguir para obtener la mejor experiencia.

Palabras clave

Punto de interés, POI, Roadtripper, Restauración, Sistema de recomendación

Abstract

In today's globalized world, traveling has never been so easy and yet so complex due to the massive amount of places and information to choose from. Because of this, millions of people resort to the use of apps with which they can program their travels and discover places they may have never found without them.

The epitome of these apps are the roadtrippers, which visually create a route based on the points of interest (POIs) selected or randomness. To this there has to be added that the user needs to eat somewhere for him to continue the route with energy, which means an additional POI to be added to the route.

All this choice can be done with the help of recommender systems, which filter the information to give the user the POIs that match his tastes.

Both the recommender system and the roadtripper need a visual representation from where to choose the options and visualize the results of those options. For this reason, for this project it has been developed a web platform which lets the user select the POIs and culinary tastes to later show the route the user should follow to obtain the best experience.

Keywords

point of interest, POI, Roadtripper, restaurant business, recommender systems

Índice

Listas

Lista de ecuaciones

Lista de figuras

Lista de tablas

1

Introducción

1.1. Motivación

El turismo hoy en día, incluso tras la pandemia que ha reducido los ingresos de 1.466.000 millones de dólares a 566.000 millones de dólares [1], sigue siendo una de las principales fuentes de ingresos de muchos países. Con la cantidad ingente de lugares e información que tiene el turista a mano, es imperativo crear medios que ayuden a estos a realizar una visita de forma sencilla para fomentar el gasto y los viajes.

La principal herramienta usada hoy en día es la generación de rutas turísticas con las que el turista pueda, de forma intuitiva y sencilla, navegar una zona y visitar los lugares que quiera. Además, el principal problema que tiene un turista cuando viaja a una zona desconocida es encajar no solo cuándo comer, sino también dónde, y además que sea algo que a él le guste.

Los recomendadores, mediante las preferencias del individuo, generan sugerencias que aplicados al ámbito de la restauración den al usuario un restaurante en el que se sienta cómodo y disfrute de una comida que ayude a mejorar la experiencia de la visita.

1.2. Objetivos

Este proyecto trata de crear una herramienta que, como se ha mencionado arriba, sea intuitiva y sencilla y realice una ruta que el usuario pueda seguir a rajatabla y tener una visita inolvidable, teniendo en cuenta restricciones temporales como la duración media de una visita, horarios de comida, duración de una ruta, etc. Para ello, se creará una interfaz web en la que el usuario pueda introducir los lugares que quiere visitar y mediante un recomendador se añada un restaurante en los horarios generales establecidos en la mayoría de culturas.

1.3. Estructura de la memoria

El documento se estructurará siguiendo el siguiente formato:

1. Introducción.

En este capítulo se habla la motivación, objetivos y estructura del proyecto.

2. Estado del arte.

En este capítulo se analizarán los detalles teóricos que sustentan el proyecto.

3. Análisis, diseño e implementación.

En este capítulo se hará un análisis técnico del proyecto, decisiones de diseño e implementación.

4. Resultados.

En este capítulo se mostrarán y comentarán los resultados obtenidos.

5. Conclusiones e implementaciones futuras.

En este capítulo se hablará de la conclusión del proyecto y las posibles implementaciones para mejorar el proyecto.

2

Estado del arte

En este capítulo se hablará de las partes principales que conforman a los ”Roadtrippers” o aplicaciones de planificación de viajes (puntos de interés, algoritmo para encontrar la mejor ruta y sistemas de recomendación) y las tecnologías web necesarias para ofrecer una interfaz intuitiva al usuario.

2.1. Puntos de interés (POI)

Los puntos de interés o POI son lugares de intereses habitualmente frecuentados durante el día, por ejemplo, restaurantes, supermercados, atracciones turísticas, etc. [2]. Empresas de Location Based Social Networks (LBSN) como Foursquare y Facebook Places utilizan estos POIs, normalmente geo-localizados mediante latitud y longitud, para desarrollar una base de localizaciones que poder recomendar mediante un sistema de recomendación.

Dado que los POIs se basan en la interacción del usuario, se deben poseer mecanismos para mantener la base de localizaciones con una calidad que pueda ser usada. Los aspectos más relevantes a tener en cuenta para crear una base de POIs bien formada son [2]:

Completitud

Presencia o ausencia de POIs de una fuente de datos comparada con la realidad. Los errores pueden surgir por encargo (existe exceso de datos en el dataset) o por omisión (faltan datos en el dataset).

Consistencia lógica

Se observa si la estructura de datos es rígida y no tiene problemas de relación, atributos, etc. [2].

Precisión geo-espacial

Precisión de la localización de un POI, expresada en latitud/longitud. Parecido a como se aborda la identificación de patrones para evaluar la consistencia lógica, algunos POIs tienen más probabilidades de estar colocados con otros tipos de POIs, y desviación del patrón pueden indicar errores en la posición.

Calidad temporal

Puede ser precisión temporal del POI (horarios) o moda del POI (cuándo fue creado o actualizado por última vez). La mayoría de abordamientos relacionados con la temporalidad son aplicables a los datos Open Street Map (OSM) POI; que permiten el análisis temporal gracias a que estas modificaciones son perseguidas y publicadas.

Precisión temática

Exactitud de los tipos de lugar encontrados en los POIs. La temática se refiere a cualquier información adicional asociada a un POI, pero normalmente se suele conectar con el nombre y el tipo de lugar.

Usabilidad

Cuántos requerimientos del usuario cumplen los datos. Para analizar esto hay que dividir los datos en distintas áreas de estudio; algunos ejemplos de usabilidad son la conexión al objeto y la conexión a la zona geográfica. Los POIs pueden incorporarse en modelos para analizar fenómenos urbanos, como la relación entre el número de cafeterías y la subida de los precios del alquiler.

Como se muestra en la figura 2.1, el número de artículos sobre la revisión de los datasets de POIs es muy pequeña, mientras que una simple búsqueda con los términos ”POI”, ”point-of-interest” y ”LBSN” devuelve 10 veces más. Esto presenta un debate sobre si estos estudios están basados sobre unos datasets bien revisados y refinados, o sencillamente es

más preferible realizar estudios, aunque no se tenga la certeza de tener unos datasets de POIs con la calidad correspondiente. Cabe destacar que la mayoría de estudios se realizan sobre datasets de POIs que pertenecen a empresas privadas como Foursquare, y esto hace difícil realizar revisiones sobre los datasets.

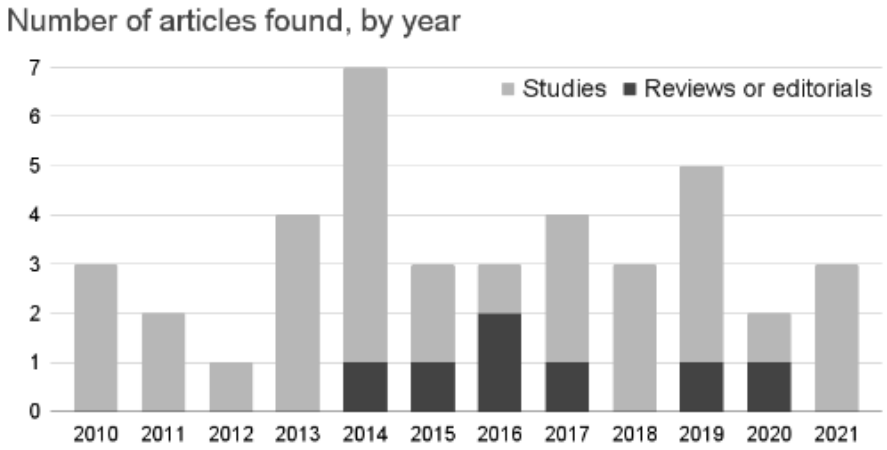

Figura 2.1: Número de artículos sobre la revisión de POIs por año [2]

Como se ha mencionado al inicio de la sección, las LBSN como Foursquare y Facebook Places basan su negocio en la recolección de registros del usuario o check-ins en POIs para guardar en su base de datos y así tener una mejor calidad en las recomendaciones siguientes. Otro tipo de aplicaciones basado en POIs son los ”Roadtrippers” como Roadtrippers o Google Maps, estas aplicaciones crean rutas óptimas para ser recorridas por el usuario basándose en los POIs que este da de entrada.

