Universidad Autónoma de Madrid
Escuela politécnica superior

Grado en Ingeniería Informática

Trabajo Fin de Grado

VISUALIZACIÓN DE RUTAS TURÍSTICAS USANDO HERRAMIENTAS OPEN SOURCE

Autor: Miguel Andrés Russián Rojas
Tutor: Alejandro Bellogín Kouki
Ponente: Fernando Díez Rubio

Febrero 2018

VISUALIZACIÓN DE RUTAS TURÍSTICAS USANDO HERRAMIENTAS OPEN SOURCE

Autor: Miguel Andrés Russián Rojas
Tutor: Alejandro Bellogín Kouki
Ponente: Fernando Díez Rubio

Dpto. de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Febrero 2018

Resumen

Resumen

A pesar de que la demanda de turismo así como de las recomendaciones basadas en las preferencias de viaje de cada usuario está incrementando, y aún teniendo cada vez más datos al respecto, las herramientas para entender los patrones de viaje de los usuarios y sus motivos todavía no han madurado. Con este proyecto proporcionamos lo necesario para analizar, extraer, y visualizar trayectorias de viaje de historiales de usuarios, detectadas de manera configurable. Además, mostramos y comparamos algunos algoritmos de detección básicos con lo que remarcamos la posibilidad de extraer rutas de usuarios que mantienen una temática común.

Para cumplir con lo propuesto primero hemos examinado las tecnologías, conocimiento científico y herramientas disponibles en la actualidad, teniendo en cuenta que la definición de una ruta es, al fin y al cabo, relativa. A partir de aquello educimos algunas perspectivas y consideraciones de diseño, que plasmamos en requisitos del producto final, y posteriormente construimos sobre ese conocimiento y software abierto al público un conjunto de herramientas fáciles de usar y configurables, que crea una transición clara de bases de datos de ubicaciones a trayectorias de usuarios manipulables interactivamente. Más tarde, comprobamos que los requisitos extraídos con anterioridad son satisfechos por el software desarrollado, y, finalmente, lo aprovechamos como un sistema completo para mostrar su potencial para segmentar un histórico de rutas.

La información de dónde se encuentra un usuario en un momento dado es relativamente sencilla de obtener, normalmente con su consentimiento, lo cual unido al hecho de que cada vez más personas utilizan un dispositivo inteligente, que está asiduamente conectado a Internet y puede dar información muy precisa de su ubicación, hace la combinación de lugar y tiempo aplicada a un usuario un recurso interesante y con potencial para ser aprovechado. Aunque otros datos como los gustos de un usuario y su actividad en redes sociales han demostrado permitir obtener buenos resultados en términos de recomendaciones de viaje, esta información no está disponible para aplicaciones finales y, por tanto, no ha sido considerada en este trabajo.

Usando un conocimiento muy limitado de los usuarios (que ni siquiera incluía su red de contactos), algoritmos de filtrado sencillos sobre una variable y unas pocas hipótesis (como que el usuario duerme 8 horas al día o que no camina más de unos cuantos kilómetros al día), nuestra investigación muestra que información útil como medios de transporte y densidad de puntos de interés puede ser aprovechada para deducir si el usuario estaba viajando como un turista y si puntos de interés específicos estaban unidos en una misma ruta.

Palabras Clave

Visualización, mapas, turismo, detección de rutas.

Abstract

Tourism and targeted recommendations based on tourist's travel interests are increasing in demand, but despite the growing amounts of user data related to it, mechanisms to understand users travel routes and their reasons are yet under-developed. In this work we provide the tools to analyze, extract, and visualize travel paths from user's history in a configurable way. Furthermore, we showcase and compare a few basic detection algorithms to underscore the possibility of extracting semantically-cohesive travel routes.

In order to do so, we first examine the technologies, research and tools available, bearing in mind that travel routes are ultimately subjectively defined. From these we create design perspectives and considerations, materialized in product requirements, and directly build upon open knowledge and developments to implement a software toolkit that is easy to use and configure and defines a clear workflow from raw geo-temporal data to interactive travel history visualization. Afterwards, requisites previously extracted are compared with the implemented product's functionality, verifying its compliance, and finally test the latter as a whole to demonstrate its travel history segmentation potential.

Information about user's location at a given time is relatively simple to obtain, normally provided they consent. Adding this to the fact that more people are using their smart devices, which are constantly connected to the Internet and can give very precise information about the user's whereabouts makes the combination of geographical coordinates and time bound to a specific user an interesting resource to take advantage of. While other resources such as tastes and social media activity has been shown to have good results as far as travel recommendation is concerned, they are not as available to applications, and so have not been considered in this work.

Despite having limited knowledge about the users (among which their contact network is disregarded) and using simple filtering algorithms that only consider a single variable and a few assumptions (such as the user sleeping 8 hours a day or not being able to walk more than a few kilometers in one day), our analysis shows that useful information such as means of transport and point of interest density could be leveraged to infer if the user was traveling as a tourist and whether specific points of interest were related in a single tour.

Key words

Visualization, maps, tourism, route detection.

Agradecimientos

A mi familia, y a mi tutor.

Índice general

Índice de Figuras IX

Índice de Tablas XI

1. Introducción 1

1.1. Motivación del proyecto . . . 1

1.2. Objetivos y enfoque . . . 2

1.3. Organización de la memoria . . . 2

2. Estado del arte y de las tecnologías 5

2.1. Sistemas de ubicación con fines sociales . . . 5

2.2. Detección de rutas . . . 7

2.3. Visualización de rutas . . . 8

2.4. Conclusiones . . . 9

2.4.1. Recorridos de usuarios . . . 9

2.4.2. Reconocimiento de rutas . . . 9

2.4.3. Visualización . . . 10

3. Diseño 11

3.1. Sistema completo . . . 11

3.1.1. Estructura general . . . 11

3.1.2. Objetivos . . . 12

3.1.3. Requisitos funcionales y no funcionales . . . 13

3.2. (Sub)sistema de reconocimiento de rutas . . . 14

3.2.1. Captación de datos . . . 14

3.2.2. Reconocimiento de rutas . . . 15

3.3. (Sub)sistema de visualización de rutas . . . 15

3.3.1. Empaquetado de rutas para visualización . . . 15

3.3.2. Visualización de rutas . . . 16

4. Implementación **17**
4.1. Detección de rutas . . . 17
4.1.1. Punto de entrada . . . 17
4.1.2. Preprocesamiento . . . 17
4.1.3. Base de datos . . . 18
4.1.4. Filtros . . . 18
4.1.5. Gráficos . . . 18
4.1.6. Rutas . . . 19
4.2. Visualización de rutas . . . 19
4.2.1. Librerías escogidas . . . 19
4.2.2. Fuentes de mapa base . . . 19
4.2.3. Barra lateral . . . 19
4.2.4. Barra de categorías . . . 20

5. Experimentos realizados y resultados **21**
5.1. Validación de requisitos . . . 21
5.1.1. Sistemas probados . . . 22
5.1.2. Procesamiento de rutas . . . 22
5.1.3. Visualización de rutas . . . 24
5.2. Comparativa de algoritmos de separación de rutas . . . 27
5.2.1. Umbrales fijos . . . 29
5.2.2. Umbrales relativos . . . 31
5.3. Experimentos del sistema completo . . . 33

6. Conclusiones y trabajo futuro **35**
6.1. Conclusiones . . . 35
6.2. Trabajo futuro . . . 35

Glosario de acrónimos **37**

Bibliografía **38**

A. Manual de utilización **43**
A.1. Sistema de generación de rutas . . . 43
A.1.1. Prerrequisitos . . . 43
A.1.2. Petición de filtrado . . . 43
A.1.3. Ejecución . . . 44
A.2. Sistema de visualización . . . 44
A.2.1. Prerrequisitos . . . 44
A.2.2. Ejecución . . . 45

Índice de Figuras

1.1. Vuelos entrantes a países de la Unión Europea en 2017 (desde la UE) 1

3.1. Componentes del sistema completo 11
3.2. Casos de uso del proyecto 13

5.1. Extracto estado del árbol de directorios tras el filtrado del dataset de BrightKite. 23
5.2. Superposición de tiempo (segundos) entre check-ins sucesivos. 23
5.3. Visualización de mapas - rutas de prueba. 24
5.4. Visualización de mapas - una ruta eliminada. 25
5.5. Visualización de mapas - rutas de prueba eliminadas. 25
5.6. Visualización de mapas - dos ficheros de prueba. 25
5.7. Visualización de mapas - rutas coloreadas. 26
5.8. Visualización de mapas - rutas coloreadas y algunas eliminadas. 26
5.9. Visualización de mapas - error al cargar. 26
5.10. Visualización de mapas - fichero con rutas sin coordenadas. 27
5.11. Visualización de mapas - categorías de puntos de interés sin rutas. 27
5.12. Visualización de mapas - rutas y algunos puntos de interés con categorías. 28
5.13. Visualización de mapas - rutas con una categoría eliminada. 28
5.14. Visualización de mapas - categoría cambiada de color. 28
5.15. Visualización de mapas - categorías con una ruta eliminada. 29
5.16. Dataset Gowalla - usuario con ID 777. 30
5.17. Dataset Gowalla - usuario con ID 18932. 30
5.18. Dataset Gowalla - usuario con ID 621. 30
5.19. Dataset Gowalla - usuario con ID 38496. 31
5.20. Dataset Gowalla - usuario con ID 4915. 31
5.21. Dataset Gowalla - usuario con ID 444. 32
5.22. Dataset Gowalla - usuario con ID 18932. 33
5.23. Dataset Gowalla - usuario con ID 777 (atípico). 33
5.24. Dataset Gowalla - usuario con ID 10972. 33

5.25. Visualización de un dataset pequeño. 34
5.26. Visualización del dataset Gowalla. 34

Índice de Tablas

1.1. Turismo de residentes en España 1

5.1. Matriz de trazabilidad de requisitos y pruebas 21

Capítulo 1

Introducción

1.1. Motivación del proyecto

España es un país en el que el turismo está muy presente, tanto para consumir servicios como para ofrecerlos, dentro y fuera del país. En primer lugar, veamos algunos datos de los viajeros por motivos personales (2016) de la Encuesta de turismo de residentes del Instituto Nacional de Estadística [1]:

Descripción	Dato
Residentes en España que viajan	65.7 %
Proporción que viaja sólo por España	68.7 %
Proporción que viaja por España y el extranjero	23.3 %
Principales motivos para no viajar	Financieros (46.8 %)

Cuadro 1.1: Turismo de residentes en España

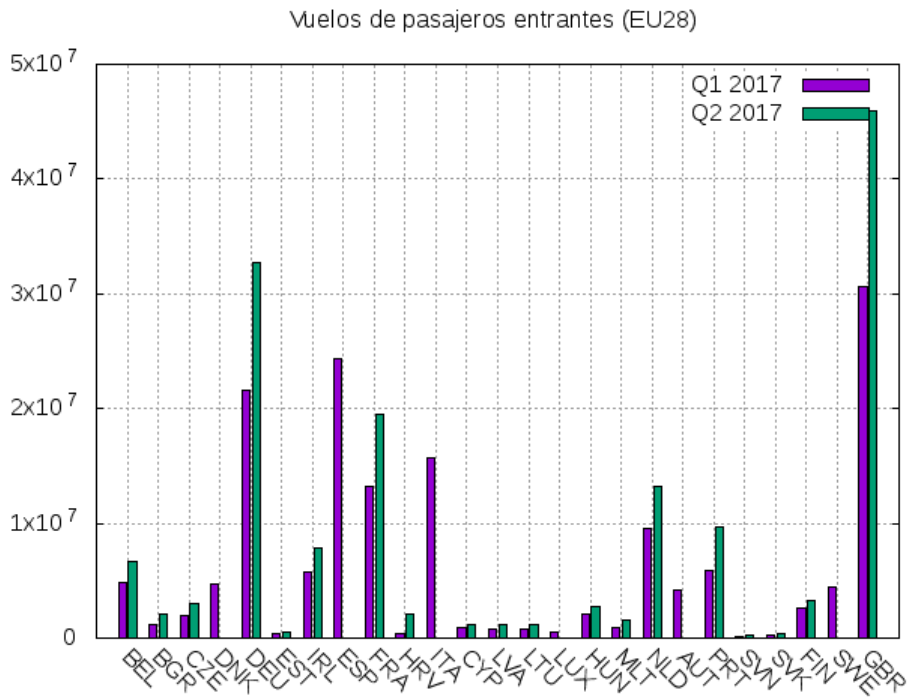


Figura 1.1: Vuelos entrantes a países de la Unión Europea en 2017 (desde la UE)

En segundo lugar, veamos el número de vuelos entrantes a países de la Unión Europea en la primera mitad del 2017. Como se puede ver en la Figura 1.1 (donde Q1 es el primer trimestre del 2017 (Enero, Febrero, Marzo) y el Q2 el segundo, según la información sacada de [2]), España está entre los principales destinos turísticos en la UE en acceso aéreo, lo cual pone al país entre los primeros 10 destinos del mundo según la OMT [3]. Además, hay que considerar que, si bien el turístico no es el único motivo para volar, tiene la mayoría absoluta como método preferido para viajar fuera del país (53.3 %).

En cuanto a las motivaciones para viajar, son bastante diversas. En Europa en 2015, el 89.3 % de las personas viajó por motivos personales, y casi la mitad (48.5 %) del total fue por vacaciones y entretenimiento principalmente. La plataforma de reservas de hospedaje Airbnb, que tiene una sección de «Experiencias» donde los usuarios locales publican planes de actividades turísticas, afirma que entre las 10 categorías más populares entre el 17 de Octubre y el 16 de Noviembre del 2017, ninguna abarcó más del 30 % de las reservas; entre éstas figuran deportes, gastronomía, música, naturaleza, moda, etc. [4].

Además, existe una marcada preferencia por encontrar planes turísticos en Internet (a dónde ir, qué hacer y dónde hospedarse) y que sean personalizados. Un estudio, también en 2015, de American Express nos revela que el 85 % de los estadounidenses (que viajaron al menos dos veces en los últimos 5 años) prefiere una experiencia personalizada, que 4 de cada 5 prefiere hacer sus reservas y planes por Internet, y que casi dos tercios está de acuerdo en que las compañías aprovechen datos sobre sus gastos en viajes por placer para personalizar su experiencia de viaje; estas proporciones aumentan monótonamente cuanto más joven es el grupo de edad [5].

Todo esto nos conduce a la necesidad de estudiar los recorridos turísticos y representarlos por medios informáticos, en particular un entorno web, para que futuros turistas puedan ver, compartir y encontrar recomendaciones sobre sus preferencias de turismo, bien por afinidad con otros internautas o por servicios especializdos.

