UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Doble Grado en Ingeniería Informática y Matemáticas

TRABAJO FIN DE GRADO

Predicción y recomendación de enlaces en redes sociales

Autor: Antonio Amor Mourelle
Tutor: Javier Sanz-Cruzado Puig
Ponente: Pablo Castells Azpilicueta

junio 2019

Antonio Amor Mourelle
Predicción y recomendación de enlaces en redes sociales

Antonio Amor Mourelle

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

Lo fácil es hacer la fórmula difícil.

Pablo Castells

Agradecimientos

En primer lugar, quiero dar las gracias a mis padres, que me enseñaron que a base de esfuerzo se puede hacer cualquier cosa. Seguidamente quiero agradecerle a alguien muy especial, que siempre a estado ahí en los momentos de dificultad, haciéndome seguir adelante. Por último a todos aquellos con los que me ha sido posible desconectar y relajarme y con los que he podido recargar fuerzas para seguir esforzándome.

Resumen

Los sitemas de recomendación se empezaron a formar como área hace más de 2 décadas. Desde entonces, grandes empresas como Amazon,Spotify, Netflix, Twitter o Facebook las utilizan para proveer de contenido a los usuarios de sus plataformas. Y a pesar del buen funcionamiento de estos sistemas, a día de hoy se sigue investigando como mejorar su funcionamiento, ya no solo desde el punto de vista del acierto en las recomendaciones sino también en otras cuestiones como la novedad que estas aportan.

Este TFG tiene como objetivo seguir profundizando en el análisis de los sistemas de recomendación para las redes sociales, poniendo énfasis en 2 cuestiones. En primer lugar, el análisis de nuevas perspectivas del valor de la recomendación, en particular la relevancia recíproca, cuyo objetivo es encontrar enlaces bidireccionales entre usuarios ya que estos enlaces aportan por lo general más valor a los usuarios de una red social así como a la propia plataforma. Este análisis se ha llevado a cabo mediante un estudio experimental en el que se observará el comportamiento de algoritmos clásicos en la recomendación de contactos, tanto algoritmos basados en vecindarios como algoritmos de filtrado colaborativo. Los resultados muestran que al medir la precisión reciproca de las recomendaciones resulta mejor emplear algoritmos basados en vecindarios, al contrario que en precisión estándar, donde resulta conveniente utilizar algoritmos de filtrado colaborativo.

Segundo, durante este trabajo se estudia el problema también desde el punto de vista de clasificación. Esto se debe a que todavía la comunidad científica no siempre es precisa en si el problema de “recomendación de contactos” se resuelve con mejores resultados si aplicamos la metodología de clasificación o la de ranking. Para analizar esta cuestión se formuló un problema de clasificación binaria (“existe enlace” o “no existe enlace”) y se predijeron todos los enlaces posibles entre 2 usuarios de la red social. Los resultados muestran que predecir con los algoritmos basados en vecindarios no proporciona grandes resultados cuando se evalúa y que utilizar métodos de filtrado colaborativo, aunque mejora los resultados, todavía resulta en un acierto escaso. Hemos comprobado no obstante que así como los mejores algoritmos en acierto simple y acierto reciproco no son los mismos, en general la comparativa entre algoritmos en términos de ranking y de clasificación tiende a concordar.

Palabras clave

Sistema de recomendación, algoritmos basados en vecindarios, filtrado colaborativo, clasificación, recomendación, precisión, contacto, recomendación de contactos, predicción de enlaces

Abstract

The recommendation systems started to be developed more than 2 decades ago. Since then, big companies like Amazon,Spotify, Netflix, Twitter or Facebook use them to provide content to the users of their platforms. And in spite of the good performance of these systems, to this day research continues on how to improve their functioning, not only from the point of view of the correctness of the recommendations but also in other issues such as the novelty they bring.

The objective of the present work is to continue deepening the analysis of recommendation systems for social networks, placing emphasis on 2 issues. Firstly, the analysis of new perspectives on the value of the recommendation, in particular the reciprocal relevance, which objective is to find bidirectional links between users as these links are more valuable to users of a social network. This analysis has been carried out by means of an experimental study in which the behaviour of classic algorithms (both neightbourhoods and collaborative filtering algorithms) will be observed. The results show that the recommendation systems captures more bidirectional links while recommending with the neightbourhood algorithms, as opposed to standard precision, where it is convenient to use collaborative filtering algorithms.

Second, during this work the problem of recommendation is also studied from the point of view of classification. This is due to the fact that the scientific community is not always clear about whether the problem of “recommendation of contacts” is solved with better results if we apply the classification methodology or the recommendation methodology. In order to analyze this question, a problem of binary classification (“link does exists” or “link doesn’t exists”) was formulated and all possible links between 2 users of the social network were predicted. The results show that predicting with neighborhood-based algorithms does not provide great results and that using collaborative filtering methods, while improving results, still results in poor success. However, we have found that just as the best algorithms in simple success and reciprocal success are not the same, in general the comparison between algorithms in terms of ranking and classification tends to coordinate.

Keywords

Recomendation system, neighbourhood algorithms, collaborative filtering, classification, recommendation, precision, contact, contact recommendation, link prediction

Índice

Listas

Lista de ecuaciones

Lista de figuras

Lista de tablas

1

Introducción

Este capítulo sirve a modo de introducción del presente TFG. En primer lugar se detallará la motivación que lleva a la realización de este trabajo y posteriormente se listarán los objetivos que se derivan de ella y la metodología seguida.

1.1. Motivación

Los sistemas de recomendación empezaron a formar a mediados de los años 90. Desde entonces, su objetivo es satisfacer la demanda que tienen los usuarios de la plataforma proporcionándoles contenido que, el sistema estima, será de sus interés. En la actualidad estos sistemas juegan un papel cada vez más relevante en las plataformas más utilizadas por el público general. Spotify y Netflix son ejemplos de servicios multimedia que utilizan recomendación y Twitter es una de las redes sociales que implementan este tipo de servicio para recomendar, entre otras cosas, contactos. Y es en este último ámbito donde se centra el presente TFG. En general, este problema ha sido estudiado desde 2 perspectivas diferentes: fundamentalmente, como un caso específico de la recomendación pero también como una tarea de clasificación, conocida como predicción de enlaces. Las diferencias entre ambas tareas son difusas y han dado lugar a diferentes enfoques, sobre todo a la hora de evaluar. Por ello, resulta de interés llevar a cabo un estudio con el objetivo de discernir las similitudes y diferencias entre ambas tareas.

En la recomendación de contactos, caben diferentes definiciones y matices de lo que significa una recomendación acertada por lo que se plantean cambios en las perspectivas de acierto a la hora de evaluar. Generalmente consideramos acierto si el usuario acepta la recomendación pero cabría exigir, además, la interacción prolongada en el tiempo entre los usuarios (más allá de la mera aceptación de la recomendación) o que cuando se acepte el nuevo contacto, este sea correspondido obteniendo una relación de seguimiento recíproca. Este trabajo se enfoca en contrastar la segunda perspectiva (los contactos recíprocos) con el acierto básico (contactos unidireccionales), realizando experimentos que analizan el comportamiento de los algoritmos habituales cuando se buscan estos enlaces más fuertes.

1.2. Objetivos

A raíz de la motivación descrita en la sección anterior se plantea como principal objetivo analizar el comportamientos de varios algoritmos de recomendación de contactos bajo diferentes perspectivas, incluyendo la perspectiva de un problema de clasificación, de un problema de ordenación de elementos en un ránking, así como otras perspectivas adicionales tales como la novedad de la recomendación. Concretamente se tratarán las siguientes cuestiones:

- Comparar los métodos de evaluación de dos tareas tan similares como la predicción de enlaces y la recomendación de contactos.
- Proponer nuevas métricas que permitan analizar propiedades adicionales de los algoritmos de recomendación de contactos. En concreto, que permitan analizar si el enlace recomendado no solamente ha sido creado, sino que, en el caso de redes dirigidas, ha sido devuelto, formándose así una nueva relación reciproca en la red social.
- Comparar los diferentes algoritmos de recomendación en función de esta nueva perspectiva.
- Analizar los motivos de las posibles distintas entre las diferentes perspectivas.

1.3. Metodología

Para lograr los objetivos descritos, se ha actuado según el siguiente proceso de trabajo. En primer lugar, se ha estudiado a fondo la literatura relacionada con la tarea de recomendación con el objetivo de obtener una primera visión y familiarizarse con los conceptos básicos, para posteriormente profundizar en la recomendación de contactos. A su vez, con el fin de realizar una comparativa entre la recomendación y predicción de contactos, se ha consultado documentación sobre la metodología de predicción, habitual en el aprendizaje automático.

Una vez se ha entendido el problema, se ha implementado un módulo en Python junto con algunos de los algoritmos de recomendaciónde contactos. A estos algoritmos se les añaden algunos tomados de librerías externas sobre los que ha habido que consultar su documentación y la interacción con las métricas que se evaluarán sobre ellos. Tras esto, se ha estudiado el conjunto de datos escogido para la evaluación de los algoritmos, transformándolo en el formato escogido por el módulo implementado.

Por último se ha procedido al diseño y ejecución de una batería de experimentos. En primer lugar, se ha realizado un barrido de parámetros con el que se busca encontrar las mejores configuraciones para la precisión recíproca y posteriormente se han comparado la metodología de predicción y la metodología de recomendación.

2

Estado del arte

Los sistemas de recomendación surgieron con la aparición de sistemas como *GroupLens* [15] durante mediados de los años 90. Desde entonces se han desarrollado una gran cantidad de algoritmos y metodología que han permitido alcanzar la gran popularidad de la que gozan a día de hoy. Este capítulo detalla el momento actual en el que se encuentran los sistemas de recomendación, empezando desde un punto de vista general y, posteriormente, focalizándolo a la recomendación de contactos y la predicción de enlaces.

2.1. Sistemas de recomendación

El principal objetivo de un sistema de recomendación es ayudar al usuario de la plataforma a descubrir nuevo contenido que le resultará de interés. Esta tarea puede parecer trivial, sin embargo debido al gran volumen de datos manejados hoy en día, estos sistemas se han convertido en una necesidad. Con los más de 400 millones de productos distintos manejados por Amazon, resulta evidente la dificultad de encontrar productos que resulten interesantes para el usuario de esta plataforma. Algo similar ocurre con Netflix, donde la suma de series y películas en EE.UU. supera las 5.900. Grandes empresas han realizado estudios [1] donde se estima que el 35 % de las compras en Amazon y el 75 % del contenido visto en Netflix se producen gracias a las recomendaciones proporcionadas a los usuarios.

Dependiendo de las características del problema planteado, existen diferentes estrategias para lograr recomendar con éxito, estas se verán en la siguiente sección.

2.2. Tipos de sistemas de recomendación

Dependiendo de los datos utilizados para llevar a cabo las recomendaciones podemos encontrar 3 grandes grupos de sistemas de recomendación.

- **Basados en contenido** [2, chap. 1.3.2], utiliza las características de los *item* para recomen-

[1] https://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers

dar contenido similar al que a sido marcado como satisfactorio previamente por el usuario.

- **Basados en filtrado colaborativo** [2, chap. 1.3.1], recomienda contenido que resultó satisfactorio para usuarios similares, para ello se utilizan las características de los usuarios, tanto explicitas (sexo, edad,...) como implícitas (gustos observados, clicks...).
- **Híbridos** [3], combinaciones de los anteriores .

2.2.1. Sistemas de recomendación basados en contenido

Los sistemas de recomendación basados en contenido surgen de la intuición de que al usuario le gustarán *items* similares a los que ya le han gustado en el pasado. Tomemos como ejemplo a un usuario de Netflix, que ve “Batman:Begins”. Ahora el sistema puede recomendarle contenido que disponga de etiquetas similares, como otras películas de acción o superhéroes. Este ejemplo se muestra visualmente en la figura 2.1.

