Universidad de Concepción

Facultad de Ciencias Físicas y Matemáticas

Departamento de Ingeniería Matemática

Estudio de algoritmos de recomendación no supervisados en redes sociales utilizando conceptos de teoría de grafos.

Tesis presentada para obtener el título de Ingeniera Civil Matemática

Autora:

Katherinne Tabilo Navarro

Profesor Guía:

Dr. Christopher Thraves Caro

Concepción, Chile
2021

CON AMOR PARA MIS PADRES, MARÍA Y JOSÉ

Una etapa que significa más que conocimientos y competencias

Al concluir estos años de esfuerzo, desvelo y aprendizaje, me tomo un pequeño tiempo para reconocer a quienes han formado parte de este proyecto tan hermoso. Me gustaría agradecer en primer lugar al Dr. Christopher Thraves, por ser responsable directo de este documento que permite finalizar esta etapa. Gracias por su apoyo, paciencia y comprensión en todo este periodo de trabajo en conjunto, por sus ideas y explicaciones que permitieron llegar a buen puerto y por sobre todo por ser un profesor tan humano que entiende las situaciones que uno vive personalmente y dispuesto a escuchar a sus estudiantes. También agradezco a cada uno de los docentes del departamento que en estos años abrieron las puertas de sus oficinas para atender nuestras dudas, quienes a través de su vocación de docentes nos permiten formar nuestro camino profesional, en especial me gustaría agradecer a los profesores Dr. Rodolfo Rodriguez y Dr. Manuel Solano, porque en tiempos de desesperación universitaria ayudaron a esta estudiante llena de dudas. Una mención especial al profesor Dr. Julio Aracena, a quien admiro mucho, y que a pesar de sufrir con cada trabajo y prueba impuesta por él, me ha llevado a pensar que hay que dar el máximo y más, exigirnos para superar las expectativas y lograr la excelencia. No quiero dejar de mencionar a la profesora Dra. Mónica Selva por su gestión para con los estudiantes de nuestra carrera y de agradecer al personal de la facultad que cada día aporta a resolver nuestros trámites y dificultades, además de compartir experiencias y consejos con nosotros.

Queridos mamá y papá, ustedes son y han sido mi mayor apoyo en este camino, gracias por estar a mi lado incondicionalmente, por creer en mí más que yo misma, sin ustedes no estaría donde estoy hoy. Gracias Beatriz y Marcos por ser unos segundos padres, por las oportunidades que me han brindado, la confianza y largas conversaciones en el trabajo, por estar presentes en tantas oportunidades que lo he necesitado. Mi Franquito Aron, no alcanzan las líneas para expresar lo significativo de tu compañía, es invaluable contar con alguien como tú.

Este paso por la universidad no solo ha dejado conocimientos, sino que experiencias y personas que han enriquecido mi vida, sería injunto enumerar olvidando alguno de ellos, pero sepan que ICM está formada de estudiantes esforzados, brillantes, con alegrías, sueños, creatividad y convicciones que compartimos día a día, y de los cuales me siento enormemente bendecida de conocer y haber compartido aunque fuese un trabajo, una charla o cursos completos con ustedes. A mis queridos *electrónicos*, sepan que son los mejores amigos que encontré en esta universidad, Pablito Torres eres el mejor

consejero, Kata amiga de mi corazón eres el mejor reencuentro que tendré, cada uno de ustedes tiene un espacio de mi corazón. La universidad no hubiese sido lo mismo sin los amigos y compañeros con los que compartimos horas de estudio, distracción, angustias, penas y alegrías.

¡Muchas gracias a todos!