1.2. Objetivos y enfoque

Este proyecto tiene dos objetivos principales: desarrollar una herramienta capaz de extraer rutas (recorridos de un usuario cuyos puntos de interés están relacionados), y otra que pueda mostrar dichas rutas de manera gráfica e interactiva. Además, estamos interesados en el escenario en el que estos dos objetivos están interconectados, de manera que la salida de la primera herramienta (una ruta concreta) pueda ser visualizada por la segunda herramienta.

Se ha hecho hincapié en usar tecnologías Open Source porque tienen menor esfuerzo de mantenimiento y para un tecero resulta más fácil adaptar sus productos para integrar con los de este proyecto (para mejorarlos y añadir funcionalidad complementaria).

1.3. Organización de la memoria

Para facilitar la lectura se ha estructurado este documento en capítulos, cada uno construyendo sobre los anteriores para terminar en los frutos de este trabajo. Los capítulos son los siguientes:

- Introducción: Este capítulo, donde ya se ha podido apreciar por qué este proyecto es interesante, qué necesidades pretende satisfacer y, a grandes rasgos, cómo.
- Estado del arte y de las tecnologías: Antes de cualquier desarrollo, es conveniente conocer las tecnologías y desarrollos existentes de modo que este trabajo aporte algo nuevo que

mejore lo preexistente. Por este motivo se realizará un estudio del status quo tecnológico, en lo relevante a los objetivos del trabajo previamente descritos.

- Diseño: Una vez conocidas las herramientas a utilizar, proseguiremos a formalizar los objetivos del proyecto y a diseñar estos productos.
- Implementación: Partiendo de los diseños analizados para cumplir con los dos objetivos principales de este proyecto, en este capítulo detallaremos cómo se desarrollaron y con qué tecnologías.
- Experimentos realizados y Resultados: Terminados los productos, y de modo que podamos comprobar en qué medida cumplen los objetivos del proyecto, realizaremos una serie de pruebas; en este capítulo se detallan estas pruebas y los resultados obtenidos.
- Conclusiones y trabajo futuro: Consultando los resultados de las pruebas, analizaremos los productos del trabajo y sus posibles mejoras.

Capítulo 2

Estado del arte y de las tecnologías

En este trabajo nos serviremos principalmente de herramientas para, primero, detectar rutas turísticas y, luego, interactuar con ellas. Para ello, en este capítulo exploramos distintas tecnologías y herramientas que nos servirán para estas dos tareas.

Si buscamos determinar, o al menos aproximarnos, al recorrido que siguen las personas y su motivación para ello, resulta de gran ayuda conocer de antemano los puntos de interés que existen en su entorno geográfico, es decir, aquellas ubicaciones que tienen mucho potencial de ser objetivos del recorrido turístico.

Como bien sabemos, la mejora de los dispositivos PC (incluidos no solo ordenadores de sobremesa y portátiles, sino también smartphones) y su uso más difundido (ya en 2015 la mitad de la población tenía un smartphone, mientras que en 2006 no estaba disponbile al público [6]) ligado a la mayor accesibilidad del Internet, ha catalizado la proliferación de una gran variedad de aplicaciones y servicios, desde las más lúdicas como los juegos sociales, hasta los de vital importancia como la telemedicina.

Uno de los servicios más populares que resultaron de la cooperación del PC y el Internet es, sin duda, el de las redes sociales, algunas grandes pero otras más pequeñas y especializadas, con enfoques y funcionalidad que les distingue del resto. En concreto, la masificación del uso de las mencionadas tecnologías y la mejora de los sistemas de posicionamiento global (GPS), ha facilitado el surgimiento de las denominadas redes sociales basadas en ubicación, o LBSN. Recabar el historial de recorridos permitiría reconocer patrones tanto en rutas planificadas como espontáneas, surgidas por la interacción con la aplicación y sus usuarios.

Por supuesto, las LBSN no son las únicas fuentes de recorridos de usuarios surgidos gracias a los avances tecnológicos de la última década, como veremos a continuación.

2.1. Sistemas de ubicación con fines sociales

En general, una LBSN permite a un usuario informar al resto de la red dónde está, haciendo un check-in en un punto de interés (como un café, un cine, un bar, etc.). A su vez, puede conocer los check-ins de sus contactos, información relacionada como fotos y comentarios, e incluso qué usuarios han hecho check-in en los alrededores. Con esto, las redes sociales pueden recolectar estos recorridos y, además de recomendar contactos, emitir información a terceros para, por ejemplo, mejorar la publicidad, ofrecer promociones con negocios asociados, y realizar análisis estadísticos varios.

A continuación, describiremos varias compañías con redes sociales destacables en materia de aprovechamiento de la ubicación de sus usuarios:

- Foursquare

  En la actualidad, es una compañía que ofrece soluciones tecnológicas basadas en ubicación para empresas y usuarios [7].

  A través de su aplicación Foursquare Swarm, los usuarios pueden registrar dónde han estado [8] mediante check-ins [9], personalmente o también en nombre de otros amigos, así como publicar comentarios, fotos y listas de recomendaciones sobre los sitios en los que han estado.

  Dispone de un API para desarrolladores que deseen interactuar con esta tecnología en sus aplicaciones, permitiendo solo recolectar datos de check-ins de los dispositivos en los que residan las mismas [10].

- Gowalla

  Al igual que con Foursquare, los usuarios en la red social de Gowalla pueden compartir su ubicación con sus contactos [11], al igual que fotos e historias, pero además tener un «pasaporte» como registro de los sitios visitados [12]. Cerró en 2012 tras ser comprada por Facebook [13][12].

  Se dispone de un listado de usuarios con tiempo y ubicación de sus check-ins en un dataset elaborado como parte de un proyecto de la Universidad de Stanford, bajo el nombre SNAP.

- BrightKite

  Fundada en 2007 [14], fue una red social hasta finales de 2010 que incluía check-ins como parte del núcleo de su aplicación [15]. También se dispone de un conjunto de datos de check-ins gracias a SNAP.

- Loopt

  Aplicación abierta al público en 2006, permitía a los usuarios compartir su ubicación precisa actual a través de la propia aplicación y mediante notificaciones en otras redes sociales como Facebook y Twitter, además de ofrecer mensajería para amigos cercanos [16] [17]. También hacía posible encontrar valoraciones, comentarios y ofertas de sitios como restaurantes [18] [19].

  Fue comprada en 2012 para usar su tecnología para mejorar servicios bancarios[20].

- GeoLife

  Como parte de un proyecto de investigación de Microsoft en Asia surgió GeoLife, en Febrero de 2009. Al contrario que otras LBSN, el recorrido geográfico se captura de manera más granular, permitiendo capturar y compartir el recorrido completo en lugar de sólo puntos de interés predefinidos. A partir del historial de recorridos de sus usuarios, el servicio también crea recomendaciones de viaje genéricas y personalizadas.

  Gracias a este proyecto también tenemos trayectorias ya clasificadas, pero sin identificación de usuario[21]

Si bien las LBSN representan una aplicación importante de los servicios de ubicación, no son las únicas que han incrustado los servicios de ubicación en el núcleo de su modelo de negocio, y por tanto vale la pena explorar otras fuentes de recorridos de usuarios:

- Servicios especializados

  Un primer ejemplo es la aplicación Glympse [22], la cual permite a un usuario compartir en tiempo real su ubicación por un periodo de tiempo y a los contactos que decida, de manera

que pueda mantenerles informados. De momento, no comparte el historial de ubicaciones con terceros [23], por lo cual no es un medio aprovechable.

Otro ejemplo es Skyhook, cuyo enfoque está en exponer la ubicación precisa de "todos los dispositivos en todos los entornos", mediante la combinación de señales Wi-Fi, GPS y telefónica de manera que sea posible, mediante analítica, mejorar distintos aspectos de un negocio [24]. Según su política de privacidad, solo comparten la información de ubicación de sus usuarios con clientes asociados, y "en zonas concretas en un intervalo de tiempo concreto"[25].

Aunque no un servicio independiente, la API de Geolocalización de Google [26] permite a las aplicaciones que la utilicen conocer con mayor precisión la ubicación de sus usuarios por distintos medios, siempre y cuando residan en el dispositivo del usuario y éste conceda su permiso explícito.

- Agregación

  A veces solo es necesario unir los puntos para descubrir la trayectoria de los internautas, es decir, extraer datos de distintas fuentes y correlarlos.

  A modo de ejemplo, en 2012 se reportó que usando datos ofrecidos por Apple, era posible determinar el recorrido de uno de sus dispositivos iOS, por un lado captando las direcciones MAC que el dispositivo emite de los últimos 3 puntos de acceso Wi-Fi a los que se conectó, y por otro usando esas direcciones MAC contra los servicios de ubicación de Apple, quienes devuelven la ubicación física del punto de acceso [27].

2.2. Detección de rutas

Una vez tenemos los recorridos de los usuarios es necesario discernir algorítmicamente qué check-ins (o qué secciones de un recorrido preciso) forman parte de una ruta, cuáles no, y en el primer caso, a qué ruta pertenecen. En otras palabras, se deben extraer los objetivos de una porción de todo el recorrido de un usuario, entendiendo que la mayoría, sino todos, los objetivos de cada porción tienen una relación entre ellos.

- A través del historial de caminos del usuario, se pueden trazar los posibles caminos y reconocer los puntos de interés del usuario [28]. Esto es posible gracias a que los caminos que toma el usuario suelen ser repetitivos. Google tiene la patente [29] sobre la metodología de reconocimiento y predicción de rutas mencionado en el anterior artículo (originalmente de la Universidad de Florida Sur), específicamente para el aviso de posibles obstáculos en la vía.
- En lugar de conocer el historial de rutas de un usuario, se pueden extraer teniendo disponibles conjuntos de fotografías (o cualquier otro elemento que ubique al usuario identificable en el espacio y en el tiempo) [30]. El trabajo citado se enfoca en la recomendación, dada una ciudad, de rutas turísticas, generadas y catalogadas a partir del contenido publicado por usuarios que se deducen son turistas también.

  Es posible eliminar algunas restricciones para hacer recomendaciones por similitudes (por su historial, por las etiquetas de los recorridos, por segmentación de usuarios, etc.), crear heurísticas para evaluar los resultados de otros algoritmos de reconocimiento, etc.
- Otra posibilidad, teniendo el historial de visitas de usuarios, es generar una ruta basándose en el contenido, es decir, las características de los puntos de interés, para seleccionar un conjunto de POIs y formar un camino Hamiltoniano cuya distancia de recorrido completo sea menor que una constante predefinida, maximizando la afinidad al usuario total del conjunto y por tanto del recorrido [31].

2.3. Visualización de rutas

A la vez que se ha descubierto y valorado la información de ubicación geográfica de personas, negocios, eventos y otros puntos de interés, también ha surgido la necesidad de presentar estos datos de manera accesible a todos los públicos, con distintos formatos, grados de interactividad, facilidad de producción (para desarrolladores pero también usuarios finales), y otras características dependientes del uso para el que estuviese concebido. No tienen las mismas necesidades el software de geolocalización utilizado para llevar a un vehículo desde su ubicación actual a un único destino seleccionado, que una recomendación de restaurantes, o una simple aplicación para exportar imágenes de mapas para que puedan añadirse luego a documentos de texto, etc.

Como este proyecto también tiene unas necesidades específicas, exploraremos las distintas herramientas de visualización para seleccionar la que mejor se adapte:

- Google Maps

  Dispone de una API para JavaScript en la que se puede dibujar un mapa con marcadores, proporcionando latitud y longitud [32]. También es posible conectar estos puntos mediante una figura polilínea (similar a un polígono, pero puede no estar cerrado y no tiene área) [33], o directamente mostrando una ruta eficiente entre coordenadas [34]. Para usar la API se necesita de una cuenta de Google, y tiene transacciones limitadas para una cuenta gratuita (2.500 al mes) [35]. Por último, permite crear imágenes de mapas, pero a través de un API separada [36].

  BatchGeo facilita crear mapas de Google gratuitamente, bien interactivos o bien como una imagen de vista preliminar [37], pero está limitado a poner marcadores en el mapa, pudiendo numerarlos pero no unirlos.

- OpenStreetMap

  OpenStreetMap es un proyecto desarrollado por una comunidad en Internet para ofrecer mapas estáticos e interactivos para webs y aplicaciones. Para el objetivo de este proyecto, dispone de una variedad de librerías para distintos lenguages (C++, JavaScript, Java, Python y muchos más) y plataformas (en web, smartphone y desktop) [38].

  De entre las disponibles, si observemos solo aquellas de alta portabilidad, esto es, disponibles en distintos sistemas operativos desktop y preferentemente en smartphones también, podemos distinguir estas categorías según su enfoque:

  - Mapas para web: librerías que se sirven tecnologías web y utilizan JavaScript casi exclusivamente para mostrar y manipular mapas. Esta opción es la más accesible para todos.
  - Mapas para programas multi-plataforma: estas librerías dependen a su vez de programas que se pueden encontrar en varias plataformas, como son Qt y la máquina virtual de Java.
  - Librerías especializadas de un mismo desarrollador: exponen una base común pero ofrecen SDKs específicas de plataformas como Android e iOS. Es el caso de Mapbox y Skobbler, los cuales necesitan de claves de acceso que se pueden obtener gratuitamente. De los mencionados, sólo Mapbox impone un límite de peticiones a sus servidores. Hasta la fecha (Enero de 2018), Skobbler no está disponible para nuevos desarrolladores [39].

- HERE Maps

  Comparte con Google las características mencionadas: API de JavaScript para marcadores [40], polilíneas [41], rutas según medio de transporte, creación de imágenes de mapas [42]

y tiene un volumen de transacciones restringido para una cuenta gratuita (en este caso, 15.000 al mes) [43].

- GPSVisualizer

  Herramienta web con funcionalidad básica de visualización de coordenadas [44]. Une los puntos de la ruta automáticamente para archivos de texto de coordenadas separadas por tabulador. También permite combinar polilíneas con marcadores en el mapa (y así distinguir una ruta entre un cúmulo de coordenadas).

- QlikView

  Herramienta de escritorio para visualización de datos principalmente desde su interfaz gráfica [45], aunque también incluye su propio lenguaje de scripting para procesar los datos a visualizar [46].

  Dispone de una versión personal gratuita[47].

- Aplicaciones de análisis de datos

  Tanto Mathematica [48] como SAS Visual Analytics [49] son aplicaciones que permiten visualizar puntos en un mapa de alguna manera pero cuya funcionalidad excede con creces las necesidades de este proyecto. A pesar de su complejidad, ofrecen pruebas gratuitas de medio mes [50] [51], pero requieren un pago o subscripción para un uso prolongado [52].