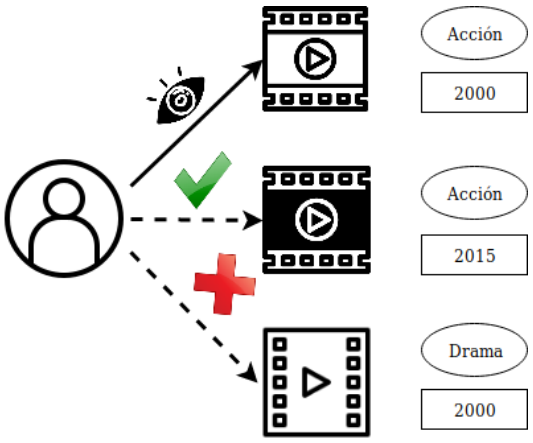


Figura 2.1: Ejemplo gráfico de recomendación basada en contenido

El principal problema de este tipo de recomendación es la percepción de encasillamiento que obtienen los usuarios al obtener recomendaciones centradas exclusivamente en ciertos temas y a lo que se le añade una pérdida del descubrimiento de nuevo contenido.

2.2.2. Filtrado colaborativo

Un sistema de filtrado colaborativo se basa en la idea que la interacción con los *items* que tienen dos usuarios parecidos es parecida. De esta forma el sistema puede estimar la valoración que un usuario puntuará a un *item* que todavía no ha consumido. Supongamos por ejemplo, dos usuarios de Netflix con gustos similares. El primero de ellos, u, lleva más tiempo en la plataforma y ha visto más contenido que el segundo usuario, v. Sin embargo, las películas que v ha marcado como de su gusto son, en su mayor parte, también del gusto de u. De esta forma el sistema reconoce que existe un parecido entre

u y v y le recomendará a v contenido todavía no visualizado por él y que hubiese sido del interés de u. La figura 2.2 ilustra este ejemplo.

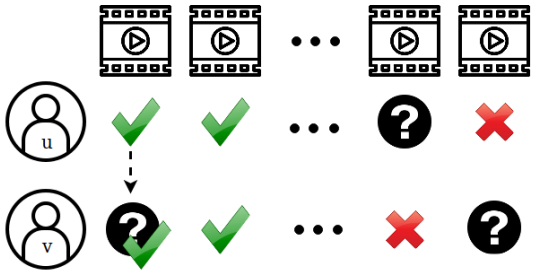


Figura 2.2: Ejemplo gráfico de recomendación por filtrado colaborativo

Su principal problema se conoce como *cold start* [18]. Este se refiere a la incertidumbre que generan los nuevos usuarios o *items* en el sistema. A estos usuarios no se les puede recomendar debido a la falta de información sobre ellos mientras que los *items* no pueden ser recomendados por la falta de valoraciones por parte de los usuarios.

2.2.3. Sistemas de recomendación híbridos

Cada uno de los sistemas de recomendación expuestos anteriormente se caracteriza por tener sus ventajas y sus inconvenientes. La creación de los sistemas híbridos viene motivada por la idea de "juntos somos más fuertes". Al combinar los diferentes recomendadores se consiguen grandes resultados gracias a que se consiguen compensar los defectos de cada uno de ellos.

Existen 2 grandes grupos de sistemas de recomendación híbridos:

- **Combinación de los algoritmos**, la recomendación la proporciona un único sistema de recomendación programado con un algoritmo resultado de combinar las dos filosofías anteriormente expuestas.
- **Mezcla de los resultados**, los resultados obtenidos se obtienen a partir de diferentes recomendadores. Por ejemplo, usar un recomendador u otro en función del contexto, o combinar linealmente los resultados obtenidos por cada uno de los sistemas de recomendación.

2.3. Formalización del problema de recomendación

Llegado este momento, merece la pena pararse un segundo a plantearse el problema general de la recomendación desde un punto de vista formal. La tarea de recomendar puede entenderse de manera sencilla como asignarle una probabilidad a que a un usuario le guste un *item*.

Para formalizar este concepto debemos realizar varias definiciones previamente:

- $\mathcal{U}$, conjunto de los usuarios a los que se le va a realizar una recomendación.
- I, conjunto de los *items* que podemos recomendar a los usuarios.
- I_u, subconjunto de I con los *items* con los que el usuario u, perteneciente a U, esta satisfecho.

Una vez llevadas a cabo estas consideraciones la formulación de recomendar un *item* a un usuario queda como muestra la formula 2.1:

$$rec_u(i) = \begin{cases} i & si\ \mathcal{P}(i \in I_u) >= x \\ \emptyset & si\ \mathcal{P}(i \in I_u) < x \end{cases} \tag{2.1}$$

donde $i \in I$, $u \in \mathcal{U}$ y $x \in [0, 1]$ un valor umbral a partir del cual se recomienda el *item*.

Pero la definición anterior puede resultar poco práctica en situaciones reales donde se pretende siempre realizar una recomendación, sobre todo debido a la presencia del parámetro umbral x donde si ningún *item* supera el valor del umbral no se realizaría ninguna recomendación. Es por ello que realizamos una modificación a la definición para deshacernos de este parámetro.

$$rec_u = \{i_j \mid max_k(\mathcal{P}(i_j \in I_u))\} \tag{2.2}$$

donde $i_j \in I$ y $u \in \mathcal{U}$.

Pero la definición 2.2 puede hacerse más general, para ello solo tenemos que sustituir la función de probabilidad $\mathcal{P}$ con cualquier otra función f que nos devuelva un valor con el que poder puntuar la afinidad que tiene un usuario por un *item*. Así, finalmente, la recomendación a un usuario puede definirse como:

$$rec_u = \{i_j \mid max_k(f_u(i_j))\} \tag{2.3}$$

donde $i_j \in I$ y $u \in \mathcal{U}$.

Con este cambio de definición hemos pasado de un problema binario de clases “recomendar” o “no recomendar” a un problema de *ranking*. La formulación genérica 2.3 da lugar a los diferentes tipos de recomendación utilizados actualmente y que se detallan en la sección 2.2.

2.4. Recomendación de contactos

Ahora queremos centrarnos en la recomendación de contactos [4]. Este tipo de recomendación tiene como objetivo descubrir relaciones entre personas, sean ya existentes o vayan a aparecer en el futuro. Su principal característica es que el conjunto de *items* a recomendar pertenece al mismo espacio que el usuario. A pesar de no ser exclusivo de las redes sociales (también existe recomendación de

personas en aplicaciones de citas online o la búsqueda de expertos por ejemplo) es aquí donde cobra especial interés.

Debido a la naturaleza de las redes sociales, recomendar contactos es claramente un problema de *ranking* ya que los diferentes contactos resultantes de la recomendación se le ofrecen al usuario. Con esta consideración la formula podemos utilizar la formula de recomendación 2.3, que queda modificada de la siguiente manera:

$$rec_u = \{v \mid max_k(f_u(v))\} \tag{2.4}$$

donde $u, v \in \mathcal{U}$.

A pesar de ser un caso tan concreto, abstrayéndose el problema de la recomendación de contactos puede verse como un problema de recomendación genérico, pudiendo entonces aplicarse cualquier tipo de recomendador de los mencionados en la sección 2.2. Sin embargo, realizar esto supone desperdiciar una gran parte de la información que nos brinda que estemos es una red social y que nuestros *items* sean los usuarios de la misma. Es por ello que durante este TFG no realizamos dicha abstracción y se utilizarán algoritmos enfocados diseñados para la recomendación de contactos.

2.5. Predicción de enlaces

La predicción de enlaces [10] consiste en encontrar nuevos enlaces (o existentes pero observados) en una red a partir del estado de una red actual. Esta red, no tiene necesariamente un carácter social, un ejemplo de esto puede ser una red criminal, o las relaciones comerciales entre diferentes empresas. Sin embargo, durante este trabajo nos restringiremos a las redes sociales con el fin de observar las diferencias en entre la tarea de recomendación de contactos explicada anteriormente y esta tarea de predicción de enlaces.

Si abstraemos el problema de recomendación de contactos (2.4), obviando que los *items* son usuarios y que las conexiones entre *items* y usuarios son relaciones personales, obtenemos el problema de predicción de enlaces en un grafo cualquiera y, a pesar de esto, los problemas son afrontados desde puntos de vista distintos. Esto se debe a que se buscan finalidades distintas, en la recomendación de contactos se pone como foco la satisfacción del usuario, ofreciéndole contactar con gente que ya conoce fuera de la red social o con nuevas personas con las que comparte ciertas características. Por otro lado, el objetivo de la predicción de enlaces se centra en el grafo en si mismo, en su topología y las características derivadas del mismo.

Debido a esto no se utilizan las mismas técnicas de ranking utilizadas en la recomendación de contactos. Cuando se plantea este problema se acuden a mecanismos habituales del aprendizaje automático, concretamente los métodos de aprendizaje supervisado.

En los métodos supervisados, se modeliza la supervisión como un problema de clasificación binaria con las clases “existe link” o “no existe link”. Para resolver este problema pueden utilizarse puede utilizarse cualquier mecanismo de clasificación, desde arboles de decisión o *random forest* hasta técnicas más avanzadas como redes neuronales [7]. Posteriormente se utilizan métricas especificas como *area under curve* (AUC) que se corresponde con el área de la curva ROC.

3

ALGORITMIA

El desarrollo de algoritmos para la recomendación de contactos y la predicción de enlaces ha proliferado enormemente en los últimos años, en los que han aparecido multitud de técnicas y herramientas para ambas tareas.

La figura 3.1 muestra una selección de dichos algoritmos, que serán explicados en esta sección y posteriormente utilizados a lo largo de los experimentos. Concretamente se han seleccionado 3 grandes tipos de algoritmos. En primer lugar, algoritmos que utilizan la estructura de la red, concretamente los vecinadrios de los usuarios que la componen. En segundo lugar, algoritmos de recomendación clásicos, basados en filtrado colaborativo. Por último, detallamos una implementación de la factorización de matrices, esta técnica proviene del filtrado colaborativo pero debido a su gran presencia en el estado del arte en los últimos años, la factorización de matrices se ha ganado la categoría de tipo de algoritmo.

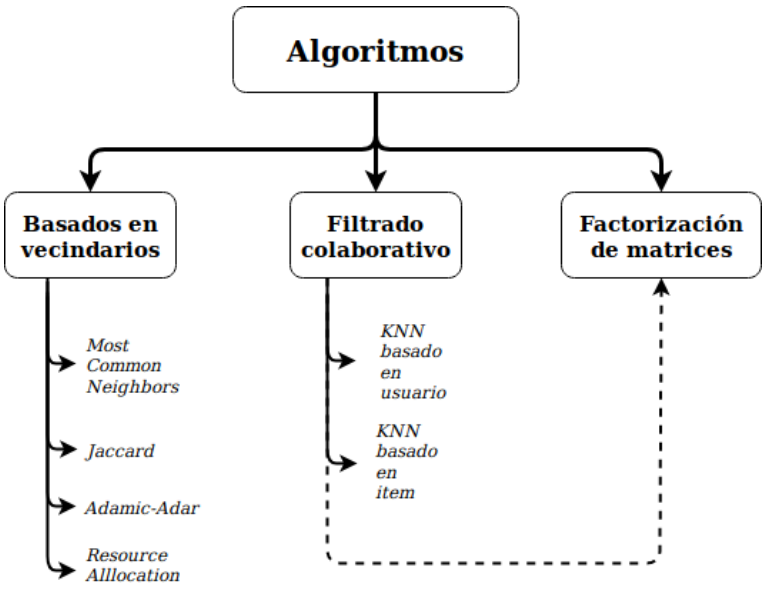


Figura 3.1: Algoritmos de recomendación de contactos tratados en el TFG

Este capítulo se divide en 2 secciones, en la primera sección (3.1) se detallarán los algoritmos implementados durante el TFG, que se corresponden con los algoritmos basados en vecindarios. En

la sección 3.2, detallaremos el resto de los algoritmos que se muestran en la figura 3.1, cuya implementación se obtiene de módulos externas.

Además, todos estos algoritmos tienen en cuenta la gran diferencia con cualquier tipo de recomendación, una vez se tiene contacto con un usuario, este no puede volver a ser recomendado.

3.1. Algoritmos implementados

Durante esta sección se detallan los algoritmos implementados en el TFG. En primer lugar, en la sección 3.1.1 introduciremos dos algoritmos utilizados como lineas base en la literatura. En primer lugar se detallarán la recomendación aleatoria y la recomendación basada en popularidad. Posteriormente (en la sección 3.1.2) se introducirán los algoritmos basados en la estructura de la red, concretamente los algoritmos basados en vecindarios.