Índice general

Índice de figuras

Capítulo 1

Introducción

Los inicios del internet se remontan a fines de los años 60, con la creación de ARPANET, una red de computadoras creadas para el Departamento de Defensa de Estados Unidos; cuyo uso era la comunicación entre las diferentes instituciones académicas y estatales. Más tarde, con la creación de la *Worl Wide Web* (dominio WWW), el uso del internet se masificó, hasta las instancias en las que nos encontramos hoy en día, estimándose la cantidad de usuarios de internet en 4.66 billones (alrededor del 60 % de la población mundial), donde 4.2 billones son usuarios de redes sociales [3]. Es imaginable que con la gran cantidad de usuarios en línea existan grandes volumenes de información (datos, publicidad, perfiles de usuarios en las redes sociales, videos, etcétera), sin ir más lejos, a Youtube, la plataforma más popular de videos, se cargan más de 48hrs de videos por minuto y reciben más de mil millones de reproducciones diarias [28]. Es a raíz de estos grandes volúmenes de datos donde surge la necesidad de optimizar los tiempos de búsqueda en la web, para así mejorar la experiencia de los usuarios.

Los *algoritmos de recomendación* son definidos por Ruchita V. Tatiya y Archana S. Vaidya en [26] como una subclase de sistemas de filtrado de información que intentan entregar al usuario una guía sobre servicios útiles en función de sus preferencias, comportamiento pasado o similitud de gustos con otros usuarios; como servicios útiles se puede entender: películas, música, libros, artículos en general, perfiles de usuarios, páginas web, etc. El algoritmo de recomendación busca predecir el interés que despierta en el usuario un ítem que este no ha considerado antes, dichas predicciones pueden estar basadas en distintos parámetros, ya sea, del usuario, del ítem o de ambos.

En este proyecto se busca desarrollar algoritmos de recomendación que sean

aplicables en redes sociales en donde a cada usuario se les recomienden otros usuarios con los que compartan intereses. Se buscará introducir nuevos *criterios* o métodos para generar recomendaciones. En particular, se buscará diseñar algoritmos basados en la idea de que los usuarios son parte de un *espacio métrico finito*, definiendo así distancias entre ellos y generando recomendaciones a partir de esas distancias.
Esta memoria tendrá un carácter experimental, por lo que parte del trabajo será recolectar datos reales a los cuales se les aplicarán los algoritmos diseñados, y sobre los cuales se evaluará la calidad de los algoritmos.
Cabe destacar que la interacción o relación de *amistad* en redes sociales la representamos a través de grafos. También evaluaremos la calidad de los algoritmos aplicando criterios basados en la teoría de grafos, sin embargo, recalcamos que se evalúa la *calidad potencial* ya que si se quisiera evaluar la calidad real de los algoritmos, cada recomendación realizada a un usuario debiese ser evaluada por el mismo usuario.
Este trabajo se divide en 4 capítulos en los que se abordan desde los conceptos matemáticos que nos permitirán modelar y evaluar nuestro problema y algoritmos (en particular los conceptos de la teoría de grafos), hasta la presentación de los resultados obtenidos al aplicar los algoritmos diseñados en una base de datos de una red social. En el Capítulo 2 presentamos las notaciones y definiciones que nos permiten contextualizar nuestro trabajo y explicar las herramientas utilizadas para evaluar la calidad potencial de los algoritmos. En las secciones 2.2 y 2.3 introducimos las *matrices de Robinson* y la *conductancia*, respectivamente, conceptos que aportan novedad al trabajo realizado para la elaboración de algoritmos y evaluarlos. En la Sección 2.4, presentamos nuestro problema de forma matemática y asociamos conceptos matemáticos con el problema real de recomendación en redes sociales. Luego, en el Capítulo 3 revisamos el *estado del arte* de cuatro tópicos que consideramos esenciales en este trabajo: matrices de Robinson, conductancia, algoritmos de recomendación y *k-means*; abordamos trabajos de académicos donde descubrimos interesantes aplicaciones sobre los tópicos mencionados que nos han llevado a integrarlos en nuestro trabajo, por ejemplo, el uso de la conductancia matemática para evaluar clusters. En los capítulos 4 y 5 presentamos lo realizado por nosotros, partiendo por los algoritmos diseñados y posteriormente los resultados obtenidos al aplicarlos sobre una base de datos de la red social *Twitter*. Dentro de los algoritmos diseñados y evaluados veremos primero los algoritmos inspirados por la cantidad de *amigos en común* en redes sociales entre dos usuarios y la proporción que estos representan de la lista actual de amigos de cada usuario. En segundo lugar, desarrollamos algoritmos basados en el *problema del ordenamiento*