Este proyecto se basa tanto en las aplicaciones de los Roadtrippers para generar una ruta de recorrido óptimo mediante la resolución del problema de enrutamiento de vehículos o VRP (explicado en la sección 2.2), como en las LBSN y su uso de recomendadores (detallado en la sección 2.3) para recomendar al usuario el restaurante donde debería comer dentro de esa ruta.

2.2. Vehicle Routing Problem

Vehicle Routing Problem (VRP) o problema de enrutamiento de vehículos es un problema NP-completo de optimización combinatoria que generaliza el Traveling Salesperson Problem (TSP), también NP-completo.

El TSP se define de la siguiente manera: dada una lista de N ciudades y una ciudad inicial, el vendedor debe recorrer todas las ciudades exactamente una vez y terminar en la ciudad inicial [3].

El VRP, por otro lado, se define mediante una lista de nodos a visitar, siendo uno de ellos inicio y fin (denominado depot o almacén) y un conjunto de viajantes (vehículos, personas, etc.) que empiecen en ese depot. En esta situación, el problema consiste en encontrar las rutas de coste mínimo (se tarde menos, requiera la menor distancia posible, requiera del menor número de viajantes, etc.) en las que cada nodo solo deba visitarse una sola vez por uno de los viajantes [4]. Se observa que la generalización del VRP respecto al TSP reside en que VRP admite varios viajantes, aunque a su vez añade una forma de optimización adicional en que se use el menor número de viajantes para completar la ruta.

Como se menciona arriba, VRP es un problema NP-completo y, por tanto, es necesario de heurísticas para reducir su tiempo de ejecución a algo polinomial. Google ofrece una librería que soluciona el VRP en tiempo polinomial mediante el uso de estas heurísticas, llamada ”ortools” [5]. La sección 2.4.1 tocará brevemente el porqué del uso de la librería y la sección 3.3 tratará más en detalle el uso de esta librería.

2.3. Sistemas de recomendación aplicados al turismo

Los sistemas de recomendación son sistemas de filtrado de información que solucionan el problema de la sobrecarga de información mediante el filtrado y la generación dinámica de información acorde a las preferencias, intereses o comportamiento del usuario [6].

Estos sistemas no solo forman parte, sino que son el núcleo, de las grandes empresas de hoy en día, como pueden ser Facebook y Twitter en el sector de las redes sociales; Amazon en el sector del comercio electrónico; o Netflix y HBO en el sector del entretenimiento.

Existen varios tipos de sistemas de recomendación, estos son [7]:

Recomendadores basados en relaciones sociales o demográficos:

El sistema recomendará en función de a que nicho demográfico pertenezca el usuario.

Recomendadores basados en el conocimiento:

La recomendación se basa en el conocimiento del recomendador acerca del uso del ítem, y como ese uso puede ayudar en el área que el usuario está pidiendo la recomendación.

Recomendadores basados en la contribución de cada miembro del grupo o colaborativos:

Hace recomendaciones al usuario basándose en lo que le ha gustado a otros usuarios similares anteriormente.

Recomendadores híbridos:

Estos recomendadores implementan dos técnicas de recomendación para, mediante las ventajas de una, eliminar las desventajas de la otra.

Recomendadores basados en el contenido:

Recomiendan basándose exclusivamente en la descripción del ítem y de un perfil con los intereses del usuario activo, teniendo en cuenta la filosofía de ”recomiéndame las cosas que he seleccionado o me han gustado anteriormente” [8]. Esto los hace los recomendadores por excelencia para rutas turísticas, dado que de forma muy sencilla obtienen buenos resultados, y son en los que nos centraremos.

Dentro de los recomendadores basados en el contenido se encuentran los sistemas de recomendación basados en el contenido conscientes de la semántica, que se basan en la asociación de palabras clave, el Modelo Espacio Vectorial (VSM) [9] es un modelo que trata un documento como un vector n-dimensional

$$\overrightarrow{d_j} = \langle w_{1j}, w_{2j}, ..., w_{nj} \rangle \tag{2.1}$$

con pesos ordenados donde w_{kj} es el peso del término t_k en el documento d_j.

Para calcular los pesos hay que utilizar la fórmula TF-IDF (frecuencia de términos-frecuencia invertida del documento)

$$TF - IDF(t_k, d_j) = TF(t_k, d_j) * log\frac{N}{n_k} \tag{2.2}$$

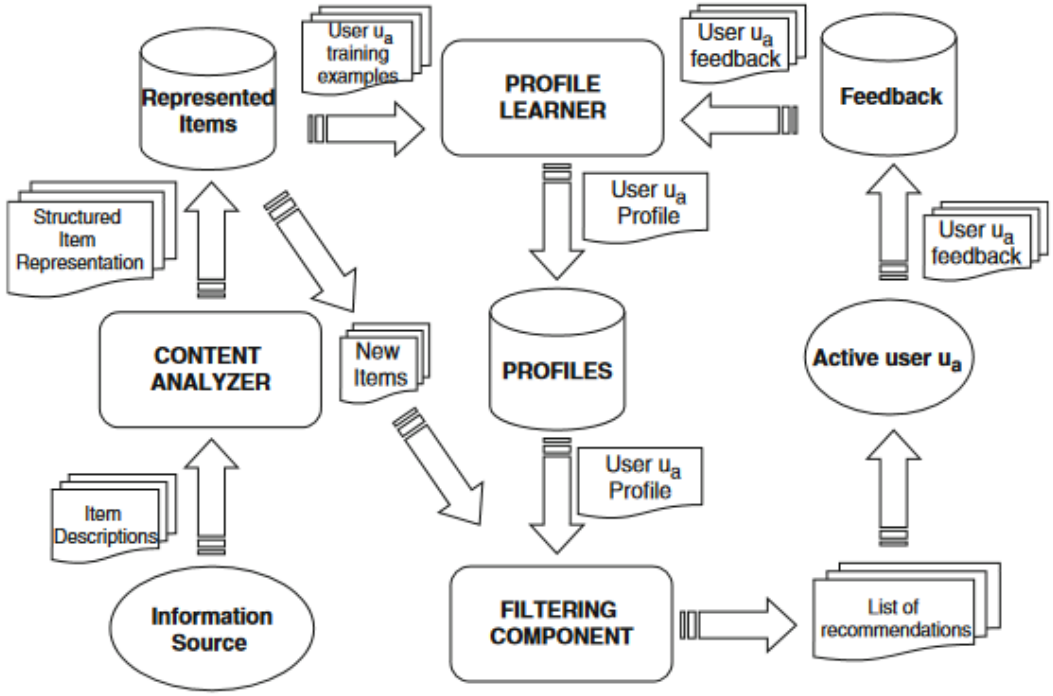

Figura 2.2: Arquitectura a alto nivel de un sistema recomendador basado en el contenido [6]

con N siendo el total de elementos. Esta fórmula permite normalizar resultados para que aquellos términos que se repitan mucho (TF) pero que no tienen relevancia en el documento actual (IDF) no influyan en el resultado.

Con ello el cálculo del peso w_{kj} quedaría como:

$$w_{k,j} = \frac{\text{TF-IDF}(t_k, d_j)}{\sqrt{\sum_{s=1}^{|T|} \text{TF-IDF}(t_s, d_j)^2}} \tag{2.3}$$

siendo t el término en el documento d y T el total de términos.

Con esto, el cálculo de similitud más común en los recomendadores basados en contenido que usan un modelo espacio-vectorial basado en palabras clave es la similitud coseno, cuya fórmula es la siguiente:

$$sim(d_i, d_j) = \frac{\sum_k w_{ki} * w_{kj}}{\sqrt{\sum_k w_{ki}^2} * \sqrt{\sum_k w_{kj}^2}} \tag{2.4}$$

donde w_{ki} son los pesos.

Los sistemas de recomendación aplicados al turismo son muy útiles a la hora de generar

dos soluciones; la primera es crear una ruta turística con POIs que el recomendador devuelva basándose en los gustos del usuario, y la segunda es generar una ruta que tenga un sentido secuencial.