2.4. Conclusiones

2.4.1. Recorridos de usuarios

Por un lado, la solución del desarrollo de una aplicación para los dispositivos de los usuarios requiere de su colaboración activa, limitando la cantidad de datos recolectables y, en consecuencia, la fidelidad de los resultados porque no son tan representativos. Por tanto, quedan descartadas las APIs y servicios para obtener la ubicación.

Por otro lado, tenemos los recorridos que, o bien podemos recolectar, o bien ya están preparados y disponibles para el consumo. El hecho de que los recorridos estén clasificados por el usuario es un beneficio añadido de algunas fuentes, ya que permiten verificar si una misma definición de ruta se ajusta a los distintos perfiles de usuario. Cabe destacar también que al no ser rutas simuladas sino recorridos reales de personas, los resultados que se obtengan del análisis serán más fieles a la realidad y por lo tanto más aprovechables.

Ya que no hay necesidad de captar datos de manera inmediata ni requisitos acerca del formato de entrada, para realizar un análisis de qué puede constituir una ruta, lo más sencillo es aprovechar los recorridos ya registrados de usuarios, con los cuales además tenemos detalle suficiente. Es por este motivo que se trabajará con los conjuntos de datos de SNAP, para las redes sociales de BrightKite y Gowalla.

2.4.2. Reconocimiento de rutas

En este aspecto es donde las herramientas disponibles son escasas sin conocer primero las características de los POIs. Es, por tanto, que se realizará un desarrollo propio, que permita incluir y probar distintas definiciones de rutas.

Es posible partir de funciones de estadística y algoritmos usados comúnmente en sistemas de recomendación para crear los primeros algoritmos de reconocimiento. Estos se explorarán en más detalle en la descripción de la implementación.

2.4.3. Visualización

Por un lado, sistemas similares como Google Maps y HERE Maps son bastante flexibles pero su uso está restringido a lo que se pague. Por otro lado, la opción gratuita de GPSVisualizer tiene una funcionalidad bastante básica, lo cual resulta poco escalable.

En resumen, OpenStreetMap ofrece el mejor balance entre variedad de herramientas disponibles, disponibilidad de uso y precio.

Capítulo 3

Diseño

3.1. Sistema completo

3.1.1. Estructura general

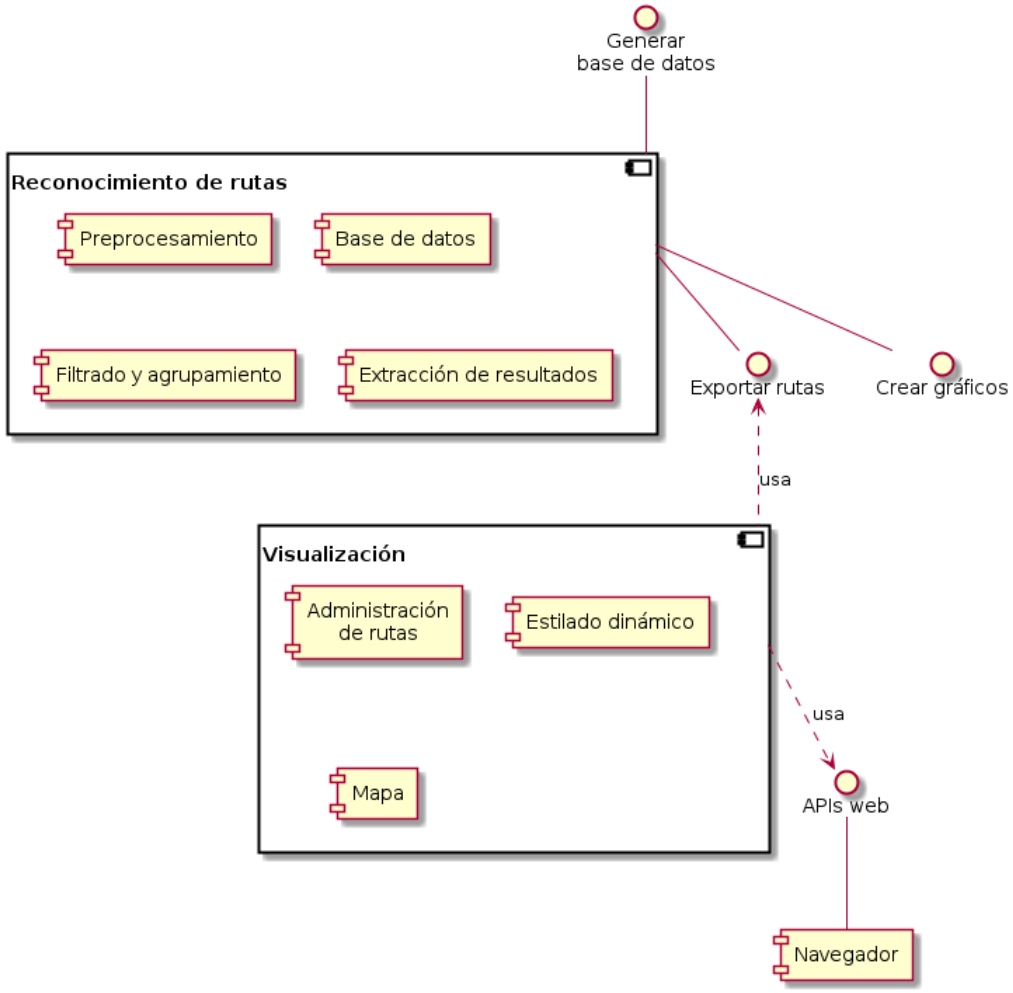


Figura 3.1: Componentes del sistema completo

Este trabajo contiene dos tareas principales claramente diferenciadas, que resolveremos mediante dos grandes módulos como se puede ver en la Figura 3.1: el reconocimiento de una ruta

y la visualización de la misma. Además, podemos distinguir tareas que satisfacen necesidades técnicas específicas, como establecer los medios para hacer uso de los resultados de un módulo en el otro, en particular, de reconocimiento a visualización.

El módulo de reconocimiento, a su vez, se ha dividido en una serie de componentes que describimos a continuación:

1. Preprocesamiento

   Una fuente de datos contiene, a priori, una cantidad indeterminada de atributos en cada uno de sus registros, pero sólo un conjunto selecto será utilizado para los algoritmos de reconocimiento que utilicemos. Este módulo se encarga no sólo de quitar los datos inutilizados, sino de plasmar en un formato más aprovechable los datos captados; una base de datos que el resto de módulos puede manipular.

2. Filtrado y agrupamiento

   A partir de aquí procedemos al reconocimiento como tal, eliminando check-ins o usuarios de los que no disponemos de suficiente información para afirmar que ha realizado una ruta concreta, y agrupando o eliminando la información restante, todo a criterio del algoritmo. Una vez determinado qué check-ins conforman las rutas de un usuario, se deben agrupar y separar del resto de check-ins para consumo de otros módulos.

3. Extracción de resultados

   Este componente tiene la tarea de presentar la información en una variedad de formatos distintos, según se requiera. Como mínimo, muestra gráficas de los campos y exporta a un formato concreto datos de la ruta.

Por otro lado, la visualización consta de los siguientes componentes:

1. Mapa de visualización

   Este componente es el que permitirá ver geográficamente ubicadas las rutas reconocidas, y obtener información relacionada si se requiere.

2. Estilado dinámico de rutas

   Para facilitar la diferenciación de usuarios y rutas por parte de un usuario humano, se establecen reglas de estilo en función de la cantidad y tipo de información que confluye en una misma región geográfica.

3. Administración de rutas

   Otro componente que mejora la experiencia de usuario es la posibilidad de poner o quitar rutas y usuarios según se necesite.

3.1.2. Objetivos

A continuación se enumeran los principales objetivos de este trabajo:

1. Reconocer rutas
   - De conjuntos de datos preparados, estáticos y con un formato específico
   - Aplicando filtros configurables
   - Emitiendo también gráficas de la distribución de la variable estudiada

2. Visualizar rutas
    - A través de un mapa interactivo multiplataforma
    - Pudiendo manipular e interactuar con las rutas, incluyendo
        - Añadir rutas desde un fichero
        - Eliminar las rutas
        - Cambiar su color
        - Incluir información adicional sobre puntos de interés en el recorrido

Los objetivos antes mencionados se pueden resumir en el siguiente diagrama de casos de uso:

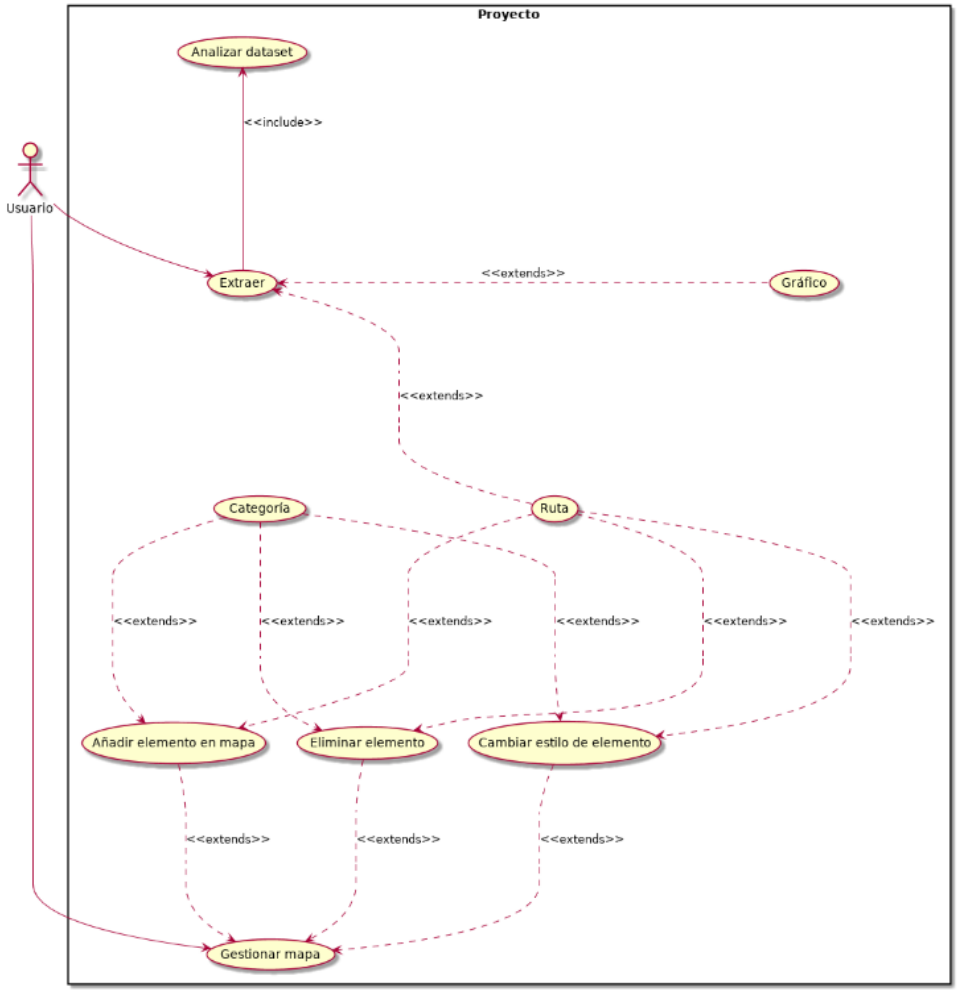


Figura 3.2: Casos de uso del proyecto

3.1.3. Requisitos funcionales y no funcionales

A partir de los objetivos descritos anteriormente, se definen los siguientes requisitos funcionales, es decir, enfocados en las capacidades de las herramientas a desarrollar:

RF1 Implementar un sistema de detección de rutas a partir de información de usuarios que interactúan con puntos de interés turísticos.

RF2 El sistema de detección debe permitir añadir y eliminar filtros de detección.

RF3 El sistema de detección debe generar gráficas de distribución de la variable utilizada para la detección.

RF4 Implementar un sistema de visualización de rutas que muestre rutas en mapas de OpenStreetMap.

RF5 El sistema de visualización debe poder añadir rutas desde un fichero.

RF6 El sistema de visualización debe poder añadir categorías de POIs desde un fichero.

RF7 El sistema de visualización debe permitir eliminar rutas del mapa.

RF8 El sistema de visualización debe permitir eliminar POIs del mapa según su categoría.

RF9 Permitir que los dos sistemas desarrollados se comuniquen entre sí.

Igualmente, se han identificado los siguientes requisitos no funcionales, es decir, aquellos que tienen que ver con el rendimiento y la usabilidad:

RNF1 El sistema de detección aceptará como mínimo un identificador de usuario, una marca de tiempo y la latitud y longitud de un punto geográfico para definir la interacción del usuario con un punto de interés.

RNF2 El sistema de detección podrá trabajar con ficheros que sigan el formato de los conjuntos de datos de SNAP.

RNF3 El sistema de detección de rutas tiene que poder funcionar con datos de hasta 100.000 usuarios o 5 millones de interacciones (check-ins).

RNF4 El sistema de visualización debe poder usarse desde un navegador web.

RNF5 Los mapas para la visualización deben ser interactivos, permitiendo ampliar, reducir y mover la vista de la zona geográfica mostrada.

RNF6 El sistema de visualización debe permitir cambiar el color de las rutas mostradas.

RNF7 El sistema de visualización debe permitir cambiar el color con que se muestran los POIs, según su categoría.

RNF8 El sistema de visualización debe dar colores distintivos a las categorías cargadas por defecto.

A continuación se pasa a describir cada uno de los dos subsistemas que conforman el proyecto y cómo se han diseñado.

3.2. (Sub)sistema de reconocimiento de rutas

3.2.1. Captación de datos

Para los conjuntos de datos de SNAP (en total casi 11 millones de check-ins), cada entrada es una línea con identificador de usuario, marca de tiempo (en UTC sin huso horario, siguiendo el formato especificado por ISO 8601 [53]), latitud, longitud e identificador de ubicación, en ese orden.

3.2.2. Reconocimiento de rutas

Existe un método de reconocimiento del camino recorrido trivial conociendo la marca de tiempo de cada coordenada, pero el objetivo es separar ese camino en rutas, cada una con su destino objetivo particular.

Alternativas

En un contexto de múltiples rutas guardadas en un mismo conjunto de datos, para distinguir entre puntos de interés reales (destinos de una ruta) y simples paradas transitorias, se puede:

1 Establecer umbrales fijos para estadísticas concretas, globales a todos los caminos:

  Como hipótesis, los tiempos entre dos sitios de rutas distintas visitados consecutivamente, por ejemplo, serán marcadamente distintos de aquellos consecutivos pertenecientes a la misma ruta, por tanto se debería apreciar un límite de distancia temporal a partir del cual, con alta probabilidad, el siguiente punto visitado pertenezca a una ruta nueva. Esta hipótesis se puede extender a distintas variables, como distancia euclideana, y factores derivados de ellas, como cantidad de sucesos en un intervalo fijo o la razón entre la distancia euclideana y la temporal.