3.1.1. Algoritmos básicos

Los algoritmos de esta sección (Aleatorio y Popularidad) tienen la utilidad de servir como elementos de control a la hora de evaluar nuevos algoritmos. Ambos se caracterizan por ser algoritmos de recomendación no personalizados. La simplicidad de la que gozan los hace rápidos de implementar y eficientes, pero en contra no obtienen grandes resultados.

Aleatorio

El algoritmo de recomendación aleatoria es el más usado como linea base comparativa en la literatura. esto se debe a que ningún algoritmo de recomendación debe funcionar peor que una recomendación aleatoria de los *items*. Con esta intuición, recomendación aleatoria nos proporciona una medida de la mejora de los algoritmos de recomendación que se desarrollen posteriormente.

Dado que no existe ningún criterio para ordenar a los usuarios a recomendar, este algoritmo se modela siguiendo una distribución de probabilidad uniforme en el intervalo cerrado [0,1].

$$f_u(v) \sim U[0,1] \tag{3.1}$$

Popularidad

Este algoritmo resulta fácil de justificar en el contexto de recomendación de contactos en el que nos enfocamos. La aparición de las redes sociales no dirigidas ha permitido a sus usuarios poder seguir a

sus personajes famosos favoritos en las redes sociales. Sin embargo, este no es el único ámbito donde la recomendación por popularidad tiene sentido. Estamos acostumbrados a ella en una gran variedad de servicios de diferente índole: en el contexto cinematográfico nos encontramos con "Número 1 en cines", en el sector *retail* con "Él más vendido", etc. Todos estos casos son ejemplos de recomendación no personalizada en la que se sigue un criterio y con él que se obtienen resultados mejores que el aleatorio. Debido a su simplicidad este algoritmo también es utilizado para establecer una comparativa inicial. La formula resultante es:

$$f_u = |\Gamma_{in}(v)| \tag{3.2}$$

3.1.2. Algoritmos basados en vecindarios

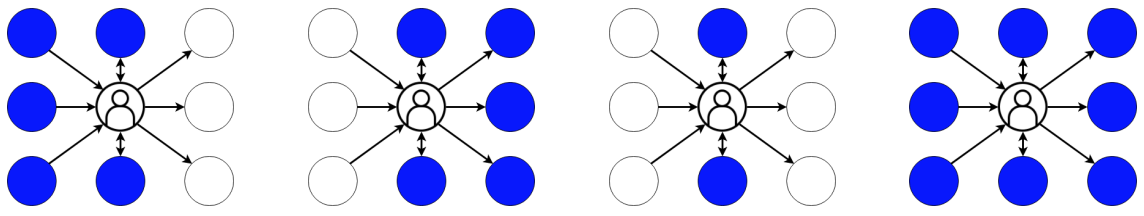
La recomendación de contactos se presta al enfoque en el que los elementos más relevantes para encontrar nuevos contactos son los contactos ya existentes en la red. Este enfoque por tanto pretende predecir la afinidad que el usuario objetivo u tendría con el usuario candidato a recomendar v en función de los contactos de ambos: $\Gamma(u)$ y $\Gamma(v)$. En una red no dirigida (por ejemplo Facebook), el vecindario de un usuario es único. Sin embargo, en las redes dirigidas (como el caso de Twitter) podemos hacer varias distinciones, las 2 más básicas son:

- $\Gamma_{in}(u)$, conjunto de usuarios que siguen a u.
- $\Gamma_{out}(u)$, conjunto de usuarios seguidos por u.

Además a partir de estos dos vecindarios podemos realizar 2 nuevas definiciones:

- $\Gamma_{int}(u) := \Gamma_{in}(u) \cap \Gamma_{out}(u)$.
- $\Gamma_{und}(u) := \Gamma_{in}(u) \cup \Gamma_{out}(u)$.

Estos vecindarios se muestran en la figura 3.2.



(a) Γ_{in} (b) Γ_{out} (c) Γ_{int} (d) Γ_{und}

Figura 3.2: Posibles vecindarios para un usuario

Dicho esto, en las siguientes subsecciones se detallan diferentes algoritmos de recomendación

basados en vecindarios. En ellos utilizaremos $\Gamma(u)$ para referirnos a un vecindario, pudiendo ser sustituido por cualquiera de los subconjuntos arriba detallados.

Most Common Neighbors

El algoritmo de *Most Common Neighbors* (MCN) [12] tiene como objetivo recomendar los usuarios que comparten un mayor número de contactos con el usuario objetivo. La intuición de este simple algoritmo es que cuantas más conexiones compartan dos usuarios, más sencillo es que quieran conectarse entre ellos. Para llevar a cabo esta tarea se sigue la formula 3.3.

$$f_u(v) = |\Gamma(u) \cap \Gamma(v)| \tag{3.3}$$

Jaccard

El algoritmo de Jaccard [10] [12] es una modificación de MCN. Surge ya que MCN solo tiene en consideración el número de vecinos comunes entre u y v. En esta situación los usuarios populares obtienen una importante ventaja. Para solucionarlo, Jaccard mide la proporción de contactos en común. La fórmula resultante es la 3.4.

$$f_u(v) = \frac{|\Gamma(u) \cap \Gamma(v)|}{|\Gamma(u) \cup \Gamma(v)|} \in [0, 1] \tag{3.4}$$

Adamic-Adar

Adamic-Adar [1] vuelve a ser una modificación de MCN. En MCN, se observa el número de contactos en común entre el usuario objetivo y de los diferentes usuarios candidatos. Esto podría entenderse como que cada uno de los usuarios de la intersección tiene un peso equivalente al de todos los demás.

Adamic-Adar cambia este aspecto, ponderando en mayor medida a los usuarios de la intersección que tengan un menor número de vecinos. La intuición que lleva detrás esta modificación es que resulta más informativo relacionarse con usuarios que eligen de manera más selectiva a sus contactos. Para llevar a cabo esto se sigue la formula 3.5.

$$f_u(v) = \sum_{w \in (\Gamma(u) \cap \Gamma(v))} \frac{1}{log|\Gamma(w)|} \tag{3.5}$$

Resource Allocation

Resource Allocation [19] es una variante de Adamic-Adar. Viene inspirada por la idea de repartir recurso finito entre los contactos de u. Concretamente u dispone de una unidad del recurso, y la divide en partes iguales entre sus contactos y posteriormente, cada uno de ellos repite este proceso. Este proceso podría realizarse un número arbitrario de veces. La fórmula 3.6 tiene como objetivo conocer cuánta cantidad de recursos le ha sido adjudicada a vecinos a distancia 2 del usuario objetivo u y usuario candidato v.

$$f_u(v) = \frac{1}{|\Gamma(u)|} \sum_{w \in (\Gamma(u) \cap \Gamma(v))} \frac{1}{|\Gamma(w)|} \tag{3.6}$$

Si comparamos la fórmula de Adamic (3.5) con Resource Allocation (3.6) observamos como en la última se ha eliminado el logaritmo. Esta modificación conlleva un aumento de la penalización sobre los vecinos altamente conectados, dado que se pierde el suavizado del logaritmo.

3.2. Algoritmos de implementación externa

Los algoritmos tratados en la sección anterior no son los únicos con los que se han realizado experimentos. Durante esta sección se detallarán los que están basados en filtrado colaborativo, en concreto veremos tres ejemplos de algoritmos colaborativos basados en memoria: K vecinos próximos, tanto basado en usuario como basado en *item* y factorización de matrices.Todos ellos implementados en RankSys [1].

3.2.1. *K vecinos próximos*

Los algoritmos de filtrado colaborativo se caracterizan por observar a los usuarios en el sistema con el fin de realizar recomendaciones a usuarios similares. Como se explicó en el sección 2.2, mediante la definición de similitudes se asocian usuarios y se recomiendan las interacciones con los *items* que resultaron satisfactorias previamente en usuarios parecidos. En concreto K vecinos próximos (KNN) [5] es uno de los algoritmos de filtrado colaborativo más extendidos en la recomendación. Y dado que la recomendación de contactos es solo un caso particular de recomendación, podemos adaptarlo de manera sencilla como se explica a continuación.

[1] http://ranksys.org

KNN basado en usuario

KNN basado en usuario tiene la idea de que usuarios similares tienen gustos similares en relación a los *items* disponibles. Llevándonos esta intuición al ámbito de la recomendación de usuarios, obtenemos que usuarios parecidos tienden a relacionarse con la misma gente. La formula para KNN basado en usuario es la 3.7.

$$f_u(v) = \sum_{w \in \mathcal{N}(u)} sim(u, w) A_{wv} \tag{3.7}$$

donde $\mathcal{N}(u)$ representa ese conjunto de usuarios que son similares a u y A_{wv} representa el elemento en la fila w y la columna v de la matriz de adyacencia del grafo, cuyas entradas toman valores binarios, siendo 1 el caso en el que hay un enlace entre w y v, 0 en el caso contrario.

Además se debe definir la similitud entre usuarios. Una manera intuitiva y clásica en la literatura, es utilizar la similitud de coseno con los vecindarios de salida de los usuarios. Esta similitud es utilizada debido al fácil cómputo que conlleva dado que se compone de sumas y productos. Esta idea se plasma en la formula 3.8.

$$sim(u, v) = \frac{|\Gamma_{out}(u) \cap \Gamma_{out}(v)|}{\sqrt{|\Gamma_{out}(u)||\Gamma_{out}(v)|}} \tag{3.8}$$

KNN basado en *item*

KNN basado en *item* utiliza los *items* y las similitudes entre ellos para generar recomendaciones. Dado un *item* candidato, el sistema coge *items* similares con los que el usuario objetivo haya interactuado previamente. El resultado de la recomendación es una combinación de las similitudes de los *items* escogidos y la satisfacción que el usuario objetivo marco para dichos *item*. Dado que los *items* tratados son también usuarios obtenemos cierta simetría en la formula si la comparamos con 3.7 como se observa a continuación:

$$f_u(v) = \sum_{w \in \mathcal{N}(v)} sim(v, w) A_{uw} \tag{3.9}$$

Como puede observarse ahora se itera sobre los vecinos del usuario (tratado como item) v y se comprueba si el usuario objetivo u tiene contacto con los vecinos w del *item* v.

De nuevo la similitud más utilizada con KNN en nuestro contexto de recomendación de contactos es la similitud coseno, y dado que os *items* son usuarios la similitud se define de la misma manera que en la formula 3.8 con la salvedad que ahora debemos sustituir el vecindario de salida Γ_{out} con el

vecindario de entrada Γ_{in}. Este cambio es debido a que ahora se observan los usuarios que apuntan al *item* mientas que antes en 3.8 se quería observar los *items* a los que apuntan los usuarios.

$$sim(u,v) = \frac{|\Gamma_{in}(u) \cap \Gamma_{in}(v)|}{\sqrt{|\Gamma_{in}(u)||\Gamma_{in}(v)|}} \tag{3.10}$$

3.2.2. Factorización de matrices

La factorización de matrices es una de las técnicas con mayor presencia en el estado del arte [9] [11]. Simplificando enormemente, pretende obtener unas características implícitas que describen tanto a los usuarios como a los *items*, permitiendo así, representar a ambos conjuntos en un mismo espacio y comparar tanto usuarios como *items*. Concretamente este proceso se lleva a cabo mediante la creación de 2 matrices, una de usuarios y otra de *items*. En la matriz de usuarios contiene la representación de los mismos por columnas. Del mismo modo, en la matriz de *items* cada una de las columnas es la representación de un *item*. Gracias a esta codificación, podemos estimar el valor de la recomendación simplemente mediante la multiplicación de los vectores del usuario y del *item*. Esto se muestra con claridad en la siguiente fórmula:

$$f_u(i) = x_u^T y_i \tag{3.11}$$

donde x_u es la columna que especifica las características del usuario u y y_i es la columna que representa al *item* i.