En general, los LBSN no tienen en cuenta la secuencialidad dado que sus algoritmos son muy básicos porque su negocio se basa en el uso de la información de los registros. Aquí se halla una divergente entre los investigadores de los sistemas de recomendación de POIs y las compañías que poseen los datos; en el que los primeros crean sistemas de recomendación mucho más refinados, mientras que los segundos sencillamente utilizan los sistemas de recomendación clásicos, pero con buenos resultados dado que poseen una gran cantidad de ejemplos que el recomendador puede usar [10].

La recomendación de rutas turísticas con recomendadores ya ha sido estudiada previamente, como puede verse en ”TripBuddy” (K-Means Clustering) [11], ”TripRec” (recomendación basada en contenido) [12] y ”Hybrid recommender system” (recomendador hibrido) [13], en el que tratan la recomendación de POIs aplicada al turismo como un planificador de rutas de POIs; pero en ningún caso se ha estudiado usar el sistema de recomendación para añadir un restaurante a la ruta para completar dicha ruta turística. Esto ha servido como motivación para estudiar cómo funcionaria un sistema de recomendación basado en contenido cuando se aplica a la recomendación de restaurantes a una ruta ya creada y, por tanto, con una serie de restricciones temporales.

2.4. Estudio de las tecnologías

2.4.1. Para procesamiento y explotación de datos

Pandas es una librería rápida, potente y flexible de código abierto para el análisis y la manipulación de datos construida sobre Python [14]. Esta librería es la que se usará para limpiar inicialmente el dataset (más detalles en 4.2.2) y para obtener los restaurantes.

También se ha utilizado en menor medida la librería NumPy [15] debido a que es más eficiente que Pandas cuando se están tratando arrays o el sub set de datos es menor que 50K líneas.

OR-Tools es una librería de código abierto para la resolución de problemas de optimización

combinatoria desarrollada por Google [5]. Esta librería es la de facto para temas enrutamiento de vehículos, lo cual es justo lo que se necesita en este proyecto.

2.4.2. Para la aplicación web

Toda la información obtenida en el roadtripper debe mostrarse de una forma intuitiva al usuario, para ello una de las tecnologías más usadas son las tecnologías web. En el caso de este proyecto se ha escogido Python como lenguaje para la programación de la aplicación (se explicara el porqué más adelante en 3.1) y, por tanto, las 2 opciones principales son Django y Flask.

Django es un framework abierto y gratuito de alto nivel en Python que permite un desarrollo con un diseño rápido, limpio y pragmático. Al ser un framework, este se encarga de toda la implementación web y deja al desarrollador con la única tarea de desarrollar la aplicación [16].

Por el contrario, Flask es un micro-framework también abierto, gratuito y en Python. La diferencia reside en que Flask te ofrece un marco mucho menos potente, pero con más libertad de uso, por ejemplo, no posee abstracción de base de datos ni validación de formularios. Por lo que es un entorno rápido y fácil de implementar su base [17].

Se ha decidido escoger Django por ser el framework más usado durante la carrera y la posibilidad de aumentar fácilmente el alcance de la aplicación, que se discutirá en la sección 5.2.

Para el frontend se ha utilizado la opción más común que es HTML con una capa de estilo de CSS. Para mostrar el resultado de forma intuitiva se ha buscado una herramienta que permita, mediante los datos, crear un mapa señalando la ruta y los POIs. Se han tenido en cuenta 2 opciones, Folium [18] y Geopandas [19].

Folium es una librería externa de Python que mediante el uso de la librería Leafletjs [20] crea mapas interactivos. Geopandas extiende a la librería Pandas [14] para permitir operaciones geo-espaciales en tipos geométricos mediante Fiona [21] y Matplotlib [22].

Aunque en este proyecto se ha usado Pandas para tratar los datasets, se ha decidido usar la librería Folium por su facilidad de uso y resultado más llamativo hacia el usuario final, dado que permite interactuar intuitivamente con el mapa, además de añadir información adicional fácilmente.

3

Análisis, diseño e implementación

En este capítulo se detallarán los requisitos del proyecto. También se explicarán las decisiones de diseño tomadas, el diseño de la aplicación, el sistema recomendador y la aplicación web.

3.1. Análisis

En esta sección se realizará un análisis de los requisitos del proyecto y los algoritmos utilizados. Para entender mejor la aplicación, la figura 3.1 describe los módulos y el flujo de estos.

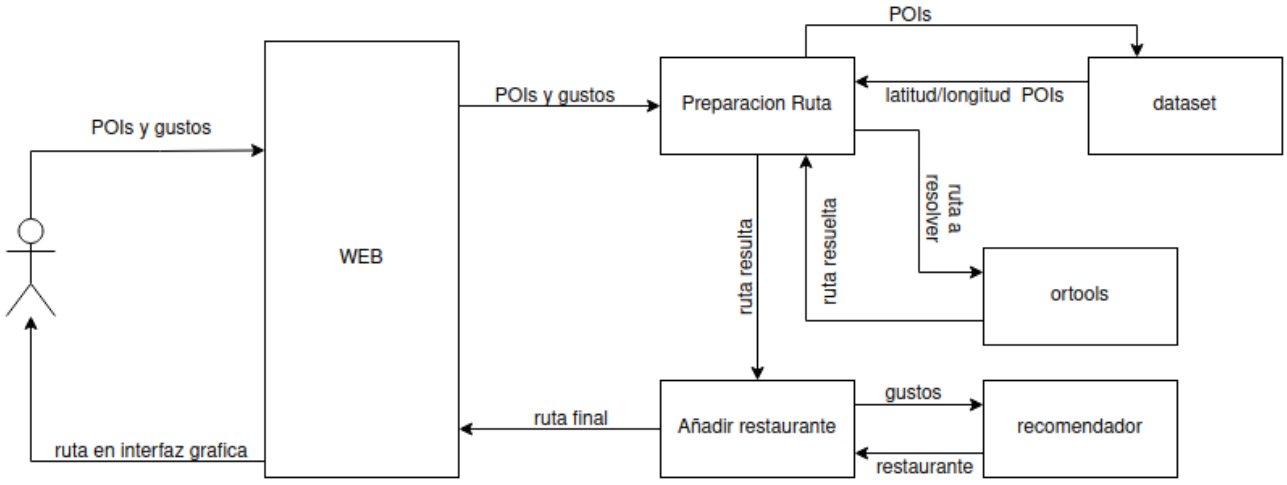

Figura 3.1: Estructura y flujo general del proyecto

Los módulos que conforman la aplicación son el preparador, que transforma los datos para el solucionador; el solucionador, que obtiene la ruta óptima con los datos del preparador; el recomendador, que contiene los modelos de recomendación y obtendrá el mejor restaurante

en el POI correspondiente; y el módulo Web, que contiene el front-end de la aplicación.

3.1.1. Requisitos funcionales

RF-1.– El modelo de recomendación debe ser basado en gustos.

RF-2.– La API deberá mostrar en una pantalla las diversas opciones de creación de ruta al usuario.

- **RF-2.1.–** Debe mostrar una lista con los nombres de los lugares para seleccionar el punto de inicio de la ruta.
- **RF-2.2.–** El usuario solo podrá escoger 1 opción de la lista obtenida a partir del requisito RF-2.1.
- **RF-2.3.–** Debe mostrar otra lista con los nombres de los lugares para visitar.
- **RF-2.4.–** El usuario tendrá la posibilidad de escoger múltiples opciones de la lista obtenida a partir del requisito RF-2.3.
- **RF-2.5.–** Debe mostrar otra lista con los tipos de restaurantes para seleccionar los gustos.
- **RF-2.6.–** El usuario tendrá la posibilidad de escoger múltiples opciones de la lista obtenida a partir del requisito RF-2.5.

RF-3.– Al terminar la selección, la API deberá mostrar un mapa interactivo con la ruta resaltada y los POI resaltados con marcadores que contendrán información al seleccionarlos.

RF-4.– La API deberá mostrar información acerca de su uso.