2 Determinar localmente un umbral.

  Por ejemplo, a partir de las últimas N paradas, aquel check-in que se distancie mucho de la media para cierta variable se consideraría otra ruta.

3 Usar técnicas tradicionales en sistemas de recomendación y aprendizaje automático

  A partir de métricas de afinidad, comúnmente discutidas en el ámbito de los sistemas de recomendación, es posible partir de un algoritmo de reconocimiento de rutas, que tras ser aplicado a usuarios afines, compare a su vez la afinidad de las rutas, lo cual permita refinar parámetros del algoritmo que a su vez aminoren la diferencia entre afinidades para los usuarios y sus respectivas rutas.

En cualquier caso, para enriquecer los resultados se pueden incluir en el cálculo variables contextuales como cantidad de datos por región, densidad de medios de transporte por tipo y zona, clustering de puntos de interés (conectados por aceras y/o carretera), etc., lo cual da cabida a una gran variedad de métricas y consecuentes heurísticas para evaluar la pertenencia a una ruta y la relevancia de la misma en la detección de otras.

Cabe mencionar que cuando los datos de los que disponemos son check-ins, y no una secuencia contínua en el tiempo de coordenadas, la interpretación de qué conforma una ruta puede cambiar ligeramente. Si bien un check-in es, supuestamente, un punto de interés en sí, cabe la posibilidad de tomar todos los check-ins entre el comienzo y el final de la ruta como parte de la ruta. No obstante, también es posible filtrar algunos, entendiendo que se registraron porque estaban en el camino o cerca, a pesar de que no concuerde con la temática de la ruta.

3.3. (Sub)sistema de visualización de rutas

3.3.1. Empaquetado de rutas para visualización

Como parte de un diseño modular y extensible, es deseable que las rutas que resulten del reconocimiento se presenten en algún formato bien definido, preferiblemente difundido y estándar

para que programas ya existentes sean capaces de interpretar y manipular. Además, es valorable que dicho formato pueda incluir información arbitraria, útil para implementaciones futuras y mejoras.

3.3.2. Visualización de rutas

Ya que este módulo es el que más interactúa con el usuario final, se debe hacer un especial énfasis en la usabilidad, la facilidad de interacción entre el programa y el usuario a la hora de realizar los casos de uso más comunes. Por este motivo el mapa debe ser interactivo (pudiendo añadir elementos, quitarlos y cambiar de vista con efecto inmediato), y los medios para manipular los elementos (no solo insertarlos, sino también cambiar su presentación) deberían poder realizarse por lotes y de manera gráfica (frente a la manipulación mediante texto).

Capítulo 4

Implementación

Como se vió en el capítulo anterior, este proyecto está diseñado partiendo de dos subsistemas principales: el de detección de rutas y el de visualización. En este capítulo, vamos a describir cómo se han desarrollado estos dos subsistemas.

4.1. Detección de rutas

Por la facilidad de manipulación de ficheros y flujos de datos, se escogió el lenguaje sh para conectar los distintos componentes del módulo, implementados en Python. Para que las rutas pudiesen ser consumidas por cualquier tipo de aplicación de visualización, se optó por encapsular sus datos de coordenadas en un formato estándar ya concebido para este fin, el formato GeoJSON (RFC 7946) [54]. Mediante este formato, no sólo podemos distinguir rutas de manera explícita y legible, sino que podemos añadir información sobre la ruta, como fecha, autor, estadísticas varias, y cualquier otro dato relevante, sin necesidad de cambiar aplicaciones preexistentes.

De esta forma, la herramienta de detección de rutas se compone de scripts que se encargan de reconocer variables, aplicar filtros sobre ellas y generar los gráficos y ficheros de rutas para su posterior uso. En ningún momento se encarga de reconocer, manipular o exportar categorías de puntos de interés, pero se pueden usar los filtros de rutas para este efecto como subproducto.

4.1.1. Punto de entrada

Esta frase no se entiende El punto de entrada de la herramienta es la única interfaz que se expone al usuario en todo momento (los desarrolladores pueden recibir ayuda sobre un script particular ejecutándolo sin argumentos). Es un script en el lenguaje sh (en su versión que cumple con el estándar POSIX), y está encargada de la ejecución de toda la herramienta (véase Anexo A para información de uso).

4.1.2. Preprocesamiento

Dado que el resto de scripts no tiene medios para reconocer semánticamente los campos con los que trabaja, se realiza con anterioridad la creación de una base de datos que representa todos los datos utilizables del conjunto de datos proporcionado al inicio del programa.

4.1.3. Base de datos

Los datos aprovechables se almacenan en una base de datos jerárquica basada en texto. Se estructura de la siguiente manera:

- Carpeta 'unfiltered': Representa la base sobre la cual los filtros ejecutan sus algoritmos. Aquí se guarda un fichero por cada usuario, teniendo por nombre su respectivo identificador, y dentro los campos aprovechables, separados por caracteres de espacio (tabulador, espacio), en una línea por check-in, y check-ins consecutivos se almacenan en líneas consecutivas, de manera que los filtros puedan conocer la sucesión de eventos, y a qué usuario pertenecen.
- Carpeta 'tmp': Representa el producto del filtrado. Se recomienda que los filtros mantengan la jerarquía de carpetas para el correcto funcionamiento del resto de scripts de la herramienta. No obstante, si se considera oportuno es posible añadir y quitar campos a los check-ins, incluso cambiarlos de orden, pero se debe tener en cuenta que otros filtros que se adhieran al orden por defecto pueden dejar de funcionar como consecuencia.
- Carpeta 'graphs': Aquí se almacenan los gráficos, identificados por un nombre compuesto por la variable filtrada y el nombre del filtro utilizado.
- Carpeta 'json': Dentro de esta carpeta habrá una por cada magnitud filtrada, con su nombre.
    - Magnitud: De la misma manera, dentro se guarda, por filtro ejecutado, una carpeta con su respectivo nombre.
        - Filtro: aquí se guardan, un fichero por usuario, todas las rutas que se han reconocido del mismo. El fichero es GeoJSON válido, tiene como nombre el identificador del usuario y extensión json.

4.1.4. Filtros

Los filtros son archivos de código Python, almacenados en la carpeta codigo/scripts/filters. Se espera que, como mínimo, reciban por parámetro el nombre de una carpeta de entrada (de donde extraerán los check-ins de usuarios) y una de salida (donde almacenarán los check-ins ya filtrados, siguiendo la nomenclatura de la base de datos); todos los demás argumentos son opcionales y se suplen desde la petición de filtrado, sin modificación.

Para que, dentro de un fichero de usuario, se distinga una ruta de otra, deberán estar separados sus check-ins por una línea en blanco. Los check-ins de una misma ruta se consideran que forman un camino dirigido en el orden en el que se leen en el fichero.

4.1.5. Gráficos

Los gráficos se generan, para los filtros ejecutados con éxito, mediante la herramienta gnuplot. La representación gráfica es una superposición de los valores de la primera columna de cada archivo de usuario, siendo el eje de abscisas el número de fila y el eje de ordenadas el valor de la celda.

Los gráficos se generan tras la ejecución de cada filtro, por lo que cada combinación de magnitud y filtro generará una imagen PNG el gráfico correspondiente al contenido de la carpeta 'tmp' en ese momento.

4.1.6. Rutas

Cada ruta de usuario se genera tomando las coordenadas de sus check-ins, en el orden en que aparecen, y empaquetándolos en GeoJSON describiendo un LineString sin atributos adicionales. Posteriormente se agrupan las rutas de un mismo usuario como un FeatureCollection, también sin atributos adicionales, que finalmente es lo que se guarda en el fichero correspondiente.

4.2. Visualización de rutas

La visualización se implementa como una única página web, pero se ha estructurado su código para añadir otras páginas en el futuro. Los archivos que pertencen exclusivamente a la página principal son aquellos de nombre 'index' (HTML, CSS y JS).

Los ficheros relacionados con la funcionalidad del mapa se encuentran en la carpeta maps, tanto las funciones para interactuar como aquellas para obtener las celdas del mapa.

Consúltese la documentación de Leaflet.js para información sobre cómo cambiar de servidor de celdas de mapa u otros usos.

4.2.1. Librerías escogidas

En vista del soporte para integración que tiene JavaScript en todo tipo de plataformas, así como la flexibilidad de programación que da el lenguaje, resulta una buen candidato para desarrollar el sistema de visualización.

En cuanto a librerías JavaScript, para OpenStreetMap han sido desarrolladas varias por distintas colaboradores (el listado completo se encuentra en [38]), pero pocas se encuentran en mantenimiento y presentan la funcionalidad deseada. Es por eso que se ha escogido Leaflet [55] para implementar la visualización, la cual permite a través de un API sencillo importar rutas de JSON y arrays de coordenadas, mostrarlas en el mapa con estilos configurables, permitir interactuar con ellas, etc.

4.2.2. Fuentes de mapa base

Los sistemas de visualización no estarían completos sin la imagen del mapa, en la que se plasman las características de la geografía como relieve y calles. Este mapa se divide en celdas, y se presentan de manera distinta según el nivel de aumento o zoom.

Para una gestión eficiente, las celdas son proporcionadas por un servidor dadas las coordenadas X,Y y Z (latitud, longitud y nivel de aumento, respectivamente), bien comercialmente bajo contrato o de manera gratuita por voluntad de terceros.

En nuestro caso, para la demostración de la funcionalidad del proyecto, se ha optado por una solución gratuita por cortesía de Stamen, un estudio de diseño enfocado, entre otras cosas, a la cartografía [56, 57].

4.2.3. Barra lateral

La barra lateral presenta información sobre el estado del mapa y de las rutas que contiene:

1 En la parte superior se encuentra el div de identificador #map_coords, que indica las coordenadas actuales del centro del mapa (útil para tener una noción de en qué coordenadas

del globo se encuentra cada zona geográfica). Por defecto no tiene texto, pero una vez que se mueva el mapa aparecerá.

2 Debajo de las coordenadas se encuentra un formulario para subir archivos, que consta de un elemento 'input' para ficheros (uno a la vez), y otro para enviar el formulario. Los ficheros se procesan mediante JavaScript para evitar cambiar de página, de manera que se mantengan todas las variables, elementos y estado del mapa que hayamos añadido previamente.

3 Finalmente, tenemos el listado de ficheros y rutas cargados. Internamente se gestiona una variable en JavaScript (variable global loaded_paths) que contiene el árbol de rutas y sus ficheros (raíz ->nombre de fichero ->ruta), del cual el listado gráfico es un reflejo. El color actual de cada ruta está registrado por la librería Leaflet, y basta con tener una referencia a la ruta y leer o escribir el atributo 'options.color' para conocer o modificarlo.

4.2.4. Barra de categorías

La barra de categorías se encuentra en la parte inferior de la página, debajo del mapa. Su principal función es listar las categorías cargadas y permitir modificar sus atributos (como el color de sus puntos de interés).

Las categorías se encuentran en un flex-box (un 'div' con 'display: flex'), cada una un 'div' con el nombre de la categoría y un 'input' para cambiar el color. De nuevo, la gestión se realiza mediante eventos JavaScript y no se formaliza una petición ni se cambia de página.

Dado que Leaflet no tiene medios documentados para establecer el color del punto de interés, y en su lugar habilita asignar una clase CSS, para cada categoría se genera una clase, cuyo atributo 'background-color' es cambiado de valor cada vez que se desea cambiar el color de una categoría, cambio que se hace visible de manera inmediata. También se mantiene una variable global en JavaScript ('categories') con la asociación de nombres de categorías a nombres de clases CSS, para agilizar modificaciones posteriores.

Los colores de las categorías son generados por defecto de manera balanceada, es decir, tomando puntos equidistantes en el espectro de color, para minimizar la dificultad de distinguir unas categorías de otras en el mapa. En todo momento la saturación es máxima, y sólo varía el matiz (por lo que los colores siempre son brillantes y destacan).

Capítulo 5

Experimentos realizados y resultados

De modo que podamos extraer conclusiones fidedignas a partir del software desarrollado, demostraremos primero que el comportamiento del software es el deseado para los casos de uso tenidos en cuenta durante el diseño, y posteriormente compararemos algunos algoritmos de filtrado que pueden ser usados a futuro para reconocer las rutas de usuarios, o crear algoritmos derivados con este propósito.

5.1. Validación de requisitos

Una vez implementado el sistema, debemos proceder a verificar que las necesidades que debe validar este producto se satisfacen mediante las herramientas desarrolladas. Dado que las mismas se describen por separado, en la siguiente tabla se describe en qué sección del documento se realizan estas comprobaciones.

Requisito funcional	Validado en
RF1	5.1.2
RF2	5.1.2
RF3	5.1.2
RF4	5.1.2
RF5	5.1.2
RF6	5.1.3
RF7	5.1.3
RF8	5.1.3
RF9	5.1.3
RF10	5.1.2
RF11	5.3
RFN1	5.1.2
RFN2	5.1.2
RFN3	5.1.3
RFN4	5.1.3
RFN5	5.1.3
RFN6	5.1.3
RFN7	5.1.3

Cuadro 5.1: Matriz de trazabilidad de requisitos y pruebas

5.1.1. Sistemas probados

El proyecto fue desarrollado y probado en los siguientes entornos:

- Firefox 57 en Linux Mint 18.1 64bits (Kernel 4.4.0-87-generic)
- Firefox 56 en Fedora 26 64 bits (Kernel 4.13.5-200.fc26.x86_64)
- Google Chrome 61 en Windows 10 Pro

Las pruebas que informen sobre el rendimiento durante la ejecución de las herramientas fueron realizadas sobre el siguiente hardware:

- CPU: Intel Pentium G3258 a 3.2GHz
- RAM: 4GB DDR3
- Disco duro: magnético 1TB SATA6

5.1.2. Procesamiento de rutas

Pruebas con filtros por defecto

Para este módulo debemos probar que aportando un dataset y una descripción de filtrado, crea los productos esperados, es decir, las rutas organizadas por usuario y un gráfico del conjunto de datos completo por cada combinación de variable y filtro. Por simplificar, ejecutaremos todas las pruebas desde la carpeta de código, que dentro contiene una de scripts (para detección de rutas principalmente) y otra dedicada al módulo de visualización.