y matrices de Robinson, conceptos que desarrollaremos a lo largo del presente trabajo. Por último, planteamos un método de recomendación para redes sociales usando el algoritmo *k-means* para generar vecindades. Posteriormente se evalúan estas recomendaciones y se comparan entre sí. Para finalizar, presentamos las conclusiones respecto a los resultados y trabajo en general en el Capítulo 6, en particular se explicitan conclusiones respecto a los resultados de: promedios de corte normalizado, promedios de conductancia, promedios de la expansión de la recomendación, entre otras; las cuales nos llevan a concluir que los algoritmos propuestos presentan métricas similares a nuestro referente *k-means*, además, presentan ventajas cualitativas como el poder personalizar la cantidad de perfiles en la recomendación que se quieren obtener para un usuario y generar recomendaciones para todos los usuarios de la red, lo cual no siempre se obtiene con *k-means*.

Capítulo 2

Notaciones y Definiciones

En este capítulo presentamos las notaciones usadas a lo largo del documento y las definiciones necesarias para entender el mismo. Primero, repasamos los conceptos básicos de la teoría de grafos que nos permitirán modelar nuestro problema de la vida real y cotidiana de forma matemática, además de poder representar esto mismo de una forma amigable para nuestras herramientas computacionales. Luego, definimos las matrices de Robinson, las que nos ayudarán a obtener un orden de los usuarios en la red social y que en el transcurso del trabajo entenderemos el por qué de esto. Después se definen otros conceptos de la teoría de grafos, los cortes y la conductancia, ellos nos permitirán evaluar las recomendaciones que buscamos generar. Por último, en las secciones 2.4 y 2.5, escribimos matemáticamente el problema de generar una recomendación a un usuario de una red social y las métricas para evaluarlo. Además, explicamos cómo los conceptos vistos en las secciones 2.1, 2.2 y 2.3 están relacionados con el modelamiento del problema.

2.1. Grafos, digrafos y matrices asociadas

Denotamos un grafo (no dirigido) a través de $G = (V, E)$, donde V es un conjunto de vértices y E es un conjunto de pares no ordenados de elementos distintos de V. Los elementos de E se llaman aristas y las denotaremos por uv, donde u y v son elementos de V. En ocasiones usaremos $e = uv$ para llamar e a la arista uv. Los vértices u, v se dicen extremos de la arista $e = uv$ y e se dice incidente a los vértices u y v.
Dado un grafo G, se denota $V(G)$ y $E(G)$ como los conjuntos de vértices y aristas de G, respectivamente. El orden y tamaño de un grafo G está dado por $|V(G)|$ y $|E(G)|$, respectivamente.

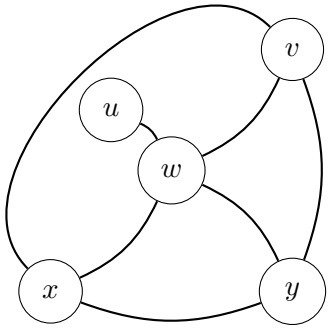


Figura 2.1: Un dibujo de $G = (V, E)$, con $V(G) = \{u, v, w, x, y\}$ y $E(G) = \{uw, vw, xw, vy, vx, wy, xy\}$.

Un grafo $G = (V, E)$ puede ser representado gráficamente por un dibujo donde cada vértice es un círculo etiquetado por el vértice y cada arista $e = uv \in E$ es representada por una curva que une los vértices u y v.

Ejemplo 1. *Dado un grafo G con $V(G) = \{u, v, w, x, y\}$ y $E(G) = \{uw, vw, xw, vy, vx, wy, xy\}$, un dibujo de G será el mostrado en la Figura 2.1.*

Además, dado un grafo G, para todo vértice $v \in V(G)$, se define la *vecindad* de v en G por:

$$N_G(v) = \{u \in V(G) : uv \in E(G)\}$$

Si $u \in N_G(v)$, entonces u y v se dicen *adyacentes*. Por otro lado, denotamos $N_G[v] = N_G(v) \cup \{v\}$, la vecindad cerrada de v en G. Si no hay confusión, podemos escribir $N(v)$ en lugar de $N_G(v)$.
A continuación, presentamos la definición de grafo ponderado, valorado, o con pesos, el cual corresponde a un grafo en el que las aristas tienen asociado un valor, o peso. Este tipo de grafos es utilizado para el modelamiento de redes sociales con relaciones valoradas.