RF-5.– La interfaz tendrá 3 páginas:

- **RF-5.1.–** Una página principal de inicio con información acerca de la aplicación y el autor.
- **RF-5.2.–** Una página para crear las rutas con las opciones mencionadas en RF-2.
- **RF-5.3.–** Una página donde se muestre la documentación mencionada en RF-4.

3.1.2. Requisitos no funcionales

RNF-1.– El lenguaje de programación utilizado en el back-end será Python versión mínima 3.X para el uso de las librerías de Folium y ortools.

RNF-2.– El front-end se desarrollará en Django.

RNF-3.– La aplicación debe ser web-responsive para poder ser usada con cualquier dispositivo y resolución.

RNF-4.– El tiempo de respuesta debe ser menor a 3 segundos.

3.2. Diseño

En esta sección se detallará el diseño de la aplicación, incluyendo la estructura modular y la comunicación entre estos módulos.

Como se observa en la figura 3.1 el proyecto consiste en la unión de varios módulos que contienen funcionalidades distintas. Con este formato, se facilita la implementación y el posterior mantenimiento y añadido de funcionalidad. A grandes rasgos, el proyecto contiene cuatro

módulos importantes: preparador (”Preparación Ruta”), recomendador, solucionador (”ortools”) y Web.

3.2.1. Preparador

Este módulo contiene la lógica para transformar los datos obtenidos de la API en un formato que pueda pasarse al solucionador, accediendo al dataset que contiene toda la información de los POIs de la ciudad. Como puede verse en la figura 3.1, si se cambia el dataset de la ciudad, tenemos una aplicación diseñada para otra ciudad distinta sin necesidad de hacer modificaciones mayores.

Dentro de este módulo se encuentra un submódulo, ”generador ruta”, que a partir de una lista de nodos, creará mediante la librería Folium [18] un fichero .html con un mapa resaltando la ruta y el restaurante, que el módulo web cargará para enseñar el resultado final al usuario.

3.2.2. Solucionador

Este módulo contiene el solucionador del problema VRP descrito en 2.2. Mediante la librería ortools [5] que aplica el VRP y el sistema de recomendación basado en contenido 2.3 se genera una ruta de recorrido de POIs óptima que, mediante el recomendador, introduce el restaurante recomendado en la ruta teniendo en cuenta la hora de inicio de la ruta [23], la hora aproximada media en el que una persona come (13:00-15:00), la duración media de una ruta y el tiempo de duración de la visita a un POI [24] y la velocidad media de una persona [25].

Cabe destacar que la librería ortools tiene distintas aplicaciones a la solución del VRP, la primera que se probó fue Vehicle Routing Problem with Time Windows (VRPTW) o enrutamiento de vehículos con ventanas de tiempo, para tener en cuenta los horarios de los POIs y restaurantes; pero tras varias pruebas, debido a la falta de datos en el dataset utilizado y una comparativa de ambos modelos, se optó por el VRP básico.

3.2.3. Recomendador

Este módulo contiene los modelos de recomendación, actualmente solo tiene un modelo, que implementa el modelo de recomendación basado en contenido consciente de la semántica, definido en 2.3. Sería muy sencillo cambiar, gracias a la modularidad del proyecto, el

modelo de recomendación por cualquier otro de los mencionados en 2.3.

Otro modelo que podría haberse utilizado sería el modelo basado en la demografía, dado que en el ámbito del turismo se pueden crear grupos como visitantes de museo, visitantes de monumentos, etc. pero no se ha implementado debido a que es necesario establecer estos grupos mediante un sistema de usuarios, y este sistema de usuarios no ha sido implementado y será algo a tener en cuenta en el futuro (5.2).

3.2.4. Web

Este módulo contiene la implementación del front-end que, como se indica en los requisitos funcionales descritos en 3.1.1, se implementará en Django, por la facilidad de aumentar el tamaño de la aplicación y los conocimientos sobre este framework, motivos descritos en la sección 2.4

3.3. Implementación

Esta sección detallará la implementación de cada módulo, así como la interacción entre ellos mostrada en la figura 3.1.

3.3.1. Preparador

Mediante un conjunto de POIs obtenidos mediante un formulario en el módulo Web, obtendrá sus latitudes y longitudes del dataset y, como se detalla en la sección 3.3.2, mediante la fórmula de Haversine (ecuación 3.1) se obtendrá el tiempo que se tarda en llegar de un POI a otro; además, se añadirá un tiempo extra que será el tiempo medio que un turista se queda visitando un POI (aproximadamente de una hora, descrito en [24]), con esto se creará la matriz $N \times N$ que usará el solucionador de ortools para obtener la ruta óptima (detallado en 3.3.2).

Código 3.1: Función que transforma los POIs del dataset a una matriz de distancias.

```
distance_matrix = []
    for iIndex, iRow in muestra.iterrows():
        linea = []
        for jIndex, JRow in muestra.iterrows():
            linea.append(round(haversine.Haversine([iRow['Latitude'],iRow['Longitude']],[JRow['Latitude
                '],JRow['Longitude']]).meters))
        distance_matrix.append(linea)
```

Recordar que dentro de este módulo se encuentra el submódulo ”generador ruta”, que mediante la solución final del módulo ”solucionador” y la librería Folium [18] creará un mapa interactivo resaltando la ruta a seguir; marcando los POIs y el restaurante (código 3.2).

Código 3.2: Creacion del mapa interactivo mediante la libreria Folium [18]

```
m = folium.Map(location=loc[0],
               zoom_start=15)
folium.PolyLine(loc,
                color='red',
                weight=15,
                opacity=0.8).add_to(m)

for i, v in enumerate(solver[0][:-1]):
    folium.Marker(loc[i], popup="<i>{"+"{}".format(df.iloc[v]['Venue ID'])+"}</i>", tooltip=None).
        add_to(m)

text = "Tiempo total: {}h".format(tsp.objectiveValue)
folium.map.Marker(
    loc[0],
    icon=DivIcon(
        icon_size=(200,86),
        icon_anchor=(0,0),
        html='<div style="font-size: 24pt">{}</div>'.format(text),
        )
    ).add_to(m)

m.save("map.html")
```

3.3.2. Solucionador

Para poder utilizar la librería de ortools, se le debe pasar una matriz $N \times N$ con los POIs como índices (código 3.3) y como valores una unidad que los relacione, en nuestro caso será la distancia de Haversine (ecuación 3.1) donde φ es la latitud, λ la longitud y R el radio de la tierra (6,371 km); con la distancia de Haversine en metros.

Código 3.3: Función que recoge los datos de entrada del VRP [5]

```
def create_data_model():
    data = {}
    data['distance_matrix'] = [[A,B,C],
                               [A,B,C],
                               [A,B,C]]
    data['num_vehicles'] = 1
    data['depot'] = 0
    return data
```

$$d = 2r \cdot arcsin\left(\sqrt{sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + cos(\varphi_1)cos(\varphi_2)sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right) \tag{3.1}$$

Con la matriz de $N \times N$ el módulo solucionador que contiene la librería de ortools obtendrá la ruta óptima (código 3.5, recordemos que óptima en un problema VRP es aquella que recorre todos los nodos en la menor distancia/tiempo posible), y de ahí obtendremos el nodo en el cual el usuario esté a la hora de la comida (código 3.4), mencionado en el diseño del solucionador (sección 3.2.2), que se obtiene transformando las distancias obtenidas en metros a minutos mediante la velocidad media de una persona a pie [25].