De entre la gran variedad de algoritmos de factorización de matrices, el escogido se caracteriza como recomendación implícita, en la que no se dispone necesariamente de valoraciones de los usuarios [8]. Este obtiene la representación de los usuarios y los *items* optimizando la siguiente función:

$$\min_{x,y} \sum_{u,i} c_{u,i}(p_{ui} - x_u^T y_i)^2 + \sum_u \lambda ||x_u||^2 + \sum_i \lambda ||y_i||^2 \tag{3.12}$$

donde:

- El valor de preferencia: p_{ui} del usuario u por el *item* i. Es una variable binaria que toma el valor 1 si el usuario tiene una valoración r_{ui} positiva sobre el *item* y 0 en caso contrario.
- El nivel de confianza para p_{ui} : c_{ui} se calcula como $c_{ui} = 1 + \alpha * r_{ui}$ siendo α un parámetro libre positivo.
- λ es un parámetro libre positivo para evitar el sobre ajuste.

4

Evaluación y Métricas

La evaluación de los sistemas de recomendación es todavía objeto de estudio. Una primera distinción en los tipos de evaluación es la evaluación *online* u *offline*.Para poder realizar la evaluación *online* se debe disponer de un sistema en producción y realizar una división de los usuarios de la plataforma para que una sección prueba los cambios y novedades. A este proceso se le conoce como Test A-B. Este tipo de evaluación es útil, ya que conseguimos el *feedback* directamente de los usuarios. Sin embargo, es difícil poder realizar este tipo de recomendación, ya que se necesita un sistema en producción.

Dado que no se dispone de acceso a Twitter, lugar del que se han obtenido los datos utilizados durante los experimentos (que se detallan en el capítulo 5), la evaluación que realizaremos será *offline*. Este tipo de evaluación se basa principalmente en la técnica de particiones de entrenamiento y test. Esta técnica se basa en la división de los datos en 2 secciones. Para realizar esta división se pueden utilizar cantidad de criterios (aleatoriamente, por usuarios, por *items*, etc.) pero cuando los datos tienen carácter temporal (como es el caso), se recomienda hacer este tipo de división. Una vez segmentados los datos se ejecutan los algoritmos sobre el conjunto que se haya denotado como de entrenamiento y el resultado obtenido es evaluado contra el conjunto de test.

Además, el estudio puede afrontarse como un problema de clasificación binaria, en el que el contacto recomendado se considera un acierto cuando el usuario objetivo acepta la recomendación (y en caso contrario un fallo). Bajo este enfoque habitualmente se emplea metodología de predicción de enlaces y con sus correspondientes métodos de evaluación. En la primera sección de este capítulo (4.1) se explicará el funcionamiento de la métrica *area under curve* (AUC) clásica en este contexto.

En la sección 4.2, el problema de recomendación se resuelve mediante la utilización de *rankings*. Este enfoque se basa en que para cada usuario predeciremos una lista de enlaces con los que, en orden descendente, tendrá afinidad para conectar. De esta forma se obtiene una solución procedente del paradigma de la recuperación de la información (IR) en la que, a una consulta (en nuestro caso un usuario) le asignamos un conjunto de documentos relevantes en orden descendente de importancia (usuario con los que es posible que se establezca contacto). Junto a este enfoque se introduce la metodología de IR y con ella todo un nuevo conjunto de métricas. Los experimentos llevados a cabo en el capítulo 5 se evaluarán siguiendo la metodología IR, con especial atención a las métricas de Precisión,

EPC y la métrica propuesta por este TFG, la Precisión Recíproca. Posteriormente se realizará una comparación de los resultados obtenidos con las métrica de IR contrastando los resultados con los obtenidos mediante la utilización de la métrica de AUC.

4.1. Área bajo la curva

La métrica de área bajo la curva (AUC) recibe su nombre por cuantificar el área que queda por debajo de la curva ROC [6]. Esta curva es una manera gráfica ampliamente utilizada para representar el acierto en modelos de predicción dentro del paradigma del aprendizaje automático. Previa su definición, debemos plantear una serie de definiciones básicas necesarias al hablar de modelos de predicción.

Empecemos definiendo la matriz de confusión. Esta matriz se utiliza para representar los aciertos de un modelo de predicción. Si este modelo solo dispone de dos clases (supongamos que se llaman Positivo y Negativo) obtendríamos una matriz de confusión 2x2 como la mostrada a continuación.

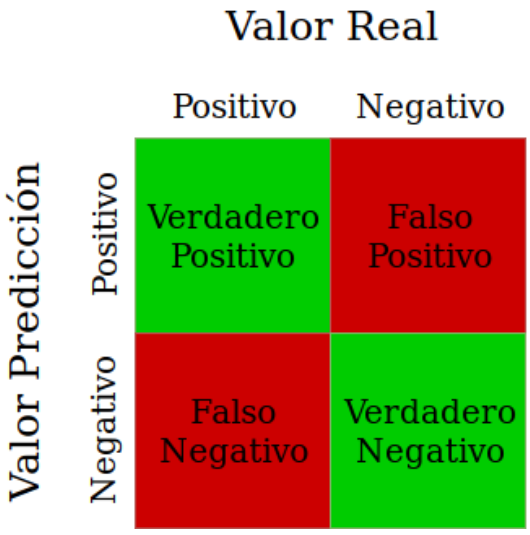


Figura 4.1: Matriz de confusión

Como se observa en la figura 4.1 obtenemos 4 divisiones en función del valor real del objeto y la predicción realizada. Cuando el objeto era de clase positivo y la predicción se realizó correctamente se llama Verdadero Positivo, del mismo modo, cuando la clase era negativo y se predice la misma clase se llama Verdadero Negativo. Y se producen fallos en la predicción cuando la clase la clase real y la predicción no coinciden. A estas situaciones las conocemos como Falso Positivo (se pronosticó Positivo y era Negativo) y Falso Negativo (el objeto era Positivo y la predicción erró). Es interesante diferenciar estas 4 posibilidades dado que las situaciones en las que nos sitúa cada una de ellas son muy distintas, un ejemplo de ello lo encontramos en este artículo de Google [1].

[1] https://developers.google.com/machine-learning/crash-course/classification/true-false-positive-negative?hl=es-419

Una vez aclarados estos conceptos podemos introducir a la curva ROC. En esta curva se representan 2 medidas relacionadas con la matriz de confusión. En primer lugar, la Tasa de Verdaderos Positivos (TVP), esta mide cuantos ejemplo se han catalogado correctamente como positivos entre todos los positivos que había realmente. De manera análoga, la Tasa de Falsos Positivos (TFP), mide de entre todos los casos que realmente eran Negativos, que cantidad se catalogo erróneamente (cuantos fueron Falsos Positivos).

Para construir la curva ROC debemos tener un ranking con todos los enlaces a clasificar y las puntuaciones proporcionadas por el algoritmo. A continuación recorremos en orden descendente el ranking, estableciendo umbrales. Entonces la curva ROC representa la tasa de verdaderos positivos y la tasa de falsos positivos para casa uno de esos umbrales.

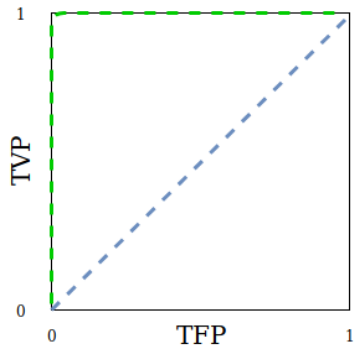


Figura 4.2: Curva ROC

En la figura 4.2 se observa el resultado de pintar 2 curvas ROC. La linea punteada verde es la curva ROC ideal, en la que todos los ejemplos se han clasificaron correctamente. La linea punteada azul se corresponde con un clasificador aleatorio con el que siempre se espera tener una TVP igual a la TFP.

A pesar de que las curvas ROC son un gran instrumento para un análisis visual, solo con ellas se vuelve complicado la comparación entre diferentes modelos de predicción. Por ello se diseño AUC. A través del calculo del área que deja por debajo esta curva conseguimos un valor numérico con el que realizar las comparaciones de los diferentes modelos. A mayor área, mejores resultados son alcanzados por el modelo.

La manera más sencilla para hallar el valor de AUC es mediante la integración de Riemann [2]. Por ello, sean x e y, los vectores con las coordenadas del eje de abscisas y ordenadas respectivamente AUC se calcula como se muestra en la fórmula 4.1.

$$AUC = \frac{1}{2}\sum_{i=1}^{N}[x_{i+1} - x_i][y_{i+1} + y_i] \tag{4.1}$$

[2] https://es.wikipedia.org/wiki/Integracion_de_Riemann

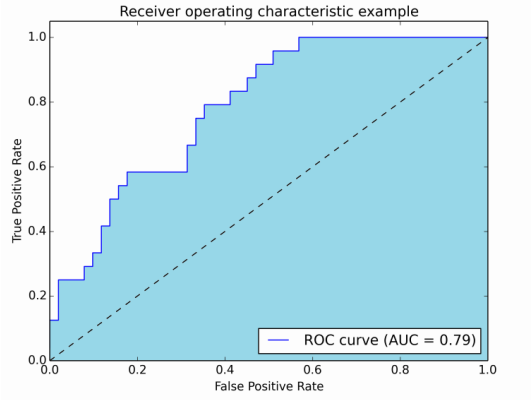


Figura 4.3: Ejemplo de AUC

4.2. Métricas de recomendación

En el área de la recuperación de información se han desarrollado multitud de métricas debido al auge de los buscadores en la época de los 90. En un primer momento, parecía suficiente con medir el acierto de los resultados obtenidos, y a día de hoy, esto sigue siendo básico y fundamental. Sin embargo, con el paso de los años se han empezado a tener en cuenta otros factores como la novedad y la diversidad. Todo el avance en investigación llevado a cabo en el campo de IR se ha aplicado en los sistemas de recomendación. Por ello, si afrontamos la evaluación de sistema de recomendación con la perspectiva de la recuperación de información buscamos la eficiencia no solo en términos de precisión sino también en cuestiones como la novedad. Por ello en esta sección se detallarán 3 métricas, la precisión (clásica pero indispensable), la precisión reciproca (como enfoque novedoso de precisión más fuerte) y *expected popularity complement* (EPC) como métrica para evaluar la novedad de los sistemas de recomendación.

4.2.1. Precisión

Llamamos precisión a la métrica que mide la media los aciertos que consigue el sistema al recomendar al conjunto de todos los usuarios. Para lograr esto se aplica la fórmula 4.2 a cada usuario por separado y posteriormente se realiza un promedio. Además esta métrica también puede interpretarse como la probabilidad media de que una recomendación resulte satisfactoria.

$$P = \frac{|Relevante \cap Devuelto|}{|Devuelto|} \quad \in [0, 1] \tag{4.2}$$

Es habitual que al usuario solo se le muestre una pequeña parte de los resultados, las k mejores recomendaciones (el *top* k del ranking). Esto deriva en una variación de la métrica de precisión, apodada “Precisión a corte k” y representada por P@k.

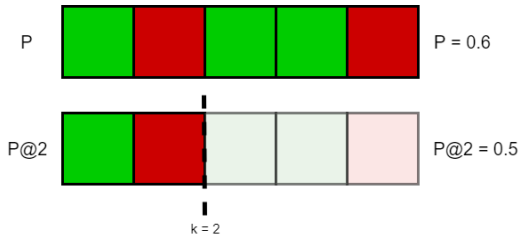


Figura 4.4: Precisión y Precisión @ k

Esta variación resulta interesante ya que en entornos reales de trabajo se realizan n recomendaciones a las que luego se le aplica un corte a las k mejores. En concreto, en los experimentos del capítulo 5 se calculan 100 recomendaciones para cada usuario y las evaluaciones se efectúan con k = 10.

El flujo seguido por el sistema de recomendación se muestra en la figura 4.5. En primer lugar se capturan todos los casos donde el usuario candidato v no tenga ningún contacto previo con el usuario objetivo u. Es en este caso cuando el sistema, siguiendo los algoritmos detallados en el capítulo 3 realiza las recomendaciones. Por último, evaluan esas recomendaciones: se considerará un acierto cuando la recomendación propuesta se encuentre en el conjunto de test, en caso contrario, la recomendación se considerará fallida.

