UNIVERSIDAD AUTONOMA DE MADRID

Escuela Politécnica Superior

21|22

Trabajo fin de máster

Estudio y análisis de minería de trayectorias para recomendación de rutas deportivas

Álvaro Sosa Herrera

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

Proyecto para la obtención del título de
Máster en Investigación e Innovación en
Inteligencia Computacional y Sistemas Interactivos
(MU I2-ICSI)
por la Universidad Autónoma de Madrid

Tutor del trabajo fin de máster:
Alejandro Bellogin Kouki

Estudio y análisis de minería de trayectorias para recomendación de rutas deportivas

Álvaro Sosa Herrera

Álvaro Sosa Herrera
Estudio y análisis de minería de trayectorias para recomendación de rutas deportivas

Álvaro Sosa Herrera
C\ Francisco Tomás y Valiente Nº 11

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

A mi familia y amigos.

Agradecimientos

A mis padres Emilia y José por ayudarme a llegar hasta aquí.

A mi tutor Alejandro por su apoyo y guia continua.

Y por último pero no menos importante a mis compañeros Sergio, Álvaro, Victor y Alfonso. Gracias por haberme acompañado todos estos años.

Resumen

El uso de aplicaciones con las que monitorizar el rendimiento deportivo se ha convertido en una costumbre muy arraigada en las sociedades actuales. En estas aplicaciones las personas comparten sus marcas, fotos, comentarios y recorridos a fin de mostrarles a los demás sus progresos y capacidades. Aunque tratándose de un sector en auge, no existen muchos mecanismos capaces de sugerirle a un usuario recorridos que podrían interesarle en una ciudad nueva o ayudarle a descubrir nuevas rutas que le puedan interesar de una manera personalizada y acorde a sus capacidades y características.

Así, en este trabajo se abordan dos puntos fundamentales. Primeramente, la construcción y estudio de un conjunto de datos obtenidos a partir de una red social de carácter deportivo. Seguido de la implementación de un sistema capaz de generar recomendaciones a los usuarios de dicho conjunto sobre qué rutas o recorridos podrían resultarles de interés acorde a sus preferencias.

Para obtener los datos se han estudiado y comparado un grupo de sitios webs que poseyeran una interfaz de programación de aplicaciones de acceso gratuito y en la cual se encontrasen datos y características de usuarios con sus recorridos registrados en la aplicación. Estos datos posteriormente son estudiados y tratados para poder ser empleados por los algoritmos de recomendación elegidos.

Los algoritmos de recomendación empleados se han seleccionado acorde a la naturaleza de los datos y con el fin de optimizar y aprovechar sus características intrínsecas de la forma más intuitiva posible. Estos han sido la factorización de matrices y las máquinas de factorización.

Las métricas con las que se han evaluado los métodos han sido *precision* y *recall*, las cuales a pesar de su sencillez permiten obtener un amplio abanico de conclusiones y variedad de resultados. Finalmente, a pesar de que los porcentajes de acierto en las predicciones no resultan particularmente altos, los métodos han demostrado reaccionar mejor o peor ante ciertas características, mostrando mayor o menor afinidad hacia ciertos tipos de datos, sentando así las bases para trabajos futuros en lo que a recomendación de rutas se refiere.

Palabras clave

Ruta deportiva, usuario, perfil, sistema de recomendación, algoritmo, dataset, factorización de matrices, máquinas de factorización

Abstract

The use of apps to monitor sports performance has become a deeply rooted habit in today's society. In these applications, people share their marks, photos, comments, and routes to show others their progress and capabilities. Although it is a booming sector, there are not many mechanisms capable of suggesting to a user routes that may interest her in a new city or helping her to discover new routes that may interest her in a personalized way and according to her capabilities and characteristics.

Thus, this work addresses two fundamental points. First, the construction and study of a dataset obtained from a sport social network. Then, the implementation of a system capable of generating recommendations to users of this dataset about which routes or tours may be of interest to them according to their preferences.

To obtain the data, we have studied and compared a set of web pages that have free access application programming interfaces, where we could find data and characteristics of users with their registered routes in the application. These data are studied and processed to be used by the selected recommendation algorithms.

The exploited recommendation algorithms have been selected according to the nature of the data and in order to optimize and take advantage of their intrinsic characteristics in the most intuitive possible way. These techniques, at the end, have been Matrix Factorization and Factorization Machines.

The metrics with which the methods have been evaluated were Precision and Recall, which despite their simplicity allow a wide range of conclusions and variety of results to be obtained. Finally, although the percentages of correct predictions are not particularly high, it has been shown that the methods react better or worse to certain characteristics, showing more or less affinity for certain types of data, which lays the groundwork for future work in terms of route recommendations.

Keywords

Sport path, user, profile, recommender system, algorithm, dataset, matrix factorization, factorization machines

Índice

Listas

Lista de ecuaciones

Lista de figuras

Lista de tablas

1

Introducción

Las Redes sociales (RRSS) se han consolidado como un elemento de interconexión nunca antes visto [1]. Dentro de esta inmensidad encontramos múltiples redes acerca de multitud de temas, podemos visualizar y compartir vídeos, canciones, noticias, reseñas, opiniones... Entre este amplio abanico de opciones encontramos las redes sociales de carácter deportivo, las cuales son empleadas por los usuarios para compartir sus recorridos, comentar sus actividades, comparar sus mejores marcas o planificarse entrenamientos. Como por ejemplo el sitio web Strava [1]. Este se trata de una red social de carácter deportivo que conecta atletas de multitud de puntos del globo, llegando a contar en 2021 con 74 millones de usuarios de más de 200 países [2]. Esta enorme cantidad de usuarios genera un importante torrente de información, que puede aprovecharse para mejorar la funcionalidad de estas aplicaciones y ampliar el horizonte de posibilidades que pueden ofrecer a sus usuarios.

Entre esta información generada por los usuarios encontramos las rutas, parte fundamental de las aplicaciones deportivas. Una ruta dentro del contexto de estas RRSS consiste en una sucesión de puntos geográficos o coordenadas conectadas entre sí formando un camino o vía que puede ser empleada por un usuario para recorrer una área, y que conforman el elemento central en torno al que giran muchas de estas aplicaciones. Las rutas no solo se recorren, sino que se puntúan, se comentan y se comparten haciendo en estos casos las veces de componente de interés en torno al que se reúnen los usuarios de estas plataformas. Las características que definen una ruta son muy complejas: duración aproximada, elevación máxima y mínima, desnivel promedio o hasta incluso el terreno donde se desarrolla son solo algunas de las propiedades que pueden evaluarse para conformar el perfil de una ruta. Esta dificultad ha convertido a las rutas en un auténtico reto en términos de planificación y recomendación, existiendo poca base previa en lo que a investigación y trabajos se refiere. Por ello, un sistema capaz de analizar, comparar y ofrecer rutas que podrían ser del interés de un usuario y a las que no podría acceder por sus propios medios se ha convertido en un objetivo interesante a perseguir a día de hoy.

[1] Strava.com, https://www.strava.com/ [Fecha de acceso: 24-08-2021]

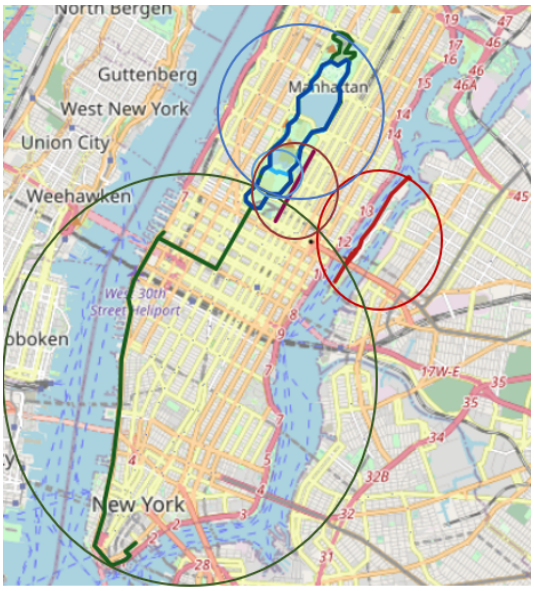

Figura 1.1: En colores, rutas de actividades deportivas realizadas por usuarios en distintas calles de la ciudad de Nueva York y registradas por la aplicación Srava.

Para ello, en este Trabajo de fin de Máster (TFM) se propone el empleo de Sistemas de recomendación (RS). Los RS se tratan de una serie de técnicas y herramientas software empleadas para sugerir artículos a un usuario particular sobre algún dominio o campo en el que se encuentre interesado [3]. Así, los RS juegan un rol protagonista en multitud de sitios web con una enorme cantidad de tráfico de usuarios de múltiples tipos de servicios, como Amazon, Spotify, Alibaba o Facebook, que los emplean para ayudar a sus usuarios a encontrar la mejor alternativa entre la vasta oferta que proporcionan [4]. Todas estas aseveraciones hacen de los RS la herramienta ideal con la que alcanzar el objetivo que este TFM persigue.

Así, se ha desarrollado un sistema de recomendación de rutas deportivas a través de dos aproximaciones de estos, las Máquinas de factorización (FMs) y los métodos de Factorización de matrices (MFs). Esto se ha conseguido mediante un análisis y estudio de la Interfaz de programación de aplicaciones (API) del sitio web Strava para la obtención de los datos, un estudio de los mismos datos con el fin de obtener similitudes, perfiles y características, un proceso de minería de datos para transformar estos desde el formato en crudo en que se reciben hasta una versión manejable y comprensible por un recomendador. Ya finalmente el empleo de varias librerías en lenguaje Python con las que inferir recomendaciones.

Este TFM se organiza en una estructura de capítulos. El capítulo 2 tratará acerca del estado del arte de los recomendadores, técnicas y aproximaciones de recomendación de rutas existentes. El capítulo 3 abarca la implementación de todo el proceso anteriormente mencionado, desde la obtención de los datos a la recomendación pasando por el perfilado y minería de los datos. Finalmente en el capítulo 4 se exponen los resultados alcanzados y se discuten los mismos, siendo el capítulo 5 en el que se hace retrospectiva de cara a mostrar la evolución del proceso.

2

Estado del arte

Actualmente no existe una aproximación con la cual se cubra exactamente el área de este TFM de una manera plena y concisa. Por lo que esta sección va a dar una visión general de los algoritmos, técnicas y paradigmas empleados con el fin de proporcionar una base sobre la que poder asentar el trabajo posteriormente realizado. También existen una serie de estudios relacionados con la compleción de la prueba de la maratón, en los cuales se desarrollan ideas que conforman el grueso de los estudios actualmente publicados que se relacionan con rutas deportivas y cómo trabajar con ellas y recomendarlas. Estos trabajos también se explicarán, a fin de proporcionar un contexto de la situación de los sistemas que trabajan con rutas deportivas en la actualidad.

2.1. Sistemas de recomendación

Como ya se ha comentado en la introducción, durante la historia reciente la masificación de los datos que se manejan en internet han popularizado el uso de los recomendadores. Estos forman parte intrínseca de multitud de páginas que visitamos a diario [5] que los utilizan para crear filtros para los consumidores o navegantes inexpertos en un campo. Haciendo así que estos no se sientan abrumados ante la cantidad de oferta disponible y aumentando de esta manera la accesibilidad a multitud de entornos y productos que un usuario no podría conocer a priori. A continuación se enumeran los conceptos y estrategias más extendidas en lo que a sistemas de recomendación se refiere.

2.1.1. Conceptos básicos

Los RS consisten en una serie de técnicas o herramientas software creadas con el fin de generar recomendaciones que puedan resultar de interés para un usuario sobre un campo concreto.

Así, un sistema de recomendación se compone de 3 elementos fundamentales, usuarios *u*, ítems *i* y valoraciones o recomendaciones *r(u,i)*, que establecen una relación entre usuarios e ítems. Siendo normalmente el objetivo de estos sistemas el de obtener la predicción de la puntuación de un item *i* para un usuario *u* al que dicho ítem le resulta desconocido [6].

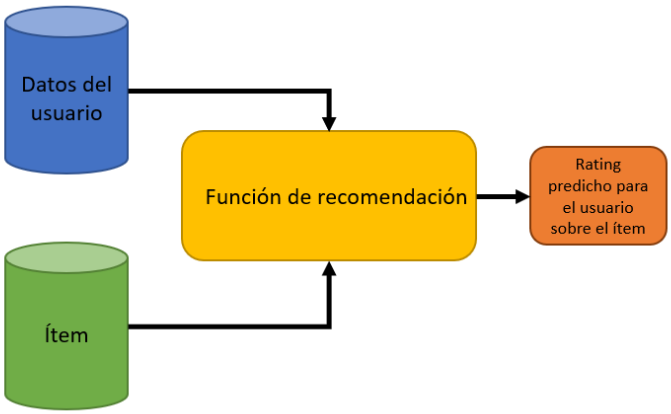

Figura 2.1: Los recomendadores se basan en un estudio detallado de los datos, tanto de usuarios como de ítems, a fin de generar relaciones con las cuales obtener información con la que poder crear recomendaciones para los usuarios. Para esto existen multitud de técnicas distintas en función del campo y las características de los datos sobre los que se están trabajando.

2.1.2. Tipos de sistemas de recomendación

Con el fin de realizar su función, identificar y ofrecer ítems que pueden resultar interesantes para un usuario, un RS debe *predecir* cuales ítems resultarían interesantes para recomendar a un usuario. Para esta tarea existen multitud de aproximaciones que varían en función del dominio en el que se trabaja y de los datos de los que se dispone, ofreciendo alternativas para multitud de escenarios. Las principales técnicas actuales son las que se describen a continuación:

Basadas en contenido: En estas aproximaciones la funcionalidad del sistema consiste en recomendar ítems que sean similares a otros que le hayan gustado al usuario en el pasado. Para ello se extraen las características asociadas a los ítems del usuario, información de cualquier tipo que estos puedan poseer, estructuras que los den forma o metadatos que los identifiquen a fin de disponer de estos para crear un histórico del usuario que poder relacionar con nuevos ítems cuyas características se asemejen a las correspondientes a los ítems del usuario. Las recomendaciones basadas en contenido clásicas generan emparejamientos utilizando palabras claves concretas de entre los datos extraídos mientras que otras técnicas como el indexado semántico emplea conceptos en su lugar. Dentro de este segundo grupo existen técnicas conocidas como de *top-down* y de *bottom-up*, basándose las primeras en la integración de fuentes de datos externos como ontologías o páginas web de carácter enciclopédico mientras que las segundas se basan en la representación semántica ligera basada en la suposición de la importancia de las palabras en función del número de veces que se repitan en los datos de los ítems. Para estas técnicas no es necesario disponer de puntuaciones previas sobre un ítem, simplemente se pueden generar recomendaciones a partir de los propios ítems sin características externas. Como contrapartida, necesitan de una enorme cantidad de información de un usuario, la cual, en caso de ser poco diversa puede llevar al sistema a ofrecer escasa variedad de temáticas [7].

Figura 2.2: En la figura se ilustra el esquema de trabajo de los recomendadores basados en contenido. Estos extraen las características de los ítems del usuario y tras compararlos con los que radican en el sistema mediante una serie de técnicas distintas, proporcionan recomendaciones.

Filtrado colaborativo: Esta metodología genera recomendaciones para los usuarios del sistema basándose en los ítems que marcaron como favoritos en el pasado otros usuarios con perfiles similares al de este. Para calcular la similitud entre los gustos de dos usuarios de cara a establecer una relación entre ellos, estas técnicas pueden emplear el histórico de valoraciones de los usuarios [8]. Entre estas técnicas encontramos las basadas en vecindarios, las cuales pueden concentrarse o bien en las relaciones entre los usuarios o entre los ítems que estos han valorado, decidiendo entre unas u otras en función de la cantidad de usuarios e ítems en el sistema. Mientras que por un lado las relaciones basadas en los ítems proporcionan recomendaciones más eficientes son las basadas en usuarios las que generan recomendaciones más originales haciendo más orgánica y satisfactoria la experiencia original. Por su eficiencia y efectividad el uso de estas técnicas se encuentra muy extendido a día de hoy [9]. Sin embargo, por otro lado, las limitaciones llegan en términos de escasez y cobertura muy limitada, problemas para los cuales se han explorado soluciones relacionadas con reducción de la dimensionalidad y grafos. Las primeras proporcionan una representación más compacta de los usuarios y los ítems entendiendo y capturando sus características más representativas, haciendo las relaciones anteriormente mencionadas realmente significativas. Las segundas, las basadas en grafos, explotan la transitividad de los datos, siendo así que se evitan los problemas ya mencionados de escasez y de limitada cobertura.

Demográficas: Estas técnicas recomiendan basándose en el perfil demográfico de los usuarios. La idea subyace detrás de que las personas han demostrado poseer unos u otros nichos de gustos en función de su perfil demográfico. De esta manera multitud de sitios web ofrecen diferentes recomendaciones en función de la edad de los usuarios, de su idioma o del país donde residen. Estas técnicas se justifican bajo la idea de que, por regla general, usuarios con características personales similares quizás tengan también gustos similares [10].

Basadas en conocimiento: En este escenario las recomendaciones se generan basándose en un dominio de conocimiento específico con el que esclarecer en qué grado un ítem resulta atractivo o en última instancia, necesario, para un usuario. Uno de los principales tipos dentro de estas técnicas son

los conocidos como los basados en casos, donde se estudia hasta qué punto las necesidades de un usuario encajan con las recomendaciones que se le están proporcionando. Estos sistemas son en ocasiones propuestos a fin de evitar problemas relacionados con arranque frío pero si no se encuentran correctamente configurados para aprender correctamente acerca de los usuarios pueden verse superados por otras aproximaciones [11]. Siguiendo esta línea encontramos los basados en comunidades, sistemas basados en los gustos e histórico de los allegados al usuario. Basándose en la idea de que los seres humanos son más propensos a seguir recomendaciones de ítems que han gustado a sus conocidos, que pueden proporcionar testimonios de primera mano, estos sistemas recolectan información del entorno social del usuario [12]. Con las preferencias de los amigos del usuario estos sistemas pueden establecer recomendaciones para el mismo, siendo especialmente utilizados estos sistemas en redes sociales de distinta índole por lo similar de la naturaleza de estas con el paradigma con el que trabajan. Finalmente encontramos los basados en restricciones, los cuales se diferencian de los basados en casos en que estos primeros recomiendan basándose en una serie de reglas predefinidas de cómo se ha de encajar una necesidad de un usuario con las características que ofrecen un ítem.

Técnicas híbridas: Finalmente estas técnicas engloban algunas de las anteriormente mencionadas, tratando de subsanar las deficiencias de una con las virtudes de la otra. Así bien en estas técnicas se persigue aplicar conceptos que son interesantes para un grupo de técnicas en los procesos de otras. Conceptos como bien pudiera ser el contexto o el entorno, que algunas aproximaciones dejan de lado en su idea inicial, pero que posteriormente se ha demostrado que pueden aportar información y variedad. Algunos de los diferentes algoritmos que incorporan esta información son los de prefiltrado, postfiltrado y modelado de contexto [13].

2.1.3. Algoritmos más comunes

Así, con los distintos tipos de técnicas y familias de sistemas de recomendación descritas en el apartado anterior, esta sección se centra en describir varios algoritmos concretos que pertenecen a estas familias. Estos serán divididos en algoritmos personalizados y no personalizados.

No personalizados

Paradigma de algoritmos que no tienen tanto en cuenta el historial del usuario de cara a realizar recomendaciones y se basan en elementos ajenos al historial de valoraciones de este. Estos algoritmos pueden emplearse o bien sobre escenarios en los que existe escasez de información o bien sobre escenarios en los que se quieran realizar pruebas contra el sistema. Entre los más ampliamente extendidos encontramos:

Aleatorios: algoritmos que recomiendan una serie de ítems al azar entre el conjunto disponible. A pesar de no resultar particularmente interesantes por sí mismos permiten establecer baselines útiles con las que comparar algoritmos más elaborados y permiten establecer una idea más clara acerca de hacia dónde está avanzando el recomendador y de distinguir resultados a priori interesantes de recomendaciones aleatorias [14].

Basados en popularidad: un concepto muy simple para obtener rankings consiste en tener en cuenta las valoraciones generales de un ítem de manera que los ítems más recomendados serán aquellos que cuentan con mejor acogida en general entre los usuarios. Todo esto de nuevo respaldado bajo la idea de que es más probable que un usuario se interese por un ítem en el que se están interesando muchos otros. De esta manera, las recomendaciones generadas por este tipo de algoritmos resultan competentes en acierto promedio pero flaquean en cuestiones como por ejemplo personalización, donde se generarán los mismos rankings para multitud de usuarios puesto que estos se generan en función de la recepción del ítem y no del perfil del usuario o de cómo este se adapta al ítem en particular [14].

Personalizados

Estos algoritmos tienen en cuenta el historial previo del usuario y usan estos datos para generar recomendaciones más originales y novedosas que los mencionados en el apartado anterior.

Recomendación basada en contenido: La mayoría de los recomendadores basados en contenido usan modelos relativamente simples de recuperación como por ejemplo el emparejamiento de palabras o Vector Space Model (VSM). Este último es una representación espacial de documentos de texto en el cual cada documento es representado en un vector de espacio n-dimensional donde cada dimensión corresponde a un término del vocabulario general de la colección de documentos dada. Con ello, cada documento es representado como un vector de pesos donde cada peso indica el grado de asociación entre documento y término. Así, cada documento d_j es representado como el vector (w_{1j}, w_{2j}, ..., w_{nj}) donde w_{kj} es el peso del término k en el documento j. El esquema más comunmente empleado en estos casos resulta ser Term Frequency Inverse Document Frequency (TFIDF) [6, p. 140] cuya fórmula podemos ver a continuación:

$$\textit{TF-IDF}(t_k, d_j) := TF(t_k, d_j) \log \frac{N}{n_k} \tag{2.1}$$

Recomendación basada en vecindarios: tipo particular dentro de los RS de filtrado colaborativo los cuales pueden dividirse en 2 tipos de modelos, basados en usuarios y en ítems. En los primeros se obtienen los n usuarios más parecidos al que está siendo evaluado mientras que en los segundos el valor del ítem se genera en función de los valores de otros ítems similares al mismo. Lo cual quiere decir que mientras que en unos se comprueban los usuarios similares al que se quiere recomendar, en los

otros lo que se comprueba son los ítems. Sus numerosas virtudes han hecho de estos unos algoritmos ampliamente utilizados, entre ellas podemos destacar su simplicidad, la cual lleva a que sean sencillos de implementar a parte de la tolerancia que demuestran a la adhesión de nuevos artículos. Así, a pesar de estas virtudes este tipo de algoritmos no produce recomendaciones tan acertadas como pudieran ser por ejemplo las basadas en modelos [15]. Para calcular las similitudes tanto entre usuarios como entre ítems pueden emplearse una serie de métricas, entre las que podemos destacar la Similitud Coseno y la Correlación de Pearson:

Similitud coseno: como su nombre indica la idea que subyace detrás de esta métrica consiste en transformar el elemento que se esté comparando en un vector y aplicar la formula de la similitud coseno. Esta puede emplearse para calcular similitudes tanto entre usuarios como entre ítems. Para calcular la similitud entre usuarios se ha de representar al usuario *u* como el conjunto de ítems sobre los que ha generado una valoración mientras que para el ítem *i* su representación consistirá en el conjunto de los usuarios que lo han evaluado. La ecuación siguiente representa la similitud coseno entre dos ítems *i* y *j* representando *r(u,i)* la valoración del usuario *u* sobre el ítem *i*.

$$\cos(i,j) = \frac{\sum_{u \in U} r(u,i) r(u,j)}{\sqrt{\sum_{u \in U} r(u,i)^2 \sum_{u \in U} r(u,j)^2}} \tag{2.2}$$

Correlación de Pearson: método similar al anterior pero en el que se subsana la ausencia de control sobre la media y la varianza de los usuarios, variantes que no se contemplan en la similitud coseno. Esta correlación se muestra en la siguiente ecuación:

$$\mathit{pearson}(i,j) = \frac{\sum_{u \in U} (r(u,i) - \overline{r}(i))(r(u,j) - \overline{r}(j))}{\sqrt{\sum_{u \in U} (r(u,i) - \overline{r}(i))^2 \sum_{u \in U} (r(u,j) - \overline{r}(j))^2}} \tag{2.3}$$

2.1.4. Factorización de matrices

Algunas de las aproximaciones relativas a factores latentes más exitosas se encuentran basadas en Factorización de matrices (MF). En su descripción más básica puede definirse la MFs como la caracterización de ambos elementos, usuarios e ítems, por sendos vectores de factores inferidos de los patrones de calificación de los ítems. Una alta correspondencia entre los factores de ítems y de usuarios lleva a una recomendación. Estos métodos han cobrado particular protagonismo recientemente debido a su buena escalabilidad e indice de acierto al que podemos sumar la flexibilidad de adaptación que ofrecen a numerosas situaciones. Esencialmente en los recomendadores una dimensión de la matriz representará al usuario mientras que la otra representará los ítems [16]. La figura 2.3 representa un ejemplo de matriz usuario-ítem.