Definición 1. *Sea $G = (V, E)$ un grafo. Una **función de asignación de pesos** corresponde a una función $\omega : E \to \mathbb{R}$, tal que a cada arista le asigna un número real.*

Definición 2. *Un **grafo ponderado** corresponde a un trío ordenado $G = (V, E, \omega)$, donde V es el conjunto de vértices, E son sus aristas y ω es una función de asignación pesos.*

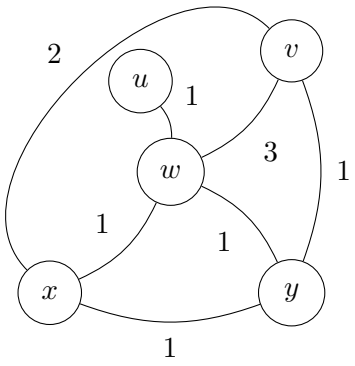


Figura 2.2: Un dibujo de $G = (V, E, \omega)$, con $V(G) = \{u, v, w, x, y\}$ y $E(G) = \{uw, vw, xw, vy, vx, wy, xy\}$ y con la siguiente asignación de pesos: $\omega(uw) = 1$, $\omega(vw) = 3$, $\omega(xw) = 1$, $\omega(wy) = 1$, $\omega(vy) = 1$, $\omega(xy) = 1$, $\omega(vx) = 2$.

Ejemplo 2. *En la Figura 2.2 se muestra un dibujo de un grafo ponderado $G = (V, E, \omega)$. En este caso, hemos asignado pesos a cada arista del grafo mostrado en la Figura 2.1 a través de la siguiente función de pesos: $\omega(uw) = 1$, $\omega(vw) = 3$, $\omega(xw) = 1$, $\omega(wy) = 1$, $\omega(vy) = 1$, $\omega(xy) = 1$, $\omega(vx) = 2$.*

Por otro lado, definimos un ***subgrafo inducido***. Dado un grafo G y un conjunto $S \subset V(G)$, el subgrafo inducido por S, denotado por $G[S]$, es el grafo cuyo conjunto de vértices es S y cuyo conjunto de aristas son todas las aristas en $E(G)$ con sus dos extremos en S.

$$G[S] := (S, \{uv \in E(G) : u, v \in S\}).$$

También definimos los digrafos, los cuales serán utilizados en la elaboración de los algoritmos más adelante.
Un ***digrafo***, o grafo dirigido, es un par $D = (V, A)$ donde $V = V(G)$ es un conjunto de vértices y $A = A(G)$ un conjunto de pares ordenados de vértices llamados *arcos*. Denotaremos los arcos por (u, v), con u y v elementos de V. Además, u se llama cola y v se llama cabeza del arco. Un arco de D es llamado *bucle* si la cola y la cabeza del arco son el mismo vértice.
Además, denotamos la vecindad de entrada y vecindad de salida como los siguientes conjuntos respectivamente:

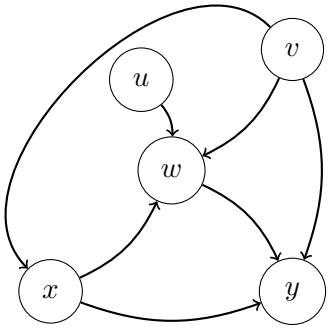


Figura 2.3: Un dibujo de $D = (V, A)$, con $V(D) = \{u, v, w, x, y\}$ y $A(D) = \{(u, w), (v, w), (x, w), (v, y), (v, x), (w, y), (x, y)\}$.

$$N_D^-(v) := \{u \in V(D) : (u, v) \in A(D)\},$$
$$N_D^+(v) := \{u \in V(D) : (v, u) \in A(D)\}.$$