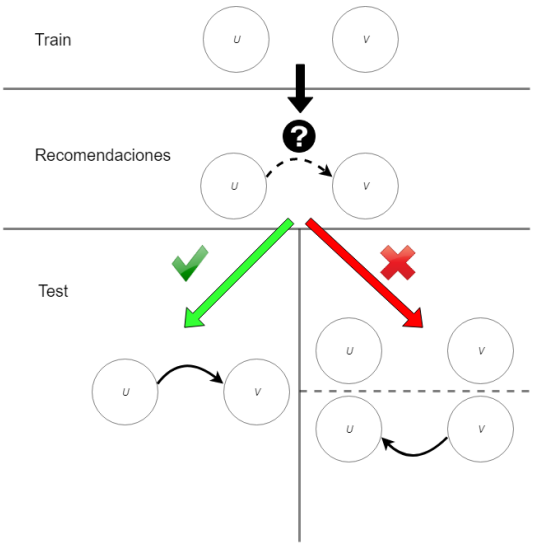


Figura 4.5: Proceso de recomendación en Precisión

4.2.2. Precisión Recíproca

La métrica de precisión recíproca viene motivada por la búsqueda de relaciones más fuertes en la redes sociales ya que en la actualidad muchas ocasiones se aceptan recomendaciones por no ofender a quien la propone pero no tenemos intención de relacionarnos con la persona. Existen estudios previos

relacionados con esta métrica, concretamente esta forma de medir el acierto tiene gran importancia en las redes de citas pero el problema que proponen no es idéntico ya que en su caso no se dispone de una estructura de grafo [13] [14].

Uno de los objetivos de este TFG es ver como afecta la exigencia de reciprocidad a la hora de realizar recomendaciones en el mundo de las redes sociales, con la intención de sentar las bases para trabajos futuros: estudiar como cambia la evolución de las redes sociales al imponer esta restricción, observar que cambios ocurren en las interacciones futuras de los usuarios al imponer esta restricción en el momento de establecer un primer contacto y analizar los cambios en la transmisión de información en contactos que disponen de este tipo de relación más fuerte.

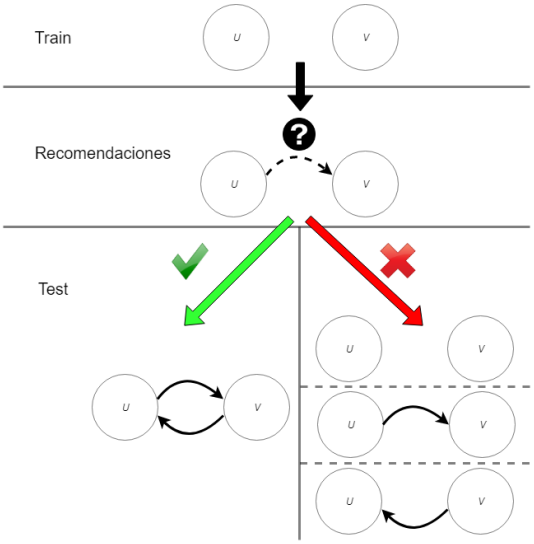


Figura 4.6: Proceso de recomendación en Precisión Recíproca

La precisión recíproca (RP) es un caso particular de la precisión estándar que solo cobra sentido cuando los *items* a recomendar son usuarios, de manera que estos puedan también recibir y aceptar recomendaciones [2, chap. 13.8.3]. En esta métrica se considera una recomendación como acierto cuando a un usuario u se le recomienda un candidato v, y posteriormente en el conjunto de test se observa un doble enlace entre ellos. Este proceso se observa de manera gráfica en la figura 4.6.Nótese que no es necesario que v reciba a u como recomendación pero podría darse el caso. Este matiz difiere de las redes de citas en las que previo al "match" al usuario u se le ha de recomendar v y a v se le ha de recomendar u. Este cambio se permite dado que pretendemos evaluar la frecuencia con la que se crean este tipo de enlaces con doble sentido en redes dirigidas.

4.2.3. Novedad

Una métrica comúnmente utilizada para estimar el grado de novedad que aporta una recomendación es la llamada *Expected Popularity Compliment* (EPC) [17]. El objetivo de esta métrica es estimar la

probabilidad media de que el usuario no conociese el *item* (en nuestro caso, otro usuario) que se le ha recomendado. Para ello, estimamos la probabilidad de que sí conozca al usuario que fue recomendado y transformamos la probabilidad en la del suceso complementario (que no se conocieran), de ahí el nombre. La formula para EPC es la media en todos los usuarios de la siguiente fórmula:

$$EPC(u) = \frac{1}{|\mathcal{R}|} \sum_{v \in \mathcal{R}} \mathcal{P}(conocido|v) = \frac{1}{|\mathcal{R}|} \sum_{v \in \mathcal{R}} 1 - \frac{|\Gamma_{in}(v)|}{|\mathcal{U}|} \tag{4.3}$$

donde $\mathcal{R}$ se corresponde con el ranking de recomendaciones proporcionado a u y $\mathcal{U}$ representa el conjunto de usuarios total.

Debido a como se ha construido esta métrica, EPC siempre toma valores pertenecientes al intervalo [0,1]. Por ello, EPC puede interpretarse como la cantidad porcentual de “poca popularidad” de los usuarios recomendados.

5 Experimentos

Este capitulo consta de 4 secciones. En la sección 5.1 se detallan se han realizado los experimentos. La segunda sección relata el estudio de parámetros llevado a cabo para los diferentes algoritmos (tanto basados en vecindarios como en filtrado colaborativo) bajo las métricas de P@10 y RP@10. A continuación se realiza una comparativa con los resultados en la sección anterior, y se introduce la cuestión de novedad, donde veremos que correlación existe entre la novedad de los recomendaciones y la utilización de los diferentes algoritmos. Además se estudia el desempeño obtenido al utilizar el enfoque del aprendizaje automático, mediante métricas clásicas como AUC. Por último, se sintetizan las conclusiones de los diferentes apartados en una conclusión final que cierra el capítulo de experimentos.

5.1. Configuración del experimento

Datos utilizados

Los experimentos llevados a cabo se han realizado con datos reales de la red social Twitter. Estos datos se obtuvieron mediante su API publica. Es un conjunto de datos de los muchos tratados en [16, chap. 7.1].

Para los experimentos estos datos se han dividido en dos conjuntos, conjunto de entrenamiento y conjunto de test. La división de estos datos se realiza siguiendo su orden natural de creación en la red social, dado que realizar un tipo de corte que no fuese temporal resultaría en la creación de situaciones ficticias que podrían simular escenarios inverosímiles. La separación temporal realizada se detalla en la tabla 5.1.

	No. usuarios	No. aristas
Train	9.964	475,530
Test	8,180	98,519
Total	10.000	582.172

Tabla 5.1: Detalles de la red generada por los datos

Metodología

Con los datos divididos, el proceso de estudio de parámetros se basa en proporcionarle a los algoritmos los datos del conjunto de entrenamiento. El resultado obtenido es comparado con los datos reales presentes en el conjunto de test. Se miden las métricas de precisión y precisión recíproca, ambas en su variante en corte 10.

Para el experimento en el que se utiliza predicción de enlaces se medirá que resultado obtienen las recomendaciones que realizan los algoritmos mediante la métrica de AUC.

Algoritmos

Los algoritmos que se utilizaran en los experimentos son todos los mencionados en el capítulo 3, tanto los basados en vecindarios (de implementación propia) como los de filtrado colaborativo (de librerías externas).

Para los algoritmos de implementación propia se comprobará su eficiencia al cambiar los vecindarios introducidos para cada uno de los usuarios relacionados (usuario objetivo u, usuario candidato v y si cuando sea necesario w, usuario vecino tanto de u como de v).

Por su parte, para los algoritmos de filtrado colaborativo también realizaremos un barrido de parámetros para encontrar las mejores configuraciones. Concretamente, para KNN, tanto basado en usuario como basado en *item*, recorreremos los diferentes valores de numero de vecinos k. Para la factorización de matrices, debido al amplio espacio de parámetros que dispone, se realizará primero el barrido en la dimensión del numero de características k y posteriormente se barrerán conjuntamente el parámetro de afinidad α y el parámetro de sobreajuste λ.

5.2. Estudio de parámetros

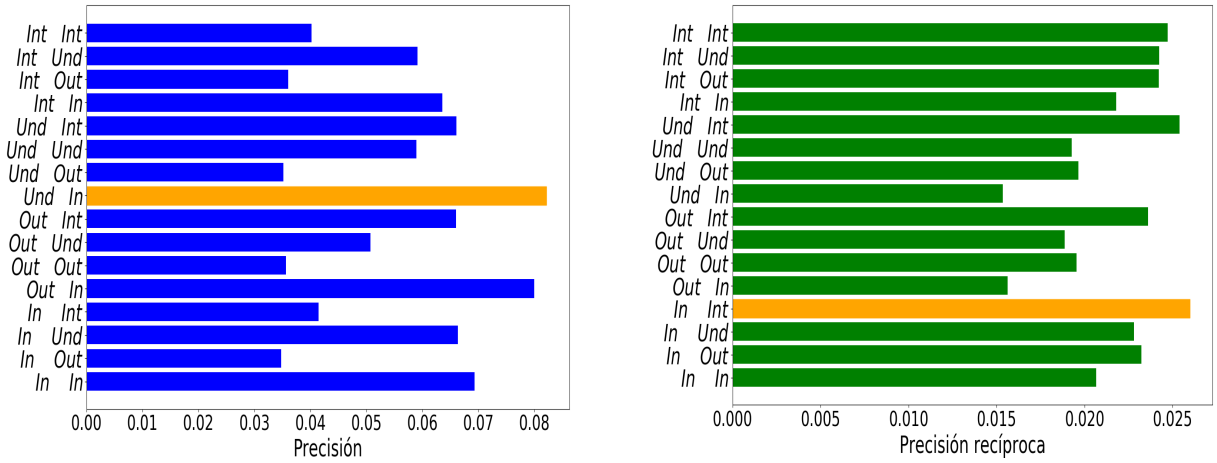
5.2.1. Algoritmos basados en vecindarios

En primer lugar queremos medir el comportamiento de las diferentes configuraciones de los algoritmos explicados en la sección 3.1.2 cuando se realiza una evaluación con precisión recíproca, concretamente se evaluará RP@10. Además, se realizará de manera simultanea una comparación de dichos resultados con los obtenidos al medir la precisión estándar, de nuevo P@10.

Concretamente veremos como funcionan los algoritmos al variar los vecindarios sobre los que realizan los cálculos. Como ya se detalló en la sección 3.1.2, podemos tratar con los vecindarios de entrada (Γ_{in}), de salida (Γ_{out}), en vecindario Γ_{int}, resultante de la intersección de los dos anteriores y por último el vecindario unión (Γ_{und}).

MCN

En la figura 5.1(a) se muestra la precisión obtenida al evaluar la recomendación de MCN sobre el fichero de test. Como puede observarse, el máximo se alcanza con Γ_{und} para el usuario objetivo y Γ_{in} para el usuario candidato con una precisión de 0.084.



(a) Precisión para MCN

(b) Precisión Recíproca para MCN

Figura 5.1: Resultados para la búsqueda en rejilla para Precisión y Precisión Reciproca

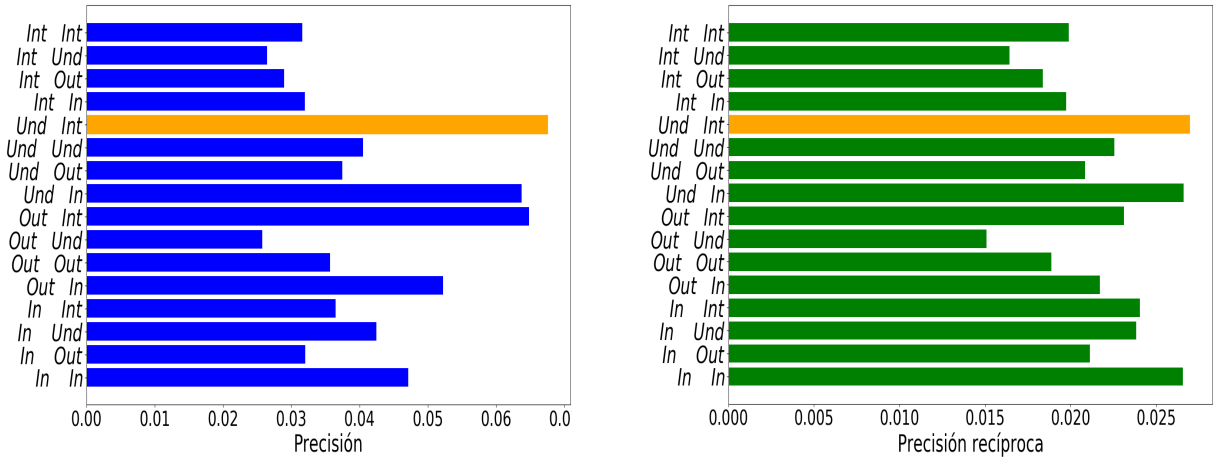
Del mismo modo la figura 5.1(b) se muestra la precisión recíproca obtenida al evaluar. En este caso, los vecindarios que maximizan esta métrica son Γ_{in} para u y Γ_{int} para v, obteniendo 0.026. Esto ya resulta interesante, dado que la intuición de que el vecindario Γ_{int} resultaría de interés al evaluar con RP se cumple ya en el algoritmo de vecindarios más simple. Además ocurre otro fenómeno que despierta curiosidad, se puede observar un comportamiento cíclico: para cualquier elección del vecindario de u, se alcanza una mayor precisión recíproca cuando utilizamos el vecindario Γ_{int} para v. Esto puede deberse a que RP busca la reciprocidad en la recomendación y el vecindario Γ_{int} captura la facilidad con la que el usuario candidato v tiene este tipo de relación.