Id usuario	Id carrera	11419250	11417385	19852236	8847877	14727097	14726926	8891972	134122	14263905	11419239	11419214	13531885	9603082	8847882	8856859
10262188		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3370924		1	0	1	0	0	1	0	0	1	1	1	0	1	1	1
9032675		0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0		0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
23665645		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11763137		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figura 2.3: En la figura podemos observar un ejemplo de cómo se relacionan los ítems de un conjunto con los usuarios del sistema. Los 1 simbolizan las carreras que han corrido los atletas de este trabajo en la ciudad de Nueva York mientras que los 0 simbolizan que el usuario no ha corrido la carrera con ese ID concreto. De ella podemos extraer una gran cantidad de información, con la que podemos obtener tanto conjeturas para seguir trabajando posteriormente, como sugerir que dos de los atletas solo han pasado fortuitamente por la ciudad y no residen en ella como afirmaciones inequívocas, por ejemplo, que 3 de los usuarios, con ids 10262188, 23665645 y 11763137 no han corrido nunca en Nueva York.

Existen multitud de modelos populares en lo que a MFs se refiere. En el área de la recuperación de la información Singular Value Decomposition (SVD) se encuentra bien establecido para identificar factores semánticos latentes. Sin embargo aplicarlo sobre clasificaciones explícitas en el dominio del filtrado colaborativo puede dificultarse debido a la elevada cantidad de valores ausentes.

Singular Value Decomposition: los modelos de factorización matricial asignan tanto a los usuarios como a los ítems a un espacio factorial latente de dimensionalidad *f* siendo que las interacciones usuario-ítem son modeladas como el producto interno en ese espacio [6]. El espacio latente trata de explicar las clasificaciones a través de la caracterización de ambos, productos y usuarios, haciendo inferencias con el feedback del usuario. De esta manera cada ítem *i* se encuentra asociado a un vector $q_i \in R^F$ y cada usuario *u* se encuentra asociado a un vector $p_u \in R^F$. Para un ítem *i* su vector q_i cuantifica la medida en la que posee estos factores, bien sea positiva o negativa. Mientras que para el usuario *u* su vector p_u simboliza su grado de interés por los artículos que son altos en sus factores correspondientes. De esta manera el producto de ambos captura la interacción entre los mismos, pudiendo formularse la siguiente regla para la calificación de un usuario *u* sobre el ítem *i*

$$\hat{r}_{ui} := \mu + \beta_i + \beta_u + q_i^T p_u \tag{2.4}$$

Así, con el objetivo de aprender el modelo de los parámetros (b_u, b_i, p_u y q_i) minimizamos el error cuadrático.

$$\sum_{(u,i)\in\kappa}^{n} (r_{ui} - \mu - \beta_i - \beta_u - q_i^T p_u)^2 + \lambda_4(\beta_i^2 + \beta_u^2 + ||q_i^2|| + ||p_u^2||) \tag{2.5}$$

La constante λ_4 que controla el grado de regularización se determina bajo validación cruzada. La minimización sin embargo se lleva a cabo bien por descenso de gradiente o bien de manera estocástica.

Singular Value Decomposition ++ (SVD++): método con el que se mejora el SVD anterior considerando también el feedback implicito que proporciona información adicional del usuario. Este caso es especialmente de ayuda en aquellos usuarios que han proporcionado mucho más feedback implícito que explícito. De esta manera se consigue una precisión mayor de la capaz de alcanzarse con el SVD convencional.

2.1.5. Máquinas de factorización

Las Máquinas de factorización (FMs) se tratan de un modelo que combina las ventajas de los Support Vector Machines (SVM) con los modelos de factorización. Los SVM se tratan de unos de los predictores más popularizados en los campos de la minería de datos y el aprendizaje automático. Sin embargo estos no juegan un papel tan protagonista en campos como el filtrado colaborativo, esto es debido a su incapacidad para aprender parámetros interesantes o confiables en espacios complejos de datos. Así, surgen unos nuevos modelos de predictores, las FMs. Estas se tratan de nuevo de un predictor general como en el caso de los anteriormente mencionados pero que han demostrado ser capaces de estimar parámetros como confiables o no confiables sobre escenarios con mucha ausencia de datos. Entre las ventajas que proporcionan las FMs a parte de lo ya mencionado está el hecho de son linealmente complejas, esto quiere decir que son capaces de escalar a datasets de dimensiones enormes como el de Netflix, que cuenta con hasta 100 millones de elementos de entrenamiento. También pueden trabajar con cualquier vector de características real, en contraste con otros modelos de factorización más estandarizados que solo pueden trabajar en escenarios muy estrictos [17].

La ecuación que representa el modelo de una máquina de una FMs de grado 2 es la siguiente:

$$\hat{y}(X) := w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} (V_i, V_j) x_i x_j \tag{2.6}$$

Donde los parámetros del modelo a estimar son los que se muestran en la ecuación 2.7

$$w_0 \in R, \mathbf{W} \in R^n, \mathbf{V} \in R^{n \times k} \tag{2.7}$$

La fórmula 2.6 representa la interacción entre las variables w_0, el cual se trata del bias global, w_i el cual modela la i-ésima variable y por último (V_i, V_j) el modelo que representa la relación entre la i-ésima y j-ésima variables. Bajo este modelo las FMs son capaces de aplicarse en distintas tareas de predicción entre las que encontramos:

Regresión: $\hat{y}(x)$ puede ser empleado directamente como el predictor siendo el criterio optimizador por ejemplo el error cuadrático medio.

Clasificación Binaria: se usa el signo de $\hat{y}(x)$ y los parámetros son optimizados en función de distintas funciones de pérdida como la logística o la de bisagra.

Ranking: Finalmente se usa el score de $\hat{y}(x)$ para ordenar los vectores de X.

A parte del modelo cerrado mostrado en la ecuación 2.6 los parámetros de modelo de las FMs pueden ser aprendidos de manera eficiente por métodos de gradiente descendiente como por ejemplo Stochastic Gradient Descent (SGD) en una gran variedad de tipos de funciones de pérdida, como la cuadrada, la logística o la de bisagra ya mencionadas. Posteriormente en la figura 2.4 se muestra un ejemplo de vectores de características reales basándose en unos datos de ejemplo. En ella puede observarse el formato que toma con las distintas partes que lo componen y el objetivo o *label* situado al final.

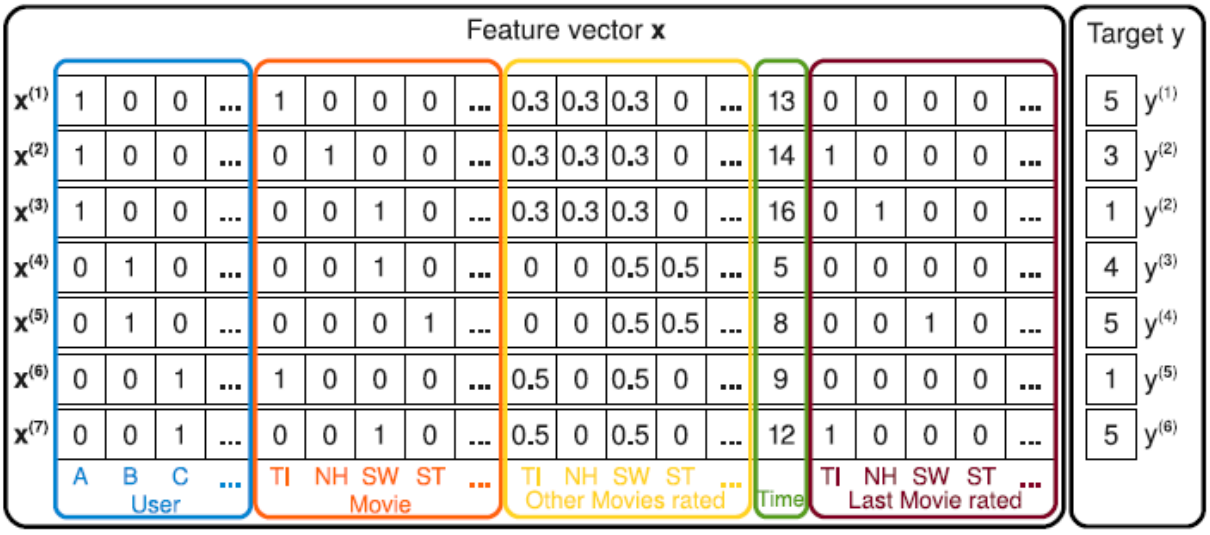

Figura 2.4: En cada fila encontramos un vector de características x(n) con su correspondiente objetivo o label y(i). Las primeras 4 columnas, en azul, representan variables indicadoras del usuario activo. Las 5 siguientes en rojo representan variables indicadoras del ítem activo. Las 5 posteriores, en amarillo contienen indicadores de otros ítems valorados por usuarios distintos al activo. La columna en verde representa el tiempo en meses y las últimas 5 columnas en marrón tienen indicadores del último ítem que el usuario ha calificado antes del activo, en rojo. La columna más a la derecha es el objetivo [17, p. 996].

2.2. Recomendación usando datos de carreras

Esta sección del capítulo está destinada a explorar entre algunos de los trabajos que han profundizado en los conceptos de las rutas y de la recomendación de manera conjunta en la historia reciente. Por recomendación entendemos la aplicación de conceptos que se han explicado en las secciones anteriores de este capítulo, como pudiera ser el filtrado colaborativo. En ella se abordarán las finalidades de estos trabajos, la metodología que emplean, recursos que necesitan y las conclusiones que alcanzan con el objetivo de dar una idea de cómo se han planteado estas aproximaciones por distintos grupos de investigación independientes.

2.2.1. A Collaborative Filtering Approach to Successfully Completing the Marathon

Objetivos: escrito por Jakim Berndsen, Barry Smyth y Aonghus Lawlor en este paper se aborda el proceso de cómo generar una planificación o estrategia para llevar a un corredor a completar la prueba de la maratón, que comprende una distancia de 42,195 kilómetros, evitando el conocido como efecto Muro utilizando filtrado colaborativo. Muro es el término que se utiliza en el argot para definir un efecto que produce un descenso repentino en el rendimiento de un atleta durante una carrera. Así, en este trabajo exploran las maneras de relacionar rutas entre sí y también mediante el estudio de los usuarios de cómo pueden asimilar sus perfiles las recomendaciones que se les proporciona [18].

Recursos empleados: Cuentan con los datos Global Positioning System (GPS) de cada entrenamiento de un total de 6.000 corredores que han practicado sus actividades en las ciudades de Nueva York, Londres y Dublin entre los años 2014 y 2017. Para las predicciones utilizan XGBoost [19] implementado en Python

Incovenientes encontrados: Definen una estrategia de carrera de la siguiente forma: *cómo de rápido un usuario tiene que correr un segmento u otro de manera que pueda administrar sus energías*. Sin embargo alcanzar esta meta cuenta con multitud de problemas que resolver, como el hecho de generar un tiempo realista para el usuario, asegurarse de que este no es demasiado excesivo por exigencia o demasiado laxo para las capacidades del atleta son solo algunos de los problemas a los que se enfrenta este trabajo. Existen bases previas con las que se han abordado trabajos de este estilo las cuales subyacen en sistemas de razonamiento por casos.

La figura 2.5 representa un esquema general de cómo funciona su método.

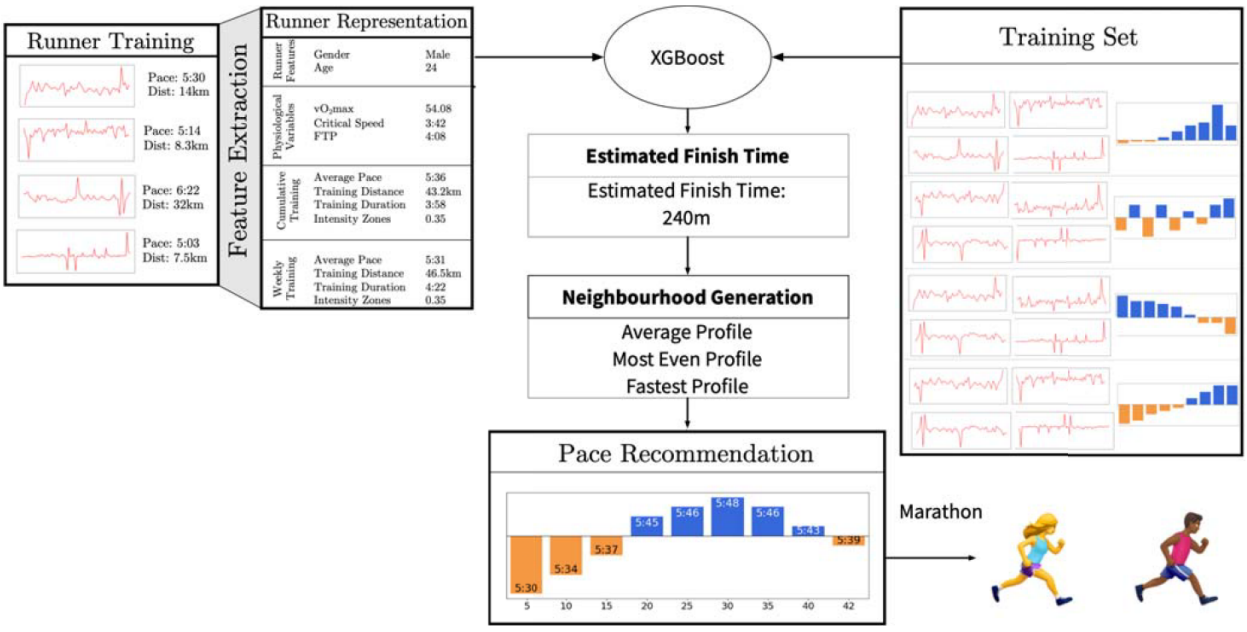

Figura 2.5: Extrayendo características fisiológicas y de entrenamiento de los usuarios generan un perfil con el que predecirán utilizando la libreria anteriormente mencionada un tiempo estimado en función de las competencias en donde encaje el atleta. Posteriormente proceden a evaluar el acierto mediante una proporción de entrenamiento y test del 90/10 empleando validación cruzada con 10 folds alcanzando unos resultados con un error absoluto de 11.3 minutos, situándose así como uno de los de métodos más acertados de la actualidad. De cara a la generación de la estrategia emplean 3 tipos de filtrados colaborativo usando KNN. Cada una de las aproximaciones genera una estrategia que funciona mejor o peor según el perfil que muestre el atleta [18, p. 654].

2.2.2. Pace My Race: Recommendations for Marathon Running

Objetivos: escrito por Jakim Berndsen, Barry Smyth y Aonghus Lawlor la propuesta de este segundo paper consiste en crear recomendaciones durante la propia maratón para los atletas que participan en ella. De nuevo el objetivo que persiguen es asegurar la finalización de la prueba pero en esta ocasión optimizando mediante recomendaciones durante la misma y no generando una estrategia previa que el usuario deba seguir desde el principio hasta el final [20].

Recursos empleados: disponen del ritmo en minutos por kilómetro, frecuencia cardíaca en pulsaciones por minuto y cadencia en pasos por minuto de 13.000 corredores que han corrido las maratones de Nueva York, Dublin y Londres. Estos datos los dividen en ventanas de 1 km, 5 km y toda la carrera para poder observar los mismos desde distintos marcos. Usan XGBoost [19] de Python para generar un modelo de regresión con el que poder trabajar.

Incovenientes encontrados: la información de la que disponen en su dataset no proporciona información exacta acerca de si los usuarios han sido capaces de terminar la prueba correctamente o han cumplido sus expectativas para con la misma. Otra limitación consiste en que carecen de información previa de los corredores, información que según comentan sería de mucha utilidad para conocer si están ajustándose o no a las capacidades totales que ha demostrado en toda su trayectoria. La explicabilidad, es decir, cómo orientar al usuario con los pasos a tomar, ha resultado también un reto. La figura 2.6 muestra un ejemplo de interfaz para explicar a los usuarios cómo seguir sus indicaciones.

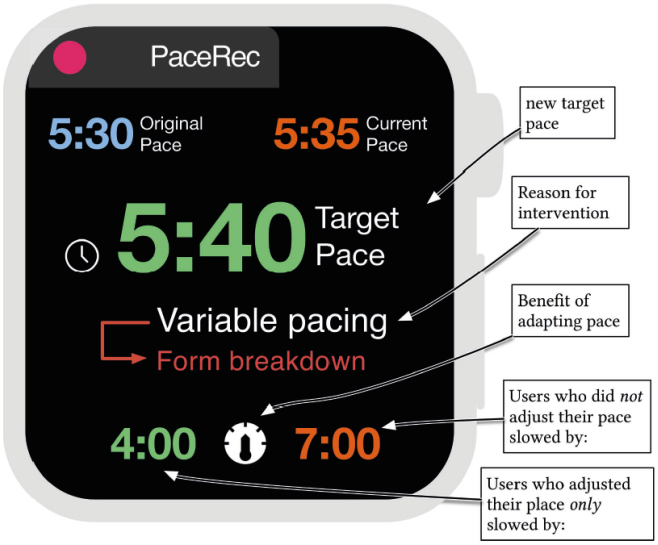

Figura 2.6: Para construir un modelo generan las características anteriormente mencionadas en intervalos de 500 metros. De esta manera construyen un modelo para cada intervalo de 500 metros mediante el uso de la mencionada XGBOOST [19]. Posteriormente para calcular el acierto de la predicción lo comparan con un baseline que generan de manera paralela. Para predecir el ritmo por otro lado se encargan de controlar las reducciones de velocidad de los atletas, adaptando las recomendaciones en función de la información que recolectan en cada intervalo [20, p. 249].

2.2.3. Providing Explainable Race-Time Predictions and Training Plan Recommendations to Marathon Runners

Objetivos: escrito por Ciara Feely, Brian Caulfield, Aonghus Lawlor y Barry Smyth. En este trabajo proponen, utilizando datos de aplicaciones fitness y sensores portátiles, generar entrenamientos personalizados basados en la similitud del usuario activo con corredores de hábitos similares con ideas de razonamientos basado en casos. Su objetivo consiste en generar predicciones durante el propio entrenamiento basándose en sus sesiones de entrenamiento recientes, para que este mismo pueda calibrar sus objetivos diarios y entender la efectividad de su entrenamiento [21].

Recursos empleados: el trabajo es evaluado empleando los datos recolectados de 21.000 maratonianos a través del mundo.

Incovenientes encontrados: la mayoría de los entrenamientos enfocados a una maratón requieren de complejos sistemas de entrenamiento repartidos en varias sesiones por semana durante varias semanas, donde se involucran conocimientos sobre fisiología humana y mecánicas de carrera. Por ello proponen emplear Case Based Reasoning (CBR), debido a que esta metodología permite generar recomendaciones sin necesidad de un conocimiento en profundidad del campo de conocimiento concreto. Han tenido que enfrentarse a la identificación de características descriptivas, las cuales son críticas en cualquier sistema CBR. Distinguir entre cuáles son realmente significativas ha sido un proceso delicado dentro del trabajo.

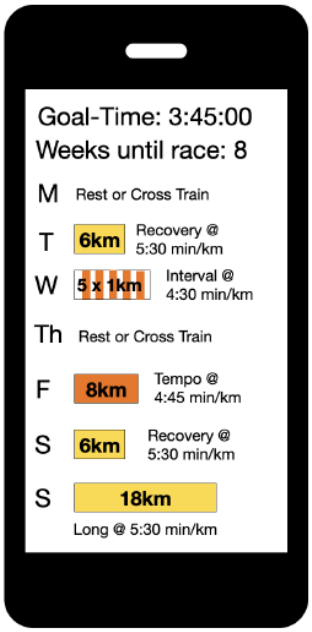

Figura 2.7: En primer lugar usan un proceso de selección de características para identificar indicadores importantes en el rendimiento del entrenamiento y también cómo emplear datos de entrenamientos pasados. En segundo lugar muestran esta representación nueva y transparente de recomendación de entrenamientos [21, p. 249].

2.2.4. Mining Marathon Training Data to Generate Useful User Profiles

Objetivos: escrito por Jakim Berndsen, Barry Smyth y Aonghus Lawlor el objetivo de este estudio se centra en abordar las carencias de los sistemas para recomendar estrategias de carrera tradicionales mediante la generación de perfiles de usuario. Estos perfiles representan tanto la aptitud fisiológica de un corredor como una amplia descripción de su entrenamiento para la maratón. El trabajo pretende demostrar que estos perfiles de usuario son capaces de predecir con exactitud el tiempo de llegada a lo largo de varios puntos del entrenamiento y demostrar cómo, junto con investigaciones anteriores, estos perfiles de usuario pueden utilizarse para ayudar a los corredores a prepararse para la maratón [22, p. 113-125].

Recursos empleados: En total, cuenta con datos detallados de más de 650.000 sesiones, obtenidas durante 196.642 semanas de los entrenamientos de 12.627 atletas que participaron en las maratones de Londres, Nueva York y Dublin entre los años 2014 y 2017.

Incovenientes encontrados: el mayor reto que ha encontrado este trabajo ha consistido en cómo generar un perfil de usuario consistente con el que poder generar recomendaciones y evaluar de manera eficaz. Para abordar esta tarea han tenido que plantear cómo minar y filtrar las características fisiológicas más importantes, volumen de oxígeno máximo, velocidad crítica, distancia máxima total y ritmo umbral funcional, donde han generado un perfil que representa al atleta semana a semana. Posteriormente han tenido que estudiado el progreso semanal en el entrenamiento, compuesto por otra serie de nuevas y diferentes características a lo largo de las semanas. Ambos perfiles se combinan generando un perfil final con un total de 125 características por usuario con el que se generarán recomendaciones.