Ejecutamos la herramienta de detección como se explica en el Anexo A, sin una solicitud de filtrado. Una ejecución exitosa creará la base de datos al mismo nivel que el directorio 'scripts' (véase capítulo anterior). Visualmente podemos comprobar que existen estas carpetas con el siguiente comando, cuya salida se puede ver en la Figura 5.1:

```
$>  tree -L 3 --filelimit 100 .
```

Uno de los filtros por defecto (n_within) es aquel que toma las rutas de al menos N check-ins consecutivos, que para una magnitud entre check-ins dada ninguno se distancia más de una cantidad concreta del resto. Un ejemplo de gráfica generada para este filtro se puede ver en la Figura 5.2

Para este mismo filtro, podemos ver que se ha creado una carpeta suya bajo la carpeta json, dentro de la carpeta de la magnitud de tiempo. El otro filtro (group_by) no tiene una carpeta correspondiente porque no solo ha filtrado, sino agregado los resultados y por tanto no representa check-ins.

Todas estas pruebas se han condensado en un único script, route_detect.sh en la carpeta test, con mensajes de error si llegan a ocurrir.

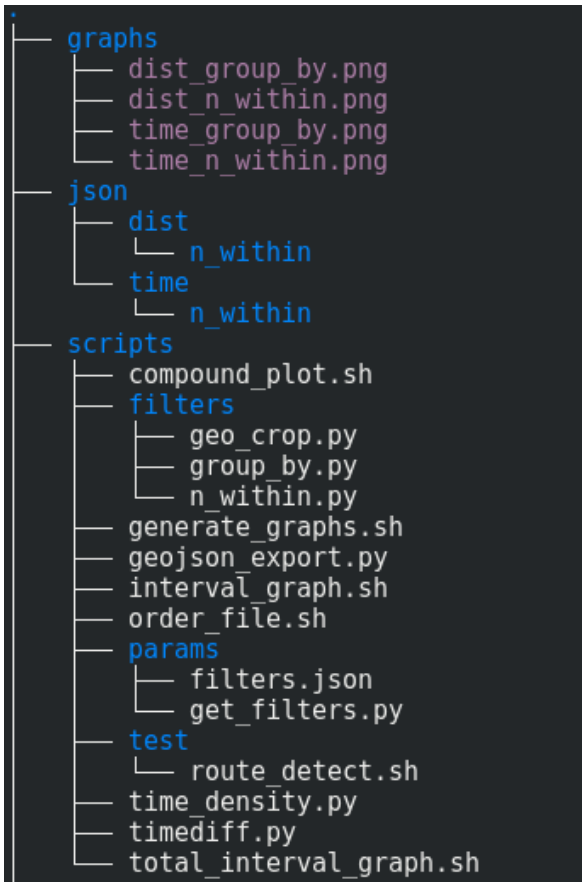


Figura 5.1: Extracto estado del árbol de directorios tras el filtrado del dataset de BrightKite.

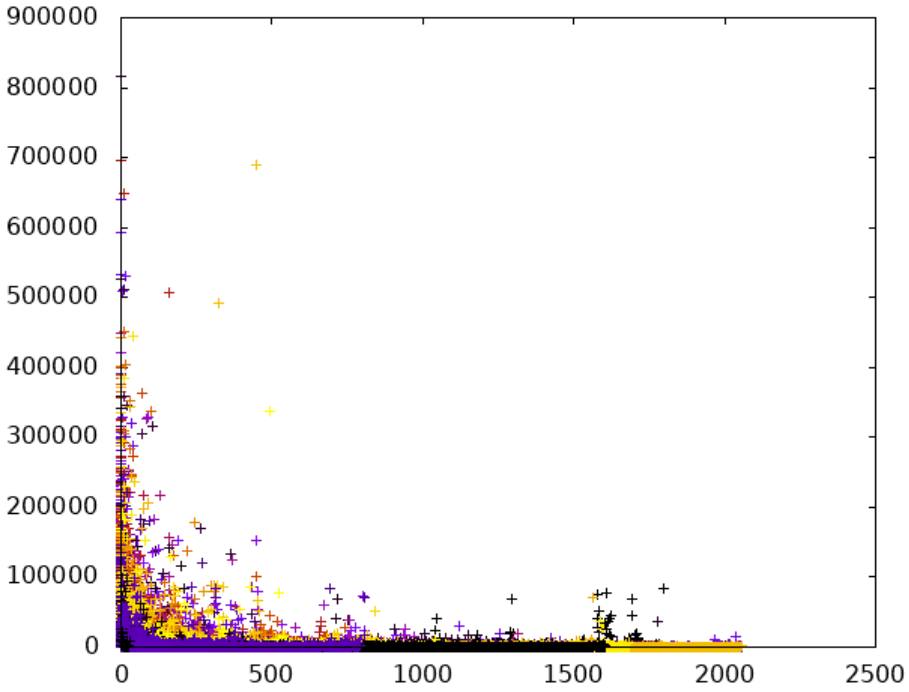


Figura 5.2: Superposición de tiempo (segundos) entre check-ins sucesivos.

Prueba con filtros específicos

La siguiente prueba aporta una solicitud de filtrado, en un fichero json, sustituyendo la solicitud por defecto.

Ejecutamos en una terminal el siguiente comando:

```
$>  sh generate_graphs.sh <nombre dataset> <nombre solicitud>
```

De nuevo, los ficheros deben ser accesibles por el nombre o ruta especificados para cada uno. Con el script route_detect.sh podemos ejecutar esta prueba.

5.1.3. Visualización de rutas

Aquí probamos las herramientas de visualización y manipulación de rutas e información añadida, como categorías, por tanto entra en juego el uso del navegador.

Administración de rutas en el mapa

Recordemos que, cumpliendo los objetivos del proyecto, este módulo debe permitir insertar rutas cargando un fichero, eliminarlas, y cambiar su color en el mapa.

Para empezar, cargamos un fichero con rutas válidas (ver Figura 5.3).

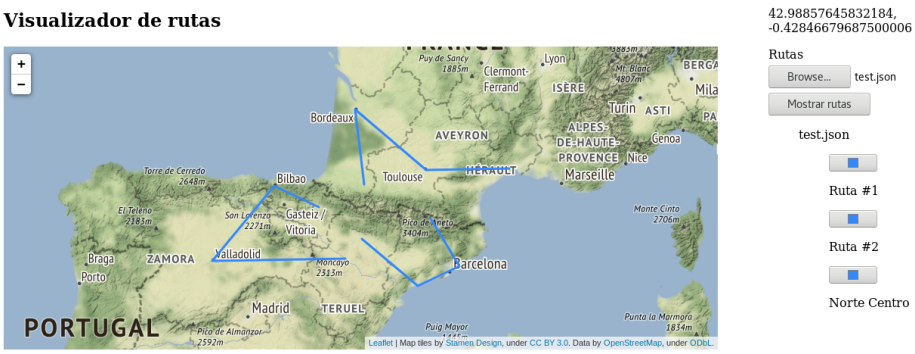


Figura 5.3: Visualización de mapas - rutas de prueba.

Aquí podemos destacar los nombres de las rutas, ya que las que no especificaban un nombre fueron numeradas en su lugar. Como se esperaba, las rutas se muestran en el mapa.

Ahora probaremos a eliminar una ruta (ver Figura 5.4). Vemos que no solo no aparece en el mapa, sino que las rutas sin nombre restante han cambiado de numeración, manteniéndose consistente con las visibles en el mapa.

Probamos a eliminar todas las del fichero, haciendo click en el nombre del fichero cargado (la salida se muestra en la Figura 5.5). Y vemos que desaparecen tanto las rutas del mapa como toda referencia a las mismas en el panel lateral.

Añadimos el mismo fichero, y otro más (ver Figura 5.6). Cada vez que añadimos un fichero nuevo, el mapa cambia de vista para abarcar las rutas nuevas (pero no las antiguas necesariamente).

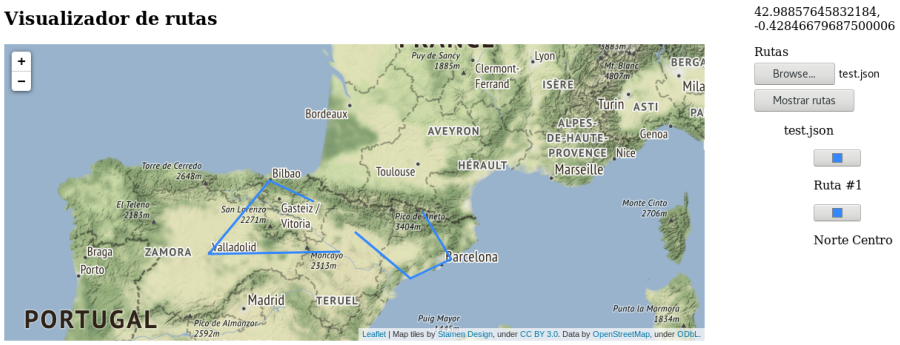


Figura 5.4: Visualización de mapas - una ruta eliminada.

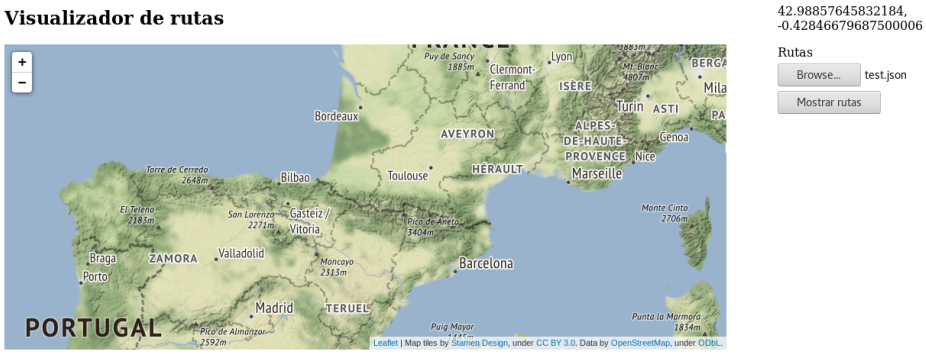


Figura 5.5: Visualización de mapas - rutas de prueba eliminadas.

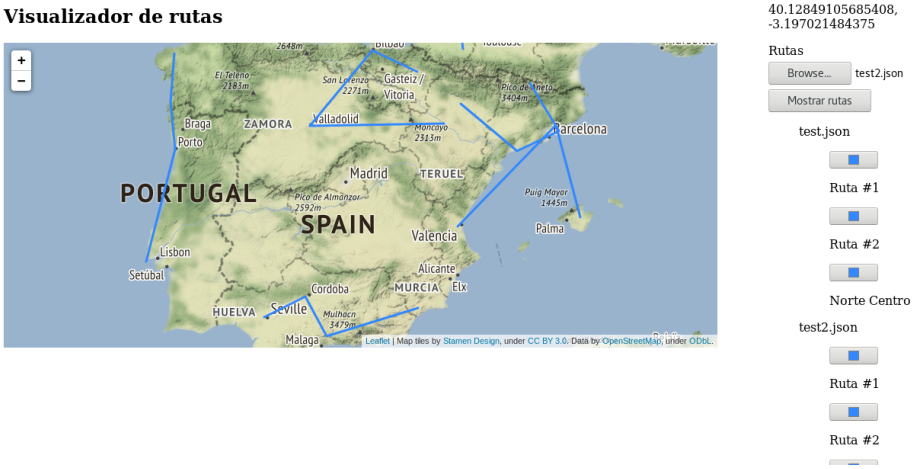


Figura 5.6: Visualización de mapas - dos ficheros de prueba.

En la Figura 5.7, mostramos el resultado después de cambiar los colores de algunas rutas. Para ver mejor las rutas se ha movido manualmente la vista del mapa, haciendo click y arrastrando sobre el mismo.

Si ahora probamos a eliminar una ruta (Figura 5.8), comprobamos que el resto mantiene sus colores asignados. Además, como se demostró anteriormente, cambia la numeración de las rutas sin nombre.

Por último, mostramos cómo se procesan los errores en la interfaz. La Figura 5.9 muestra el mensaje de error que aparece cuando añadimos un fichero inválido.

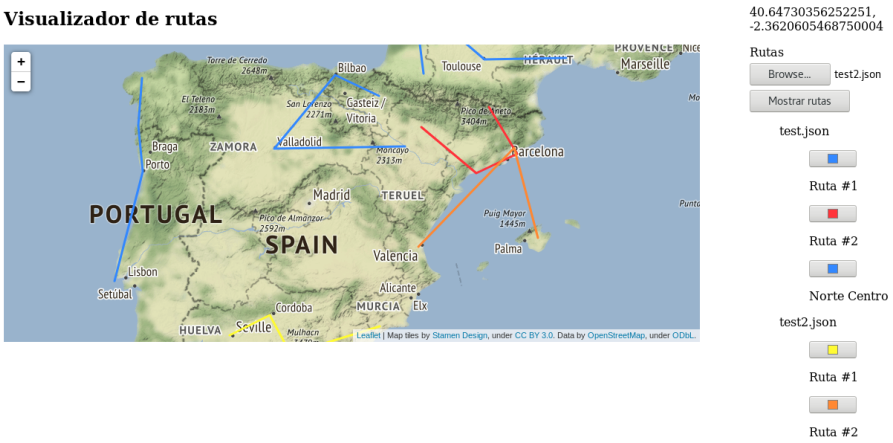


Figura 5.7: Visualización de mapas - rutas coloreadas.

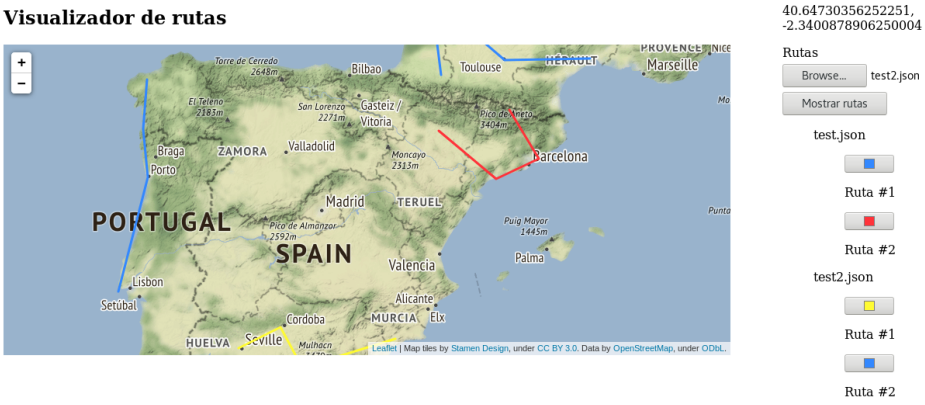


Figura 5.8: Visualización de mapas - rutas coloreadas y algunas eliminadas.