Código 3.4: Código que obtiene el nodo donde estara el usuario a la hora de la comida, teniendo en cuenta la hora media de comida y la hora media de incio de ruta [23]

```
recorrido = [] # lista con tiempos de cada movimiento en minutos
for i in solver[0][:-1]:
    recorridoSingle = distance_matrix[solver[0][i]][solver[0][i+1]]
    recorridoTiempo = round((recorridoSingle/(5000))*60)
    recorrido.append(recorridoTiempo)

# devuelve lugar de la ruta que estariamos para la hora de la comida
horaComida = 14*60
horaInicio = 10*60
horaRecorrido = 0+horaInicio
rutaComida=0
for i, v in enumerate(recorrido):
    horaRecorrido+=v
    if(horaRecorrido >= 13*60 and horaComida <=15*60):
        rutaComida = i
        break
```

Código 3.5: Código de la libreria ORTOOLS para solucionar el problema VRP [5]

```
def VRP_solver(self):
    transit_callback_index = self.routing.RegisterTransitCallback(self.distance_callback)
    self.routing.SetArcCostEvaluatorOfAllVehicles(transit_callback_index)
    search_parameters = pywrapcp.DefaultRoutingSearchParameters()
    search_parameters.first_solution_strategy = (
        routing_enums_pb2.FirstSolutionStrategy.PATH_CHEAPEST_ARC)
    solution = self.routing.SolveWithParameters(search_parameters)

    if solution:
        self.objectiveValue = solution.ObjectiveValue()
        self.print_solution(solution)
        return self.get_routes(solution)
```

Esta hora de la comida es una generalización, y con un sistema de usuarios podría añadirse un campo personal para indicar esta hora, pero este sistema no ha sido añadido al proyecto (se hablará de ello en 5.2). Estos datos, además de los gustos culinarios del usuario, se pasarán al recomendador para obtener el restaurante en el que se le sugiere al usuario para comer.

Una vez obtenido el restaurante (ver Código 3.6), se añadirá a la ruta inmediatamente después del nodo que está dentro del intervalo de la hora de la comida y mandará la información al submódulo ”generador ruta” (dentro del módulo preparador) que generará una representación visual de la ruta.

Código 3.6: Código que obtiene el restaurante, aplicando el recomendador.

```
rastreo = {}
for i, row in restaurantes.iterrows():
    rastreo[i] = round(haversine.Haversine([row["Latitude"],row["Longitude"]],[latitudRastreo,
        longitudRastreo]).meters)

restaurantes1km =[]
for k, v in rastreo.items():
    if v <= 1000:
        restaurantes1km.append((k,v))

recomendacionIndex = None
for gusto in gustos:
    for restaurante in restaurantes1km:
        if restaurante[0] in restaurantes.index:
            if restaurantes.at[restaurante[0],'Venue␣category␣name'] == (gusto):
                recomendacionIndex = restaurante[0]
                break
if recomendacionIndex is None:
    recomendacionIndex = random.choice(restaurantes1km)
```

La primera implementación que se consideró utilizaba el VRPTW (VRP con restricciones

de ventanas temporales) pero no se pudo finalizar esa implementación debido a que faltaban los datos de horarios en el dataset; y no se consiguió encontrar uno que tuviera ese dato.

Código 3.7: Función que recoge los datos de entrada del VRPTW [5]

```
def create_data_model():
    data = {}
    data['distance_matrix'] = [[A,B,C],
                               [A,B,C],
                               [A,B,C]]
    data['time_windows'] = [[A,B,C],
                            [A,B,C],
                            [A,B,C]]
    data['num_vehicles'] = 1
    data['depot'] = 0
    return data
```

Como puede observarse, la entrada de datos es casi idéntica al código 3.3 utilizado, simplemente se añade otra matriz (”time_windows”) que contiene una unidad temporal. Esta implementación daría resultado a una ruta mucho más realista y restrictiva debido a que tendría que tener en cuenta el horario de los POIs a visitar, por lo que resultaría en una modificación del solucionador actual (ruta más corta) con posibles tiempos de espera, etc.

3.3.3. Recomendador

Una vez el módulo ”solucionador” ha detectado el nodo en el que el usuario estará durante el intervalo de la hora de la comida, el recomendador realiza un barrido en el dataset de todos los restaurantes que estén a 1 km a la redonda del nodo mediante la diferencia de las distancias de Haversine entre el nodo y el restaurante. Obtenidos todos los restaurantes a esa distancia y con los gustos culinarios del usuario obtenidos en un formulario HTML del módulo Web, se realizará una recomendación de similitud coseno (descrita en 2.3) sin cálculo de pesos, para ver si se puede obtener un restaurante que contenga el tipo de comida que el usuario desea. En caso de que no se encontrase, se recomendará un restaurante aleatorio dentro del área de 1 km del nodo elegido.

3.3.4. Web

Este módulo implementa todo el front-end de la aplicación. Tiene una página inicial y una página para la creación de rutas, ”Crear Ruta”, donde habrá un formulario HTML para selec-

cionar el origen de la ruta, los POIs que se desea visitar y los gustos culinarios del usuario. Estos datos se pasarán al preparador y recomendador para crear la ruta y, una vez generada la solución completa, cargará el fichero .html creado por el submódulo ”generador ruta” mediante la librería Folium y lo mostrará en pantalla (código 3.8).

Código 3.8: Código HTML para cargar el mapa creado por Folium

```
{% extends 'base.html' %}
{% load static %}
{% block body_block %}
<html>
  <body>
    <div>
        <object type="text/html" data="../static/templates/map.html"></object>
    </div>
  </body>
</html>
{% endblock %}
```

Las siguientes figuras (3.2, 3.3 y 3.5) muestran las pantallas web de la aplicación.

Figura 3.2: Página principal de la web

Figura 3.3: Página de la creación de rutas.

Figura 3.4: Página donde se muestra el mapa generado por Folium.

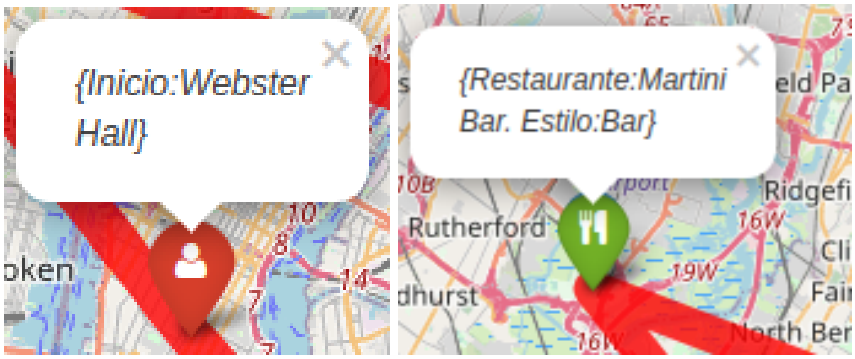

Figura 3.5: Información adicional al hacer click sobre los iconos.

4

Pruebas y resultados

En este capítulo se presentan qué pruebas y cómo se han preparado, así como el análisis de los resultados obtenidos. También se hará una breve validación de requisitos para comprobar que la aplicación funciona como se ha querido en un principio.

4.1. Validación de requisitos

La siguiente tabla presenta la matriz de trazabilidad del proyecto.

4.2. Pruebas

En esta sección se indicarán las características del entorno usado para las pruebas, así como qué dataset y cómo se ha refinado este mismo junto con las distintas pruebas realizadas.

4.2.1. Entorno para las pruebas

Las pruebas se han realizado en un ordenador con las siguientes características (Tabla 4.2):

4.2.2. Limpieza del dataset

El dataset utilizado para las pruebas es el dataset de Dingqi [26], que contiene check-ins de la ciudad de Nueva York. Todo el proceso de limpieza, al igual que la obtención de los restaurantes a 1 km, está detallado en 3.3.2. Se hará mediante la librería Pandas [14].

Este dataset está divido por las siguientes columnas: ”ID usuario” (anonimizado), ”ID del

Requisito	Validado en
RF-1	3.3.3
RF-2	3.3.4
RF-2.1	3.3.4
RF-2.2	4.2
RF-2.3	3.3.4
RF-2.4	4.2
RF-2.5	3.3.4
RF-2.6	4.2
RF-3	3.3.4
RF-4	3.3.4
RF-5	3.3.4
RF-5.1	3.3.4
RF-5.2	3.3.4
RF-5.3	No validado (se ha decidido que esta memoria sea la documentación)
RNF-1	4.2.1
RNF-2	4.2.1
RNF-3	4.2
RNF-4	4.2

Tabla 4.1: Matriz de trazabilidad de los requisitos del proyecto.