A la vista de estos primeros resultados nos preguntamos si el vecindario Γ_{int} volverá a maximizar la precisión reciproca o ha sido un caso aislado.

Jaccard

En la figura 5.2(a) se muestra la precisión obtenida al evaluar la recomendación de Jaccard sobre el fichero de test. En esta ocasión, la configuración que alcanza la máxima precisión es Γ_{und} para u y Γ_{int} para v, con un resultado de 0.68.

Al comprobar el rendimiento de las diferentes configuraciones al medir RP@10, vemos en la figura 5.2(b) como se alcanza el máximo con Γ_{und} para u y Γ_{int} para v obteniendo 0.027.



(a) Precisión para Jaccard

(b) Precisión Recíproca para Jaccard

Figura 5.2: Resultados para la búsqueda en rejilla para Precisión y Precisión Reciproca

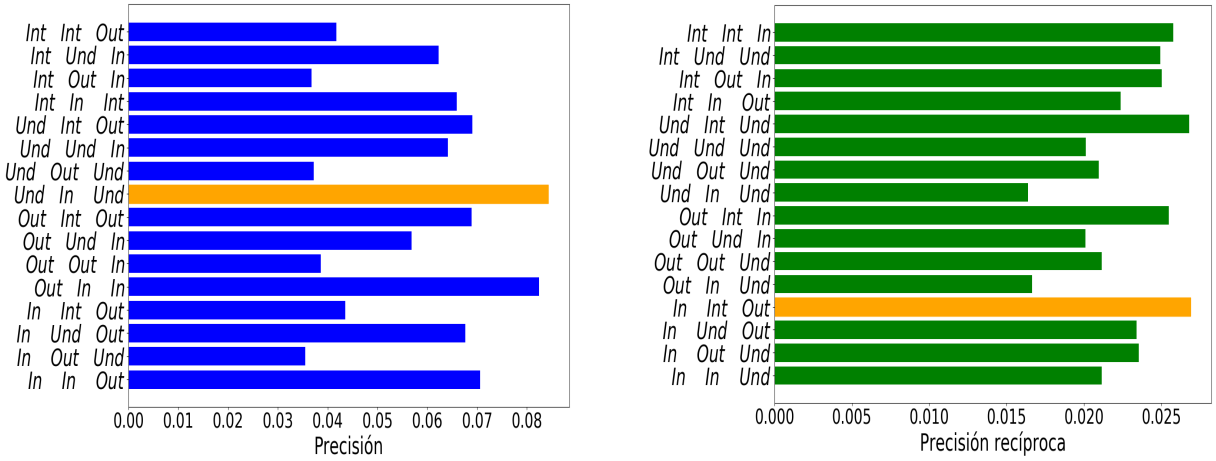
La figura 5.2 muestra por tanto dos cuestiones destacables, en primer lugar, RP@10 vuele a maximizarse con Γ_{int} como vecindario del usuario candidato, coincidiendo además con el mejor resultado para la precisión estándar. Segundo, a pesar de que las escalas de las figuras 5.2(a) y 5.2(b) son distintas, las gráficas muestran la misma estructura cualitativa, lo que da lugar a pensar en la existencia de proporcionalidad entre ambas métricas para el algoritmo de Jaccard. Este comportamiento de ambas métricas puede deberse a que el algoritmo de Jaccard penaliza a los usuarios más populares (al normalizar entre un gran tamaño de su vecindario) y la métrica de precisión recíproca busca una devolución del seguimiento, comportamiento que los usuarios populares no acostumbran a realizar.

Adamic-Adar

En el algoritmo de Adamic-Adar (ver sección 3.1.2) no solamente hay que tener en cuenta los vecindarios de los vecinos objetivo y candidato ($\Gamma(u)$ y $\Gamma(v)$), sino que, además, en el barrido de parámetros tenemos que tener en cuenta el vecindario de los vecinos en la intersección de ambos ($\Gamma(w)$). Por tanto ahora el barrido de parámetros se realiza a través de las 64 combinaciones posibles de los vecindarios. Sin embargo, se ha observado que con $\Gamma(u)$ y $\Gamma(v)$ fijos, variar el vecindario escogido para $\Gamma(w)$ varia escasamente los resultados obtenidos tanto en P@10 como RP@10. Por ello, los resultados observados en la figura 5.3 solo constan las configuraciones que para cada par $\Gamma(u)$ $\Gamma(v)$ fijos, $\Gamma(w)$ maximiza el valor de la métrica.

En la figura 5.3(a) destacan con claridad dos grupos de configuraciones: $\Gamma_{und}(u)$ $\Gamma_{in}(v)$ y en segundo lugar $\Gamma_{out}(u)$ $\Gamma_{in}(v)$, ambos grupos con independencia del valor del vecindario de w. Este comportamiento también se observa en los algoritmos anteriores, de lo que se puede deducir de manera empírica que se maximiza la métrica de precisión cuando modelizamos los gustos del usuario objetivo u como el total de sus contactos (Γ_{und}) o, en su defecto, como los usuarios a los que sigue (Γ_{out}), y

modelizamos al usuario candidato v como el conjunto de usuarios que siguen a v Γ_{in}. Esto resulta intuitivo ya que el usuario u tenderá a seguir a usuarios similares a los que ya sigue y modelando de esta manera al usuario v conseguimos este efecto.



(a) Precisión para Adamic-Adar

(b) Precisión Recíproca para Adamic-Adar

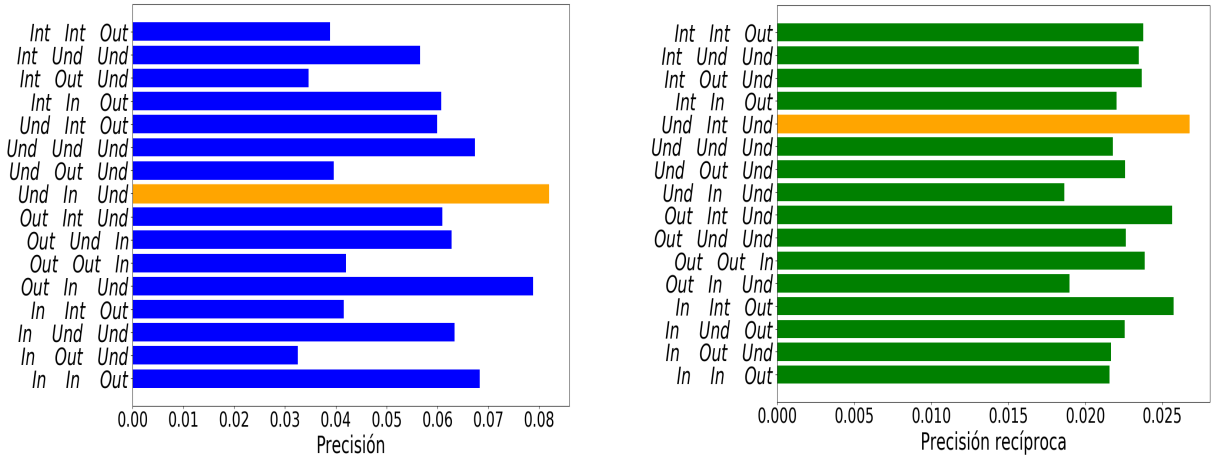
Figura 5.3: Resultados para la búsqueda en rejilla para Precisión y Precisión Reciproca

En la figura 5.3(b), se estudia el comportamiento de RP@10. En ella podemos encontrar dos grupos de configuraciones que consiguen un RP@10 mayor que los demás: $\Gamma_{in}(u)$ $\Gamma_{int}(v)$ y $\Gamma_{und}(v)$ $\Gamma_{int}(v)$, ambos grupos con independencia del valor del vecindario de w. Y, nuevamente, este comportamiento se ha repetido en algoritmos anteriores, por lo que podemos pensar que lo idóneo para conseguir reciprocidad en las recomendaciones es modelizar al usuario objetivo u como los usuarios que le siguen (Γ_{in}) o como la unión de todos sus contactos (Γ_{und}) mientras que al usuario candidato conviene modelizarle como su vecindario (Γ_{int}). La intuición detrás de esta idea reside en que si modelizamos a v como los usuarios con los que tiene una enlace recíproco estamos modelizando la predisposición de v a devolver el seguimiento propuesto por u.

Resource Allocation

En la figura 5.4 se muestra los resultados del barrido de parámetros para P@10 y para RP@10. De nuevo, en *Resource allocation* (ver sección 3.1.2) tenemos que barrer 64 parámetros, pero por hacer más fácil la lectura de la gráfica solo mostraremos las 16 configuraciones con las que el vecindario de w maximiza los pares $\Gamma(u)$ y $\Gamma(v)$.

Lo más interesante que se puede observar en la figura 5.4 es que los planteamientos propuestos por los resultados obtenidos a lo largo de los barridos anteriores se confirman nuevamente: para obtener una P@10 máxima debemos utilizar los vecindarios $\Gamma_{und}(u)$ y $\Gamma_{in}(v)$ sin tener un gran impacto que vecindario elijamos para el vecindario de w. Lo mismo ocurre en le caso de la precisión recíproca, donde volvemos a obtener el máximo con la configuración $\Gamma_{und}(u)$ $\Gamma_{int}(v)$ y en segundo lugar



(a) Precisión para *Resource Allocation*

(b) Precisión Recíproca para *Resource Allocation*

Figura 5.4: Resultados para la búsqueda en rejilla para Precisión y Precisión Reciproca

$\Gamma_{in}(u)\ \Gamma_{int}(v)$.

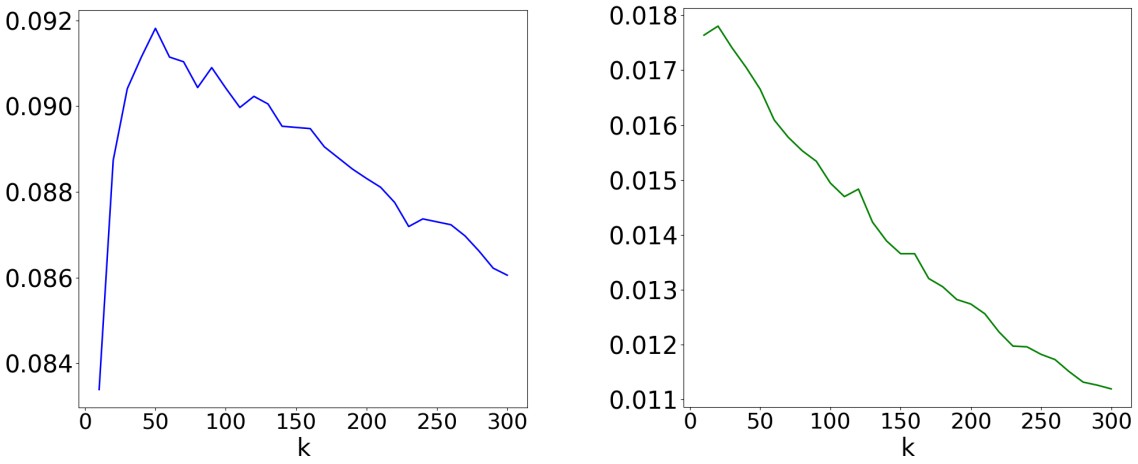
5.2.2. Algoritmos de filtrado colaborativo

En esta sección se realiza el barrido de parámetros correspondiente a los algoritmos de filtrado colaborativo, el fin de observar bajo que configuraciones observamos un mejor rendimiento al medir tanto precisión como precisión recíproca ambos con corte 10.