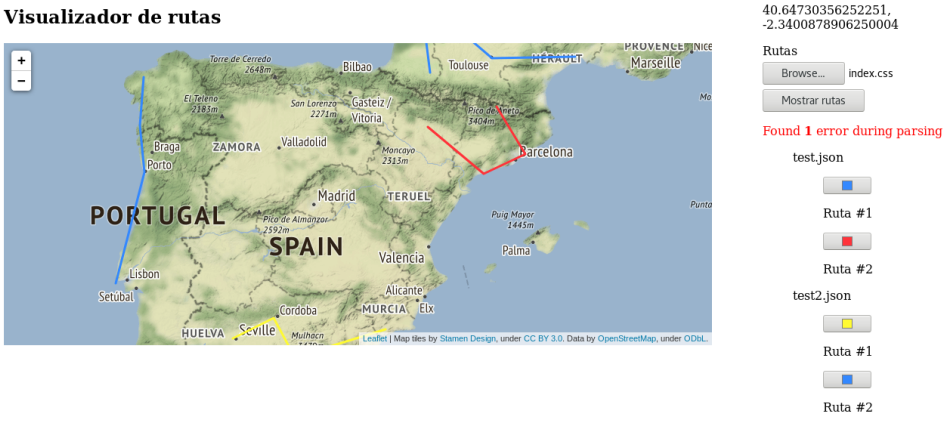


Figura 5.9: Visualización de mapas - error al cargar.

Por otro lado, la Figura 5.10 prueba a cargar un fichero válido, pero sin rutas, tras eliminar las existentes. Nótese que este mensaje es para rutas sin coordenadas, un fichero vacío generaría un error de procesamiento.

Figura 5.10: Visualización de mapas - fichero con rutas sin coordenadas.

Administración de categorías en el mapa

Para que las categorías de puntos de interés de las rutas se carguen correctamente, dichas rutas deben estar cargadas con anterioridad. No obstante, la Figura 5.11 muestra lo que ocurre cuando probamos a cargar las categorías sin sus rutas.

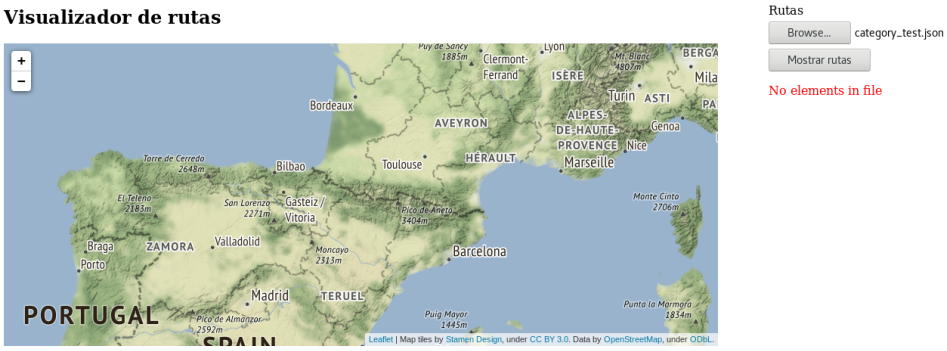


Figura 5.11: Visualización de mapas - categorías de puntos de interés sin rutas.

Si ahora cargamos primero las rutas y, posteriormente, las categorías, se puede ver en la Figura 5.12 que se han cargado tres categorías. También vemos que uno de los POIs de la parte inferior está gris y no tiene categoría. Esto se permite porque queremos ser capaces de ver puntos de los que se desconoce su categoría.

Probamos ahora a eliminar una categoría. La Figura 5.13 muestra que, además de desaparecer el punto correspondiente en el mapa, el resto de categorías en la zona inferior se distribuyen el espacio restante.

En esta prueba cambiamos el color a una de las categorías restantes (ver Figura 5.14). Como vemos, su color cambia tanto en el mapa como en la zona inferior.

Si quitamos una ruta (Figura 5.15), vemos que los puntos de interés asociados desaparecen. Como, además, los puntos de interés eliminados fueron los últimos de sus respectivas categorías, observamos que desaparecen las opciones para cambiarles de color.

5.2. Comparativa de algoritmos de separación de rutas

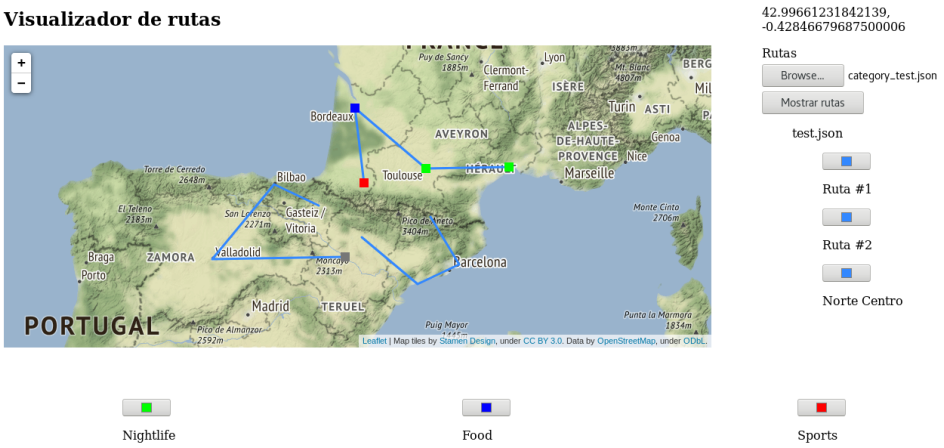


Figura 5.12: Visualización de mapas - rutas y algunos puntos de interés con categorías.

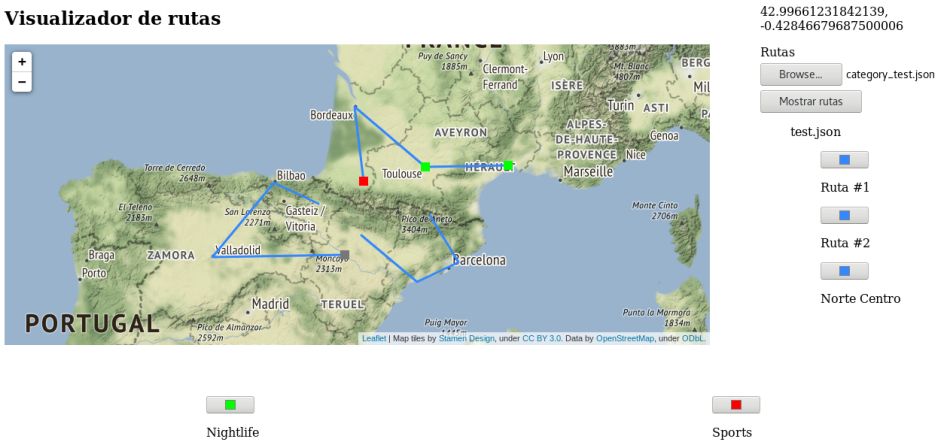


Figura 5.13: Visualización de mapas - rutas con una categoría eliminada.

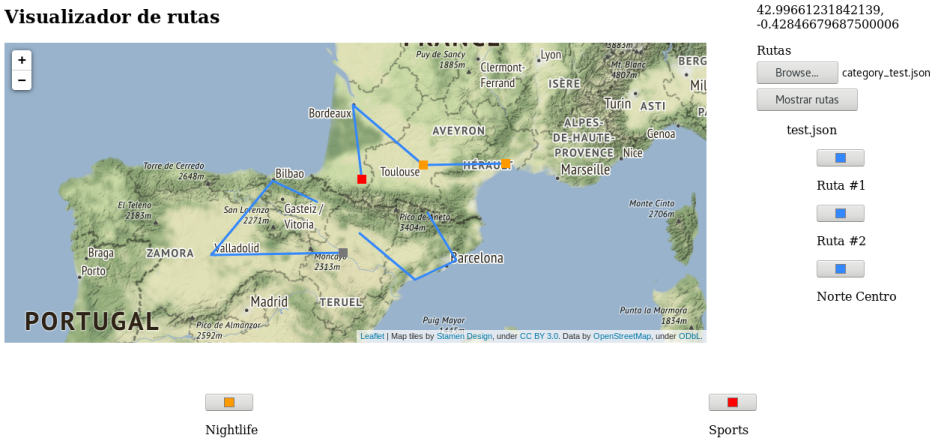


Figura 5.14: Visualización de mapas - categoría cambiada de color.

En función de la definición de ruta que se amolde a las necesidades del usuario se pueden emplear distintas heurísticas, o algoritmos de reconocimiento de rutas. Dado que una ruta representa un recorrido secuencial entre puntos de interés, cada algoritmo separará una ruta de otra comparando check-ins consecutivos.

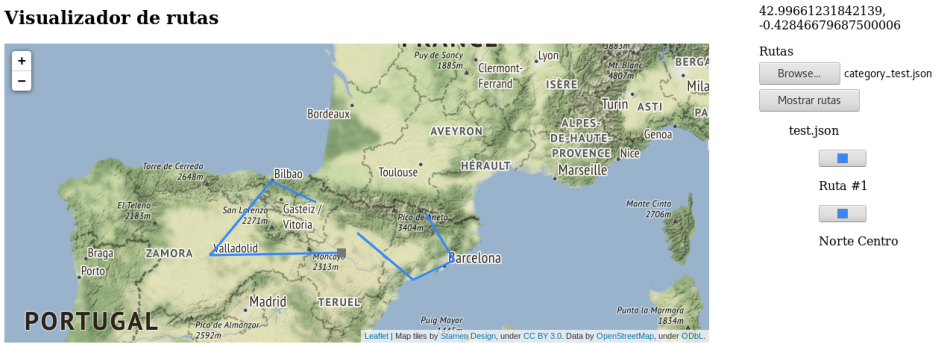


Figura 5.15: Visualización de mapas - categorías con una ruta eliminada.

Las siguientes secciones describirán los resultados del reconocimiento a partir de un mismo objetivo, determinado mediante la configuración de parámetros y aplicado al dataset de Gowalla (que tiene más usuarios que Brightkite). Las imágenes mostradas corresponden a las de mayor número de check-ins para ese usuario, lo cual permite visualizar las distintas formas de detección de ruta que el filtro es capaz de reconocer, a la vez que nos permite comprobar visualmente si lo reconocido concuerda con la definición de ruta buscada. Para facilitar la distinción entre rutas, se han aleatorizado los colores de las mismas.

5.2.1. Umbrales fijos

Estos son los filtros más básicos, que separan una ruta de la siguiente cuando una magnitud concreta supera un valor fijo.

Por su naturaleza son filtros estrictos con una definición muy concreta de ruta. Esto es una ventaja si la definición de ruta está bastante acotada, pero a medida que se aumente el umbral, también lo hará el número de falsos positivos.

Total menor que constante

Cuando la suma de cierta magnitud entre check-ins (por ejemplo, tiempo o distancia) supera un umbral, ya consideramos que el siguiente check-in pertenece a una ruta distinta. Un ejemplo claro de filtro es el de un día natural, en el que el recorrido de un turista puede ser válido para otros en la misma zona geográfica. Aunque el día tiene 24 horas, al menos 4 deberían consagrarse al sueño, por tanto veamos qué rutas resultan de poner un rango máximo de 20 horas (ver Figura 5.16).

Al no haber restricciones geográficas, es posible apreciar este fenómeno del usuario 'haciendo escalas' hasta que transcurran las 20 horas configuradas.

Un segundo usuario (Figura 5.17) también tuvo unos pocos viajes de larga distancia, pero la mayoría se concentran en Estocolmo, con varios viajes a otras ciudades importantes de los alrededores. De manera similar, el usuario que se muestra en la Figura 5.18 sería el caso más aprovechable a la hora de buscar rutas turísticas, dado que el usuario no ha salido de Japón, pero ha hecho bastantes recorridos por el centro del país, muchos de ellos pasando por la capital, Tokio.

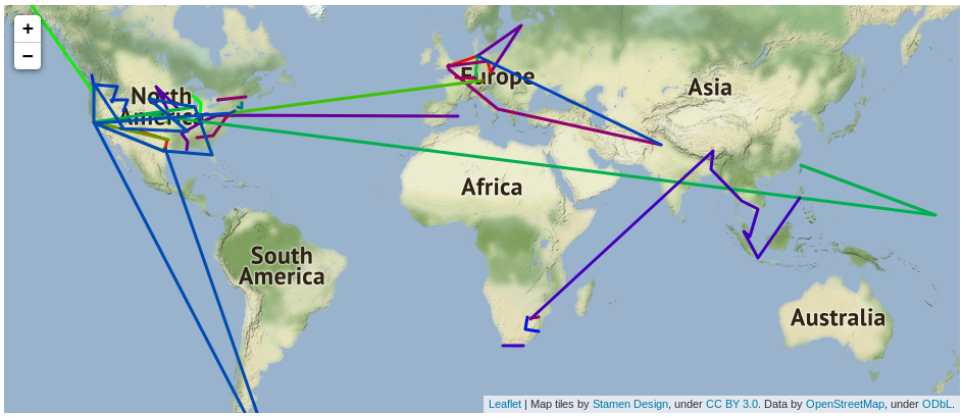


Figura 5.16: Dataset Gowalla - usuario con ID 777.

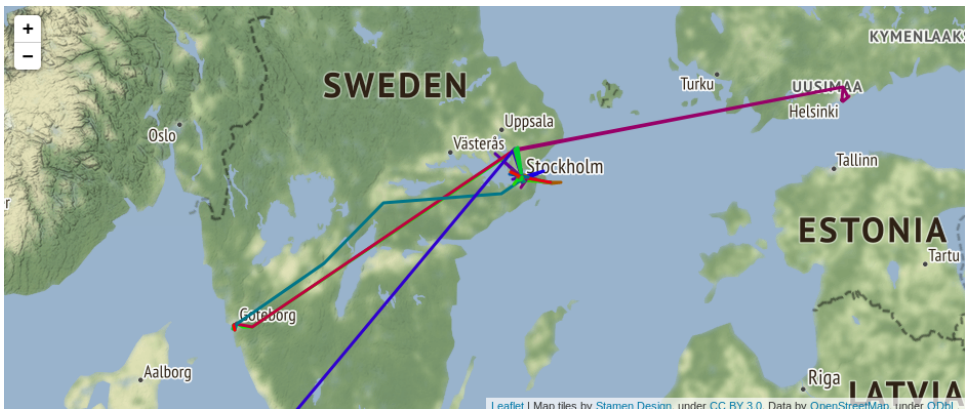


Figura 5.17: Dataset Gowalla - usuario con ID 18932.