Elemento	Característica
OS	Ubuntu 20.04.2 LTS (Focal Fossa)
RAM	16GB DDR4 3200MHZ
CPU	AMD Ryzen 5 3500X 6-Core
GPU	GeForce GTX 1660 SUPER 6GB
Versión Python	3.8.10
Versión Django	2.2.12

Tabla 4.2: Características del ordenador utilizado para el desarrollo y pruebas.

```
Index(['User ID', 'Venue ID', 'Venue category ID', 'Venue category name',
       'Latitude', 'Longitude', 'Timezone offset in minutes', 'UTC time'],
      dtype='object')
   User ID                 Venue ID         Venue category ID  \
0      470  49bbd6c0f964a520f4531fe3  4bf58dd8d48988d127951735
1      979  4a43c0aef964a520c6a61fe3  4bf58dd8d48988d1df941735
2       69  4c5cc7b485a1e21e00d35711  4bf58dd8d48988d103941735
3      395  4bc7086715a7ef3bef9878da  4bf58dd8d48988d104941735
4       87  4cf2c5321d18a143951b5cec  4bf58dd8d48988d1cb941735

   Venue category name   Latitude  Longitude  Timezone offset in minutes  \
0  Arts & Crafts Store  40.719810 -74.002581                        -240
1               Bridge  40.606800 -74.044170                        -240
2      Home (private)  40.716162 -73.883070                        -240
3      Medical Center  40.745164 -73.982519                        -240
4           Food Truck  40.740104 -73.989658                        -240

                       UTC time
0  Tue Apr 03 18:00:09 +0000 2012
1  Tue Apr 03 18:00:25 +0000 2012
2  Tue Apr 03 18:02:24 +0000 2012
3  Tue Apr 03 18:02:41 +0000 2012
4  Tue Apr 03 18:03:00 +0000 2012
```

Figura 4.1: Columnas y datos originales del dataset.

lugar" (Foursquare), "ID de la categoría del lugar" (Foursquare), "Nombre de la categoría del lugar" (Fousquare), "Latitud", "Longitud", "Diferencia de zona horaria offset en minutos y Tiempo UTC".

Para empezar, las columnas ID usuario, ID de la categoría del lugar, Diferencia de zona horaria, offset en minutos y Tiempo UTC son columnas que no nos interesan; por lo que haremos un "pandas.DataFrame.drop" sobre estas columnas para eliminarlas del dataset, limpiando y mejorando la eficiencia del uso de este.

A continuación, habrá que transformar la columna "ID del lugar" para obtener el nombre del POI, para ello se ha realizado un pequeño programa adicional 4.1 que realiza un bucle que recorre el dataset entero que llama a la API de Foursquare [27] y sobreescribe la cadena actual (que es una cadena de números) por el nombre real del lugar.

Código 4.1: Funcion que sobreescribe los IDs de los lugares por su nombre real llamando a la API de Foursquare [27]

```
for index, row in dingqiNombres.iterrows():
    VENUE_ID = row['Venue ID']

    url = 'https://api.foursquare.com/v3/places/{}?fields=name&session_token=sesison'.format(
            VENUE_ID)

    headers = {

        "Accept": "application/json",

        "Authorization": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
        # por motivos de confidencialidad la clave API esta tapada por 'X's
    }

    response = requests.get(url, headers=headers)
    print(response.status_code)
    print(response.text)
    # si existe el sitio, reemplazar Venue ID
    if response.status_code == 200:
        res = json.loads(response.text)
        dingqiNombres.at[index,'Venue ID']='{}'.format(res['name'])
        dingqiNombres.to_csv('./transformer.csv')
    # si no existe el sitio, eliminar fila
    else:
        dingqiNombres = dingqiNombres.drop(index)
```

Con esta limpieza ya tendríamos un dataset funcional para el programa principal. El total de filas del dataset ha quedado finalmente como se puede ver en la Tabla 4.3.

Dataset original	Dasaset refinado
227.428	7.534

Tabla 4.3: Número de filas del dataset antes y después de la limpieza.

Se puede observar que se ha reducido enormemente la cantidad de filas del dataset, esto seguramente se deba a que durante la pandemia muchísimos sitios han cerrado por bancarrota.

4.2.3. Pruebas realizadas

Dado que esta aplicación se basa en la generación de rutas turísticas con restauración, la mejor forma de probarla ha sido ponerse en contacto con usuarios reales, mezclando usuarios no expertos, así como usuarios expertos para tener una mezcla de opiniones bien basada.

Pruebas con usuarios expertos

Se ha intentado poner en contacto con usuarios expertos que conozcan la ciudad utilizada para las pruebas (Nueva York), pero tras varias peticiones denegadas en foros de turismo de esta ciudad solo se ha podido encontrar un experto.

Las pruebas con experto han consistido en realizar varias rutas y que el experto diese su opinión al respecto, teniendo en cuenta las siguientes cuestiones: ¿es la ruta óptima (visita todos los puntos deseados de la forma más rápida)?, ¿el restaurante recomendado está acorde a los gustos seleccionados?, y ¿es el restaurante una buena recomendación, o recomendarías otro en la zona?.

4.2.4. Pruebas con usuarios comunes

Para estas pruebas, se ha puesto en contacto con personas ajenas al proyecto y se les ha pedido que realizasen la ruta que ellos deseasen y diesen su opinión respecto a esta con las siguientes cuestiones: ¿esta ruta recorre todos los lugares seleccionados?, ¿el restaurante recomendado está acorde a los gustos seleccionados?, ¿tomarías esta ruta si visitases la ciudad?.

A ambos grupos también se les hizo una pregunta adicional relacionada con la facilidad de uso, algo que es necesario en una aplicación de estas características.

4.3. Resultados

En esta sección se verán los resultados obtenidos de las pruebas realizadas a los usuarios expertos y comunes, y se analizará si la aplicación ha sido un éxito. Para cumplir con el artículo 72.p) ”Infracciones consideradas muy graves.” de la Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales [28] se han mantenido los usuarios tanto expertos como comunes en la anonimidad.

4.3.1. Resultados de las pruebas con usuarios expertos

Usuario experto

El usuario experto es una persona nacida en Manhattan que encontré por un foro de internet, y me ha ayudado a realizar una ruta que recorre varios sitios típicos de Nueva York (las respuestas a las preguntas están transcritas y traducidas del inglés). Para la comida ha escogido todo lo relacionado con la comida basura americana para probar los platos típicos (hamburguesa, alitas, etc.).

Inicio: Times Square.

Lugares a visitar: Madison Square Garden, Rockefeller Center, Memorial park, Metropolitan Museum of Art, Central Park.

Gustos culinarios: Food Truck, American Restaurant, Wings Joint, Hot Dog Joint, Fried Chicken Joint.

Figura 4.2: Resultado del usuario experto.

¿es la ruta óptima (visita todos los puntos deseados de la forma más rápida)? Sí, aunque

sería mejor coger el metro para ir a Memorial park.

¿el restaurante recomendado está acorde a los gustos seleccionados? Sí.

¿es el restaurante una buena recomendación, o recomendarías otro en la zona? El sitio seleccionado es un lugar típico de Nueva York porque es un sitio aparentemente sucio, pero tiene un pollo de calidad con una receta clásica; por lo que es recomendable.

4.3.2. Resultados de las pruebas con usuarios comunes

Usuario común 1

Al usuario común 1 le gusta visitar zonas con naturaleza y ver a la gente local desarrollar su día a día; por ello ha escogido varios parques. Para sus gustos culinarios no tiene preferencia, por lo que ha escogido un poco de todo.

Inicio: Webster Hall.

Lugares a visitar: Madison Square Park, Washington Square Park, St.Patrick's Cathedral, Fulton Park, Mateos Park.

Gustos culinarios: Burger Joint, Spanish Restaurant, Asian Restaurant, Sushi Restaurant

¿Esta ruta recorre todos los lugares seleccionados? Sí.

¿El restaurante recomendado está acorde a los gustos seleccionados? Sí.

¿Tomarías esta ruta si visitases la ciudad? Sí.

Usuario comun 2

El usuario común 2 es creyente y le gustan las iglesias, por lo que ha seleccionado varias de estas. Al ser una persona suramericana ha seleccionado burritos y arepas como restaurante principal, además de los otros dos como opciones adicionales.