Ejemplo 3. *Consideremos nuevamente el conjunto de vértices usado en el Ejemplo 1, pero en este caso el conjunto de aristas serán reemplazados por arcos como hemos definido previamente, es decir, definimos el digrafo mostrado en la Figura 2.3 como* $D = (V, A)$ *con* $V(D) = \{u, v, w, x, y\}$ *y* $A(D) = \{(u, w), (v, w), (x, w), (v, y), (v, x), (w, y), (x, y)\}$.
Además, podemos ver que las vecindades de entrada y salida del vértice w *son respectivamente:*

$$N_D^-(w) := \{u, v, x\},$$
$$N_D^+(w) := \{y\}.$$

Observación 1. *Notemos que de igual forma como se ha definido para un grafo simple, los digrafos pueden tener pesos asociados a sus aristas, obteniendo así un* ***digrafo ponderado*** $D = (V, A, \omega)$*, con* ω *función de pesos. Un ejemplo de esto se muestra en la Figura 2.4.*

En lo que sigue de la sección, utilizaremos v_i con $i = 1, \ldots, n$ para denotar a los vértices de los grafos y digrafos, indistintamente, pues es útil para expresar un orden en los conjuntos y fácil de asociar a filas y columnas de las matrices utilizadas para representar grafos, las cuales describimos a continuación.

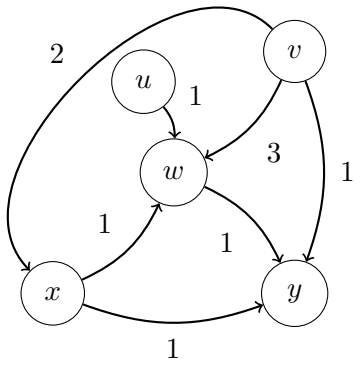
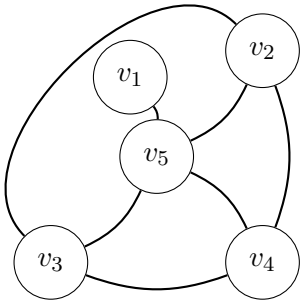


Figura 2.4: Un dibujo de un digrafo ponderado $D = (V, A)$, con $V(D) = \{u, v, w, x, y\}$ y $A(D) = \{(u, w), (v, w), (x, w), (v, y), (v, x), (w, y), (x, y)\}$, y función de pesos dada por: $\omega((u, w)) = 1$, $\omega((v, w)) = 3$, $\omega((x, w)) = 1$, $\omega((w, y)) = 1$, $\omega((v, y)) = 1$, $\omega((x, y)) = 1$, $\omega((v, x)) = 2$.

Sea un grafo $G = (V, E)$ con $V = \{v_1, \ldots, v_n\}$ su conjunto de vértices. Definimos la ***matriz de adyacencia*** de G, denotada por $Ad(G)$, matriz de tamaño $n \times n$, cuyas entrada $a_{i,j}$ de la fila i y columna j toman valores 0 y 1 de acuerdo con:

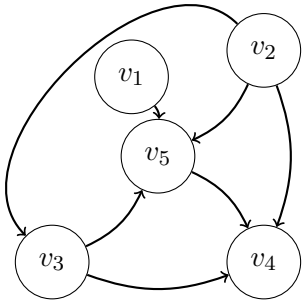
$$\forall i, j \in \{1, \ldots, n\}, \quad a_{i,j} = \begin{cases} 1 & si \quad v_i v_j \in E \\ 0 & si \quad v_i v_j \notin E \end{cases}$$