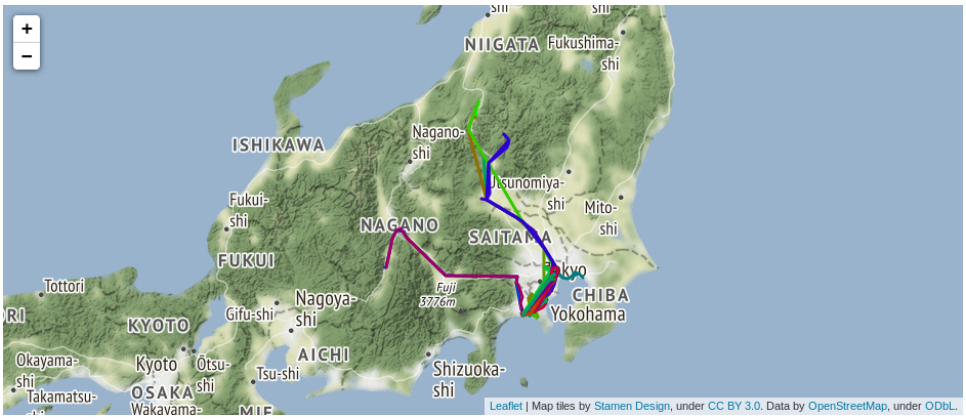


Figura 5.18: Dataset Gowalla - usuario con ID 621.

Distancia entre check-ins menor que constante

Este método separa dos rutas si la distancia (u otra magnitud dada) entre dos check-ins supera un valor. Una posible aplicación es el reconocimiento de rutas entre varias regiones: al acabar la ruta se recorrerá poca distancia en mucho tiempo, por tanto el siguiente check-in ya pertenecerá a otra. Nótese que en este caso la magnitud es compuesta (distancia frente a tiempo).

Siguiendo el ejemplo de una ruta de un día, si definimos que una ruta tiene al menos 3 check-ins, en el caso más extremo tendremos 4 horas 40 minutos entre check-ins. Siempre y cuando

los puntos de interés sean específicos (que un punto de interés sea Palacio de Cristal en lugar de Parque del Retiro), es razonable no poner un umbral más alto porque, en el caso de check-ins más separados aún, significaría que los puntos de interés se concentran en un área más reducida (lo cual ya detecta el filtro), o que un punto de interés consume mucho tiempo en disfrutar, en cuyo caso no es una ruta.

Con este límite superior ejecutamos el algoritmo sobre el dataset de Gowalla. La Figura 5.19 muestra algunas rutas extraídas. El usuario aquí representado sólo tiene tres rutas, lo cual muestra que ha estado en zonas cuya densidad de puntos de interés ha sido alta, en promedio unos 3,700 por ruta.

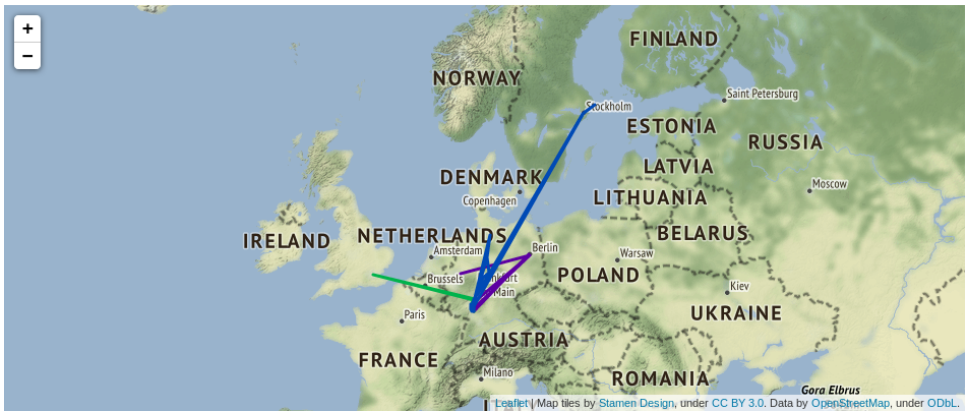


Figura 5.19: Dataset Gowalla - usuario con ID 38496.

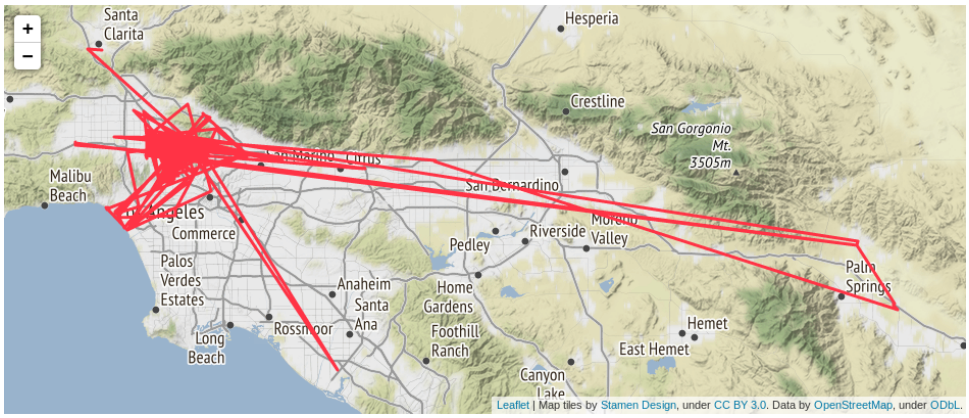


Figura 5.20: Dataset Gowalla - usuario con ID 4915.

Más extremo es el caso del usuario mostrado en la Figura 5.20, con una única ruta, pero que sin embargo mantuvo el ritmo para registrar muchos puntos de interés de California, Estados Unidos. Por otro lado, en la Figura 5.21 se ven todas las rutas del usuario, con la salvedad de unas pocas rutas en un viaje a Miami. El motivo de la omisión no es otro que para resaltar la concentración de rutas en Kansas City, donde es más probable que se pueda extraer una ruta verdaderamente turística.

5.2.2. Umbrales relativos

Estos filtros son capaces de reconocer una variedad mayor de rutas, ya que no se rigen por una constante sino por la tendencia que tiene un recorrido en un momento dado.

Figura 5.21: Dataset Gowalla - usuario con ID 444.

Comienza en atípico

De nuevo, teniendo en cuenta una magnitud, agrupamos check-ins a medida que se leen, verificando que ningún valor nuevo se distancie de la media más de una constante (es decir, el cambio porcentual para llegar desde la media a ese valor es menor que una constante), y que la varianza del conjunto completo se mantenga por debajo de un umbral.

Al controlar que los valores sean más o menos dispares controlamos el margen de 'regularidad' de la ruta, lo cual significa que, por ejemplo, con poco margen, un turista solo realice recorridos cortos, frente a otro que podría cambiar de ciudad unos días y volver.

En función de la implementación, se pueden reconocer varios tipos de ruta con el mismo filtro, los mismos parámetros y los mismos check-ins, como pueden ser rutas a varias escalas de la misma magnitud. Así es posible sacar check-ins de distintas rutas, posiblemente conectados con atípicos para cualquiera de esas rutas, para detectar otros tipos de turismo (por ejemplo, con otros medios de transporte).

Volviendo al ejemplo de una ruta de un día, para detectarla tendremos que asumir alguna propiedad sobre el comportamiento del turista en estos casos. Suponiendo que duerme 8 horas seguidas de las 24 del día, la desviación típica máxima que puede tener el conjunto es de 8 (con x check-ins a la hora de despertarse y los mismos a la hora de dormir), en cuyo caso la desviación de la media máxima de un nuevo elemento sería del 100 %. Todo check-in que cambie la desviación típica o supere la distancia de la media, por tanto, pertenecerá a otra ruta.

Probando estos parámetros hemos ejecutado el filtro para el dataset de Gowalla, y he aquí algunas rutas extraídas. Es interesante observar en la Figura 5.22 que, a pesar de todos los check-ins (casi $4,300$), y la distancia entre ellos, se ha reconocido solo una ruta. Esto es posible gracias a que el algoritmo sólo restringe la variabilidad de los datos, no sus cantidades, y por tanto, por ejemplo, haciendo un check-in todos los días a la misma hora, la desviación típica se puede mantener mínima y sin embargo viajar por todo el mundo.

Por otro lado, el efecto antes mencionado es el motivo por el cual en la Figura 5.23 también se reconoce sólo una ruta, convirtiéndola más bien en una bitácora de viaje (que puede ser otro tipo de recorrido a estudiar). Este usuario es el mismo que el mostrado en la Figura 5.16, pero en aquel caso se reconocieron 56 rutas, que se asemeja más al concepto de ruta turística tradicional.

Por último, aunque en la Figura 5.24 también se encuentra una única la ruta, la zona geográfica es más reducida, y a partir de aquí se podría aplicar otro algoritmo para separar en rutas verdaderamente turísticas el recorrido de un usuario.

Figura 5.22: Dataset Gowalla - usuario con ID 18932.

Figura 5.23: Dataset Gowalla - usuario con ID 777 (atípico).

Figura 5.24: Dataset Gowalla - usuario con ID 10972.

5.3. Experimentos del sistema completo

La primera prueba que se realizó para el sistema completo fue con un dataset creado a mano, con el que fuera posible comprobar visualmente que las rutas detectadas por el filtro eran las esperadas.

El filtro era de total fijo para la magnitud de tiempo, es decir, la ruta es el conjunto máximo de check-ins consecutivos tal que las marcas de tiempo mínima y máxima no superan un umbral fijo. En este caso, además debe haber en cada conjunto al menos tres check-ins.

En 0.3 segundos (según el comando 'time') procesó 10 check-ins, generando las carpetas y archivos con rutas pertinentes. El resultado de la visualización se puede ver en la Figura 5.25, y muestra correctamente tres rutas distintas; los colores fueron aplicados a mano.

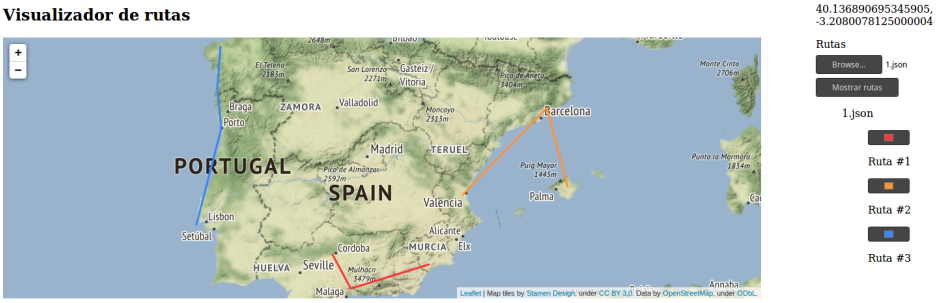


Figura 5.25: Visualización de un dataset pequeño.

La segunda prueba se llevó a cabo con uno de los conjuntos de datos reales (Gowalla) y el anterior filtro, para las magnitudes de tiempo y distancia euclideana fueron generadas todas las rutas del conjunto, y luego cargadas a mano algunas de las rutas reconocidas.

El filtro recibió los parámetros para filtrar rutas de mínimo 5 check-ins en un día o en 2 kilómetros, según la magnitud estudiada.

En 14 minutos 6 segundos, la herramienta de detección fue capaz de procesar los 6.4 millones de check-ins sin fallo visible. Posteriormente cargamos varias rutas a mano (ver Figura 5.26), de las dos magnitudes, para un total de 86 archivos con 2,286 rutas.

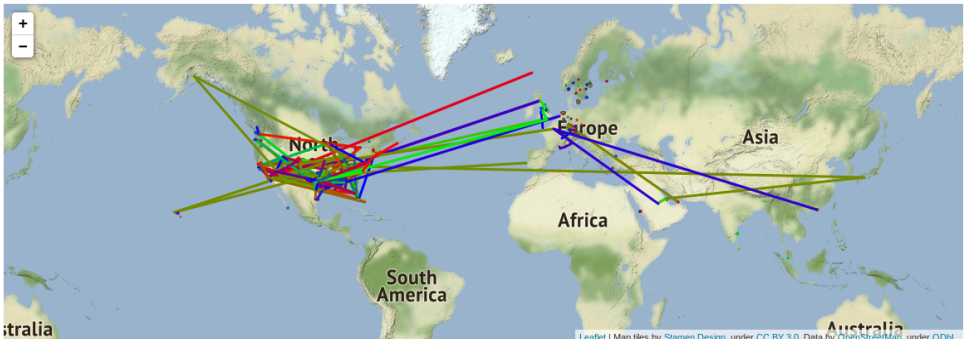


Figura 5.26: Visualización del dataset Gowalla.

Las rutas acotadas por distancia resultaron, con diferencia, mucho más cortas que la mayoría de rutas acotadas por tiempo (de hecho, por el nivel de aumento, solo se ven las segundas), lo cual nuevamente recalca que la disponibilidad de medios de transporte de más largo alcance disminuye la representatividad del tiempo, por tanto las variables de distancia deben ser más valoradas a la hora de detectar rutas turísticas.

Capítulo 6

Conclusiones y trabajo futuro

6.1. Conclusiones

El principal fruto de este proyecto ha sido la posibilidad, a partir de algoritmos sencillos, de extraer algunas rutas consideradas turísticas, por lo cual es certero concluir que analizando los check-ins de un usuario como un grafo dirigido ponderado es posible extraer subgrafos con relación semántica.

En el transcurso del proyecto se ha hecho uso de técnicas de la ingeniería del software para estructurar este documento, conocimientos de minería de datos y sistemas de recomendación para, a grandes rasgos, estudiar tanto las posibilidades de las herramientas y otras investigaciones como la relevancia, tras aplicar algunos filtros, de los resultados frente al objetivo de buscar rutas turísticas, y nociones de análisis de algoritmos para hacer algunas mejoras de rendimiento de la herramienta (cuya optimización estaba fuera del ámbito del proyecto). Por ello, este trabajo me ha servido para aplicar muchas de las asignaturas aprendidas durante la carrera, así como para especializarme y profundizar en el conocimiento de otras áreas concretas.

6.2. Trabajo futuro

Durante el desarrollo de este proyecto surgieron distintas alternativas que, por falta de tiempo, no se pudieron abordar. En esta sección describimos algunas de ellas, así como extensiones de las mismas o de las técnicas y herramientas descritas en este documento.

Para empezar, la herramienta de detección de rutas es mejorable en varios aspectos, ya que consiste en un prototipo enfocado a corroborar o desmentir la hipótesis de que se pueden reconocer rutas minimizando información contextual. Entre las posibles para este subsistema del proyecto se encuentra la creación de un lenguaje de dominio específico (aquel creado no para propósitos generales, sino para exclusivamente los objetivos del producto), de modo que sea posible aplicar varios filtros secuencialmente en el mismo conjunto de datos, siguiendo condicionales especificados en la propia solicitud. Esto enriquecería los resultados permitiendo aprovechar filtros ya creados con interacciones complejas sin necesidad de implementar un único filtro mucho más complejo.