2.2.5. Recommending routes in the context of bicycling: algorithms, evaluation, and the value of personalization

Objetivos: Escrito por Reid Priedhorsky, David Pitchford, Shilad Sen y Loren Terveen en este trabajo se exploran aproximaciones distintas a las más tradicionalmente usadas en lo que a recomendación de rutas se refiere. Frente al estudio conciencudo de distancias o tiempos en este documento se centran en analizar las preferencias personales y sobre todo y más importante, subjetivas, de cada corredor. Para ello emplean el término “bikeability” que se trata de la predisposición que muestra una ruta a ser recorrida en bicicleta. Así estudian qué predisposición tiene un usuario a realizar un tipo de ruta, en función de la “bikeability” de la misma, bien por tramos o en general. Haciendo de este proceso una recomendación personalizada la cual demuestra proporcionar interesantes resultados frente a los métodos más comunes.

Recursos empleados: Cuentan con un dataset compuesto de usuarios e ítems del sitio web Cyclopath, una wiki geográfica que proporciona servicios de búsqueda de rutas para ciclistas de donde

han extraído 482 usuarios con un total 35.608 ítems, en este caso aristas de rutas, evaluados. Han aplicado en sus experimentos técnicas relacionadas con Aprendizaje Automático y Recomendadores de Comunidades.

Incovenientes encontrados: En este trabajo han encontrado 3 retos fundamentales relacionados con la personalización. En primer lugar la estructura ítem/matriz de ratings les resulta poco favorable por la forma de su dataset, el cual es 13 veces más escaso que algunos con los que han comparado como el caso de MovieLens. En segundo lugar las restricciones tanto de los artículos como de la salida se encuentran sometidos a importantes restricciones, las topologías de aristas que emplean tienen relaciones muy estructuradas que no es sencillo de encajar y convertir en una ruta. Y por último relacionado con los algoritmos de búsquedas, los cuales necesitan un peso por arista para producir puntos finales.

2.2.6. Conclusiones acerca de las lecturas

Como se ha podido comprobar, no se ha encontrado ningún trabajo que abarque el objetivo de este trabajo de manera específica. Aunque muchas de estas lecturas han resultado interesantes en lo que a metodologías se refiere, la enorme cantidad de datos de los que disponían estos trabajos, extensos tanto en tiempo de estudio como en números, no es equiparable con lo que puede obtenerse en unos pocos meses partiendo desde 0. De esta manera no se disponía ni por asomo de los recursos necesarios para replicar con éxito ni las técnicas ni los experimentos aquí explicados de manera realmente interesante. Todos se enfocan en la realización de una prueba específica o en una preparación con la que completar una distancia concreta, camino un poco alejado pues del objetivo de este trabajo, el cual consiste en generar recomendaciones de rutas o trayectorias para los usuarios de una aplicación deportiva como puede ser Strava.

2.3. Evaluación de sistemas recomendación

Todos los métodos anteriormente propuestos precisan de ser evaluados con el fin de obtener el nivel de acierto que se ha obtenido sobre un usuario. Existen multitud de métricas con las que obtener esto, desde las más tradicionales a las más modernas. A continuación se detallarán las más importantes y las que tienen mayor recorrido hasta el momento. En primera instancia se explicarán dos métricas que comparan de manera directa el valor predicho en el sistema con el valor real dado por el usuario en el test.

Error medio absoluto (Mean Absolute Error (MAE)): se aplica el valor absoluto sobre la distancia entre el valor real con el valor predicho.

$$\textit{Error absoluto medio} = \frac{1}{R_{test}} \sum_{r_{(ui)} \in R_{test}} |f(u,i) - r_{ui}| \tag{2.8}$$

Error cuadrático medio (Root Mean Square Error (RMSE)): se obtiene de manera análoga a MAE pero haciendo la raíz cuadrada de la diferencia de la distancia al cuadrado.

$$\textit{Error cuadrático medio} = \sqrt{\frac{1}{R_{test}} \sum_{r_{(ui)} \in R_{test}} (f(u,i) - r_{ui})^2} \tag{2.9}$$

En estas expresiones *f(u,i)* representa el valor predicho por el recomendador sobre el usuario *u* para el ítem *i*. Además r_{ui} representa el rating proporcionado por el usuario u para el item i. Sin embargo existen ocasiones en las que estas expresiones no resuelven todas las situaciones posibles, existen escenarios donde los ratings del usuario no se encuentren disponibles y que sólo disponemos de información implícita, como por ejemplo los items que ha consumido. De esta forma deberemos cambiar el planteamiento en pro de obtener una evaluación. Por lo tanto en este nuevo entorno no se compararán los ratings predichos con los proporcionados por el usuario si no que se comparará una lista de items consumidos con otra de items recomendados.

Precision: esta métrica se basa en obtener el conjunto de intersección entre los documentos relevantes *T(u)* y los recomendados *L(u)* dividiendo este conjunto de intersección entre el total de los items recomendados.

$$\textit{Precision} = \frac{1}{U} \sum_{u \in U} \frac{L(u) \bigcap T(u)}{L(u)} \tag{2.10}$$

Recall: métrica similar a **precision** pero en esta ocasión en lugar de dividir entre el conjunto de recomendados dividiremos entre el conjunto de relevantes.

$$\textit{Recall} = \frac{1}{\upsilon} \sum_{u \in U} \frac{L(u) \bigcap T(u)}{T(u)} \tag{2.11}$$

En estas métricas el conjunto de los relevantes será el de los items realmente valorados por el usuario mientras que el de los recomendados serán aquellos propuestos por el recomendador para el usuario. Lo habitual es limitar los recomendadores a un tamaño de ranking, en cuyo caso se denota como metrica@N donde N es el tamaño máximo del ranking, por ejemplo P@5 se tratará de la precisión con tamaño de ranking = 5.

Existen otras métricas relevantes que también tienen cabida en el panorama actual.

Rango recíproco promedio (Mean Reciprocal Rank (MRR)): en esta métrica se compara la posición del primer relevante recomendado con su posición entre los relevantes. Los recomendados con puntuaciones más altas se valorarán mejor

$$MRR = \frac{1}{\mathcal{I}} \sum_{i=1}^{\mathcal{I}} \frac{1}{rank_i} \tag{2.12}$$

Ganancia acumulada con descuento (Normalized Discounted Cumulative Gain (NDGC)): en esta ocasión se obtiene la relevancia de los artículos recomendados, calculando para ello el Discounted Cumulative Gain (DGC) en primera instancia y comparándolo con el escenario ideal de esta métrica donde los elementos estarían ordenados perfectamente según el orden de relevancia, representado por Ideal Discounted Cumulative Gain (IDGC)

$$DCG = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{log2_{(i)}} \tag{2.13}$$

$$DCG = \frac{NDCG}{IDCG} \tag{2.14}$$

3

Diseño e Implementación

A fin de conseguir un sistema capaz de recomendar rutas deportivas para un usuario en una ciudad se necesitan satisfacer una serie de requisitos. Este capítulo se utilizará para explicar el trabajo de desarrollo llevado a cabo a fin de conseguir este objetivo. Desde el proceso para obtener los datos, el tratamiento con el que se ha tratado a los mismos, cómo se han empleado para obtener recomendaciones y finalmente su evaluación a fin de observar cómo de acertados resultan los métodos propuestos.

3.1. Diagrama general

La figura 3.1 muestra un diagrama de alto nivel del trabajo donde se muestra el ciclo del vida del proyecto. En el bloque *Minería de datos* se engloban los procesos destinados al estudio y obtención de los datos, desde el *Estudio de APIs* donde se selecciona Strava como la fuente de información pasando por la *Obtención de datos* donde se habla acerca de la metodología para descargar los datos de Strava. Mientras tanto en el bloque de *Recomendación* se desarrolla cómo se pasa de los datos en crudo descargados de la web para darles un formato que finalmente haga que tenga sentido someterlos a un proceso de *Entrenamiento* y *Test*. Finalmente los datos destinados a entrenamiento serán procesados por los *Métodos* propuestos para finalmente proceder a su evaluación.

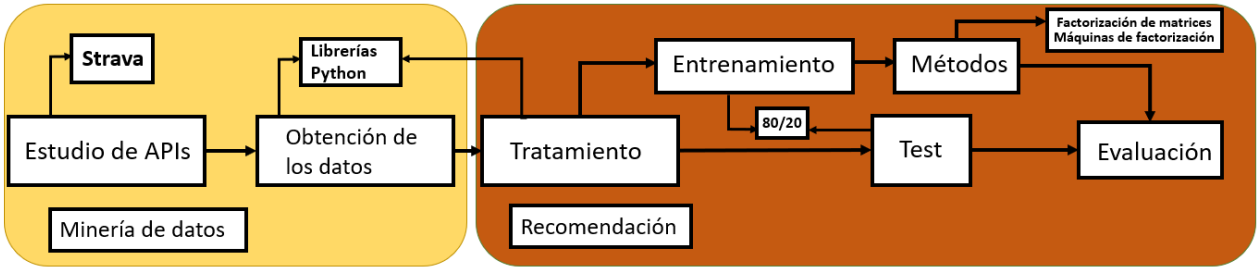

Figura 3.1: Esquema de alto nivel acerca de cómo se estructura este TFM, en él pueden distinguirse dos bloques, uno destinado a la parte de minería de datos en amarillo y otro a recomendación en naranja oscuro.

3.2. Obtención de los datos

Puesto que el dataset de este TFM se ha construido completamente desde cero, una parte del esfuerzo se ha tenido que destinar a seleccionar un sitio web a partir del cual obtener todos los datos necesarios para la compleción del trabajo. Otro porcentaje de tiempo también fue dedicado al estudio de cómo acceder a los datos con los distintos tipos de recursos actuales:

API: interfaz de acceso propia del sitio web a través de la cual es posible obtener datos siguiendo un proceso estipulado por el propio sitio. En este método tienen que tenerse en cuenta factores como número máximo de llamadas, tiempos de espera mínimos entre llamadas para evitar baneos, formato de salida de los datos y ubicación de los mismos dentro de las propias funciones.

Crawling: el cual es un proceso por el cual un robot web o araña navega sistemáticamente a través del sitio explorándolo y siguiendo links con el propósito de recoger el contenido deseado en otro sistema. En este método también existen factores relacionados con inconvenientes relacionados con tiempos de espera, formato y procesamiento de los datos. Inconvenientes a los que se suman problemas relacionados con el formato de la página, que puede ser cambiante, dejando al *crawler* completamente obsoleto.

3.2.1. Selección del sitio web

En primera instancia se buscaron páginas que poseyesen una API pública, que fuese gratuita y de uso para el público general. Entre todas las que se plantearon en un principio las principales páginas que se barajaron fueron:

Strava: Strava [2] es una red social basada en Internet y GPS enfocada a deportistas como pueden ser ciclistas y corredores y una aplicación de seguimiento GPS deportiva.

Komoot: Komoot [23] es una app de navegación guiada por voz que también proporciona un espacio donde poder compartir información y consejos sobre rutas, lugares interesantes, senderos, etc.

RideWithGps: RidewithGPS [24] es una aplicación que permite al usuario tanto crear sus propias rutas como registrar sus recorridos en la web.

El resumen fundamentalmente consiste en que todas las páginas ofrecen una lista con multitud de rutas en múltiples ciudades alrededor del mundo. Cada una de estas actividades corresponden a una lista más o menos consistente de datos. El mapa con el recorrido de la ruta, distancia, elevación y gráficas con las que figurarse la clasificación de la propia ruta. Se puede acceder a rutas a través de la página o a través de los propios usuarios. Cada página dentro de su interfaz concreta ofrece lo mismo. La información quería obtenerse desde una API por las ventajas que esto aporta: formateo intrínseco de los datos (JavaScript Object Notation (JSON)), menor riesgo ante baneos por múltiples

accesos o por comportamiento inapropiado por los términos del sitio. Finalmente, tanto por la robustez y accesibilidad de su API como por cantidad de datos y usuarios, la página seleccionada fue Strava.

3.2.2. Proceso de uso de la API

La API de Strava cuenta con un total de 90 funciones en su haber, para acceder a ellas en primera instancia es necesario crear una aplicación en el sitio web, la figura 3.2 muestra la aplicación creada para obtener los datos. Tras crear la aplicación se llevaron a cabo una serie de llamadas de prueba mediante el cliente de escritorio de POSTMAN [25]. Tras dicha verificación se procedió a estudiar las llamadas de las cuales podían obtenerse datos puesto que Strava trata los datos de los perfiles privados como confidenciales para algunas aplicaciones de terceros, entre las que se encuentran las de este trabajo, y no permite su libre descarga y tratamiento.

Figura 3.2: Figura donde aparece la aplicación creada para acceder a la API de Strava a fin de obtener datos para este trabajo. Puede observarse el número de peticiones máximo por día y los permisos de los que dispone.

Con este último dato en cuenta se indagó en la API del sitio hasta dar finalmente con una llamada que permitía acceder a un listado de carreras marcadas como públicas de las que sí se podía extraer correctamente información según la normativa de Strava. De entre todas las llamadas iniciales que proporcionaba la API finalmente sólo se terminan empleando 3, las cuales resultan suficientes para, junto con un tratamiento de estos datos, se pueda generar un dataset. Las llamadas son:

List Running Races (getRunningRaces): devuelve la lista de carreras a pie dado un criterio, en este caso un año específico.

Get Running Race (getRunningRaceById): devuelve el detalle de una carrera a pie.

Get Route (getRouteById): devuelve el detalle de la ruta de una carrera.

Una polilinea consiste en una lista de puntos en la que se dibujan segmentos de línea entre puntos consecutivos. Es el tipo de dato con el que Strava representa las rutas deportivas que representan el recorrido de sus carreras. Puesto que el fin último de este proceso es obtener polilineas, se desarrolló un proceso destinado a relacionar las funciones anteriormente expuestas entre sí para pasar desde una llamada inicial hasta un fichero con las rutas de distintas carreras de distintas ciudades alrededor del mundo. La figura 3.3 muestra el flujo de llamadas que se realizan en la API de Strava para obtener polilineas.

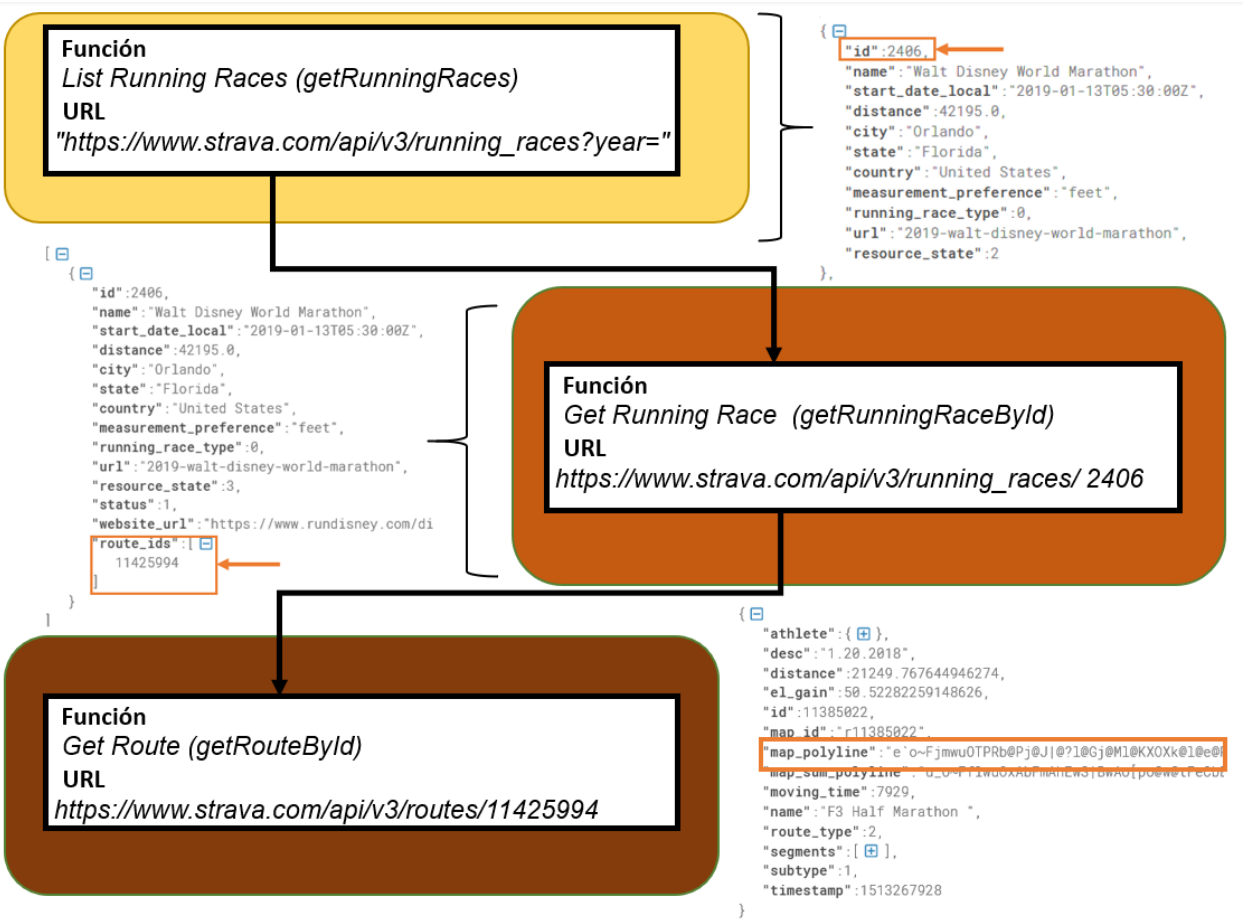

Figura 3.3: En primera instancia se obtiene el listado total de carreras a pie existentes durante los años 2020, 2019, 2018 y 2017 (getRunningRaces), se guardan sus identificadores y para cada uno de ellos se obtiene el detalle de la carrera (getRunningRaceById), donde se encuentra el identificador de su ruta, de manera análoga se llama al detalle de la ruta (getRouteById) y se obtiene finalmente la polilinea que representa la carrera.

Finalmente la figura 3.4 muestra el número de llamadas necesarias para obtener todos los datos del dataset final, getRouteById se llama 1 vez por cada año entre 2020 y 2017 mientras que getRunningRaceById se llama tantas veces como carreras a pie se publicaron en cada uno de los respectivos años. Finalmente se seleccionan las 3 ciudades con más carreras de la lista, Chicago, Sao Paulo y Nueva York con un total de 83 carreras entre las 3.

Función	Número de llamadas a la API	Resultados
getRouteById	4	1896
getRunningRaceById	1896	1896
getRouteById	83	83

Figura 3.4: Número de llamadas y cantidad de datos manejado en el proceso para obtener los datos que conforman el dataset con el que se ha desarrollado este trabajo.

3.3. Tratamiento de los datos

A continuación se abordan las implementaciones llevadas a cabo sobre los datos para terminar adaptándolos a un formato capaz de ser empleado por un sistema de recomendación. Tras convertir las polilineas obtenemos una lista de coordenadas, que no son más que una lista de tuplas con latitud y longitud. La naturaleza de estos, simples sucesiones de puntos en un plano en forma de lista, sin valoraciones, sin comentarios y con la única relación establecida real con los usuarios siendo la de si este ha corrido o no la ruta ha llevado a múltiples maneras de experimentar con estas a fin de obtener relaciones que permitan generar recomendaciones.

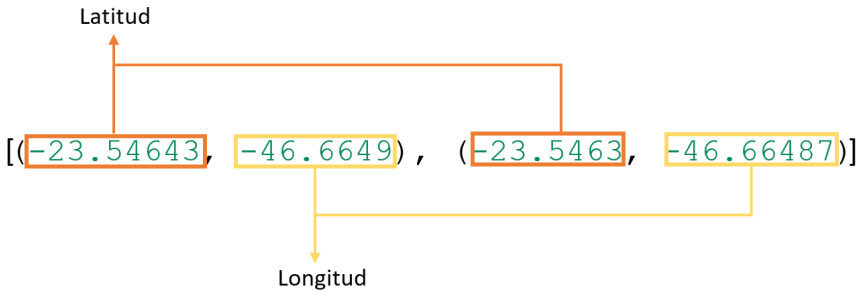

Figura 3.5: Coordenadas 0 y 1 de la primera carrera de SAO PAULO, en ella podemos ver la forma de tupla con latitud y longitud que representa la coordenada o punto, una sucesión de estas tuplas en una lista conforma una ruta.

3.3.1. Similitud entre carreras

Con unos datos tan simples en contenido y forma la primera aproximación que se ideó para obtener relaciones entre los mismos fué la de estudiar cómo de parecidas son dos rutas en base a la forma de su recorrido. Puesto que una ruta se trata de una sucesión de tuplas que la representa, se experimentó con varios métodos para calcular el grado de similitud de dos listas. Similitud coseno o similitud de Jaccard fueron algunas de estas aproximaciones. Pero finalmente el método elegido fué la Dynamic time warping (DTW) [26] mediante la librería de Python FastDTW [27].

DTW: se trata de una técnica que encuentra la alineación óptima entre dos series temporales o dos colecciones de datos. En este trabajo DTW se emplea para comparar la similitud o distancia entre dos rutas, que son simplemente la sucesión de tuplas en una lista.

Puesto que el valor de FastDTW es menor cuando más parecidas son dos listas, se implementa una función la cual considera que dos carreras son similares cuando el valor de esta función se encuentra debajo de cierto umbral. La figura 3.6 muestra dos rutas que el sistema considera similares.

Figura 3.6: Las dos primeras rutas de la ciudad de Sao Paulo comparadas en un mapa de la ciudad, una en verde y otra en azul. En la figura podemos ver cómo las rutas se superponen prácticamente en todo momento salvo un pico puntual de unos cuantos metros en la sección norte del recorrido, que se remarca en rojo con un circulo. El zoom muestra de manera ampliada el detalle de cómo existe un tramo donde ambas rutas circulan paralelas siendo prácticamente idénticas. El sistema extrae que estas carreras tienen un gran parecido entre sí y las considera, por tanto, similares.

3.3.2. Tokens de similitud por ciudad

Una vez realizado el paso anterior se necesitaba una manera de esclarecer qué rutas de cada una de las ciudades se parecían entre sí. De esta manera surgió la idea de generar unos *tokens*, que no son más que etiquetas para *clusters* compuestos de carreras que el sistema considera como similares. En estos el objetivo es el de resumir toda la información de una carrera a fin de obtener los elementos que serán útiles posteriormente.

Estos *tokens* se generar a partir de la lista de similitudes de carreras. En este proceso era imprescindible evitar el problema de transitividad que surgía en este escenario, la transitividad consiste básicamente en una relación binaria sobre un conjunto, en el que:

$$\forall a, b, c \in A : aRb \wedge bRc \rightarrow aRc \tag{3.1}$$

Esto podía resultar particularmente perjudicial ya que se creaban largos *clusters* de carreras similares pero en las que realmente podría haber carreras que no se relacionasen entre si, si no a través de alguna relación transitiva del conjunto. De esta manera cada ruta de la ciudad se compara individualmente con el resto y los *clusters* se crean a partir de las parejas que han demostrado un valor más alto, asegurando que todos los elementos que forman parte de un token tengan un elevado porcentaje de similitud entre sí y no a partir de relaciones heredadas de sus rutas similares.