KNN

Para KNN solo es necesario iterar la medición en los valores del parámetro k, el número de vecinos a considerar. La figura 5.5 muestra los resultados de las métricas de precisión y precisión recíproca en el KNN basado en usuario. Los resultados muestran como P@10 alcanza su valor máximo con 50 vecinos mientras que RP@10 empieza a perder rendimiento a partir de los 20 (donde alcanza su máximo). Lo realmente llamativo de la gráfica 5.5 es la pendiente de las rectas a partir de sus máximos. Resulta interesante como aumentar vecinos supone mucha mas perdida para la precisión recíproca.

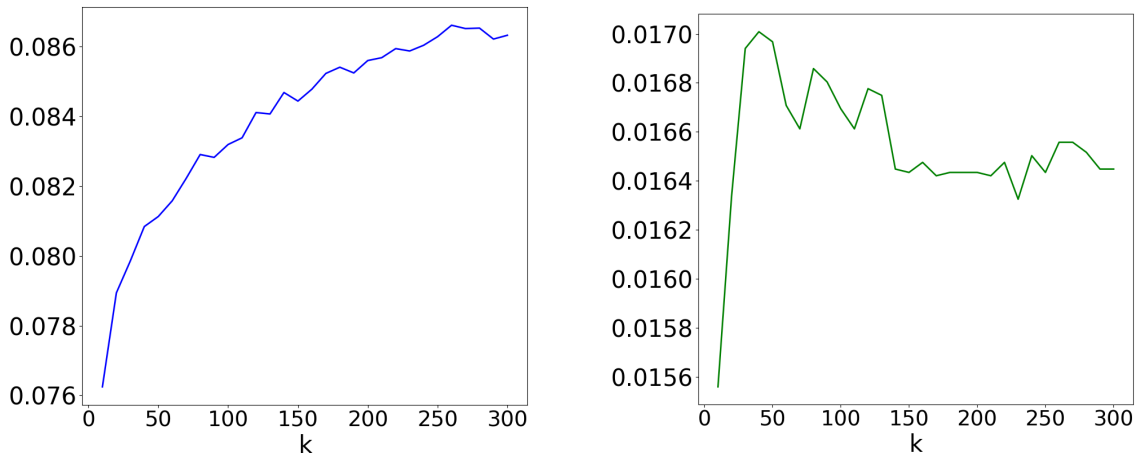
El caso de el KNN basado en *item* resulta llamativo. Como puede verse en la figura 5.6, la precisión aumenta a medida que aumentamos el número de *items* similares (alcanzando el máximo con k = 280) , sin embargo, con RP@10 observamos el comportamiento habitual al utilizar KNN, en el que si aumentamos excesivamente los *items* obtenemos una bajada en el acierto de la métrica.



(a) Precisión para KNN basado en usuario

(b) Precisión Reciproca para KNN basado en usuario

Figura 5.5: Resultados para la búsqueda en rejilla para Precisión y Precisión Reciproca



(a) Precisión para KNN basado en *item*

(b) Precisión Reciproca para KNN basado en *item*

Figura 5.6: Resultados para la búsqueda en rejilla para Precisión y Precisión Reciproca

Factorización de matrices

El barrido de parámetros de la factorización de matrices consta de tres variables: k, número de dimensiones del espacio de representación, α, factor de afinidad hacia los *items* y λ, factor para prevenir el sobreajuste.

Para la factorización de matrices solo se realiza el barrido de parámetros para la métrica RP@10 ya que existen estudios previos [16, chap. 7] con el mismo conjunto de datos que realizan el barrido de parámetros para P@10. Consideramos como posibles valores de k el intervalo [10, 300] y para el muestreo se realizarán incrementos de 10 unidades. Tanto α como λ consideramos que tienen valores viables que van desde 0.001 a 1000, su muestreo se realizará en en potencias de 10. Estos intervalos han sido los mismos que los utilizados en el estudio mencionado anteriormente.

Debido al amplio espacio de parámetros que tiene este algoritmo (y al gran coste computacional que conlleva), el barrido no se ha realizado en todas las posibles configuraciones. En primer lugar se ha variado el parámetro k, y se han dejado fijos $\alpha = 40$ y $\lambda = 150$,valores que maximizan la factorización de matrices en P@10 como se muestra en [16, chap. 7]. Tras realizar el barrido de esta primera dimensión, se ha obtenido una RP@10 máxima con k = 240. Posteriormente se ha realizado un barrido de parámetros simultaneo para α y λ. En la figura 5.7 se muestra el mapa de calor resultante. El valor máximo se alcanza con α = 10 y λ = 0.01. Por último se ha realizado una nueva iteración en k, esta vez con los nuevos valores de α y λ, con el objetivo de comprobar si volvemos a obtener el mismo valor de k. Esta iteración nos reafirma en que el el parámetro k óptimo es 240. Por consiguiente, la factorización de matrices se maximiza para la métrica de RP@10 con los valores k = 240 ,α = 10 y λ = 0.01.

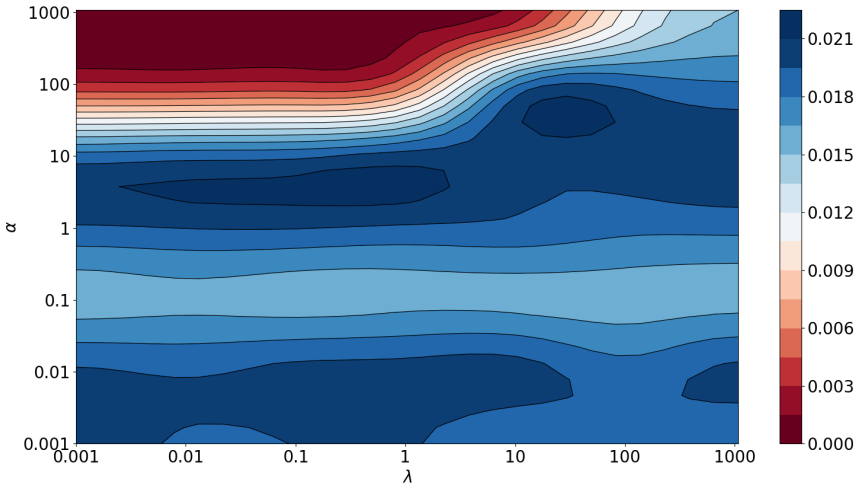


Figura 5.7: Resultados para la búsqueda en rejilla para Precisión Reciproca

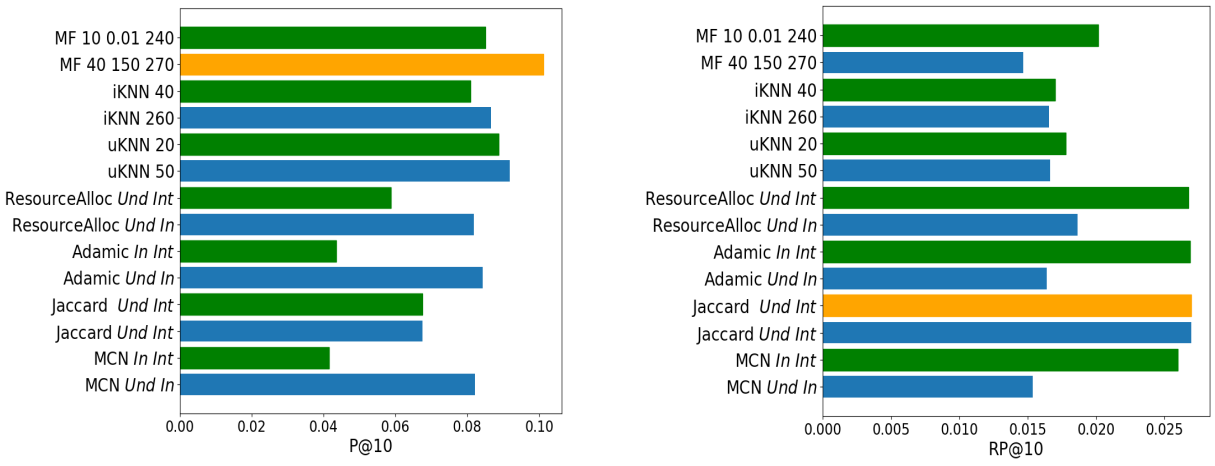
5.3. Comparativa de algoritmos

5.3.1. Precisión

Partiendo del barrido de parámetros de las secciones 5.2.1 y 5.2.2 ahora se realizará una comparativa de la eficacia obtenida por los diferentes algoritmos al evaluar las métricas de Precisión y de Precisión Reciproca para los casos donde se maximizan sendas métricas.

En la figura 5.8 se muestra en verde las configuraciones óptimas para los algoritmos en RP@10, en azul las configuraciones que maximizan P@10 y en amarillo el máximo para cada una de las métricas. La figura 5.8(a) muestra los resultados de medir con P@10 cada una de las configuraciones señaladas. Como puede verse, en todos los casos la precisión obtenida por la configuración que maximiza RP consigue peores resultados (salvo en Jaccard, donde es la misma configuración). La factorización de matrices es el algoritmo que mejor resultado consigue en P@10. De hecho, se nota una clara diferencia en el resultado obtenido aún comparándolo con el Adamic, algoritmo de vecindarios que mayor P@10 alcanza. Esta clara diferencia muestra por qué, a pesar de ser un algoritmo costoso computacionalmente, es una de las técnicas más utilizadas actualmente.

Por contra, en la figura 5.8(b) podemos observar como los algoritmos que utilizan vecindarios para operar consiguen mejores resultados para RP@10, que los de filtrado colaborativo. Concretamente Jaccard, que es el que peor P@10 consigue aplicando la configuración máxima, es quien obtiene el mejor resultado en RP@10 (aunque la diferencia con Resource Allocation o con Adamic-Adar resulta despreciable).



(a) Precisión para las configuraciones de los algoritmos

(b) Precisión Recíproca para para las configuraciones de los algoritmos

Figura 5.8: Comparativa de precisión y precisión recíproca a corte 10

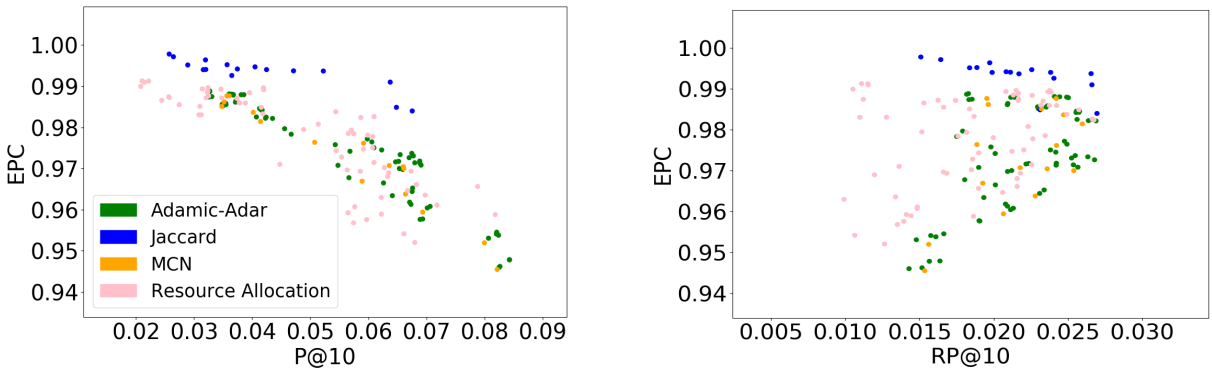
Un detalle no resaltado hasta ahora ha sido las dimensiones de los ejes. En P@10 obtenemos

resultados que en el mejor de los casos alcanzan el 10 % de acierto. Para RP@10, dado su naturaleza más restrictiva obtenemos resultados considerablemente menores, alcanzando en el mejor de los casos resultados cercanos al 2.5 %.