Ejemplo 4. *Consideremos un grafo $G = (V, E)$, con conjuntos $V(G) = \{v_1, v_2, v_3, v_4, v_5\}$ y $E(G) = \{v_1v_5, v_2v_5, v_3v_5, v_2v_4, v_2v_3, v_5v_4, v_3v_4\}$. Luego, obtenemos su matriz de adyacencia. Determinamos las entradas de la matriz siguiendo la definición antes dada, con lo cual se obtiene la matriz $Ad(G)$ mostrada en la Figura 2.5.*
Aquí es importante observar que para un grafo no dirigido, la matriz de adyacencia será una matriz simétrica, sin embargo, para el caso de un digrafo esta puede no ser simétrica.
Consideremos el digrafo definido a partir del conjunto de vértices que se definió en las primeras líneas de este ejemplo y con conjunto de arcos igual al conjunto de aristas que usamos previamente. En tal caso, se obtiene el digrafo y la matriz de adyacencia mostrados en la Figura 2.6.
Como vemos, la matriz de adyacencia del digrafo no es simétrica a diferencia de la matriz de adyacencia del grafo.



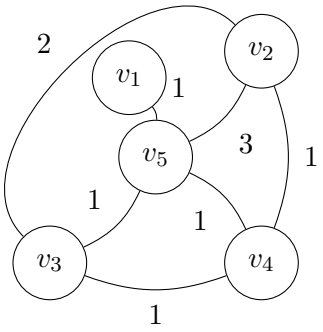
$$Ad(G) = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

Figura 2.5: A la izquierda podemos ver el grafo G definido en el Ejemplo 4 y a la derecha su respectiva matriz de adyacencia.



$$Ad(D) = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Figura 2.6: A la izquierda vemos el dibujo del digrafo $D = (V, A)$, con $V(D) = \{v_1, v_2, v_3, v_4, v_5\}$ y $A(D) = \{(v_1, v_5), (v_2, v_5), (v_3, v_5), (v_2, v_4), (v_2, v_3), (v_5, v_4), (v_3, v_4)\}$ y a la derecha su respectiva matriz de adyacencia.



$$W(G) = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 2 & 1 & 3 \\ 0 & 2 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 3 & 1 & 1 & 0 \end{pmatrix}$$

Figura 2.7: A la izquierda podemos ver el grafo G definido por $V(G) = \{v_1, v_2, v_3, v_4, v_5\}$, $E(G) = \{v_1v_5, v_2v_5, v_3v_5, v_2v_4, v_2v_3, v_5v_4, v_3v_4\}$ y asignación de pesos: $\omega(v_1v_5) = 1$, $\omega(v_2v_5) = 3$, $\omega(v_3v_5) = 1$, $\omega(v_5v_4) = 1$, $\omega(v_2v_4) = 1$, $\omega(v_3v_4) = 1$, $\omega(v_2v_3) = 2$; y a la derecha se muestra su matriz de pesos.

Dado un grafo ponderado $G = (V, E, \omega)$ con $V(G) = \{v_1, \ldots, v_n\}$. Definimos la **matriz de pesos** del grafo ponderado G como la matriz $W(G)$ de tamaño $n \times n$, cuyas entradas $w_{i,j}$ de la fila i y columna j toma valores en $\mathbb{R}$, tal que

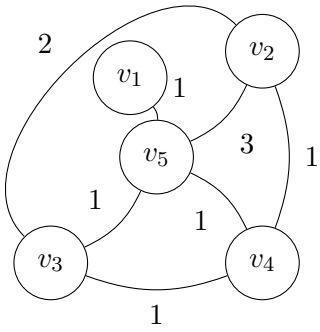
$$\forall i, j \in \{1, \ldots, n\}, \quad w_{i,j} = \begin{cases} \omega(v_iv_j) & si \quad v_iv_j \in E \\ 0 & si \quad v_iv_j \notin E \end{cases}$$

Ejemplo 5. *Consideramos el grafo ponderado $G = (V, E, \omega)$ con $V(G) = \{v_1, v_2, v_3, v_4, v_5\}$, $E(G) = \{v_1v_5, v_2v_5, v_3v_5, v_2v_4, v_2v_3, v_5v_4, v_3v_4\}$ y con la siguiente asignación de pesos: $\omega(v_1v_5) = 1$, $\omega(v_2v_5) = 3$, $\omega(v_3v_5) = 1$, $\omega(v_5v_4) = 1$, $\omega(v_2v_4) = 1$, $\omega(v_3v_4) = 1$, $\omega(v_2v_3) = 2$ y obtenemos su matriz de pesos, graficándolos en la Figura 2.7.*