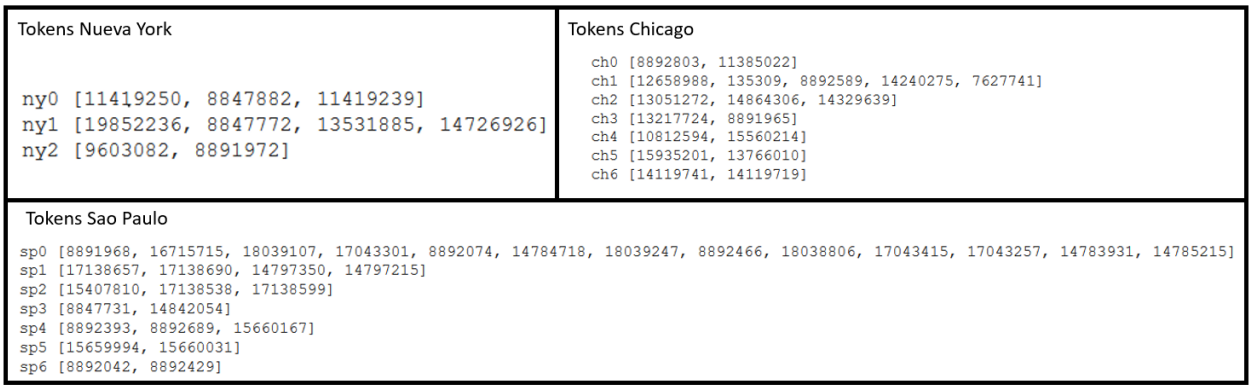

Figura 3.7: Figura representando los *tokens* de las 3 ciudades del dataset, el texto representa la etiqueta del *clusters* mientras que la lista de identificadores hace las veces de grupo de carreras similares entre sí.

3.3.3. Generador de perfiles de usuarios

A parte de rutas, el sistema cuenta con otro elemento implícito dentro de los datos de las mismas: los corredores. A pesar de no haber realizado un proceso para obtenerlos a ellos en concreto, estos aparecían dentro de la información de las rutas, representando a la persona que había llevado a cabo la misma. Por ello se consideró que merecía la pena trabajar con ellos a fin de obtener conclusiones adicionales que ayudasen al proceso y para dar sentido a las aproximaciones de recomendación propuestas en el trabajo. Puesto que de la API de Strava se obtienen rutas, y los atletas forman parte de los elementos de las mismas no se cuentan con datos explicitos de los mismos, no obstante sí que se podía generar un perfil de sus gustos en base a las rutas registradas en el sistema. El perfilador de atletas se trata de un módulo que, recorriendo las rutas que este ha llevado a cabo devuelve la distancia, elevación y tiempo de actividad promedio del corredor. Esta aproximación permite representar al atleta de manera similar a una ruta, haciendo estos promedios las veces de algunos de los valores

intrínsecos que componen una carrera, como pueden ser su distancia, su elevación y el tiempo que un corredor ha requerido para completarla. La figura 3.8 muestra un ejemplo de un perfil de atleta para la aplicación. De sus datos también podían establecerse en qué ciudades habían corrido cada uno, la figura posterior 3.9 enseña en qué ciudades han corrido los atletas del dataset.

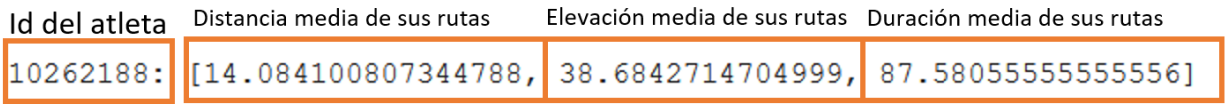

Figura 3.8: Distancia, duración y elevación promedio de las actividades de un atleta, estableciendo el perfil del mismo. Todos los datos son obtenidos al procesar las carreras asignadas al id de cada usuario

Identificador atleta	Chicago	Sao Paulo	Nueva York
10262188	SI	NO	NO
3370924	SI	SI	SI
9032675	SI	NO	SI
0	SI	NO	SI
23665645	SI	SI	NO
11763137	NO	SI	NO

Figura 3.9: Tabla representando dónde ha llevado a cabo cada atleta sus rutas. Solo existe un corredor que haya corrido en las 3 ciudades y los otros 5 se reparten en las distintas combinaciones de haber corrido en dos y no en una y al contrario.

3.3.4. Clusters de puntos

El último elemento del dataset con el que se consideró interesante trabajar fueron los puntos. Como ya se vio anteriormente, una ruta es básicamente una sucesión de tuplas (latitud, longitud) en forma de lista. En el trabajo hasta el momento se ha comparado cómo de parecidas son dos carreras pero no cómo de parecidos son dos puntos. Considerando que dos carreras similares compartirán puntos entre sí se llegó a la conclusión de que se podrían extraer nuevas e interesantes conclusiones, a parte de eliminar una gran cantidad de ruido del dataset, si se desarrollase un método que relacionase los puntos de las rutas de una ciudad que estén a unos pocos metros entre sí. Este método se ejemplifica en la figura 3.10 que nos muestra cómo dos rutas, a pesar de no estar directamente relacionadas en sus recorridos pueden compartir puntos, y esta información se refleja dentro de los *clusters* de puntos.

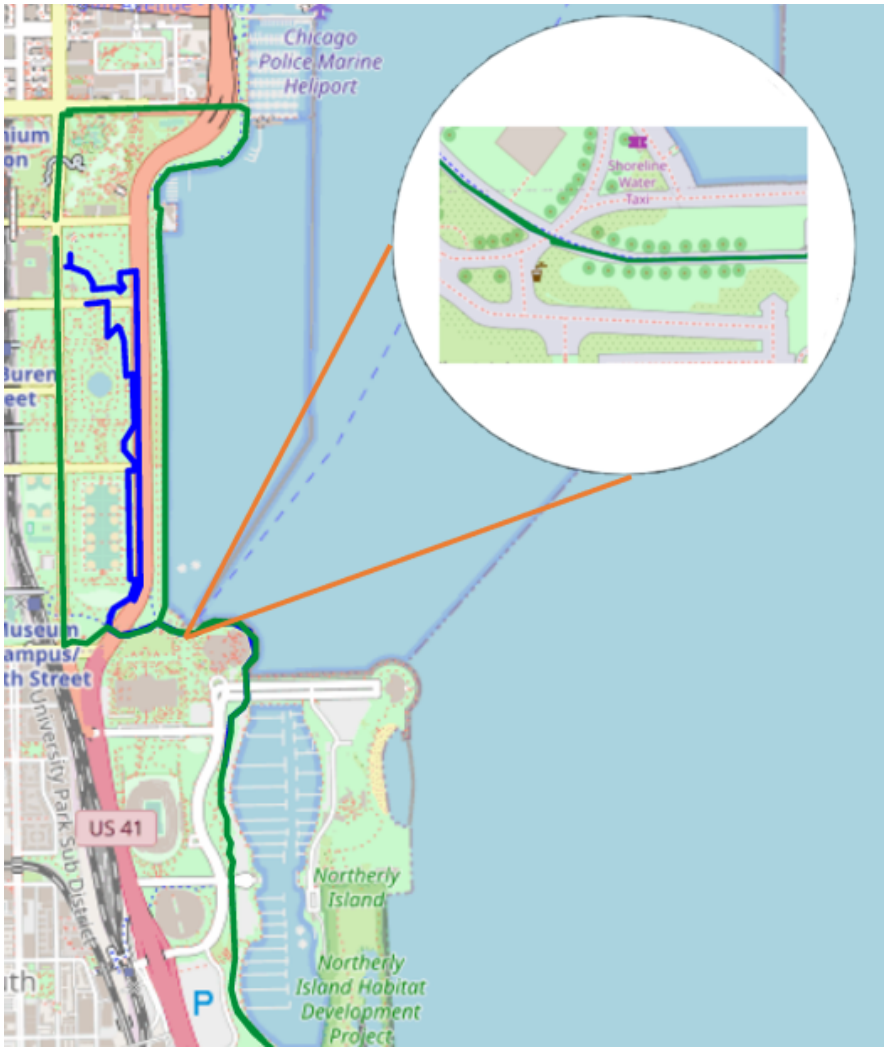

Figura 3.10: Detalle de dos carreras de la ciudad de Chicago, aparentemente muy distintas entre sí pero que cuentan con un segmento donde ambas son prácticamente idénticas. Este se considera un ejemplo adecuado de la utilidad de los *clusters*, gracias a ellos se relacionan de alguna manera dos rutas de las que a priori podrían descartarse ningún parecido gracias a estos.

Siguiendo el proceso de manera similar a como se implementaron los *clusters* de las ciudades en los tokens (ver Sección 3.3.2), los *clusters* de puntos permiten condensar la información de las coordenadas de las ciudades desde el conjunto total de miles de puntos a solo unas estrechas relaciones de similitud. Los *clusters* se establecen entre todos los puntos de las carreras de una ciudad que estén como mucho alejados 2 metros entre sí. Condensándose así miles de puntos dispersos entre sí a unas pocas decenas de *clusters* de información condensada. Los datos concretos se mostrarán en detalle en la sección resultados, más adelante.

Figura 3.11: Ejemplo con la estructura de cómo se reparten los *clusters* de puntos en un ejemplo con la ciudad de Chicago. En él podemos ver que los puntos 332 y 22 se relacionan (están a menos de dos metros entre sí) en el primer *cluster*, mientras que en el segundo caso son los puntos 1803 y 46 los que comparten relación. Los *clusters* son mínimo de 2 elementos y máximo de 4 con la cantidad de metros elegida. Para asignar los identificadores se crea una lista plana con todos los puntos de una ciudad y se le asigna un identificador en base a su posición en la lista total. Posteriormente se buscan en sus respectivas rutas para conocer las relaciones que establecen.

3.4. Implementación de algoritmos de recomendación

El objetivo último de este trabajo es el de generar recomendaciones. La naturaleza de los datos obtenidos y todo el proceso de trabajo hasta el momento ha desembocado en la selección concreta de los siguientes algoritmos.

3.4.1. Factorización de matrices

Partiendo de la hipótesis de que el concepto de ruta ejecutada o llevada a cabo por un atleta se asemeja al concepto clásico en recomendación de producto consumido o valorado por un usuario, en este contexto, la factorización de matrices se consideró una aproximación viable a explorar. Dentro de la misma, la forma y estructura de los datos permitía crear varios escenarios en función de qué relación se estableciese entre usuarios e ítems. Como veremos a continuación, y según los procesamientos que ya hemos discutido, se pueden establecer varias relaciones: usuario-carrera, usuario-punto (coordenada de carrera) y usuario-cluster, donde a su vez los cluster se pueden calcular entre carreras (ver Sección 3.3.1) o entre puntos (ver Sección 3.3.4) Así, este trabajo plantea 3 matrices a factorizar utilizando la función de reducción de la dimensionalidad SVD de Scikit-learn. También se contemplaron otros métodos de esta librería, como Non-Negative Matrix Factorization (NMF), se utilizaron en pruebas preliminares pero dieron peores resultados. No obstante, la funcionalidad y el desarrollo implementado es totalmente transparente a esta decisión.

Matriz Usuario - Carreras: Primera y más simple de las aproximaciones planteadas. En ella se establece una relación usuario, rutas que ha corrido, donde se ponen a 1 las celdas donde un usuario ha corrido la carrera. Esta primera aproximación es la más simple y es la que tiene unas matrices de menor tamaño. Sin embargo tiene que tenerse en cuenta una particularidad de los datos y es que las carreras son ejecutadas únicamente por un usuario, es decir no hay dos identificadores de corredor en la lista de atletas que han corrido una ruta mientras que sí ocurre al revés, un atleta cuenta con multitud de carreras ejecutadas. Por ello, aunque se trate del método más sencillo también se prevee que sea el más susceptible a la alta dispersión de los datos.

Id usuario \ Id carrera	16715715	17043257	17043301	17043415	17884257	17884119	18038806	18039107	18039247	17138657	17138690	17138538	17138599	15407810	14783931	14784718	14842054	8891968	8892689	8892714	14785215	14797350	14797215	15660167	15659994	15660031	8892466	8892787	8856864	8892512	8892429	8847803	8892393	8892042	8892074	8847731
10262188	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3370924	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1	1	1	0	0	1	1	1	1	1
9032675	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23665645	1	1	1	0	1	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11763137	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0

Id usuario \ Id carrera	11419250	11417385	19852236	8847877	14727097	14726926	8891972	134122	14263905	11419239	11419214	13531885	9603082	8847882	8856859
10262188	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3370924	1	0	1	0	0	1	0	0	1	1	1	0	1	1	1
9032675	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
23665645	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11763137	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Id usuario \ Id carrera	11385022	135309	8892803	14329639	13217724	8892589	15560214	15935201	7627741	12799360	12589750	13051272	13216858	13217146	12658988	12589299	13766010	14864306	14890145	8892684	14119719	14119741	14240275	14240371	8891965	10551473	10328939	10812594	16715691
10262188	1	0	0	1	1	0	1	1	0	1	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
3370924	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0	1	1	0	1	0	1	0	0	0	0
9032675	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23665645	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
11763137	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figura 3.12: Ejemplo de cómo se ven 3 matrices usuario-carrera para una de las proporciones de entrenamiento generadas aleatoriamente. Como puede verse en la imagen, estas matrices tienen tantas filas como usuarios con más de una carrera hay en el sistema y tantas columnas como carreras tiene la ciudad. Estas matrices se conforman a partir de los datos que hay en el conjunto de entrenamiento inicialmente generado. Es decir, el conjunto real cuenta con 1 en más posiciones que las que podemos ver en la figura. Esta explicación y esquema aplica para el resto de configuraciones de esta sección.

Matriz Usuario - Puntos de carreras: Segunda de las aproximaciones. En este caso la relación se establece entre el usuario con las coordenadas de las carreras que ha completado. Dada la enorme cantidad de puntos esta aproximación es la que más dispersión demuestra ya que los usuarios con menos carreras tienen miles de puntos en 0 y los pocos 1 que tienen pueden estar alejados miles de celdas entre sí.

Matriz Usuario - Clusters de puntos: Última aproximación en la que se termina de explorar las matrices con los clusters de puntos similares. Esta matriz concentra mucha más información que las anteriores ya que se forman a partir de comprobar los puntos que componen los clusters de las ciudades y poner un 1 sobre aquellos que tengan un punto entre los pertenecientes a los recorridos por un usuario, siendo capaces de establecer una relación directa entre el usuario con puntos nuevos que puede no haber recorrido.

Una vez presentados los 3 tipos de matrices el siguiente paso es explicar cómo se obtienen las predicciones. Con la matriz se inserta un modelo SVD con el que que hacer el fit, que se guarda en la variable *W* y se obtienen los componentes del modelo, que se guardan en la variable *H*. Tras esto se ejecuta el producto escalar de ambas variables para cada usuario en cada ciudad con cada una de las combinaciones, es decir, se recorre la matriz pertinente, se ejecuta el proceso descrito y se guarda. La siguiente fórmula ilustra el proceso.

$$estimation = model.fit(row_n) \bullet model.components \tag{3.2}$$

Ahora con las estimaciones guardadas se procede a obtener un ranking (ver Sección 3.4.3) y a compararlo con los elementos que se hayan quedado dentro del test, una vez obtenido esto se calculan las métricas de *Precision* y *Recall*.

3.4.2. Máquinas de factorización

Esta sección está dedicada a la aproximación mediante máquinas de factorización. En el momento en el que se crearon los perfiles de usuarios y se vió que estos devolvían un formato muy similar al que tenían las carreras se empezó a barajar alguna implementación que pudiese explotar esta similitud. Como se argumentó antes, el concepto y ejecución de las máquinas de factorización resultó ideal como línea de trabajo. Así, para explotar esta aproximación se crean los vectores usuario-carrera.

Vectores Usuario - Carrera: esta idea consiste en relacionar las características que representan a un usuario es decir, su perfil, con las características de las carreras que ha corrido y, a parte, para ampliar datos con las características de las carreras similares a las que ha corrido. Para ello se emplean los tokens de ciudades anteriormente generados, se recorre la lista de carreras del usuario y se extraen aquellas carreras similares a las de este sin duplicados y generan vectores de la forma: perfil de usuario, datos de su carreras y aproximadas y finalmente un flag final con valor 1 o 0.8 en función de si la ruta es una realmente recorrida por el usuario o sin embargo es una aproximada a las reales del usuario.

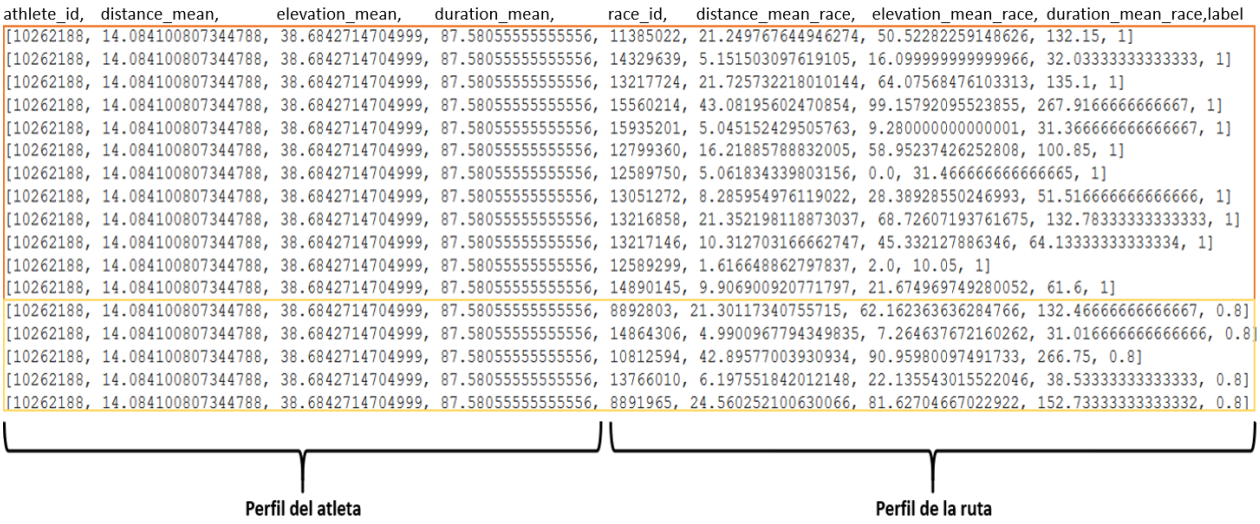

Figura 3.13: Ejemplo de un vector usuario carrera para un usuario, en naranja podemos ver aquellas rutas que han corrido el usuario mientras que en amarillo aparecen las aproximadas de sus carreras. La primera parte que va desde el campo *athlete_id* hasta el campo *duration_mean* consta del perfil, id, media distancia, media de elevación y media de duración del usuario mientras que la segunda parte es homónima pero en la carrera.

Finalmente la librería empleada para trabajar con este nuevo tipo de datos ha sido DeepCTR [27], una librería de aprendizaje profundo escrita en Python que ejecuta el modelo de Keras. A esta librería se le pasan los vectores ya divididos en los subconjuntos de entrenamiento y test y se encarga de ejecutar las predicciones. Posteriormente estas predicciones se ordenan y someten a un cutoff que se enfrenta al conjunto de test, con lo que finalmente se obtienen resultados. El detalle de estos resultados se detalla en la sección 4.3.1

3.4.3. Obtención del ranking

A continuación se detalla cómo se ha obtenido el ranking de cada una de las aproximaciones propuestas en el sistema, estas son, las 3 matrices (ver Sección 3.4.1) y las máquinas de factorización (ver Sección 3.4.2). Este proceso conlleva un par de puntualizaciones que han de comentarse de cara a tener todo completamente ilustrado. La primera y más obvia es que, una vez obtenido el ranking, se han de retirar de los resultados las carreras que han formado parte del entrenamiento de cada usuario, ya que el objetivo del recomendador es ofrecer sugerencias que sean nuevas para el mismo.

La segunda tiene que ver con la propia naturaleza del dataset. Como puede verse en la figura 3.9 no todos los atletas han corrido en todas las ciudades, lo que conlleva un pequeño control posterior. Han de retirarse todas las carreras de las ciudades donde un usuario no haya corrido ninguna ruta, puesto que se le estarán recomendando carreras que nunca se podrán comprobar en el test, ya que pertenecen a ciudades donde nunca ha estado, es decir, solo se testeará con las ciudades que alberguen al menos una ruta donde haya corrido el usuario. Una vez aclaradas estas particularidades, que son iguales para todos los métodos y por eso se exponen desde el principio, se explican las diferencias a la hora de obtener todos los rankings.

Ranking matriz Usuario - Carreras: Para este caso simplemente se toma el ranking generado por el método SVD ya limpio y ordenado y sobre el mismo se aplica un *cutoff*, cuyo valor variará entre 1,3,5 y 10, en el que se discierne si entre los elementos de este *cutoff* están los elementos de test.

Ranking matriz Usuario - Puntos: Este caso es un poco más complejo, en él se toman los puntos ya rankeados, es decir ordenados según estime el proceso SVD, una vez hecho esto a dicho punto se le otorga un valor que se corresponde al tamaño del conjunto de recomendaciones menos su posición en el propio conjunto, siendo de esta manera el primero el que más valor tenga, y así se busca a qué carrera pertenece sumando a esta el valor de todos sus puntos en el conjunto, de esta manera las carreras que tengan los puntos más altos en el ranking serán las que gozen de una puntuación más alta. Así se ordena esta nueva lista de carreras y se realiza un proceso idéntico al anterior donde las carreras más avanzadas en el ranking estarán ahí por la suma del valor individual de sus puntos.

La última de las aproximaciones de las matrices, usuario - cluster (ver Sección 3.3.4) da lugar a dos métodos distintos de generar los rankings. De la estimación de *clusters* se hace de nuevo un ranking ordenado, pero esta vez de *clusters*, es decir, de conjuntos de puntos, de esta manera los puntos que forman parte de los *clusters* mejor evaluados serán los que mayor valor tengan finalmente. Para ello se implementan dos maneras diferentes:

Ranking matriz Usuario - Cluster puntos: En este caso se toman los *clusters* y se ordenan, posteriormente se extraen los puntos de estos generando una lista plana y ya ordenada y de manera similiar a la anterior se genera un ranking de puntos, estos obtienen un valor en base a su posición en el conjunto y asignando dicho valor a la carrera de la que forma parte se obtienen las recomendaciones

finales.

Ranking matriz Usuario - Cluster carreras: Este caso final es ligeramente distinto, en este caso simplemente se recorren los puntos de los *clusters*, y las carreras se someten a una especie de votación, es decir, si entre los *clusters* recomendados existen *n* puntos en total de una carrera, esta recibe un valor *n*. Este proceso se hace para todas las carreras y se ordena en base a esta *n*. Con la lista de carreras del ranking *n* se realiza el proceso de la misma manera que en los casos anteriores.

Una vez terminado con la sección de las MFs se detalla el proceso final para obtener los rankings de las FMs que de nuevo conlleva un par de particularidades por la naturaleza de la misma.

Ranking máquinas de factorización: Para este escenario la metodología es similar a la que se usa en el método de **Ranking matriz Usuario - Carreras** con una pequeña particularidad. Esta es que las predicciones generadas, a diferencia de las de aquel método, no se generan individualmente por atleta, si no que se generan para todos los del conjunto de golpe, esto lleva que antes de poder ordenar se deban filtrar los rankings de cada corredor concreto dentro de la lista. Una vez se cuenta con estos factores ya separados, el proceso de obtención del ranking es idéntico: se ordena según el valor predicho por la Máquina de Factorización (FM) y se somete al *cutoff* correspondiente.