5.3.2. Novedad

Observando los resultados de la sección anterior, llama la atención que el algoritmo Jaccard, uno de los que peor funciona en P@10 sea el mejor cuando queremos medir RP@10. Con el objetivo de averiguar a qué se debe esto, puesto que Jaccard es el algoritmo de los estudiados que, en principio, penaliza más la popularidad, buscamos ver si, efectivamente, existe una correlación entre dicha penalización y RP@10. Para ello utilizaremos la métrica EPC (ver sección 4.2.3), que recordemos, entiende entiende la novedad como falta de popularidad.

Este experimento pretende comprobar si existe correlación entre los algoritmos detallados en el capítulo 3 con la métrica EPC. El análisis se lleva a cabo mediante la realización de 2 *scatter plots* comparando P@10 y RP@10 con EPC. Estos se muestran el la figura 5.9.



(a) Correlación P@10 y EPC

(b) Correlación RP@10 y EPC

Figura 5.9: Correlación de las métricas de precisión con la métrica EPC

En la figura 5.9(a) se muestra la correlación de la P@10 con la métrica EPC. Como puede observarse, en todos los algoritmos basados en vecindarios P@10 tienen una correlación negativa con EPC, es decir, para maximizar la precisión de las recomendaciones el sistema se ve obligado a recomendarnos usuarios más conocemos mientras que, con el fin de obtener una mayor novedad es necesario sacrificar parte del acierto del sistema de recomendación. En cambio, la 5.9(b) nos muestra una ligera correlación positiva entre RP@10 y EPC, este resultado induce al hecho de que la reciprocidad en los enlaces se da con gente con pocos seguidores.

5.3.3. Predicción de enlaces

Este experimento recurriremos la metodología de predicción de enlaces (ver sección 2.5), por ello diferenciaremos dos tipos de enlaces, “existe” o “no existe”. El proceso de este experimento consiste en la clasificación de todos los enlaces posibles en red social que no se encuentren presentes en el conjunto de entrenamiento. Dado que el conjunto de datos consta de 10.000 usuarios, esto da un total de 100 millones de enlaces a clasificar, para cada configuración de los algoritmos del capítulo 3. Por ello, en vez de realizar un barrido de parámetros como en los experimentos anteriores, se ha optado por ver los resultados que proporciona AUC para cada una de las configuraciones que maximizan P@10.

	AUC	P@10
Fact. Matrices	0.692	0.101
KNN usuario	0.682	0.091
KNN *item*	0.678	0.086
Adamic-Adar	0.596	0.084
Res. Allocation	0.581	0.082
MCN	0.536	0.081
Jaccard	0.532	0.067

Tabla 5.2: Resultados de AUC

La tabla 5.2 muestra los resultados de calcular AUC como se explica en la sección 4.1. Como puede verse, utilizando el enfoque de predicción de enlaces los algoritmos de filtrado colaborativo consiguen un mejor resultado que los algoritmos basados en vecindarios. Concretamente la factorización de matrices alcanza 0.692, casi un 0.1 más que Adamic-Adar que, siendo el mejor de los algoritmos de vecindarios se queda en e 0.596. En la tabla se pueden observar que los algoritmos obtienen mejores resultados en AUC en el mismo orden que los obtienen para P@10.

5.4. Conclusiones de los experimentos

En el primer experimento se realiza un estudio de parámetros para los algoritmos detallados durante el capítulo 3 al medir las métricas de precisión (P@10) y precisión recíproca (RP@10) (ver sección 4.2). Este análisis resulta de interés debido al parecido de ambas métricas. Los resultados indican que pedir reciprocidad cambia en gran medida la solución que optimiza P@10.

Para los algoritmos basados en vecindarios P@10 se optimiza (en la mayoría de los casos) utilizando el vecindario de unión ($\Gamma_{und}(u)$) para u y el vecindario de entrada ($\Gamma_{in}(v)$) para v. Esto intuitivamente significa que los gustos de u se pueden ver reflejados en todos los usuarios con los que ya está relacionado y que las características de v las vemos representadas como los usuarios que le siguen.

Cuando ambos conjuntos tienen una gran intersección podemos afirmar que existe una afinidad entre ellos y por eso el algoritmo recomendará ese nuevo enlace. Pero la precisión recíproca exige que v devuelva el seguimiento a u, esto implica la necesidad de un cambio en cómo se modela a los usuarios que están involucrados en la recomendación. Ahora los gustos de u se modelizarán como el conjunto de vecinos que siguen a u, es decir $\Gamma_{in}(u)$ mientras que para v se utilizará el vecindario intersección $\Gamma_{int}(v)$. El cambio de vecindario para u es el más significativo. La intuición tras este cambio es modelar las características que vuelven a u un usuario atractivo para los demás, de manera que v quiera devolverle el seguimiento. Y la utilización de $\Gamma_{int}(v)$ se puede entender como la predisposición que tiene v de devolver el seguimiento a otros usuarios.

Durante el segundo experimento, la comparativa de algoritmos, se observaron 2 resultados relevantes. En primer lugar, al medir RP@10, los algoritmos de filtrado colaborativo (KNN y factorización de matrices) obtienen unos resultados un 20 % peores que los algoritmos basados en vecindarios, mientras que en P@10 pasa justo lo contrario. La segunda cuestión llamativa es que el algoritmo que maximiza la precisión recíproca es Jaccard, que para la precisión estándar obtiene los peores resultados.

Para entender mejor este fenómeno comparamos los resultados de ambas métricas de precisión con la métrica de novedad EPC. Los resultados indican que la tendencia de Jaccard a penalizar a los usuarios populares favorecen los enlaces recíprocos. Esto puede deberse a que los usuarios populares no devolver el contacto.

Por último, se comparó la eficiencia de los algoritmos bajo la métrica AUC, clásica en el aprendizaje automático. Los resultados muestran que la utilización de estos algoritmos bajo en enfoque de predicción no genera grandes resultados (obteniendo en el mejor de los casos un AUC del 70 %). Sin embargo, merece la pena resaltar que se mantiene el orden de los algoritmos a la hora de maximizar AUC y P@10.

6

Conclusiones

Este TFG se ha enfocado al análisis de las redes sociales desde el punto de vista de la recomendación de contactos y el paradigma de la predicción de enlaces. Haciendo además hincapié en el análisis del acierto recíproco con el fin de encontrar enlaces más significativos para los usuarios y para el fomento de la actividad en una red social. Para llevar esto a cabo, se ha implementado un módulo de recomendación de contactos y una colección de algoritmos basados en vecindarios. Estos algoritmos, junto a otros de filtrado colaborativo de implementación externa, nos han permitido realizar un profundo análisis de datos reales de la red social Twitter cuyas conclusiones quedan reflejadas a continuación.

6.1. Resumen y contribuciones

En primer lugar se han comparado 2 escenarios distintos de evaluación, uno en el que consideramos acierto si el enlace aparece en la red y otro en el que consideramos acierto si tanto el enlace recomendado como su reciproco aparecen. Este segundo escenario surge de la intuición de que si un enlace es reciproco, resulta en un enlace más significativo para los usuarios de la red social. Para realizar este análisis se ha utilizado la métrica de precisión en el primer caso y una variación más fuerte de precisión en el segundo, apodada precisión recíproca.

Los resultados obtenidos muestran que las soluciones para maximizar cada tipo de acierto varían significativamente. Para los algoritmos basados en vecindarios, los gustos del usuario candidato pasan de modelarse como todos sus contactos a solo aquellos que le siguen a él. Esto intuitivamente se debe a que así conseguimos modelizar que vuelve atractivo al usuario objetivo y de esta forma maximizamos la probabilidad de reciprocidad.

Para los algoritmos de filtrado colaborativo, los resultados muestran como estos métodos son perjudicados al requerir reciprocidad. Mientras que estos obtienen los mejores resultados en precisión estándar, en los resultados en precisión recíproca quedan detrás de los algoritmos basados en vecindarios. Esto abre una nueva vía de investigación: cómo adaptar estos algoritmos para que no resulten perjudicados por este nuevo requisito a la hora de evaluar.

En la comparativa de los métodos de predicción con los mecanismos de recomendación, desta-

camos que utilizando métricas como AUC los resultados indican los métodos de predicción para algoritmos basados en vecindarios, si bien aciertan, no proporcionan grandes resultados, consiguiendo resultados similares a los de un clasificador aleatorio. Los algoritmos de filtrado colaborativo muestran una mejora sustancial, obteniendo en el mejor de los casos un 20 % más de acierto, con la factorización de matrices. Lo más destacable es que el orden al medir AUC se corresponde con el de la métrica de precisión cuando utilizamos procedimientos de recomendación.

6.2. Trabajo futuro

Los experimentos realizados durante este trabajo y las conclusiones obtenidas plantean nuevas cuestiones y sientan la base para futuros estudios, pero además el trabajo realizado durante este TFG puede pulirse y perfeccionarse.

Concretamente, existen una gran cantidad de algoritmos para recomendación de contactos no utilizados durante este TFG. Existe un interés en implementarlos para la librería existente y ampliar este estudio con ellos. Además pueden investigarse nuevos algoritmos (o variantes de algoritmos ya existentes) cuyo objetivo sea maximizar la métrica de precisión recíproca, ya que los enlaces que estos algoritmos recomendarían intuitivamente resultan más satisfactorios para los usuarios.

Desde el punto de vista teórico este TFG abre un amplio campo de investigación. Se plantean cuestiones relacionadas con la evolución de una red social al pedir reciprocidad (u otras nuevas restricciones) en los enlaces, o como varia la transmisión de la información (o el contagio de opiniones) cuando existen estas relaciones entre usuarios y cuando no.

Bibliografía

[1] Lada A Adamic and Eytan Adar. Friends and neighbors on the web. *Social networks*, 25(3), 2003.

[2] Charu C. Aggarwal. *Recommender Systems: The Textbook*. Springer International Publishing, 2016.

[3] Robin Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4), 2002.

[4] Jilin Chen, Werner Geyer, Casey Dugan, Michael Muller, and Ido Guy. Make new friends, but keep the old: recommending people on social networking sites. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2009.

[5] Christian Desrosiers and George Karypis. A comprehensive survey of neighborhood-based recommendation methods. Springer, 2011.

[6] Tom Fawcett. An introduction to ROC analysis. *Pattern recognition letters*, 27(8), 2006.

[7] Mohammad Al Hasan, Vineet Chaoji, Saeed Salem, and Mohammed Zaki. Link prediction using supervised learning. In *In Proc. of SDM 06 workshop on Link Analysis, Counterterrorism and Security*, 2006.

[8] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, Washington, DC, USA, 2008. IEEE Computer Society.

[9] Mohsen Jamali and Martin Ester. A matrix factorization technique with trust propagation for recommendation in social networks. ACM, 2010.

[10] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7), 2007.

[11] Aditya Krishna Menon and Charles Elkan. Link prediction via matrix factorization. Springer, 2011.

[12] Mark EJ Newman. Clustering and preferential attachment in growing networks. *Physical review E*, 64(2), 2001.

[13] Luiz Pizzato, Tomasz Rej, Joshua Akehurst, Irena Koprinska, Kalina Yacef, and Judy Kay. Recommending people to people: the nature of reciprocal recommenders with a case study in online dating. *User Modeling and User-Adapted Interaction*, 23(5), 2013.

[14] Luiz Pizzato, Tomek Rej, Thomas Chung, Irena Koprinska, and Judy Kay. Recon: A reciprocal recommender for online dating. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, New York, NY, USA, 2010. ACM.

[15] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, CSCW '94, New York, NY, USA, 1994. ACM.

[16] Javier Sanz-Cruzado and Pablo Castells. Contact recommendations in social networks. In *Collaborative Recommendations: Algorithms, Practical Challenges and Applications*. World Scientific Publishing, November 2018.

[17] Saúl Vargas and Pablo Castells. Rank and relevance in novelty and diversity metrics for recommender systems. In *Proceedings of the fifth ACM conference on Recommender systems*. ACM, 2011.

[18] Zi-Ke Zhang, Chuang Liu, Yi-Cheng Zhang, and Tao Zhou. Solving the cold-start problem in recommender systems with social tags. *EPL (Europhysics Letters)*, 92(2), 2010.

[19] Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang. Predicting missing links via local information. *The European Physical Journal B*, 71(4), 2009.