Inicio: St. Patrick's Cathedral.

Lugares a visitar: Trinity Church Cemetery, First Presbyterian Church, Saint Thomas Church, New Hope Lutheran Church.

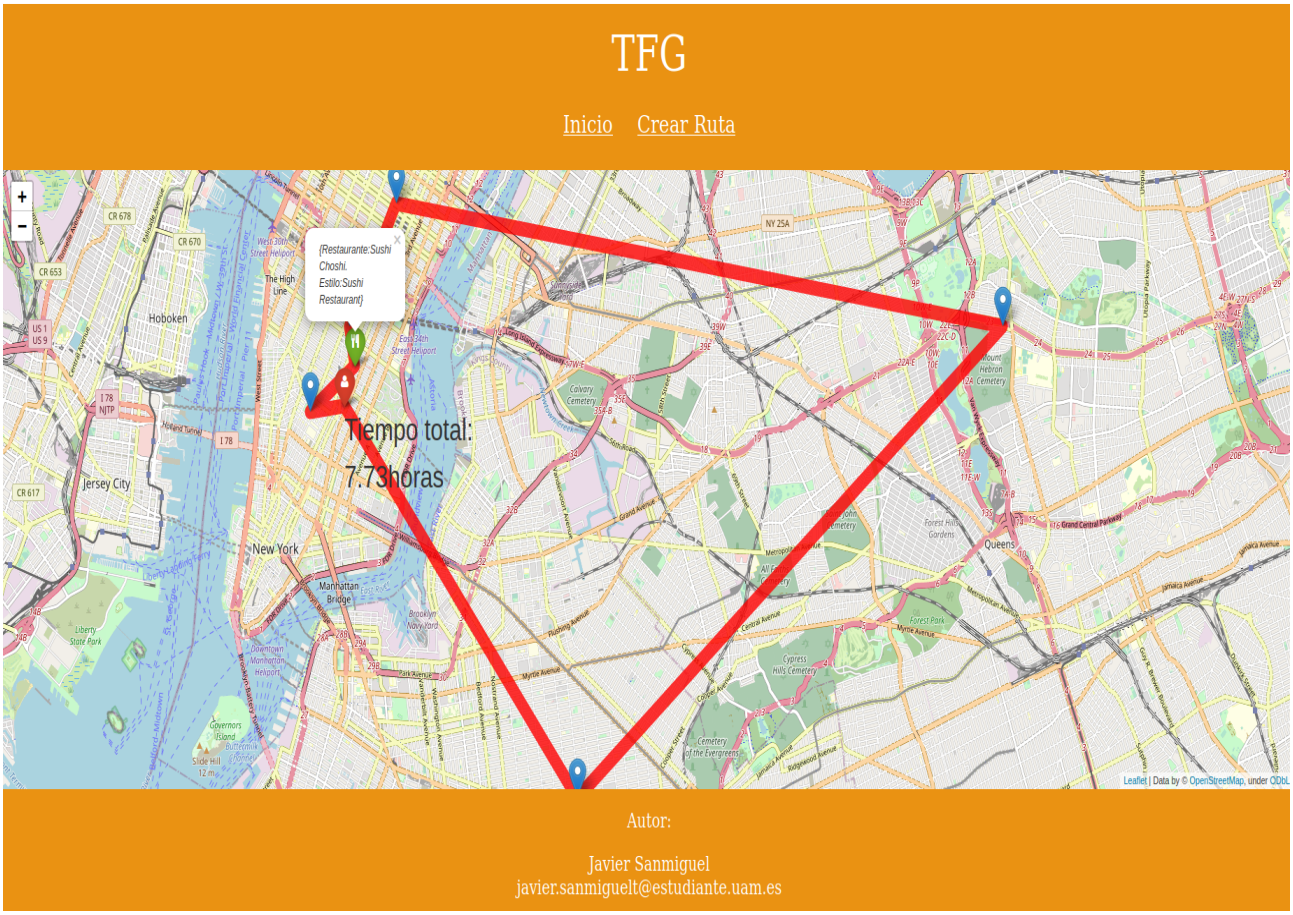

Figura 4.3: Resultado del usuario común 1.

Gustos culinarios: Pizza Place, BBQ Joint, Burrito Place, Arepa Restaurant.

Figura 4.4: Resultado del usuario común 2.

¿Esta ruta recorre todos los lugares seleccionados? Sí.

¿El restaurante recomendado está acorde a los gustos seleccionados? Sí.

¿Tomarías esta ruta si visitases la ciudad? Sí.

Usuario común 3

Al usuario tres le gusta el arte y la cultura, por lo que ha seleccionado varios museos y galerías de arte. El usuario es vegetariano, por lo que ha escogido aquellos sitios que pueden asegurar ese tipo de comida.

Inicio: Thomas Kinkade Gallery

Lugares a visitar: Museum of Arts & Design, New York Transit Museum Annex, Syndicate Gallery, Art In FLUX Harlem.

Gustos culinarios: Salad Place, Vegetarian / Vegan Restaurant.

Figura 4.5: Resultado del usuario común 3.

¿Esta ruta recorre todos los lugares seleccionados? Sí.

¿El restaurante recomendado está acorde a los gustos seleccionados? Sí.

¿Tomarías esta ruta si visitases la ciudad? Sí.

4.3.3. Análisis general de los resultados

En general, todos los usuarios están contentos con la ruta generada. Al ser personas no familiarizadas con Nueva York, no han podido dar respuestas muy específicas. Pero cabe destacar al usuario experto que ha nacido y vive en la zona y ha dado una nota positiva a la ruta, por lo tanto, aunque el número de pruebas es bajo, creo que sirve como un comienzo exitoso para el proyecto. En la tabla 4.4 se agrupan las respuestas para una mejor visibilidad.

Respecto a la facilidad de uso, todos respondieron lo mismo. El selector de lugares a visitar

debería ser algún tipo de buscador o, al menos, que la caja de selección fuera más grande; esto se puede hacer con un script en Javascript o cambiando el fichero HTML un poco, pero no se ha hecho por falta de tiempo. Otra sugerencia recibida fue que los marcadores deberían tener indicado en el símbolo el número del recorrido, este no ha podido realizarse dado que en la base de símbolos de Font Awesome no se encontraron números.

Pregunta	Positivo	Negativo
¿Esta ruta recorre todos los lugares seleccionados?	4	0
¿El restaurante recomendado está acorde a los gustos seleccionados?	4	0
¿Tomarías esta ruta si visitases la ciudad?	4	0
¿Es una aplicación intuitiva de usar?	1	3

Tabla 4.4: Agrupación de los resultados.

5

Conclusiones e implementaciones futuras

En este capítulo se hará un análisis de las conclusiones obtenidas con el desarrollo de este proyecto, así como las posibles modificaciones que podrían realizarse para mejorar el proyecto.

5.1. Conclusiones

El turismo es un motor económico muy potente en la mayoría de países, ahora que la pandemia que hemos sufrido durante dos años se ha calmado, la gente va a querer volver a visitar lugares ajenos a la zona donde han tenido que estar dos años; por ello es necesario ofrecer una forma intuitiva y eficaz para crear rutas turísticas. Este proyecto ha tratado de saciar esa necesidad creando una aplicación capaz de ofrecer rutas turísticas con restauración aplicando los sistemas de recomendación.

Tras estudiar los distintos modelos de recomendación, se ha optado por los recomendadores basados en el contenido. Los recomendadores basados en el contenido se nutren únicamente de la información proporcionada por el usuario. Dentro de estos recomendadores se encuentran aquellos que tienen en cuenta la semántica, que tienen en cuenta la semántica de la información para realizar una similitud próxima a aquel que es semánticamente similar.

Para realizar las rutas, el solucionador ortools es el más apropiado dado que es el utilizado por Google en su aplicación Google Maps, una de las mejores aplicaciones de rutas existentes, por lo que ha facilitado enormemente la implementación al no ser necesario implementar un algoritmo que solucione problemas VRP de cero.

Dado que el objetivo del proyecto era crear una ruta que el usuario pudiera entender fácilmente, se ha creado una interfaz acorde para observar la susodicha ruta con el mapa para

contextualizar la ruta en la ciudad visitada.