4

Experimentos y Resultados

Esta sección finalmente se destina a mostrar y explicar los resultados obtenidos a través de todo el proceso previamente expuesto durante este documento. En primera instancia se ejemplificarán los distintos métodos a través de una de las 3 ciudades del dataset, mostrando sus variaciones entre sí y dando una idea general de qué puede esperarse de cada una de las aproximaciones, tanto de factorización de matrices como de las FMs. Posteriormente con este primer ejemplo aclarado se ofrecerá una comparativa de cada uno de los métodos en cada una de las ciudades y finalmente una discusión acerca de los resultados mostrados.

4.1. Configuración del entorno

A continuación se describen los entornos en los que se ha implementado este TFM. La parte enfocada en extracción de datos se ha llevado a cabo en un equipo con las siguientes características sobre el Integrated Development Environment (IDE) Pycharm.

Recursos	Características
CPU	AMD Ryzen 7 3700X 8-Core
GPU	AMD Radeon RX 5700 XT
S.O	Windows 10
Version de Python	Python 3.7.4
RAM	16GB

Figura 4.1: Figura en la que se ilustran las características de los recursos sobre los que se ha implementado este trabajo.

La implementación y desarrollo se ha implementado sobre la plataforma Google Colab, un entorno potente y de fácil empleo con una cesión de RAM variable que ha permitido realizar el tratamiento de los datos sin sobrecargar el equipo principal. Posteriormente para el desarrollo de algunos scripts ligeros se ha empleado Jupyter Notebook, un entorno de nuevo simple y cómodo para ejecutar código

puntual en ocasiones necesarias. Para generar algunos gráficos se ha empleado Excel.

4.2. Metodología experimental

4.2.1. Datos utilizados

Todos los datos utilizados han sido extraídos de la API de Strava (ver sección 3.2.2), los cuales son finalmente agrupados en un fichero llamado *routes.json* que contiene 83 líneas pertenecientes a las 3 ciudades anteriormente mencionadas en el siguiente formato:

athlete: diccionario que representa al atleta que ha corrido la ruta y su información guardada durante el registro en Strava.

desc: cadena de texto con la descripción de la ruta.

distance: double con la distancia total de la ruta en metros.

el gain: double con la elevación ganada en metros.

id: entero con identificador de la ruta.

map id: cadena de texto del identificador.

map polyline: cadena de texto con la polilinea de la carrera.

map sum polyline: cadena de texto con la polilinea resumida de la carrera.

moving time: *Timestamp* en formato UNIX con el tiempo en movimiento.

route type: cadena de texto con el subtipo de la ruta.

segments: diccionario con los segmentos que conforman una ruta.

subtipo: entero que representa el tipo de la ruta, siempre será 1 en los datos.

timestamp: *Timestamp* en formato UNIX con la fecha en la que se realizó.

En la figura 4.2 podemos ver un ejemplo de salida completa desde el JSON con los datos. La ciudad de procedencia se obtiene dentro de los segmentos que la componen, los segmentos son un diccionario dentro de los cuales, entre otra información podemos observar la ciudad donde se empleza dicho segmento (ver Figura 4.3). Gracias a estos datos podemos obtener una serie de estadísticas relacionadas con los mismos (ver Figuras 4.4 y 4.5).

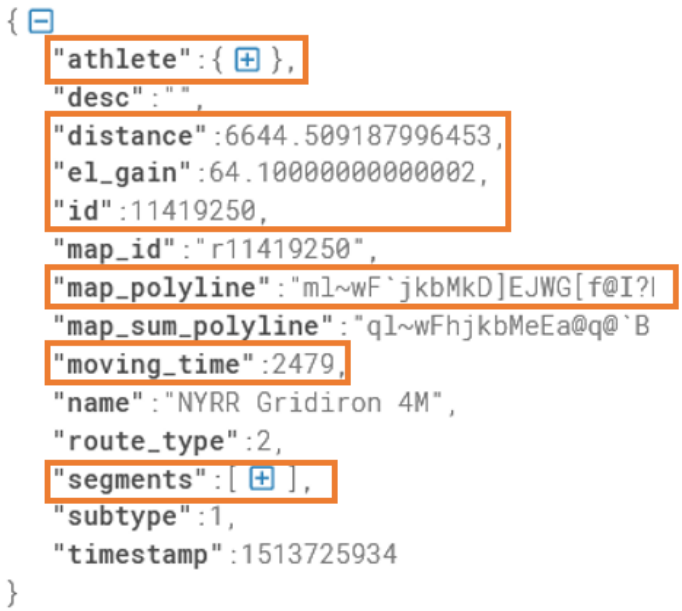

Figura 4.2: Salida para la ruta 11419250 de la API de Strava donde puede verse en naranja los campos que han sido empleados en el trabajo.

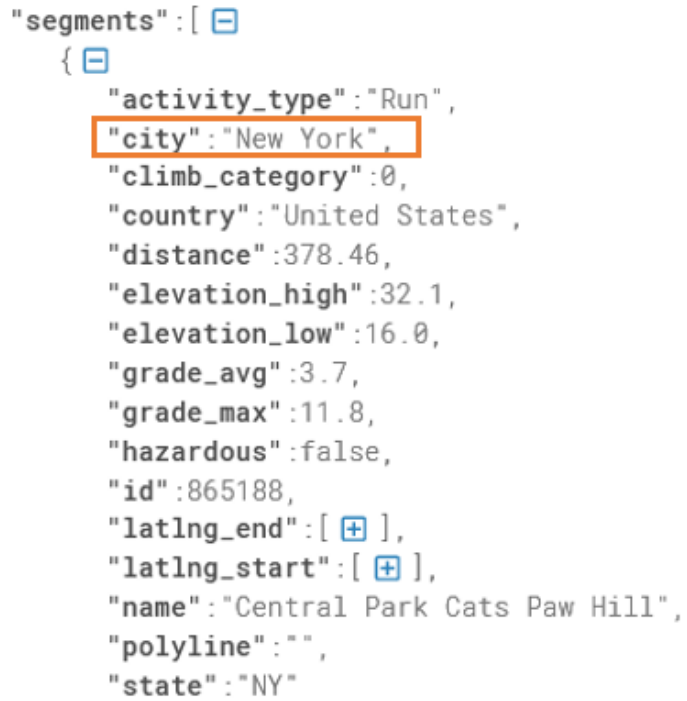

Figura 4.3: Detalle del primer segmento de la ruta 11419250 extraída de la API de Strava donde se señala en naranja la ciudad a la que pertenece el segmento.

La figura 4.4 ilustra una serie de estadísticas recopiladas a partir de los datos extraidos. Desde la lista de rutas final se recopiló el número de usuarios que han corrido las 83 carreras, dando un total de 9 usuarios. Sin embargo 3 de ellos han corrido únicamente una ruta por lo que se descartarán de cara a experimentos posteriores. La cantidad tan extremadamente baja de usuarios tiene que ver con cómo maneja Strava la información de los atletas registrados en su web, las aplicaciones de terceros que hacen uso de su API tienen muy restringido el acceso a la información de los usuarios, esto es así debido a que se maneja información muy delicada como es la ubicación de una persona, dato que puede usarse con fines ajenos al de este trabajo. Por ello los perfiles de los que dispone el trabajo se han conseguido de manera implícita, haciendo ingeniería inversa desde un dato que sí es accesible para las aplicaciones: las carreras a pie públicas. De esta forma estos nunca se han extraído a través de una llamada directa, puesto que con los permisos de los que dispone la aplicación de este trabajo, esta información es inaccesible. Debido a esto se planteó descartar el uso de la API en pro de métodos distintos de extracción de datos, pero bucles de redireccionamiento en cada enlace y una araña ubicada en el *login* protegiendo el acceso a la página hicieron que se descartase esta opción.

Gracias a la misma figura podemos ver el tipo de dataset con el que estamos trabajando, en el que un corredor alberga los datos de casi la mitad de las rutas disponibles. Seguidamente, en amarillo vemos cómo se distribuyen las carreras por las ciudades del dataset, Sao Paulo es la que cuenta con más carreras, doblando a Nueva York, la última. Finalmente se obtienen el número de coordenadas por ciudad y corredor, mientras que para los corredores el comportamiento es lineal (a más carreras más coordenadas) en las ciudades se puede observar un comportamiento ligeramente distinto. Chicago cuenta con más coordenadas entre todas sus carreras que Sao Paulo, a pesar de que cuenta con 7 rutas menos en el dataset, indicando que sus rutas tienen más segmentos que las de esta última.

De los datos anteriores también podemos extraer algunas métricas a parte como media, mediana, máximos y mínimos que se detallan en la figura 4.5.

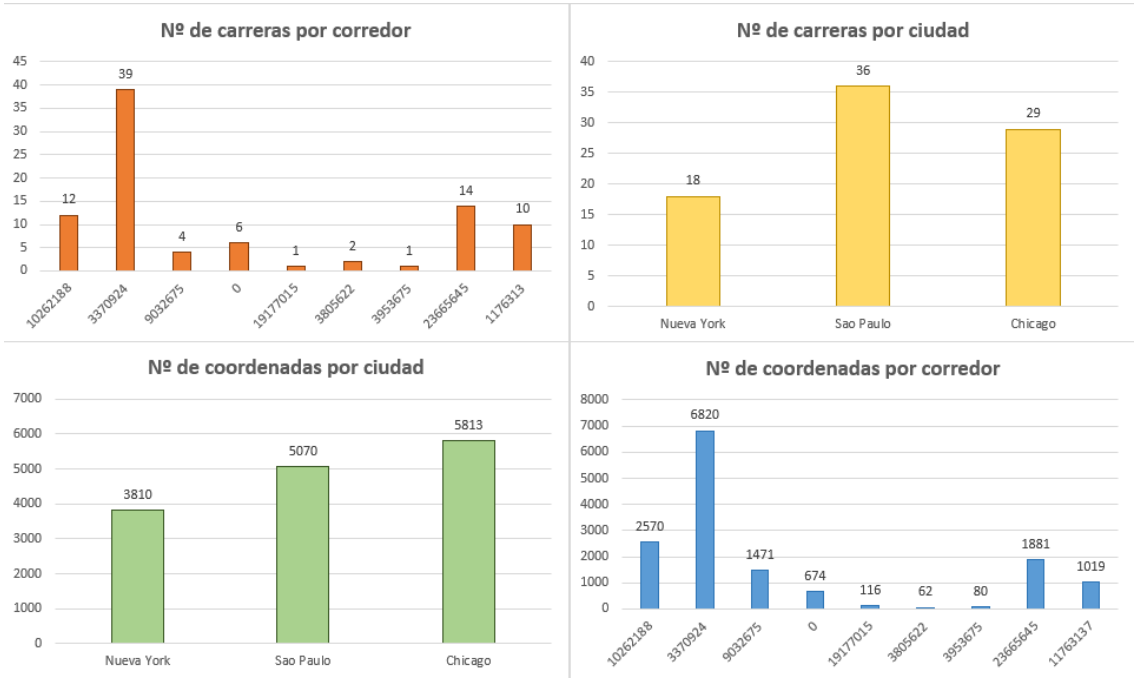

Figura 4.4: Gráficas que representan el esquema general de comportamiento del dataset. Desde el número de carreras que ha corrido cada atleta, parte superior izquierda hasta el número de coordenadas totales sumadas entre todas sus carreras en la parte inferior derecha, pasando por el número de coordenadas por ciudad, parte inferior izquierda y carreras también por ciudad, en la parte superior derecha.

	Media	Mediana	Máximo	Mínimo
Corredores/Carreras	9.88	6	39	1

Figura 4.5: Media y mediana de carreras por corredor para cada uno de los 9 atletas presentes en el dataset.

4.2.2. Partición para la evaluación

De cara a la partición para la evaluación, los datos se han dividido en dos conjuntos, uno de entrenamiento y otro de test con una proporción del 80 % y 20 % respectivamente, este proceso se realiza de manera idéntica para ambas aproximaciones, MFs y FMs no obstante a continuación se detalla cómo cada uno de los métodos reciben esta información para generar las predicciones.

Factorización de matrices: En los experimentos relacionados con este método simplemente se han tomado las carreras de cada uno de los usuarios, se han pasado por un randomizador y se ha tomado un porcentaje para entrenar y otro para probar. De este proceso bebe todo el flujo de datos de las 4 aproximaciones relacionadas con las matrices: Usuario-Carrera, Usuario-Puntos, Usuario-Cluster Carreras y Usuario-Cluster Puntos. Cada aproximación creará su modelo SVD y posterior ranking con

las carreras del entrenamiento, los cuáles posteriormente serán evaluados directamente contra los datos restantes que se quedaron en el conjunto de test.

Máquinas de factorización: De manera homónima al caso anterior las máquinas de factorización también han ejecutado sus experimentos sobre los mismos conjuntos de test y de train que el método anterior para poder generar comparaciones en igualdad de condiciones. Sin embargo en este caso se han tenido que dividir los datos en una serie de subconjuntos que se almacenarán en forma de ficheros para poder ser empleados posteriormente por el método.

- Fichero de entrenamiento: conjunto generado a partir de la partición de entrenamiento de las carreras del usuario. Se guarda linea a linea los elementos del vector de entrenamiento del atleta con el formato visto en la figura 3.13

- Fichero de test: de manera similar el porcentaje restante se escribe de la misma manera sobre el fichero de test.

Con la información en este formato las FMs son capaces de crear unas predicciones que posteriormente se clasificarán y evaluarán de manera idéntica a como se hace con las MFs.

No obstante y a diferencia de lo que pueda parecer, este último proceso de división en entrenamiento y test es muy similar al anterior de la FMs no dando de ningún modo resultados diferentes según el método. Simplemente se han adaptado los conjuntos originales que se usan normalmente a ficheros para que puedan leerse y ejecutarse correctamente por la librería seleccionada para las FMs.

4.2.3. Métricas de evaluación

Las métricas empleadas en este trabajo han sido fundamentalmente dos: *precision* y *recall*. Estas, por su simpleza y adaptabilidad se han podido ejecutar de manera indiferente para cualquiera de los 5 métodos propuestos, puesto que todos cuentan con un test y un *cutoff* que puede especificarse como se desee.

Con los rankings de cada subtipo de dato ya ordenado y almacenado, estos se comparan con los datos en el test. Es decir, para obtener en primera instancia la *precision* se divide la intersección del ranking sometido al cutoff con el subtest y con el tamaño del test mientras que para la *recall* la primera parte se hace de la misma manera pero dividiendo del tamaño del test. Tal y como se ilustra en las fórmulas del estado del arte para la sección de evaluación de sistemas de recomendación (Ver ecuación 2.10 para el caso de la *precision* y 2.11 para el caso del *recall*).

4.2.4. Parámetros de los algoritmos

Las librerías empleadas en este trabajo cuentan con algunos parámetros susceptibles de ser cambiados en pro de obtener una mayor variedad de resultados. Se han hechos cambios en:

SVD, n_parametros: el algoritmo empleado en la factorización de matrices cuenta con un argumento que representa el número de parámetros a utilizar. Este trabajo se ha cumplimentado en su totalidad con este valor asignado a 5 pero se han repetido las ejecuciones para n_parametros igual a 10 y 15, sin embargo los cambios resultan casi imperceptibles, no considerándose resolutivos para la compleción de este trabajo.

DEEPCTR, batch_size y epochs: para las ejecuciones de las FMs se han cambiado tanto el tamaño de lote como el número de épocas. Para las pruebas de este trabajo estos valores se encuentran en 256 de batch_size y 10 épocas, valores que se han aumentado hasta el doble, es decir 512 y 20, sin producir ningún cambio reseñable en los resultados, por lo que de nuevo estas ejecuciones no se considera que aporten nada especial a la explicación de este trabajo. De esta última configuración de pruebas cabe destacar además que lleva al equipo del trabajo a un punto en el que apenas puede ningún tipo de navegación sobre el mismo, sobrecargando sobremanera sus competencias.

4.3. Resultados

Esta sección finalmente se destina a mostrar y explicar los resultados obtenidos a través de todo el proceso previamente expuesto durante este documento. En primera instancia se ejemplificarán los distintos métodos a través de una de las 3 ciudades del dataset, mostrando sus variaciones entre sí y dando una idea general de qué puede esperarse de cada una de las aproximaciones, tanto de factorización de matrices como de las FMs. Posteriormente con este primer ejemplo aclarado se ofrecerá una comparativa de cada uno de los métodos en cada una de las ciudades y finalmente una discusión acerca de los resultados mostrados.

4.3.1. Análisis por algoritmo

En primera instancia se procede a analizar en detalle el comportamiento de los algoritmos propuestos en una de las ciudades; por su tamaño y cantidad de información la ciudad elegida será la de **Sao Paulo**. Estos pueden verse en la Tabla 4.1 en la cual se muestran las siglas con los nombres de todos los métodos propuestos, siendo estos en orden: **fms** las siglas para el método que encarna las FMs, **uc** para el método que emplea la matriz de Usuario - Carreras, **ucc** para Usuario - Cluster Carreras, **ucp** en el caso de Usuario - Cluster Puntos y finalmente **up** para el escenario de la matriz de Usuario - Puntos.

P@1: Para este *cutoff* el ranking solo dispone de un elemento, si este se trata de uno que se encuentre en el test entonces se considera acierto total. De esta manera podemos ver cómo los valores a pesar de ser tan bajos son en promedio los más altos de la tabla, puesto que aunque se ejecute la media entre todos los corredores en el momento en el que alguno tenga un acierto se considerará como muy valioso aumentando la estadística. El método de Usuario-Puntos (up) muestra el valor más alto mientras que el método de las FMs (fms) es con diferencia el peor no acertando ni una sola vez para ningún atleta en ninguna ejecución. También cabe destacar que 3 de los métodos empatan en acierto.

P@3: Este segundo *cutoff* muestra valores más bajos que el primero, esto se explica de nuevo con el funcionamiento de la métrica, ahora al dividir entre 3 a pesar de que se recojan más aciertos en el top, la división reduce notoriamente los valores, sin embargo este valor de *cutoff* hace que no se lleguen a obtener todos los posibles aciertos y los resultados, sin contar con un elemento incrementador como en el primer caso siguen siendo muy pobres. El método Usuario-Cluster Puntos (ucp) parece ser el mejor en esta ocasión mientras que de nuevo las FMs (fms) resultan las peores.

P@5: Para este *cutoff* los valores que se muestran parecen ser los más balanceados. Esto se debe a que a pesar de que la división se hace entre 5 para calcular la métrica, el ranking con 5 elementos es lo suficientemente amplio como para conseguir más aciertos. Usuario-Carrera (uc) resulta el que obtiene la puntuación más alta mientras que en el caso del peor encontramos a las las FMs (fms).

P@10: En este último *cutoff* los valores caen ligeramente de nuevo. A colación del argumento anterior podemos asumir que al dividir entre más elementos los resultados empeoran. De nuevo el mejor valor se obtiene para Usuario-Carrera (uc) y el peor para Usuario-Cluster Carreras (ucc). Encontramos de nuevo un empate entre dos de los elementos, las FMs (fms) y Usuario-Puntos (up) y su máximo no resulta particularmente alto ni dentro de la métrica para esta ciudad ni de los *cutoffs* empleados en la misma.

Cutoff	fms	uc	ucc	ucp	up
P@1	0	0,20	0,20	0,20	0,27
P@3	0,09	0,13	0,13	0,16	0,11
P@5	0,12	0,19	0,09	0,15	0,16
P@10	0,10	0,16	0,08	0,09	0,10

Tabla 4.1: Tabla con los distintos resultados para los *cutoffs* propuestos en la métrica de *precision* de la ciudad de Sao Paulo. En ella podemos ver en rojo que los peores valores están asignados mayoritariamente a las FMs, mientras que los valores más altos se encuentran sobre el primer *cutoff* @1.

A continuación se procede a analizar de la misma manera la métrica de *Recall* puesto que en este caso el denominador es fijo y se calcula en función de la partición los valores son mucho más altos que para la métrica anterior a medida que el *cutoff* aumenta. Estos valores pueden apreciarse directamente en la Tabla 4.2

R@1: En este caso particular los valores son increiblemente bajos. De nuevo con un triple empate el que obtiene los mejores resultados es el método de Usuario-Puntos (up) mientras que el método de las FMs (fms) es de nuevo el peor. Estos valores tan pobres se justifican ante el hecho de que normalmente el test no es de un único elemento por lo que en caso de no generarse ningún o muy pocos aciertos las medidas se empobrecen notoriamente.

R@3: Aquí puede verse cómo la tónica general mejora y aunque ningún valor es particularmente alto sí que aumentan respecto de la ejecución anterior. Usuario - Carreras (uc) es el que mejores valores demuestra y las FMs (fms) el peor, pero para este *cutoff* todos los resultados son prácticamente parejos y no existe ninguna gran diferencia muy reseñable entre los resultados.

R@5: De nuevo se puede observar cómo los resultados mejoran por el razonamiento expuesto desde la ejecución anterior. Otra vez Usuario - Carreras (uc) es aquel que mejores valores obtiene, acercándose al 40 % de acierto, Usuario - Puntos (up) lo sigue de cerca. En esta ocasión sí que resulta llamativo cómo el peor de todos, Usuario - Cluster Carreras obtiene casi la mitad de acierto que los mejores.

R@10: Salida en la que finalmente se obtiene un valor por encima del 50 % en promedio. Usuario - Carrera resulta en este caso el que mejor resultados obtiene con diferencia. Las FMs lo siguen en segundo puesto siendo la primera ocasión en la que no se ubican entre las peores. Los buenos valores de esta última ejecución en comparación pueden explicarse debido a que el *cutoff* para este caso supere el tamaño de algunos de los tests de la partición sin embargo puede verse cómo dentro de esta posibilidad existe un método que despunta notoriamente sobre el resto, Usuario - Carreras (uc) consigue un acierto por encima del 65 %.

Cutoff	fms	uc	ucc	ucp	up
R@1	0	0,07	0,07	0,07	0,09
R@3	0,11	0,14	0,14	0,14	0,12
R@5	0,26	0,36	0,17	0,21	0,35
R@10	0,46	0,67	0,36	0,25	0,43

Tabla 4.2: Tabla para el *recall* en los distintos métodos y *cutoffs* propuestos en la ciudad de Sao Paulo. Aqui puede verse como la tendencia del valor aumenta con el tamaño del cutoff y como se pasa de valores que apenas alcanzan el 0,1 para R@1 se terminan obteniendo valores casi el 0,5 en R@10.

Para un análisis exhaustivo de los algoritmos en las otras ciudades, ver Apéndice A. En la siguiente sección se analizará cómo cambian los comportamientos de los algoritmos según las ciudades en las que se ejecuten.

4.3.2. Análisis por ciudad

A continuación se exponen 2 gráficas que resumen el comportamiento de los distintos métodos para cada *cutoff* en la ciudad de **Sao Paulo**. En la primera, se muestra en la figura 4.6 la métrica de *Precision*. En ella podemos observar visualmente cómo efectivamente el método de Usuario - Puntos (up) comienza siendo el que obtiene un valor más alto pero que termina siguiendo la tendencia general del empeoramiento de los resultados a medida que el *cutoff* aumenta por las causas expuestas anteriormente. Puede verse cómo ninguno de los métodos es muy superior al resto siendo Usuario - Cluster el que parece mostrar los valores más altos en más ocasiones. Por lo demás puede verse cómo Usuario - Carreras (uc), Usuario - Puntos (up) y las FMs (fms) tienen un pico de nuevo en P@5 mientras que el resto ya en ese punto tienen una tendencia descendiente irreparable.