Una extensión que se debería abordar en el futuro es la validación de las rutas detectadas de alguna forma, a poder ser, comparándolas frente a un conjunto de datos donde las rutas ya estuvieran identificadas y etiquetadas por usuarios reales.

Otra posible mejora es el rendimiento de la ejecución: aunque en un PC con prestaciones medias es capaz de procesar más de 4 millones de check-ins en pocos minutos para filtros simples,

lo cual es suficiente para un uso personal, una aplicación a gran escala necesitaría obtener resultados con filtros más complejos (que utilicen decenas de parámetros o más) y más check-ins en menor tiempo.

En tercer lugar, y sin dejar de tener en cuenta el punto anterior, ayudaría a la escalabilidad del proyecto la unificación de los lenguajes usados para la herramienta de detección en uno (a diferencia de como está ahora, combinando lenguaje Python y consola sh). Esto disminuiría las penalizaciones temporales surgidas al guardar la información del flujo de datos en ficheros intermedios y el esfuerzo necesario para entender de manera específica cómo se manipula la información en el transcurso de la ejecución, aunque habría que tener cuidado en seguir manteniendo la separación del código en módulos que puedan intercambiarse adecuadamente de manera permanente, bien a través de configuraciones.

Glosario de acrónimos

- **API**: Application Programming Interface
- **GPS**: Global Positioning System
- **LBSN**: Location-based Social Network
- **OMT**: Organización Mundial del Turismo, de la Organización de Naciones Unidas
- **PC**: Personal Computer
- **POI**: Point of Interest
- **POSIX**: Portable Operating System Interface for Unix
- **SNAP**: Stanford Network Analysis Project
- **UE**: Unión Europea
- **Wi-Fi**: Tecnología de red inalámbrica

Bibliografía

[1] http://www.ine.es/infografias/infografia_turismo_viajeros.pdf. Accessed: 2018-01-10.

[2] http://appsso.eurostat.ec.europa.eu/nui/submitViewTableAction.do. Accessed: 2018-01-10.

[3] http://ec.europa.eu/eurostat/statistics-explained/index.php/Tourism_statistics_-_characteristics_of_tourism_trips. Accessed: 2018-01-10.

[4] https://press.atairbnb.com/whats-driving-airbnb-experiences-one-year-later-foodies-mille
Accessed: 2018-01-10.

[5] http://about.americanexpress.com/news/docs/2015x/Amex_Travel_Future_Trends_7-9-15.pdf. Accessed: 2018-01-10.

[6] https://www.economist.com/news/leaders/21645180-smartphone-ubiquitous-addictive-and-trans
Accessed: 2018-01-10.

[7] https://foursquare.com/about/. Accessed: 2018-01-10.

[8] https://support.foursquare.com/hc/en-us/articles/201908440-What-is-Foursquare-Swarm-. Accessed: 2018-01-10.

[9] https://support.foursquare.com/hc/en-us/articles/201065340-Check-ins. Accessed: 2018-01-10.

[10] https://developer.foursquare.com/docs/users/checkins. Accessed: 2018-01-10.

[11] https://snap.stanford.edu/data/loc-gowalla.html. Accessed: 2018-01-10.

[12] http://blog.gowalla.com/. Accessed: 2018-01-10.

[13] What to do when your favorite site closes. http://edition.cnn.com/2013/05/02/tech/web/when-a-site-closes/index.html. Accessed: 2018-01-10.

[14] https://techcrunch.com/2009/04/07/mobile-socializing-limbo-merges-with-brightkite-and-ann
Accessed: 2018-01-10.

[15] https://brightkite.com/blog. Accessed: 2018-01-10.

[16] https://techcrunch.com/2010/06/21/loopt-background-location-iphone/. Accessed: 2018-01-10.

[17] https://techcrunch.com/2006/09/11/loopt-to-make-mobile-presence-usable/. Accessed: 2018-01-10.

[18] https://techcrunch.com/2008/09/28/the-state-of-location-based-social-networking-on-the-i
Accessed: 2018-01-10.

[19] https://techcrunch.com/2010/02/17/loopt-another-content-partner-integrates-local-foodie-r Accessed: 2018-01-10.

[20] http://mashable.com/2012/03/09/loopt-acquired/#2ZTROF9ZL8qm. Accessed: 2018-01-10.

[21] https://www.microsoft.com/en-us/research/publication/geolife-gps-trajectory-dataset-user-guide/. Accessed: 2018-01-10.

[22] http://www.glympse.com/. Accessed: 2018-01-10.

[23] https://www.forbes.com/sites/kashmirhill/2013/06/26/glympse-is-snapchat-for-location-sharing/. Accessed: 2018-01-10.

[24] http://www.skyhookwireless.com/. Accessed: 2018-01-10.

[25] http://www.skyhookwireless.com/privacy. Accessed: 2018-01-10.

[26] https://developers.google.com/maps/documentation/geolocation/intro. Accessed: 2018-01-10.

[27] http://mashable.com/2013/05/14/apple-location-data-stalk-users/#fh6FOoSrjPqr. Accessed: 2018-01-10.

[28] https://www.locationaware.usf.edu/ongoing-research/technology/path-prediction/. Accessed: 2018-01-10.

[29] https://www.google.com/patents/US20100070171. Accessed: 2018-01-10.

[30] Munmun De Choudhury, Moran Feldman, Sihem Amer-Yahia, Nadav Golbandi, Ronny Lempel, and Cong Yu. Automatic construction of travel itineraries using social breadcrumbs. In Mark H. Chignell and Elaine G. Toms, editors, *HT'10, Proceedings of the 21st ACM Conference on Hypertext and Hypermedia, Toronto, Ontario, Canada, June 13-16, 2010*, pages 35–44. ACM, 2010.

[31] Kwan Hui Lim. Recommending tours and places-of-interest based on user interests from geo-tagged photos. In Pablo Barceló and Heng Tao Shen, editors, *Proceedings of the 2015 ACM SIGMOD PhD Symposium, Melbourne, VIC, Australia, May 31 - June 04, 2015*, pages 33–38. ACM, 2015.

[32] https://developers.google.com/maps/documentation/javascript/markers. Accessed: 2018-01-10.

[33] https://developers.google.com/maps/documentation/javascript/shapes. Accessed: 2018-01-10.

[34] https://developers.google.com/maps/documentation/javascript/directions. Accessed: 2018-01-10.

[35] https://developers.google.com/maps/documentation/directions/usage-limits. Accessed: 2018-01-10.

[36] https://developers.google.com/maps/documentation/static-maps/. Accessed: 2018-01-10.

[37] https://batchgeo.com/features/embedded/. Accessed: 2018-01-10.

[38] https://wiki.openstreetmap.org/wiki/Frameworks. Accessed: 2018-01-10.

[39] https://developer.skobbler.com/#devPlansSection. Accessed: 2018-01-10.

[40] https://developer.here.com/develop/javascript-api. Accessed: 2018-01-10.

[41] https://developer.here.com/api-explorer/maps-js/v3.0/geoshapes/polyline-on-the-map. Accessed: 2018-01-10.

[42] https://developer.here.com/develop/rest-apis. Accessed: 2018-01-10.

[43] https://developer.here.com/plans. Accessed: 2018-01-10.

[44] http://www.gpsvisualizer.com/. Accessed: 2018-01-10.

[45] http://www.qlik.com/us/products/qlikview. Accessed: 2018-01-10.

[46] https://help.qlik.com/en-US/qlikview/12.1/Subsystems/Client/Content/Scripting/script-syntax.htm. Accessed: 2018-01-10.

[47] http://www.qlik.com/us/products/qlikview/personal-edition. Accessed: 2018-01-10.

[48] https://www.wolfram.com/mathematica/new-in-10/geographic-visualization/. Accessed: 2018-01-10.

[49] http://support.sas.com/documentation/cdl/en/vaug/67500/HTML/default/viewer.htm#n1687i1a1vc6y7n1po1rpl6f9ldy.htm. Accessed: 2018-01-10.

[50] https://www.wolfram.com/mathematica/trial/. Accessed: 2018-01-10.

[51] https://www.sas.com/en_us/software/visual-analytics.html. Accessed: 2018-01-10.

[52] http://www.wolfram.com/mathematica/pricing/students-individuals.php. Accessed: 2018-01-10.

[53] https://www.w3.org/TR/NOTE-datetime. Accessed: 2018-01-10.

[54] https://tools.ietf.org/html/rfc7946. Accessed: 2018-01-10.

[55] http://leafletjs.com/. Accessed: 2018-01-10.

[56] https://stamen.com/about/. Accessed: 2018-01-10.

[57] http://maps.stamen.com. Accessed: 2018-01-10.

Apéndice A

Manual de utilización

A.1. Sistema de generación de rutas

A.1.1. Prerrequisitos

Para la correcta ejecución del generador debe cumplirse lo siguiente:

1 Tener instalado Python 2.7

2 Tener un entorno de shell adherido al estándar POSIX

3 Los archivos deben ser legibles, y los scripts ejecutables, para el usuario que ejecute el generador

4 Opcionalmente, si se desea utilizar filtros específicos se deben añadir los ficheros .py en la carpeta codigo/scripts/filters, siendo el nombre del fichero, sin la extensión, el nombre del filtro.

5 Opcionalmente, para usar filtros y/o parámetros distintos a los de por defecto, estará en una ubicación accesible por el usuario que ejecute el script el fichero JSON que describe las magnitudes, filtros y parámetros a utilizar.

A.1.2. Petición de filtrado

Una petición de filtrado es un archivo JSON que describe qué magnitudes se van a utilizar para reconocer rutas, qué filtros se aplicarán sobre ellas, y qué parámetros adicionales reciben estos filtros.

Salvo por los comentarios (el resto de cualquier línea a partir del símbolo #), esta petición es JSON válido con la siguiente estructura:

```
JSON :: { magnitudes }
magnitudes :: magnitud | magnitud, magnitudes
magnitud :: "nombre_magnitud" : [ lista_filtros ]
nombre_magnitud :: cadena
lista_filtros :: filtro | filtro, lista_filtros
filtro :: "nombre_filtro" : [ lista_parametros ]
nombre_filtro :: cadena
lista_parametros :: "parametro" | "parametro", lista_parametros
parametro :: cadena
```

Los nombres de magnitud son una conveniencia a la hora de nombrar los archivos y carpetas, por lo cual debe contener caracteres aceptables para los mismos (lo cual depende del sistema operativo y sistema de ficheros). Los campos que se utilizan para los filtros, y por tanto los gráficos, se especifican en la lista de parámetros. Por otro lado, los nombres de los filtros coinciden con el nombre del fichero a utilizar, quitando la extensión. Por último, los parámetros se pasan como argumentos al filtro, y no deben contener espacios. Estos determinan los campos que se utilizan para el cómputo tanto de rutas como gráficos.

Una posible petición de filtrado puede ser la siguiente:

```
{
"time" :
{
                #5 check-ins en un día
                "n_within" : ["5", "3", "72000", "1,3,4"],
                #5 check-ins separados por menos de 4 horas, 4 minutos
                "n_below" : ["1", "14640", "3", "1,3,4"],
                "break_outlier" : ["1", "3", "1", "8", "1,3,4"]
        },
"dist" :
{
"break_outlier" : ["2", "5", "0.75", "2500", "2,3,4"]
}
}
```

Nótese que la cantidad de filtros por cada magnitud no tiene por qué coincidir.

Aunque no es imprescindible, en la lista de parámetros se esperan por lo general al menos dos parámetros: campo por el cual filtrar, y número mínimo de check-ins para considerar un conjunto de check-ins una ruta.

A.1.3. Ejecución

En una terminal en el directorio de codigo/scripts, basta con ejecutar:

```
$> sh generate_graphs.sh <conjunto_datos> <filtros>
```

Donde <conjunto_datos>es la ubicación del conjunto de datos a procesar y <filtros>la ubicación del fichero de petición de filtrado. Este segundo argumento es opcional.

A.2. Sistema de visualización

A.2.1. Prerrequisitos

1 Navegador Google Chrome 45, Firefox 22, Opera 32, Safari 10, o superior

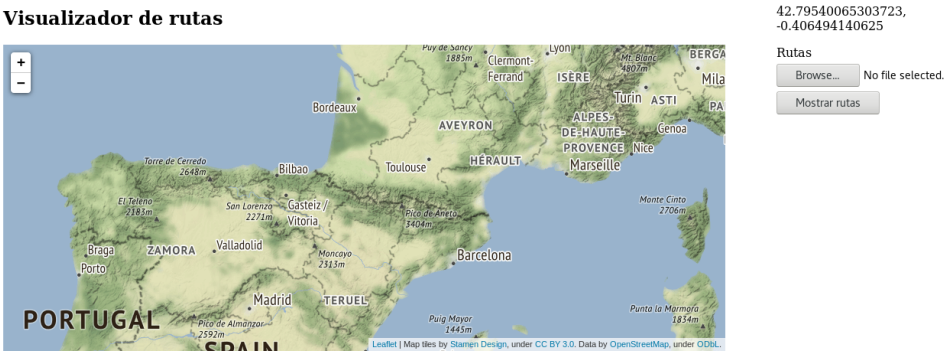
2 JavaScript habilitado

3 Pop-ups habilitados

4 Aunque no es imprescindible, se recomienda tener conexión a Internet, ya que de otro modo no se verá la geografía del mapa.

A.2.2. Ejecución

Al llegar a la vista principal por primera vez veremos una imagen como la siguiente:



Cargar rutas de un usuario

Debajo del título «Rutas» se encuentra un botón para subir ficheros. Tras hacer click en él se abrirá una ventana que permite subir un fichero de rutas (esto depende del sistema operativo). Una vez seleccionado el archivo, haciendo click en el botón de mostrar rutas se mostrarán en el mapa

Cambiar el color de una ruta

Hacer click sobre el recuadro de color situado encima del nombre de la ruta, seleccionar un color y aceptar.

Eliminar una o más rutas

Hacer click sobre el nombre de la ruta, y en la ventana de confirmación, aceptar. También se puede hacer click sobre el nombre del archivo para eliminar todas las rutas cargadas en ese archivo.

Cargar puntos de interés y categorías

El mismo método para subir rutas se puede usar para subir categorías. La aplicación distingue unas de otras por el contenido del fichero.

Cambiar el color de una categoría

Para cambiar el color de los puntos de interés de una categoría, hacer click sobre el recuadro de color encima del nombre de la categoría, seleccionar el nuevo color, y aceptar los cambios.

Los puntos de interés en gris no tienen categoría y su color no puede ser cambiado.

Eliminar puntos de interés

Para eliminar los puntos de interés de una categoría concreta, basta con presionar sobre el nombre de la categoría y aceptar en la ventana de confirmación.