Todo el código implementado se puede ver aquí:

https://github.com/JSanmigueIt/TFG.git

5.2. Implementaciones futuras

Durante el desarrollo del proyecto, se han visto áreas que podrían mejorarse con más tiempo y recursos. Dado que este proyecto tiene mucho potencial, el diseño se ha modularizado para que estas mejoras puedan implementarse de la manera más fácil posible.

La mejora más inmediata puede ser la traducción del dataset a una base de datos relacional con las distintas ciudades y dentro de cada tabla los datos que contiene el dataset, esto mejoraría los tiempos de respuesta. Esto ha sido uno de los motivos para haberse implementado con el framework de Django en vez de Flask, dado que Django está diseñado para tener bases de datos.

Otra mejora puede ser la implementación de sistemas de recomendación más refinados, aunque se ha usado un sistema de recomendación basado en el contenido sensible a la semántica, esta ha sido una implementación básica que podría mejorarse usando las partes más complejas de este tipo de recomendadores. No ha sido posible inicialmente porque el dataset utilizado no disponía de calificaciones de los sitios y, por tanto, no se ha podido aplicar la fórmula de los pesos; para poder obtener dichas calificaciones habría sido necesario acceso ilimitado a la API de Foursquare que se pidió en su momento, pero no fue concedido.

En relación con los sistemas de recomendación, una última mejora podría haber sido la implementación de un sistema de usuarios para tener un historial de las rutas para obtener qué tipos de POIs le interesan al usuario para implementar un recomendador demográfico; una lista modificable con los gustos culinarios del usuario para no tener que estar pidiéndolos continuamente en la generación de la ruta; y un campo para indicar el intervalo de la hora de la comida y salida.

También se podría crear de cero un solucionador del problema VRP en vez de usar el de la librería ortools, más centrado en la recomendación del restaurante en vez de en la optimalidad de la ruta para priorizar que el usuario coma en un restaurante que le guste en vez de recorrer rápidamente la ruta. Para ello, una posible implementación sería mirar primero los nodos y

obtener los restaurantes para aplicar la similitud coseno y, una vez obtenido el restaurante, generar la ruta para que el nodo cercano al restaurante este dentro del tramo horario de las 13:00 a las 15:00. Otra implementación del VRP, si se tuviesen los datos necesarios, sería implementar el VRPTW mencionado, lo cual daría lugar a unas rutas más realistas dado que tienen en cuenta el horario de apertura y cierre de los POIs.

Creo que estas mejoras resultarían en una aplicación profesional que podría ser usada por todo el mundo y rivalizar a las grandes aplicaciones que se usan hoy en día como Roadtrippers o Google Maps.

6

Acrónimos

LBSN

Location-Based Social Networks/Redes sociales basadas en la localización

POI/PDI

Point of Interest/Punto De Interés

TSP

Traveling Salesperson Problem/Problema del viajante

VRP

Vehicle Routing Problem/Problema de enrutamiento de vehículos

VSM

Vector Space Model/Modelo Espacio-Vectorial

Bibliografía

[1] "Evolución de los ingresos por turismo internacional en el mundo entre 1990 y 2020." https://es.statista.com/estadisticas/633163/gasto-de-los-turistas-internacionales-en-el-mundo/. Accessed: 25-05-2022.

[2] L. W. Yeow, R. Low, Y. X. Tan, and L. Cheah, "Point-of-interest (poi) data validation methods: An urban case study," *ISPRS International Journal of Geo-Information*, vol. 10, no. 11, 2021.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009.

[4] T. Caric and H. Gold, *Vehicle Routing Problem*. Rijeka: IntechOpen, 2008.

[5] "Ortools." https://developers.google.com/optimization/. Accessed: 25-05-2022.

[6] F. Ricci, L. Rokach, and B. Shapira, eds., *Recommender Systems Handbook*. Springer, 2015.

[7] F. Ricci, L. Rokach, and B. Shapira, "Recommender systems: Introduction and challenges," in *Recommender Systems Handbook* (F. Ricci, L. Rokach, and B. Shapira, eds.), pp. 1–34, Springer, 2015.

[8] Y. Pérez-Almaguer, N. Martín-Dueñas, E. Carballo-Cruz, and R. Yera, "Una revisión de los sistemas recomendadores grupales como herramienta innovadora en el área del turismo," *Revista de Ciencia y Tecnología*, vol. 35, p. 44–53, ago. 2021.

[9] M. de Gemmis, P. Lops, C. Musto, F. Narducci, and G. Semeraro, "Semantics-aware content-based recommender systems," in *Recommender Systems Handbook* (F. Ricci, L. Rokach, and B. Shapira, eds.), pp. 119–159, Springer, 2015.

[10] P. Sánchez and A. Bellogín, "Applying reranking strategies to route recommendation using sequence-aware evaluation," *User Model. User Adapt. Interact.*, vol. 30, no. 4, pp. 659–725, 2020.

[11] M. Sumardi, Jufery, Frenky, R. Wongso, and F. A. Luwinda, ""tripbuddy" travel planner with recommendation based on user's browsing behaviour," *Procedia Computer Science*, vol. 116, pp. 326–333, 2017. Discovery and innovation of computer science technology

in artificial intelligence era: The 2nd International Conference on Computer Science and Computational Intelligence (ICCSCI 2017).

[12] R. Roy and L. W. Dietz, "Triprec - A recommender system for planning composite city trips based on travel mobility analysis," in *Proceedings of the Workshop on Web Tourism co-located with the 14th ACM International WSDM Conference (WSDM 2021), Jerusalem, Israel, March 12, 2021* (C. Barbu, L. Coba, A. Delic, D. Goldenberg, T. Kuflik, M. Zanker, and J. Neidhardt, eds.), vol. 2855 of *CEUR Workshop Proceedings*, pp. 8–12, CEUR-WS.org, 2021.

[13] K. A. Fararni, F. Nafis, B. Aghoutane, A. Yahyaouy, J. Riffi, and A. Sabri, "Hybrid recommender system for tourism based on big data and ai: A conceptual framework," *Big Data Mining and Analytics*, vol. 4, no. 1, pp. 47–55, 2021.

[14] "Pandas." https://pandas.pydata.org/. Accessed: 25-05-2022.

[15] "Numpy." https://numpy.org/. Accessed: 25-05-2022.

[16] "Django." https://www.djangoproject.com/. Accessed: 25-05-2022.

[17] "Flask." https://flask.palletsprojects.com/en/2.1.x/foreword/. Accessed: 25-05-2022.

[18] "Folium." https://python-visualization.github.io/folium/. Accessed: 25-05-2022.

[19] "Geopandas." https://geopandas.org/en/stable/. Accessed: 25-05-2022.

[20] "Leafletjs." https://leafletjs.com/. Accessed: 25-05-2022.

[21] "Fiona." https://fiona.readthedocs.io/en/latest/. Accessed: 25-05-2022.

[22] "Matplotlib." matplotlib.org. Accessed: 25-05-2022.

[23] B. Mckercher, N. Shoval, E. Ng, and A. Birenboim, "First and repeat visitor behaviour: Gps tracking and gis analysis in hong kong," *Tourism Geographies*, vol. 14, pp. 147–161, 02 2012.

[24] K. Taylor, K. H. Lim, and J. Chan, "Travel itinerary recommendations with must-see points-of-interest," pp. 1198–1205, 04 2018.

[25] E. M. Murtagh, J. L. Mair, E. Aguiar, C. Tudor-Locke, and M. H. Murphy, "Outdoor walking speeds of apparently healthy adults: A systematic review and meta-analysis," *Sports Med.*, vol. 51, pp. 125–141, Jan. 2021.

[26] "Dingqi yang's foursquare dataset." https://sites.google.com/site/yangdingqi/home/foursquare-dataset. Accessed: 25-05-2022.

[27] "Dingqi yang's foursquare dataset." https://developer.foursquare.com/. Ac-

cessed: 25-05-2022.

[28] A. E. B. O. del Estado and J. del Estado, *Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales*. BOE, 2018.


UÁM
Universidad Autónoma
de Madrid