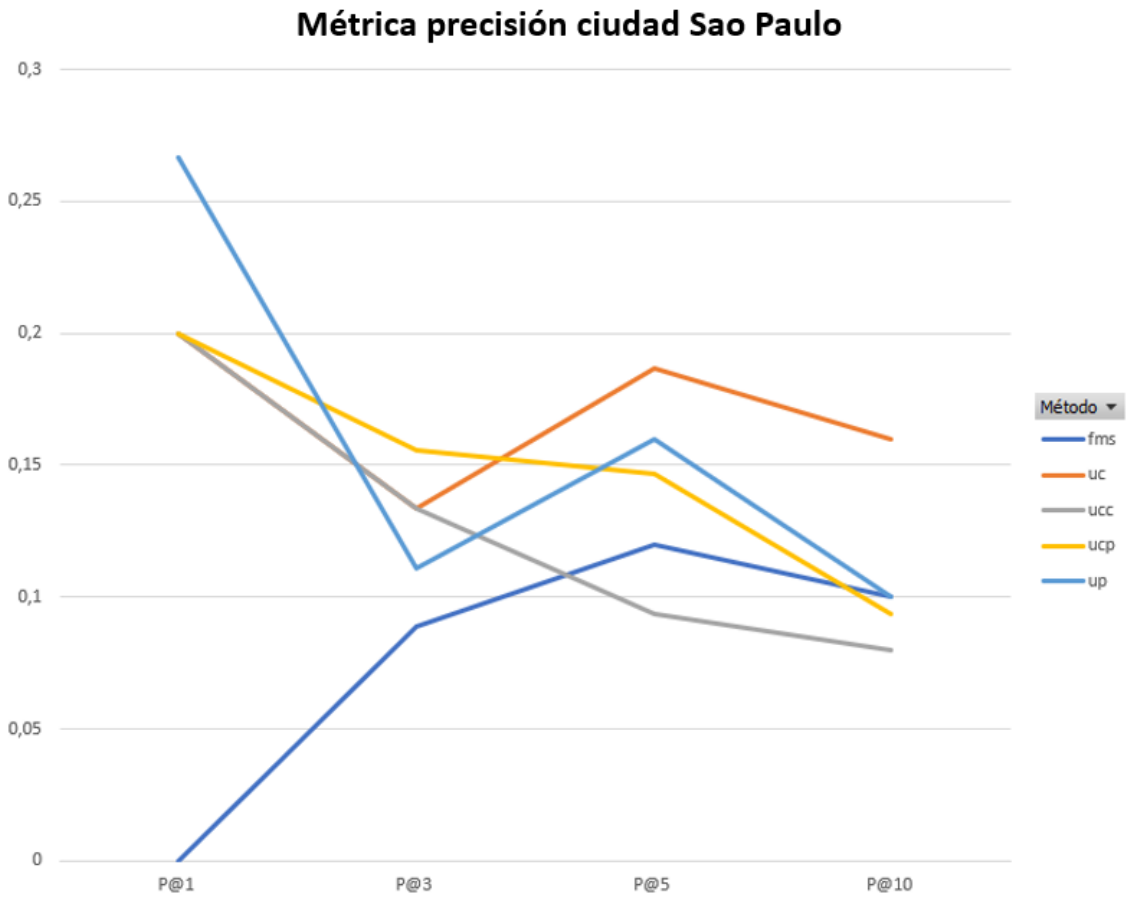

Figura 4.6: Figura que ilustra en colores el comportamiento de los distintos métodos propuestos sobre la ciudad de Sao Paulo para la métrica de la *precision*, puede verse como tras la caida inicial desde P@1 en P@5 los valores vuelven a mostrar una tendencia ascendente.

En la figura 4.7 puede observarse la evolución de los métodos para la métrica de *Recall* en la ciudad de **Sao Paulo**. En ella puede observarse una inequívoca tendencia ascendente en todos los métodos, siendo el método de Usuario - Carreras (uc) el que alcanza en R@10 un umbral por encima del 60 % de acierto. El método Usuario - Puntos (up) a pesar de comenzar el más alto en el primer *cutoff* finalmente termina siendo superado por las FMs (fms) las cuales se quedan cerca de alcanzar el 40 %. Ambos métodos relacionados con clusters muestran un rendimiento particularmente pobre en esta ciudad para esta métrica puesto que ocupan las dos peores posiciones de entre los cinco métodos.

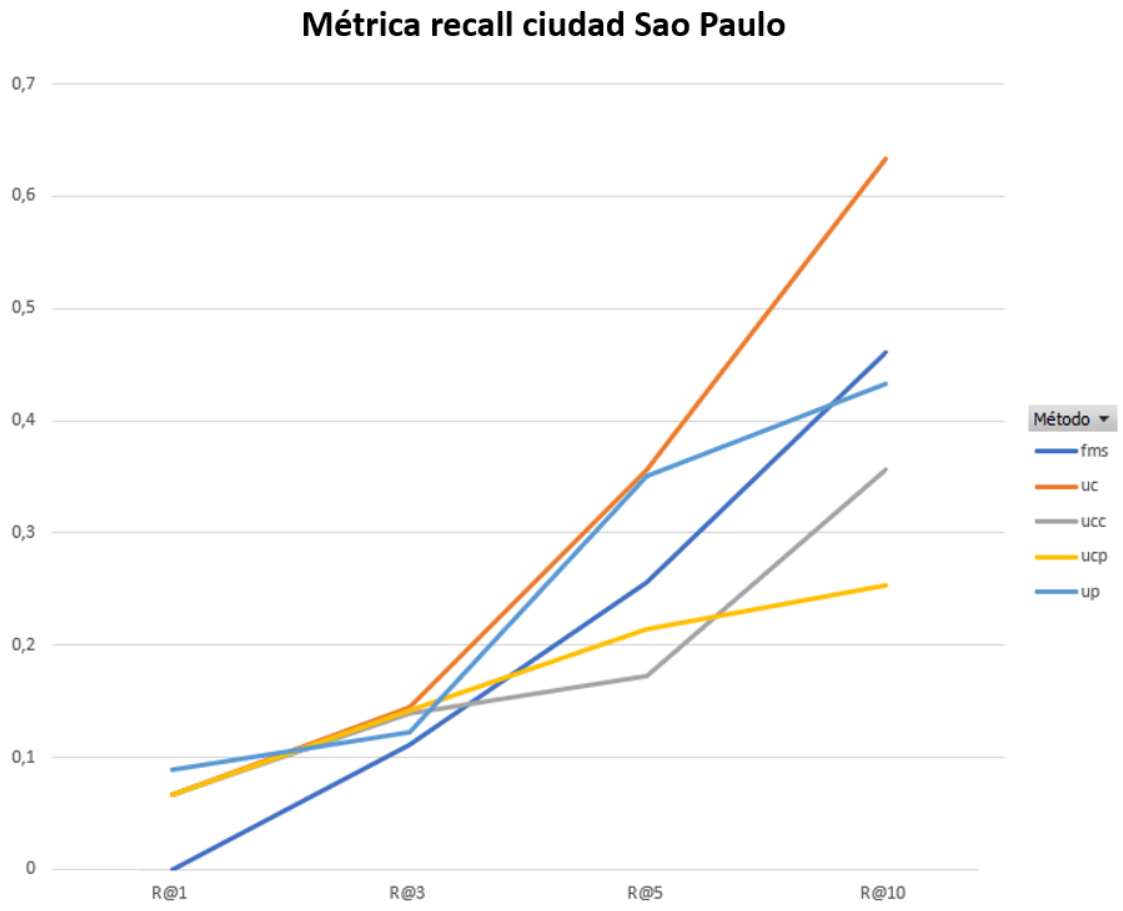

Figura 4.7: Esta figura muestra una tendencia ascendente ejemplificando los resultados de las tablas, creciendo a medida que crece el *cutoff* puede verse como todos los métodos se muestran parejas a excepción de Usuario-Carreras (uc) que se dispara respecto del resto en el último caso.

Como en el caso anterior se exponen dos figuras para ejemplificar el comportamiento de los métodos propuestos para las distintas métricas tanto en *Precision* (ver Figura 4.8) como en *Recall* (ver Figura 4.9), en este caso para la ciudad de **Nueva York**. En el caso de la precisión puede verse muy claramente el comportamiento anteriormente mencionado: en el momento en que el *cutoff* toma un valor muy grande los resultados comienza a empeorar ostensiblemente independientemente del método que se esté evaluando. El método que menos parece sufrir ante este suceso es el de las FMs (fms) pero también es el que peores resultados muestra en todo momento. De la misma manera los dos métodos que están mostrando los mejores resultados muestran una caída muy pronunciada en este último tramo. Usuario - Carreras (uc) y Usuario - Puntos (up) son los que mayor descenso sufren. También es llamativo cómo al final todos los resultados toman una tendencia a estabilizarse en torno al 20 % a pesar de la disparidad con la que comienzan.

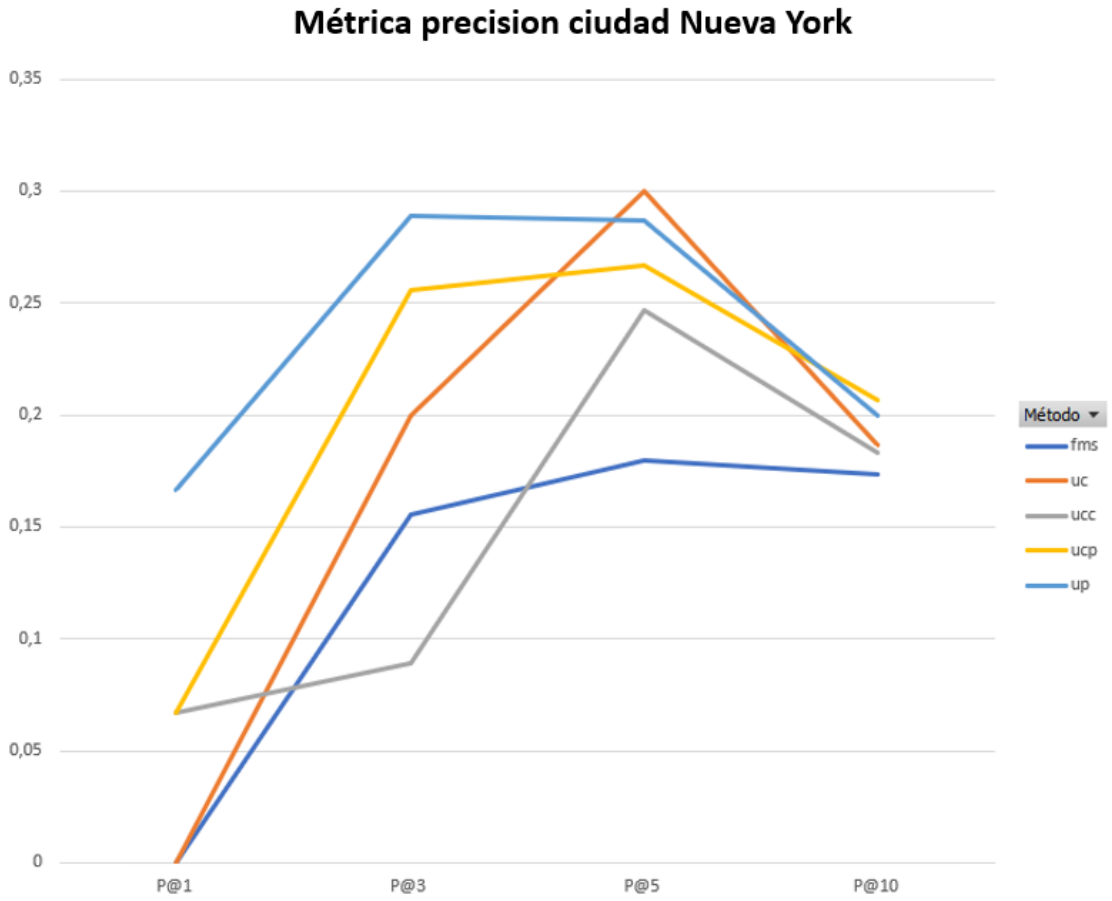

Figura 4.8: Figura que ilustra en colores el comportamiento de los distintos métodos propuestos sobre la ciudad de Nueva York para la métrica de *precision*. En ella puede verse una forma de puente en la que los cutoffs de 3 y 5 muestran resultados parejos o de menor a mayor valor pero que reciben una importante caida en P@10 donde todos los valores rondan el 20 %.

A continuación se ilustra el comportamiento de la métrica de *Recall* en la ciudad de **Nueva York**, de nuevo es evidente la tendencia creciente que muestran los valores llegan a alcanzar en el caso de dos de los métodos, Usuario - Puntos (up) y Usuario - Cluster Puntos (ucp) valores por encima del 90 % de acierto. En general todas las tendencias son positivas y las particularidades de cada una tienen que ver en lo pronunicado o no de la pendiente con los distintos *cutoffs* ya que todas al final muestran una tendencia positiva.

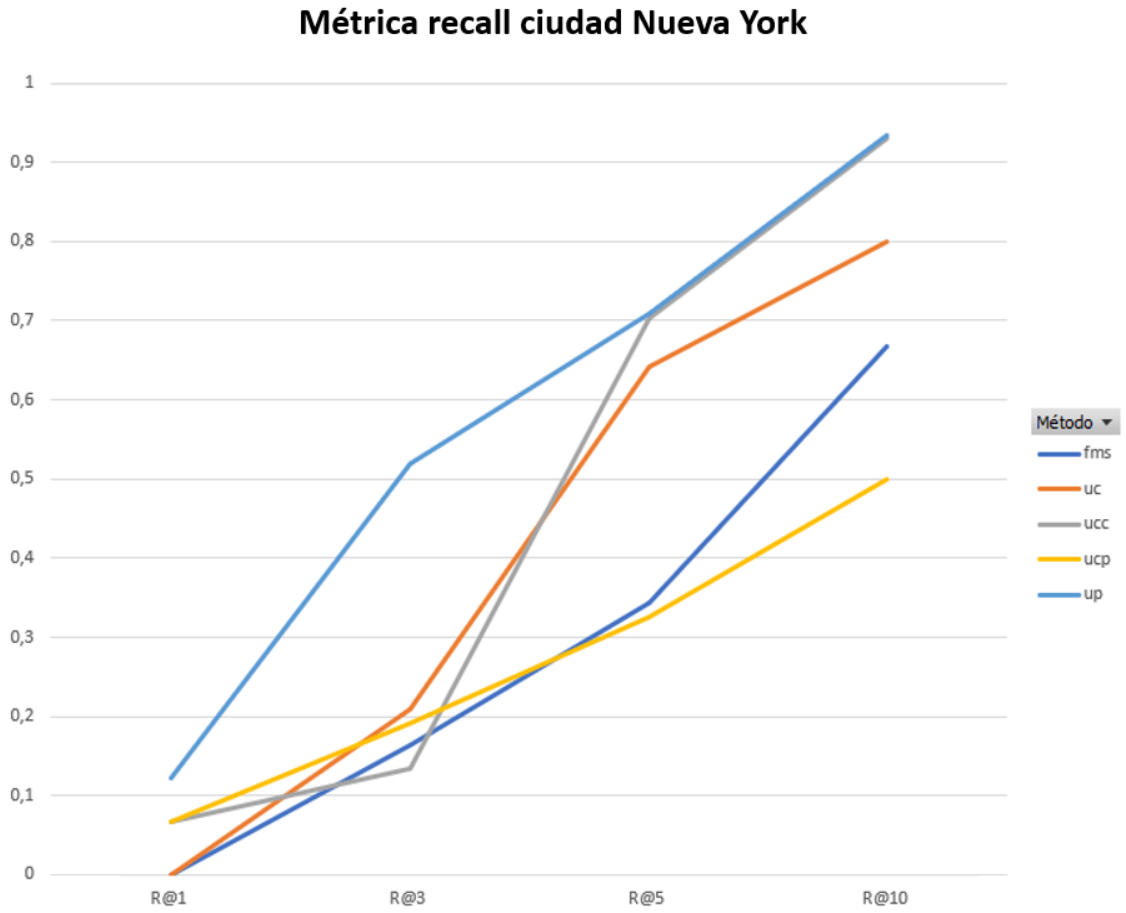

Figura 4.9: Figura que ilustra la tendencia ascendente en los distintos **cutoffs** para la métroca de *recall* en la ciudad de Nueva York. Ejemplificando los resultados de las tablas, creciendo a medida que crece el *cutoff* puede verse como todos los métodos se muestran parejos en un inicio a excepcion de Usuario Parreras que ya desde el inicio muestra una pendiente mucho más pronunciada para acabar, junto a Usuario - Cluster Carreras (ucc) tomando un valor por encima del 90 % en R@10.

Finalmente procede a analizarse la ciudad con más puntos y en cuyas calles más corredores han desarollado rutas de algun tipo, **Chicago**. De manera análoga a los dos casos anteriores en primera instancia se comentará la *Precision* mientras que posteriormente *Recall* será la que se comente. En la figura 4.10 puede verse un comportamiento un tanto errático además de los promedios más bajos en *precision* de todas las gráficas. Es particularmente destacable el aumento constante por parte de los métodos de Usuario - Carreras (uc), de Usuarios Cluster - Carreras (ucc) y de, aunque en menor medida, el método de las FMs (fms). Por contrapartida es destacable cómo los dos métodos restantes empeoran para *cutoff* 3 para luego recuperar una tendencia ascendente.

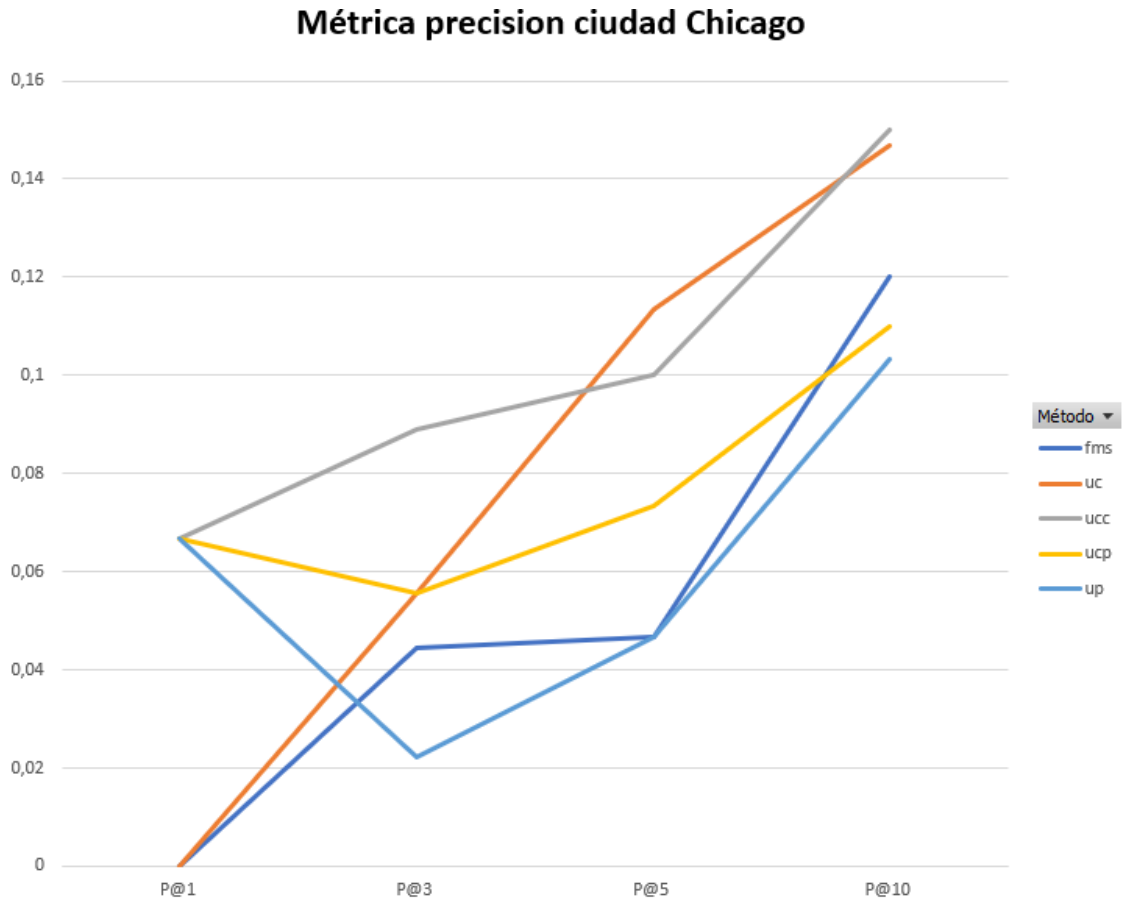

Figura 4.10: Gráfica con los distintos valores de las métodos en *precision* para la ciudad de Chicago. Puede verse la tendencia general de aumentar de valor a medida que se aumenta de tamaño de *cutoff* menos en el caso de Usuario-Cluster Puntos (ucp) y Usuario-Puntos (up) que en P@3 descienden. A pesar de eso todos los métodos crecen en mayor o menor medida para terminar obteniendo en P@10 los mayores valores de la gráfica.

La figura 4.11 muestra el progreso para la métrica de *Recall* en la ciudad de **Chicago**. De nuevo y corresponiendo con todas las tendencias de esta métrica hasta el momento, los valores crecen constantemente en mayor o menor medida, situándose Usuario - Cluster Carreras como el mejor método con diferencia sobre el resto. De esta imagen la particularidad más notoria es cómo los métodos se agrupan por parejas en sus resultados finales. Tanto Usuario - Carreras (uc) como las FMs (fms) terminan juntos en torno al 50 % mientras que Usuario - Cluster Puntos (ucp) y Usuario - Puntos (up) terminan acercándose sin alcanzarlo el 40 %.

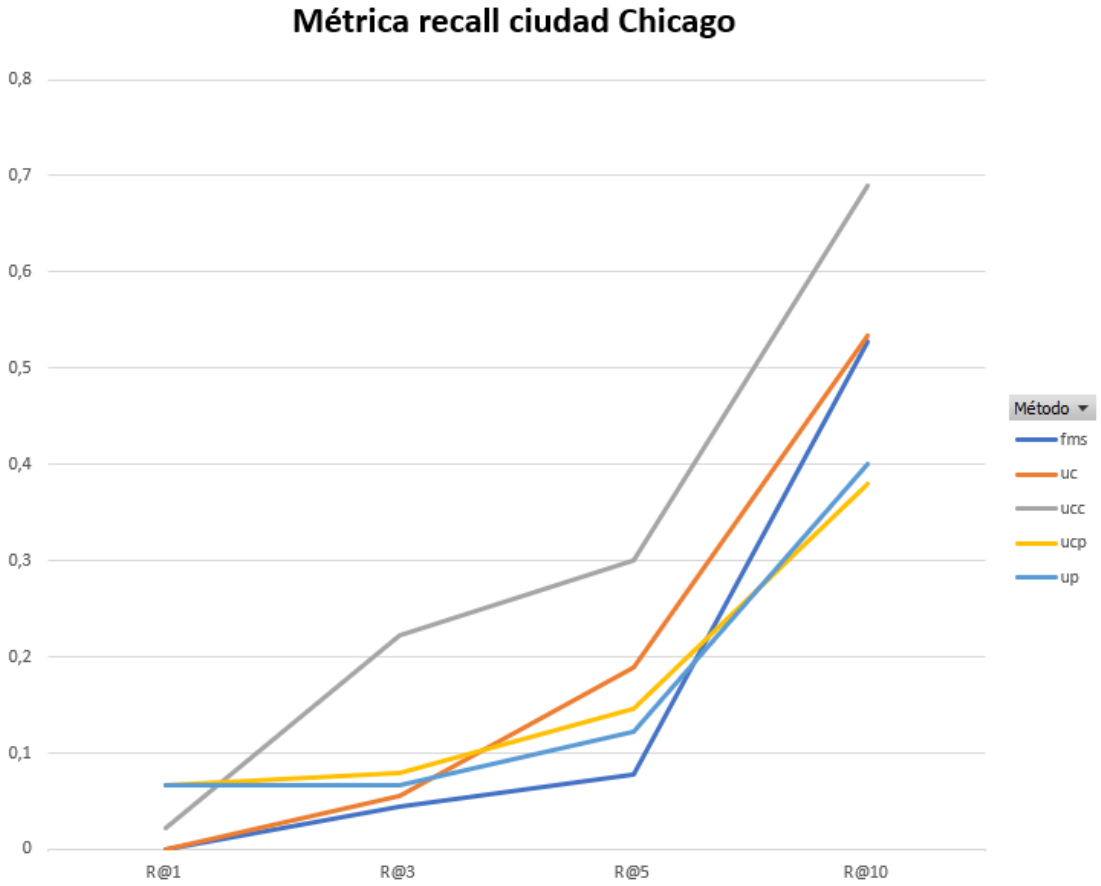

Figura 4.11: Gráfica con los valores de *recall* a lo largo de los distintos *cutoffs* en la ciudad de Chicago. Siguiendo la tónica general de esta métrica para todas las ciudades puede verse de nuevo como el valor aumenta a medida que el tamaño del *cutoff* se hace mayor, la pendiente ascendente se suaviza mucho para @5 pero en @10 puede verse como vuelve a aumentar de manera notoria para repartirse por parejas al final salvo en el caso de Usuario - Cluster Carreras (ucc) donde la tendencia a desmarcarse se puntualiza en este caso para ubicarse en un valor cercano al 70 %.

4.3.3. Baselines

Para contrastar lo bueno o malo de las predicciones anteriores se han elaborado un total de 4 tipos de *baselines* sobre los que igualmente se ha calculado **Precision** y **Recall**. Las aproximaciones sobre las que se han implementado han sido algoritmos simples que se adaptasen a la naturaleza de los datos. Estos también se han ejecutado y promediado 5 veces. Dichos métodos son:

-Aleatorio: se han tomado todas las carreras de las ciudades donde un corredor ha tenido alguna actividad, se han unido en una lista, se han eliminado aquellas que pertenecían a su conjunto de entrenamiento y se ha aleatorizado la lista restante, posteriormente de la misma manera que en los métodos anteriores se ha sometido este ranking aleatorio a un *cutoff* y se ha comparado este conjunto final con el conjunto de test del atleta, calculando a partir de estos valores *precision* y recall. Como solo se van a recomendar carreras de ciudades en las que esta persona ha participado no hace falta ningún control relacionado con no recomendar carreras de otras ciudades.

-Distancia +: en este caso en lugar de aleatorizar las carreras de las ciudades donde un atleta ha corrido se ordenan de más larga a más corta y a partir de aquí, teniendo en cuenta todo el proceso de eliminar carreras de entrenamiento y correcta aplicación del *cutoff* se calculan las predicciones.

-Distancia -: caso hermano del anterior en el que el orden en el que se organizan las carreras en lugar de ser de más larga a más corta es de más corta a más larga, recomendando las carreras con menor extensión primero.

-Elevación +: de manera casi idéntica al anterior en este método se recomienda en base a la mayor elevación, de manera que las carreras con mayor diferencia de desnivel son las que se recomiendan al atleta.

-Elevación -: de nuevo un método sencillo que de manera contraria a su semejante + recomienda primeramente las carreras con menor elevación sobre el resto.

-Duración +: para este caso se han tomado las carreras con mayor duración y en base a esto se ha construido este *baseline*.

-Duración -: finalmente y de manera homónima al resto de casos se ha recomendado en primera instancia las carreras de menor duración sobre las demás.

Con la lógica detrás de los *baselines* propuestos explicada a continuación se adjuntan las tablas para *Precision* y *Recall* de los siete *baselines* en las 3 ciudades sobre las que se ha desarrollado el proyecto. En primera instancia se encuentra la Tabla 4.3 donde puede verse los resultados en *Precision* para los *baselines* en la ciudad de **Sao Paulo**, como puede observarse algún acierto puntual produce unos números muy abultados en @1 pero el resto de valores son bastante bajos. Destaca la relación directa entre Distancia + y Duración +, que se repetirá durante todas las tablas, lógicamente a mayor distancia más se tarda en recorrer la ruta, pasando lo mismo con Distancia - y Duración -.

Cutoff	Aleatorio	Distancia +	Distancia -	Duración +	Duración -	Elevación +	Elevación -
P@1	0	0,200	0	0,200	0	0,270	0,030
P@3	0,110	0,067	0,044	0,067	0,044	0,089	0,030
P@5	0,027	0,040	0,107	0,040	0,107	0,067	0,067
P@10	0,060	0,060	0,087	0,060	0,087	0,080	0,070

Tabla 4.3: Tabla donde pueden observarse los pobres valores para la métrica de *precision* en los distintos *baselines* y *cutoffs* propuestos en la ciudad de Sao Paulo. P@1 cuenta con los mayores valores resultados de todos los *cutoffs*, los cuales descienden a medida que este valor aumenta.

Seguidamente en la tabla 4.4 pueden verse los valores en *Recall* en la misma ciudad, siguiendo la tónica general de los resultados, donde lo valores de esta métrica son más altos que los de la anterior puede verse cómo los valores, aunque aumenten, siguen sin estar cerca de los obtenidos por los métodos propuestos en este TFM. De nuevo se repiten las relaciones distancia/duración comentadas anteriormente.

Cutoff	Aleatorio	Distancia +	Distancia -	Duración +	Duración -	Elevación +	Elevación -
R@1	0	0,067	0	0	0,017	0,089	0,067
R@3	0,140	0,067	0,044	0,067	0,044	0,089	0,038
R@5	0,056	0,067	0,210	0,067	0,210	0,097	0,139
R@10	0,250	0,180	0,377	0,180	0,377	0,220	0,322

Tabla 4.4: Tabla en la que aparecen los resultados de los distintos resultados para la métrica de *recall* en los distintos *baselines* y *cutoffs* propuestos en la ciudad de Sao Paulo. En este caso concreto la tendencia aumenta a medida que se aumenta el *cutoff* rozando en R@10 el 40 %.

Ahora es el turno de la ciudad de **Chicago**, primeramente se muestra (ver Tabla 4.5) la *Precision* para los *baselines*, pero en este caso mostrando resultados muchísimo más pobres para los *cutoffs* pequeños, esto se replica también para valores más altos, encontrándose constantemente en peores números que los de los métodos propuestos. En este caso concreto la cifra es mucho menos abultada debido a que esta ciudad fue la que los métodos tuvieron mayores dificultades para recomendar en esta métrica. Puede verse cómo, salvo en aciertos puntuales para @1 en los correlacionados Distancia -, Duración - y Elevación -, el resto de resultados apenas consiguen rozar el 10 % de acierto.

Cutoff	Aleatorio	Distancia +	Distancia -	Duración +	Duración -	Elevación +	Elevación -
P@1	0,067	0,067	0,200	0,067	0,200	0,067	0,250
P@3	0,022	0,089	0,067	0,089	0,067	0,020	0,150
P@5	0,047	0,089	0,080	0,089	0,080	0,047	0,117
P@10	0,077	0,100	0,077	0,100	0,077	0,077	0,098

Tabla 4.5: Tabla con los distintos resultados para la métrica de *precision* en los distintos *baselines* y *cutoffs* propuestos en la ciudad de Chicago. Elevación + cuenta con los reusltados más pobres junto con Aleatorio mientras que Elevación - es con mucha diferencia el mejor de los baselines en cada uno de los *cutoffs*.

Seguidamente los valores en *Recall* (ver Tabla 4.6) donde pueden verse unos ligeros aumentos en los números pero de nuevo ninguno destaca particularmente sobre los métodos desarrollados. En cualquier caso, de nuevo son los métodos que pertencen al grupo de ordenar por mayor distancia o duración los que presentan los mejores números.

Cutoff	Aleatorio	Distancia +	Distancia -	Duración +	Duración -	Elevación +	Elevación -
R@1	0,022	0,022	0,067	0,022	0,067	0,022	0,088
R@3	0,022	0,220	0,067	0,220	0,067	0,022	0,118
R@5	0,078	0,270	0,130	0,270	0,130	0,077	0,156
R@10	0,344	0,450	0,190	0,450	0,190	0,344	0,220

Tabla 4.6: De nuevo la *recall* para los distintos *baselines* y *cutoffs* sobre la ciudad de Chicago muestra como el baseline Aleatorio y el de Elevación + obtienen los peores resultados frente otros como Duración + Distancia + cuya intrinseca correlación hace que presenten valores idénticos.

Los peores valores obtenidos en todas las ciudades para los *baselines* con diferencia pertenecen a **Nueva York**. Tanto *Precision* (ver Tabla 4.7) como *Recall* (ver Tabla 4.8) muestran unos valores sumamente bajos, destacando cómo no se produce ningún acierto ni siquiera para los métodos que en general en las ciudades anteriores habían estado bien posicionados dentro de los límites de los *baselines*.

Cutoff	Aleatorio	Distancia +	Distancia -	Duración +	Duración -	Elevación +	Elevación -
P@1	0	0	0	0	0	0	0
P@3	0,078	0	0	0	0	0,110	0
P@5	0,087	0	0	0	0	0,067	0
P@10	0,053	0,020	0	0,020	0	0,047	0

Tabla 4.7: Primera tabla en la que aparece una fila entera de ceros. Para la métrica de *precision* en la ciudad de Nueva York el primer *cutoff* no muestra ningún acierto. Lo mismo ocurre para los baselines de Distancia -, Duración - y Elevación -, que en ninguno de los casos obtiene ningún acierto.

Cutoff	Aleatorio	Distancia +	Distancia -	Duración +	Duración -	Elevación +	Elevación -
R@1	0	0	0	0	0	0	0
R@3	0,113	0	0	0	0	0,167	0
R@5	0,079	0	0	0	0	0,167	0
R@10	0,125	0,083	0	0,83	0	0,241	0

Tabla 4.8: La métrica de *recall* deja para los distintos *baselines* y *cutoffs* de la ciudad de Nueva York otra fila llena de ceros y varias columnas, dejando una distribución idéntica de la de *precision* pero con unos valores mucho mayores en la gran mayoría de los casos.

Cabe destacar que ninguno de los *baselines* propuestos, ni siquiera el aleatorio, se acercan al 40 % de acierto. En su caso concreto es sencillo de explicar este pobre desempeño, y es el tamaño de los test, siendo tan extremadamente pequeños que hace muy complicado que estos generen aciertos de manera consistente. Este argumento es también extrapolable, no obstante, para el resto de métodos que obtienen mejores valores. Con respecto de las diferencias entre los '+' y los '-' es que suelen ser estos últimos los que obtienen mejores resultados en la gran mayoría de ciudades salvo alguna excepción puntual aunque, como se ya se ha comentado en el análisis individual, ninguno obtiene valores particularmente altos ni mejores que los métodos implementados en este TFM.

4.3.4. Discusión

Esta sección final está destinada a comentar, comparar y obtener conclusiones acerca de los valores obtenidos en los apartados anteriormente analizados. Como ha podido verse los valores son muy dispares y salvo excepciones asociadas al comportamiento de las métricas respecto de algún *cutoff*, los resultados de las ciudades no parecen relacionarse entre sí. A continuación se desgrana cada uno de los métodos y se analiza su papel en cada una de las muestras anteriores.

Máquinas de factorización (FMs): en conjunto el peor de los métodos, trabajando excepcionalmente mal con *cutoffs* bajo esta técnica ha sido la única que no ha conseguido un resultado reseñablemente alto en ninguna de las ciudades, ni para *precision* ni para *recall*. Tanto es así que no ha conseguido ningún acierto para *cutoff* de 1 en ninguna de las combinaciones posible. El único resultado en el que no ha sido excesivamente opacado por el resto ha sido para el **R@10** de Sao Paulo, ciudad con gran cantidad de carreras y puntos y un número intermedio de corredores. No obstante puesto que su comportamiento es tan negativo y errático es dificil obtener una conclusión concreta más allá de que la implementación propuesta no saca partido de ninguna de las características de cada una de las ciudades puesto que apenas parece mostrar cambios de comportamiento ante diferencias de puntos, de corredores, de carreras de duración o recorrido. La explicación más plausible sería que las FMs necesitan de muchísimos datos para aprender bien, no siendo ese escenario el de este trabajo.

Usuario - Carreras (uc): método sorprendentemente bueno a pesar de lo simple de su implementación. Destacan los buenos valores en las precisiones de Sao Paulo y Chicago donde despunta como el mejor en 2 de los 4 *cutoffs*. En lo que respecta a *recall* es particularmente bueno de nuevo en Sao Paulo mientras que no destaca especialmente entre el resto. Esto puede estar indicando alguna clase de relación directa entre la cantidad de carreras de la ciudad y el comportamiento de este, lo cual tiene sentido por razonamiento lógico. Palidece sin embargo en *cutoffs* extremadamente bajos siendo el segundo método con menos aciertos para **@1** tanto en *precision* como en *recall* indicando que acierta poco el primer elemento mientras que consigue una mayor puntuación a medida que aumentan los *cutoffs* repartiendose entre un número mayor de opciones las posibilidades.

Usuario - Cluster Carreras (ucc): indiscutiblemente el mejor método en la ciudad de Chicago el cual sin embargo tiene pobres resultados en las otras dos ciudades para cualquiera de las dos métricas. La particularidad de la ciudad de Chicago radica en su proporción de puntos por carrera, teniendo la proporción más alta de estos por número de carreras. Así, el algoritmo seguido para obtener el ranking se ve favorecido por la naturaleza de la ciudad, la cual cuenta también con el mayor número de corredores siendo esto un indicativo de una mayor cantidad de elementos con los que trabajar. Sin embargo sus resultados se empobrecen en las otras dos ciudades, a pesar de conseguir unos muy buenos totales para **R@10** alcanzando el 93 % para *cutoffs* más bajos llega a ser el peor de todos los métodos, mostrando su concreta preferencia por el estilo de ciudad de Chicago en la que trabaja muchísimo mejor que en otras. Otro punto a destacar es que estas dos ciudades cuentan con el mayor

número de clusters, siendo Chicago la que más tiene con diferencia del resto, 138 respecto de los 57 de Sao Paulo y los 46 de Nueva York, sin embargo el número de carreras de esta última es la más baja del conjunto y la mitad que Sao Paulo. Por lo que podemos concluir que este método funciona particularmente bien en ciudades con muchos puntos correlacionados entre sí independientemente del número de carreras y corredores mientras que de la misma manera se comporta peor en escenarios que aunque dispongan de muchas carreras tienen mucha dispersión entre sus puntos.

Usuario - Cluster Puntos (ucc): método hermano del anterior que muestra un comportamiento un tanto errático a lo largo de todas las pruebas ejecutadas. A excepción de para **P@10** de Nueva York, donde obtiene el resultado más alto es el que peor se maneja de todos para *cutoffs* altos mientras que se desenvuelve mejor en *cutoffs* pequeños dando a entender que es un método que concentra pocos aciertos en puestos altos mientras que a medida que los rankings se hacen más dispersos sufre más. Sin embargo no es particularmente bueno en nada ni parece mostrar un comportamiento que pueda asociarse a ninguna tendencia concreta de los datos.

Usuario - Puntos (up): un método de implementación relativamente sencilla en comparación con algunos de los vistos en este trabajo y que parece tener los números más sólidos en general de todas las tablas. El mejor método tanto en *precision* como en *recall* de la ciudad de Nueva York, ciudad con menor cantidad de puntos, mostrando buen comportamiento para *cutoffs* bajos, siendo el mejor para todos los **@1** de todas las ciudades lo que indica buen acierto y capacidad de repartir los resultados dentro de unos *cutoffs* razonables cómo los propuestos. Es llamativo sin embargo lo mal que se desenvuelve en la ciudad con más puntos de todas (Chicago) de lo cual podemos decir que se trata de la ciudad con más segmentos, y por lo tanto puntos, más corredores, y lo más importante de todo, con mayor cantidad de clusters. Este dato anteriormente mencionado indica que esta es la ciudad con más puntos a menos de 2 metros entre sí de las tres, concluyendo por tanto que es la que más concentración de puntos muestra, siendo esto pobremente manejado por este método que encuentra en esta particularidad un descenso notable de su horizonte de trabajo, teniendo que manejarse con puntos que prácticamente no le aportan información. A colación de esto se destaca que Nueva York cuenta con una proporción de clusters muchísimo menor.

Esto podría resultar lógico teniendo en cuenta que es la ciudad con menor número de carreras, pero es que incluso cuando se calculan los clusters con una distancia de 3 metros en lugar de 2 no se obtiene ni un sólo cluster extra, cosa que sí pasa en las otras ciudades, 180 en Chicago y 128 en Sao Paulo lo que nos indica una dispersión elevada en esta ciudad, haciendo que el método disponga de información más variada con la que trabajar. Con toda esta información se puede resumir el comportamiento de este método en que requiere de elevada cantidad de dispersión entre sus puntos para trabajar mejor independientemente de la cantidad con la que trabaje y que funciona peor en escenarios con más puntos pegados.

Finalmente se adjuntan dos histogramas, uno por cada métrica, *precision* (ver Figura 4.12) y *recall* (ver Figura 4.13) donde se compara directamente el mejor valor obtenido por cada uno de los métodos con el mejor valor obtenido por cada uno de los *baselines* para *cutoff* 5. En la comparativa de la *precision* puede verse como en dos de las ciudades los métodos superan a los *baselines*. Sin embargo, Chicago, la ciudad en la que los métodos han encontrado mayores problemas para recomendar se ve superada por el *baseline* de Elevacion - aunque sea por un porcentaje ínfimo. El método Usuario-Puntos (up) en Nueva York obtiene resultados ampliamente mejores que el *baseline*, cabe destacar que en esta ciudad los *baselines* han encontrado muchas dificultados para generar recomendaciones.

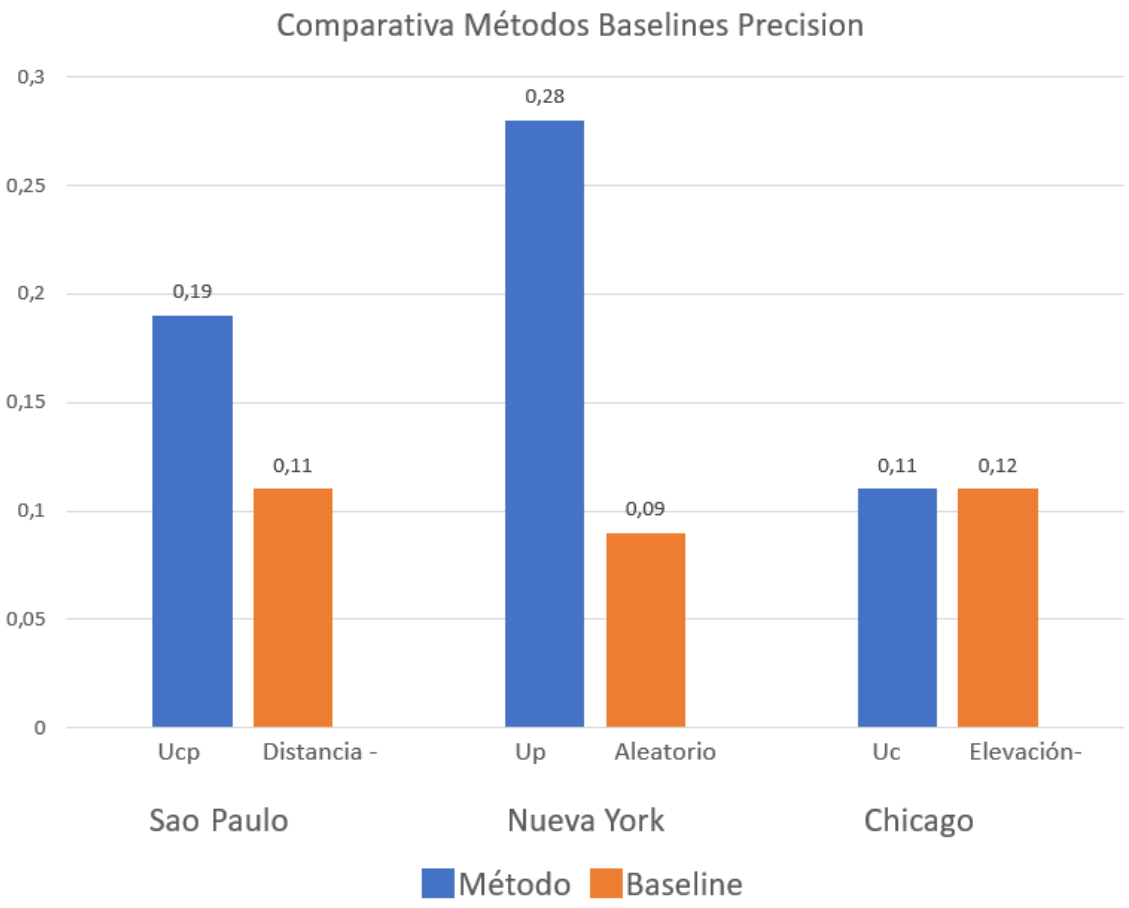

Figura 4.12: Histograma donde pueden verse los valores de los mejores métodos por ciudad comparado con los mejores *baselines* en *cutoff* 5 en *precision*. Como puede verse la tendencia general es la de que los métodos implementados superan en dos de las ciudades mientras que son ligeramente inferiores en 1 de ellas.

Finalmente en la figura 4.13 puede verse la comparativa en las mismas circunstancias para *recall* donde en este caso la diferencia es aún mayor que en el caso anterior. De nuevo puede verse la gran diferencia en Nueva York y aunque sigue siendo un valor bajo, Usuario-Cluster Carreras consigue en Chicago un valor superior al *baseline*.

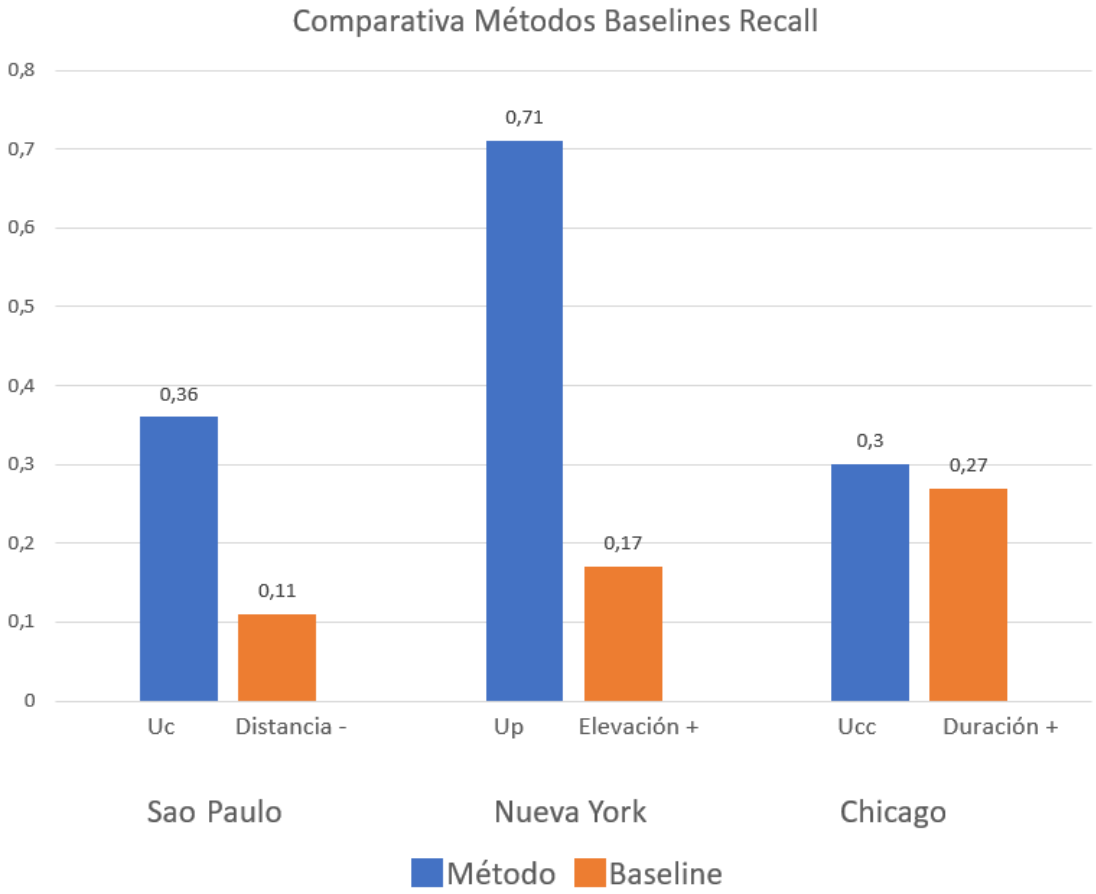

Figura 4.13: Histograma donde pueden verse los valores de los mejores métodos por ciudad comparado con los mejores *baselines* en *cutoff* 5 en *recall*. En este caso ninguno de los *baselines* supera a un método propuesto y en muchos de los casos la diferencia es increiblemente notoria, como en la ciudad de Nueva York, donde el método implementado es más de 4 veces mayor que el mejor *baseline*.

Con esta muestra puede concluirse que los métodos implementados son, en general, mejores que los *baselines* propuestos. Y aunque los máximos particulares puedan acercarse a algunos casos puntuales los valores en general son muchisimo menores para los *baselines* respecto de los métodos.

5

Conclusiones y trabajo futuro

5.1. Conclusiones

En este TFM se ha elaborado, partiendo simplemente desde una idea sin base previa ni estudios directamente relacionados, un sistema capaz de generar recomendaciones acerca de rutas a los usuarios de una aplicación o red social de carácter deportivo. Para conseguir esto se han analizado múltiples APIs de diferentes sitios web con el fin de esclarecer cuáles proporcionan los datos que se consideran más interesantes para este TFM. La API finalmente seleccionada ha sido la de la página Strava, de la cual se han obtenido los datos, rutas recorridas por diferentes atletas en multitud de ciudades a lo largo del globo. Seguidamente, estos se han estudiado en profundidad y filtrado a fin de generar un dataset de trabajo en el que poder realizar experimentos.

Durante el estudio del dataset se extrajeron estadísticas y conclusiones a través de las cuales elaborar distintos métodos para dejar los datos en un formato adecuado con el que ser empleados por dos técnicas de uso extendido y reconocida trayectoria en el mundo de la recomendación, las máquinas de factorización y la factorización de matrices. Para el empleo de la factorización de matrices se han elaborado cuatro matrices que explotan las diferentes carácteristicas que ofrecen las rutas. Estas son, una matriz usuario - ruta, que relaciona directamente un usuario con sus items de la manera más tradicional posible. Una matriz usuario - puntos, que extrae las coordenadas de las que se componen las rutas y las relaciona con los usuarios que han completado la propia ruta. Y finalmente usuario - cluster carreras y usuario - cluster puntos, dos métodos que explotan las similitudes de las rutas considerando como iguales a aquellos que se encuentren a una distancia menor de dos metros, agrupándolos a fin de obtener una mayor concentración de información. De cara a las máquinas de factorización se han generado vectores los cuales relacionan un atleta con las carreras que ha corrido y con las carreras similares a estas.

Con todas estas aproximaciones implementadas se ha utilizado el método de reducción de la dimensionalidad SVD para generar *rankings* a través de las matrices y la librería de máquinas de factorización DeepCTR para obtener *rankings* a partir de los vectores. Estos han sido evaluados a través de las métricas de *Precision* y *Recall*.

A pesar de que los resultados no resultan particularmente altos en algunos de los métodos, sí que se comportan de manera distinta en función de las caracteristicas de los datos con los que están trabajando, pudiendo extraerse conclusiones con las que plantar unas bases y hacerse una idea de qué características resultan más o menos interesantes en unas aproximaciones u otras.

5.2. Trabajo futuro

En primera instancia uno de los puntos más críticos del trabajo ha sido la ausencia de datos, las limitaciones de la API han conllevado que desde el principio la falta de datos fuera muy pronunciada. Esto sumado al hecho de que muchos de los datos no proporcionaban valor por si mismos han llevado al dataset a tener unas proporciones tan pequeñas que han condicionado todo el ciclo del trabajo. De cara al futuro sería interesante aumentar el tamaño del mismo, bien mediante la adhesión de nuevos sitios web de los que extraer datos, solicitándolos directamente a las páginas si existe la opción o contando con un conjunto de usuarios colaboradores que proporcionen de primera mano sus actividades a lo largo de distintas ciudades. Una mayor variedad de datos quizás permitiría obtener conclusiones más interesantes y certeras y daría pie a llevar a cabo unos experimentos más resolutivos y dinámicos.

Otra cuestión interesante sería la de trabajar con los datos de distintas maneras. En algunos estudios que trabajan con rutas se plantean divisiones de las mismas por segmentos en metros, con estas y otras aproximaciones a parte de las empleadas se podrían plantear distintos métodos con los que explorar aún más allá a fin de esclarecer exactamente qué datos resultan interesantes y qué datos no en lo que a recomendación de rutas se refiere.

Explorar los distintos algoritmos disponibles también sería una buena vía de trabajo futuro. Actualmente el set de experimentos es limitado, el método de reducción de la dimensionalidad SVD puede sustituirse por algún otro algoritmo relacionado con el cual se puedan obtener distintas aproximaciones y compararlas con las obtenidas por el primero. En el caso de la libreria de las FMs, DeepCTR, se deberían revisar nuevos conjuntos de parámetros que podrían arrojar nuevas conclusiones sobre el trabajo realizado. Otra posibilidad sería cambiar directamente la librería, lo cual resultaría interesante por lo mencionado en el caso anterior. Posteriormente cambiar los porcentajes de entrenamiento y test y probar con distintas cantidades de ejecuciones con la partición actual y las que se planteen puede resultar también interesante de cara al futuro del trabajo.

Otro punto que podría resultar interesante sería la ampliación del conjunto de baselines, debido a la novedad de este trabajo no se ha sabido exactamente proporcionar un baseline significativo en base a algo demostrado previamente. Un mayor entendimiento tanto del dato como del proceso podrían llevar a proporcionar baselines con mayor peso e interés con los que comparar.

También deberia barajarse el aplicar distintas métricas para testear. Con la aplicación de distintos algoritmos de recomendación podrían aplicarse también maneras distintas de testear estas prediccio-

nes, existen multitud de métricas con las que se podría probar y de esta manera, de la misma forma que únicamente con dos métricas como se usan ahora se obtiene una rica variedad en resultados con una cantidad aún más amplia de estas podrian obtenerse multitud de nuevas conclusiones con las que seguir ajustando trabajo y obteniendo conclusiones más adelante.

Finalmente otra línea de trabajo para el futuro del proyecto consistiría en crear rutas desde cero en lugar de limitarse a la recomendación de rutas ya existentes. En este documento se proponen métodos capaces de recomendar puntos para los usuarios, con lo que con ese punto de partida podría elaborarse trabajo posterior capaz de juntar algunos de estos puntos a fin de crear una ruta que pueda ser del interés de los usuarios.

Bibliografía

[1] D. Sharma, A. Alam, P. Dasgupta, and D. Saha, "A ranking algorithm for online social network search," in *Proceedings of the 6th ACM India Computing Convention*, Compute '13, (New York, NY, USA), Association for Computing Machinery, 2013.

[2] S. Bussines, "Grow your brand with strava." https://business.strava.com/, 2021.

[3] P. Resnick and H. R. Varian, "Recommender systems," *Commun. ACM*, vol. 40, p. 56–58, Mar. 1997.

[4] S. K. Mishra and M. Reddy, "A bottom-up approach to job recommendation system," in *Proceedings of the Recommender Systems Challenge*, RecSys Challenge '16, (New York, NY, USA), Association for Computing Machinery, 2016.

[5] B. Alhijawi and Y. Kilani, "The recommender system: a survey," *International Journal of Advanced Intelligence Paradigms*, vol. 15, no. 3, pp. 229–251, 2020.

[6] F. Ricci, L. Rokach, and B. Shapira, "Recommender systems: introduction and challenges," in *Recommender systems handbook*, pp. 1–34, Springer, 2015.

[7] P. Lops, M. De Gemmis, and G. Semeraro, "Content-based recommender systems: State of the art and trends," *Recommender systems handbook*, pp. 73–105, 2011.

[8] J. B. Schafer, J. A. Konstan, and J. Riedl, "E-commerce recommendation applications," *Data mining and knowledge discovery*, vol. 5, no. 1, pp. 115–153, 2001.

[9] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," *Communications of the ACM*, vol. 35, no. 12, pp. 61–70, 1992.

[10] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," *Knowledge-based systems*, vol. 46, pp. 109–132, 2013.

[11] D. Bridge, M. H. Göker, L. McGinty, and B. Smyth, "Case-based recommender systems," *The Knowledge Engineering Review*, vol. 20, no. 3, pp. 315–320, 2005.

[12] R. R. Sinha, K. Swearingen, *et al.*, "Comparing recommendations made by online systems and friends.," *DELOS*, vol. 106, 2001.

[13] R. Burke, "Hybrid web recommender systems," *The adaptive web*, pp. 377–408, 2007.

[14] R. Sánchez-Guzmán Hitti *et al.*, "Explotación de características de secuencias para su uso en sistemas de recomendación," Master's thesis, 2020.

[15] P. Sánchez Pérez *et al.*, "Estudio y aplicación de algoritmos y estructuras de datos a los sistemas de recomendación," B.S. thesis, 2016.

[16] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[17] S. Rendle, "Factorization machines," in *2010 IEEE International Conference on Data Mining*, pp. 995–1000, 2010.

[18] J. Berndsen, B. Smyth, and A. Lawlor, "A collaborative filtering approach to successfully completing the marathon," pp. 653–658, 2020.

[19] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

[20] J. Berndsen, B. Smyth, and A. Lawlor, "Pace my race: Recommendations for marathon running," pp. 246–250, 2019.

[21] C. Feely, B. Caulfield, A. Lawlor, and B. Smyth, "Providing explainable race-time predictions and training plan recommendations to marathon runners," pp. 539–544, 2020.

[22] J. Berndsen, B. Smyth, and A. Lawlor, "Mining marathon training data to generate useful user profiles," *Communications in Computer and Information Science*, vol. 1324, pp. 113–125, 2020.

[23] Komoot, "Turn your next ride." https://www.komoot.es/, 2021.

[24] R. with GPS, "Get inspired, not lost." https://ridewithgps.com/, 2021.

[25] Postman, "Postman makes api development easy." https://www.postman.com/, 2021.

[26] S. Salvador and P. Chan, "Toward accurate dynamic time warping in linear time and space," *Intelligent Data Analysis*, vol. 11, no. 5, pp. 561–580, 2007.

[27] W. Shen, "Easy-to-use, modular and extendible package of deep-learning based ctr models." https://deepctr-doc.readthedocs.io/en/latest/index.html, 2021.

Acrónimos

API Interfaz de programación de aplicaciones.

CBR Case Based Reasoning.

DGC Discounted Cumulative Gain.

DTW Dynamic time warping.

FMs Máquinas de factorización.

GPS Global Positioning System.

IDE Integrated Development Environment.

IDGC Ideal Discounted Cumulative Gain.

JSON JavaScript Object Notation.

MAE Mean Absolute Error.

MFs Factorización de matrices.

MRR Mean Reciprocal Rank.

NDGC Normalized Discounted Cumulative Gain.

NMF Non-Negative Matrix Factorization.

RMSE Root Mean Square Error.

RRSS Redes sociales.

RS Sistemas de recomendación.

SGD Stochastic Gradient Descent.

SVD Singular Value Decomposition.

SVM Support Vector Machines.

TFIDF Term Frequency Inverse Document Frequency.

TFM Trabajo de fin de Máster.

VSM Vector Space Model.

Apéndices

A

Detalle de los resultados de los métodos implementados

Este anexo se dedica a continuar con el detalle de la explicación de los resultados proporcionada en el apartado 4.3.1. En primera instancia se analizan las diferentes salidas para la ciudad de **Nueva York**. Cabe destacar que esta es la ciudad que cuenta con el menor número de carreras y de puntos. Siendo de la que menos información se dispone en el dataset. De nuevo se empieza analizando la métrica de la *Precision* que se ejemplifica en la Tabla A.1

P@1: En primera instancia puede observarse cómo los valores para este cutoff son los más bajos de cada una de las columnas, lo cual resulta llamativo teniendo en cuenta la naturaleza de la métrica lo que resulta ser indicativo de muy pocos aciertos en general. Usuario - Puntos (up) resulta ser el que obtiene una mayor puntuación frente a las FMs (fms) y el Usuario - Carreras que no consiguen ningún acierto en ninguna de las ejecuciones.

P@3: De nuevo, aumentar la cantidad de elementos en el ranking aumenta considerablemente los porcentajes, revelando que algunos métodos aunque no acierten en un top único consiguen predecir algunos valores extra a lo largo de 3 puestos, y a pesar de que se dividen entre 3 también los valores aumentan notoriamente respecto del *cutoff* anterior. Es de hecho en este *cutoff* en el que se consigue uno de los mayores aciertos de toda la tabla, Usuario - Puntos de nuevo consigue el valor más alto respecto del resto mientras que es Usuario - Cluster Carreras el método que consigue los peores valores aumentando muy ligeramente e indicando unos pobres aciertos incluso cuando se aumenta el rango total.

P@5: *cutoff* para el que se obtiene el mayor porcentaje de toda la tabla. Usuario - Carrera consigue el mayor valor seguido de cerca por Usuario - Puntos. En general se mejoran notoriamente los valores a excepción de las FMs (fms) lo que indica que a pesar del pobre arranque inicial los métodos comienzan a trabajar mejor cuando se trabaja en rangos intermedios.

P@10: Punto final en el que de nuevo puede verse cómo los valores bajan respecto del *cutoff* anterior. A pesar de los pobres números las FMs (fms) son las que reciben la menor reducción de todos los métodos, Usuario - Puntos (up) y Usuario - Cluster Puntos reciben los valores más altos de esta iteración siendo Usuario Puntos el método que demuestra un comportamiento mejor para el general de la métrica.

Cutoff	fms	uc	ucc	ucp	up
P@1	0	0	0,067	0,067	0,167
P@3	0,156	0,200	0,089	0,256	0,289
P@5	0,180	0,300	0,246	0,267	0,287
P@10	0,173	0,187	0,183	0,207	0,200

Tabla A.1: Resultados para la métrica de *precision* en los distintos métodos y *cutoffs* propuestos en la ciudad de Nueva York. Los valores más altos se obtienen en P@5, creciendo desde los dos *cutoffs* anteriores para volver a descender en P@10.

Seguidamente se trata la medida de *Recall* siguiendo la metodología anteriormente empleada. Estos valores pueden verse en la tabla A.2

R@1: De nuevo el *cutoff* con los menores valores de toda la tabla, lo cual reafirma que los métodos consiguen muy poco acierto cuando el *cutoff* se reduce hasta el extremo. Siguiendo la tónica anterior puede verse cómo Usuario - Puntos (up) cuenta con el mejor comportamiento absoluto de todas las columnas y en este tramo no es diferente obteniendo unos valores que doblan a los siguientes de las columnas.

R@3: Usuario - Puntos (up) obtiene unos resultados que superan el 50 % de acierto siendo con diferencia el que mejor porcentaje obtiene dentro de los valores observables. Por su parte el resto de métodos se encuentran al mismo nivel muy por detrás del mismo. Usuario - Cluster Carreras obtiene en este *cutoff* el peor valor de todos, de manera análoga a lo que le pasó en la métrica anterior.

R@5: De nuevo los valores se incrementan de manera contundente, 3 de los métodos superan el 60 % de acierto y dos de ellos consiguen superar incluso el 70 %. Usuario - Cluster Carreras (ucc) obtiene un valor mucho más alto que en el tramo anterior indicando que muchos de los aciertos se encontraban en las posiciones restantes. Usuario - Puntos (up) obtiene por poco el mejor resultado y Usuario - Cluster Puntos (ucp) obtiene el peor.

R@10: Los resultados más abultados de todos los métodos, el comportamiento del *cutoff* anterior se replica y todos los métodos están en el 50 % de acierto o más, dos de ellos por encima del 90 % lo que indica el buen desempeño de los métodos cuando tienen acceso a aún más posiciones del ranking. La diferencia de valores respecto de la métrica anterior nos indica que el tamaño del test es ostensiblemente menor que el *cutoff* lo cual justifica estos valores tan altos.

Cutoff	fms	uc	ucc	ucp	up
R@1	0	0	0,067	0,067	0,120
R@3	0,164	0,208	0,133	0,192	0,519
R@5	0,344	0,646	0,703	0,325	0,708
R@10	0,667	0,800	0,931	0,500	0,930

Tabla A.2: Resultados para la métrica de *recall* en los distintos métodos y *cutoffs* propuestos en la ciudad de Nueva York. En este caso pueden verse unos resultados muy altos respecto de las tablas anteriores, con unos sólidos 93 % de acierto en dos de los métodos, Usuario - Cluster Carreras (ucc) y Usuario - Puntos (up).

Finalmente procede a analizarse la ciudad con más puntos y en cuyas calles más corredores han desarollado rutas de algún tipo, **Chicago**. Se comienza con la métrica de la *Precision* cuyos resultados pueden observarse en la Tabla A.3

P@1: Con un empate triple en 3 de los métodos los resultados para los 5 métodos son terriblemente bajos, ni siquiera el hecho de estar dividiendo entre 1 hace que los números mejoren. De nuevo y replicando un comportamiento previamente visto los dos primeros métodos obtienen los peores valores, sin embargo los aciertos de los otros 3 apenas resultan reseñables.

P@3: Ampliar el *cutoff* hace que ningún método devuelva 0 de acierto, sin embargo en algunos casos el estar dividiendo entre 3 en lugar de entre 1 hace que los números se empobrezcan aún más que en el caso anterior. Usuario - Cluster Carreras (ucc) muestra los mejores valores aumentando muy ligeramente respecto de la vez anterior aunque los números siguen sin alcanzar ni siquiera el 10 % de acierto.

P@5: Los resultados siguen teniendo una tendencia anscendente pero de manera muy ligera, dos de los métodos aumentan el acierto del 10 % pero por ejemplo para las máquinas de factorización el aumento es prácticamente imperceptible, Usuario - Cluster Carreras (ucc) muestra de nuevo valores altos entre el conjunto aunque es Usuario - Carreras (uc) el que muestra los números más altos en este *cutoff* concreto. A pesar de aumentar el denominador puede verse que los números no se empobrecen en absoluto lo que indica que los valores se encuentran dispersos y alejados de los primeros puestos, los de valor más alto, del ranking.

P@10: A pesar de la tendencia tan baja de los valores es llamativo cómo los números no solo no descienden si no que aumentan incluso dividiendo entre 10, lo que refuerza la teoría de que los valores se dispersan desde las primeras posiciones del ranking en adelante. Usuario - Carreras (uc) consigue obtener de hecho el valor más alto de la tabla con un 15 % de acierto y también es destacable el ostensible aumento que puede verse en las FMs (fms) y en Usuario - Puntos (up) respecto del *cutoff* anterior.

Cutoff	fms	uc	ucc	ucp	up
P@1	0	0	0,067	0,067	0,067
P@3	0,044	0,055	0,089	0,056	0,022
P@5	0,046	0,113	0,100	0,073	0,047
P@10	0,120	0,147	0,150	0,110	0,103

Tabla A.3: En esta tabla pueden verse algunos de los resultados más bajos para la métrica de *precision* de todas las tablas. Lo cual indica lo dificultoso de recomendar sobre la ciudad de Chicago.

Para terminar con esta sección se expone la métrica de *Recall* en la ciudad de **Chicago**, que puede verse en la tabla A.4

R@1: Un *cutoff* tan reducido afecta significativamente cómo podemos ver de nuevo en los valores de recall, donde pueden observarse los peores valores en general de todos los *cutoffs*, dos de los métodos empatan pero de nuevo resultan poco reseñables respecto del que no acierta. Las FMs (fms) y el Usuario - Carreras (uc) sufren de nuevo en esta aproximación en la que no consiguen absolutamente ningún acierto.

R@3: Usuario - Cluster Carreras muestra un aumento significativo respecto tanto del *cutoff* anterior como del resto de los métodos de la interacción. Sin embargo es el único que consigue una tendencia tan alentadora. Los demás métodos muestran un pequeño aumento o incluso un estancamiento como en el caso de Usuario - Puntos (up).

R@5: Los porcentajes consiguen un ligero aumento, Usuario - Cluster Carreras sigue siendo el que mejores números obtiene mientras que Usuario - Carreras consigue el aumento más significativo de todos los observables. Usuario Cluster Puntos (ucp) y Usuario - Puntos (up) doblan sus resultados anteriores pero aún con esto siguen tratándose de valores muy bajos en comparación con los obtenidos para las ciudades anteriores en esta métrica y en este *cutoff*.

R@10: Finalmente los valores obtienen una tendencia de aumento severa. Las FMs (fms) consiguen el aumento más notorio de entre todos los métodos mientras que Usuario - Cluster Puntos (up) obtiene el peor de todos los valores. Usuario - Cluster Carreras (ucc) es finalmente el que mejores valores tienen acercándose al 70 % de acierto. De nuevo estos valores hablan de lo que se alejan los resultados de las primeras posiciones del ránking, en este ciudad es particularmente crítico ya que hasta que no se aumenta hasta 10 el valor del *cutoff* los porcentajes toman unos valores irrisorios, incluso en 5, donde las otras ciudades comenzaban a mejorar esta se mantiene estancada. Hablándonos de que Chicago se trata de la ciudad con mayores dificultades de predecir al menos con los materiales actuales.

Cutoff	fms	uc	ucc	ucp	up
R@1	0	0	0,020	0,067	0,067
R@3	0,044	0,055	0,222	0,080	0,067
R@5	0,078	0,189	0,300	0,146	0,122
R@10	0,528	0,533	0,689	0,379	0,400

Tabla A.4: Tabla en la que aparecen los resultados de los distintos resultados para la métrica de recall en los distintos métodos y *cutoffs* propuestos en la ciudad de Chicago. Usuario - Cluster Carreras (ucc) muestra una tendencia a ser el método predominante coronando en un casi 70 % de acierto en R@10.


UAM
UNIVERSIDAD AUTONOMA
DE MADRID