Ahora, definimos una ***matriz de pesos aumentada*** para el grafo ponderado $G = (V, E, \omega)$ por $W^*(G)$ de tamaño $n \times n$ y cuyas entradas $w^*_{i,j}$ de la fila i y columna j toma valores en $\mathbb{R}$ dados por:

$$\forall i, j \in \{1, \ldots, n\}, \quad w^*_{i,j} = \begin{cases} \omega(v_iv_j) & si \quad v_iv_j \in E \\ \max\limits_{v_i, v_j \in V(G)} (\omega(v_iv_j)) & si \quad i = j \\ 0 & si \quad v_iv_j \notin E \end{cases}$$

Cabe mencionar que, dados $G = (V, E, \omega)$, c una constante real distinta de cero, $W(G)$ la matriz de pesos del grafo e $\mathbf{I}$ la matriz identidad, se pue-



$$W(G) = \begin{pmatrix} 3 & 0 & 0 & 0 & 1 \\ 0 & 3 & 2 & 1 & 3 \\ 0 & 2 & 3 & 1 & 1 \\ 0 & 1 & 1 & 3 & 1 \\ 1 & 3 & 1 & 1 & 3 \end{pmatrix}$$

Figura 2.8: A la izquierda podemos ver un dibujo del grafo G definido en el Ejemplo 5 y a la derecha su respectiva matriz de pesos aumentada.

den obtener distintas matrices de pesos aumentadas a partir de la ecuación $W^*(G) = W(G) + c\mathbf{I}$, donde nuestra definición se obtiene al hacer $c = \max_{v_i, v_j \in V(G)} (\omega(v_i v_j))$.

Ejemplo 6. *En este ejemplo, obtendremos la matriz de pesos aumentada vista anteriormente en el Ejemplo 5. Para ello notamos que la entrada de mayor valor de la matriz de pesos es* 3*, por lo cual, reemplazaremos los valores de la diagonal de la matriz de pesos por* 3*, obteniendo la matriz de pesos aumentada. En la Figura 2.8 se grafica el grafo junto con la matriz de pesos aumentada obtenida al realizar dicha operación.*

2.2. Matrices de Robinson

Para definir las matrices de Robinson, primero repasaremos el concepto de *similaridad*. Una función de similaridad, o medida de similaridad, es una función de valor real que cuantifica la similitud entre dos objetos. Una función de similaridad en un conjunto finito S puede ser representada por una matriz con valores reales M, de tamaño $s \times s$, donde s el tamaño del conjunto S y tal que cada entrada de $m_{i,j}$ indica el valor de la función de similaridad para los usuarios i y j del conjunto S.
Una *matriz de Robinson* es una matriz de similaridad simétrica donde el valor las entradas de las filas y columnas no decrecen al acercarse a la diagonal.

$$\underbrace{\begin{pmatrix} 3 & 1 & 0 & 0 & 0 \\ 1 & 3 & 3 & 1 & 1 \\ 0 & 3 & 3 & 2 & 1 \\ 0 & 1 & 2 & 3 & 1 \\ 0 & 1 & 1 & 1 & 3 \end{pmatrix}}_{\text{Robinson } M^{\boldsymbol{\pi}}} = \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}}_{\text{Permutación } \Pi} \underbrace{\begin{pmatrix} 3 & 0 & 0 & 0 & 1 \\ 0 & 3 & 2 & 1 & 3 \\ 0 & 2 & 3 & 1 & 1 \\ 0 & 1 & 1 & 3 & 1 \\ 1 & 3 & 1 & 1 & 3 \end{pmatrix}}_{\text{Robinsoniana } M} \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}}_{\text{Permutación } \Pi^T}$$

Figura 2.9: En la imagen vemos una matriz Robinsoniana que a través de una permutación de filas y columnas es transformada en una matriz Robinson.

Definición 3. (**Matriz de similaridad de Robinson**) *Dada una matriz M simétrica de orden $n \times n$ sobre un conjunto S, M se dice* matriz de similaridad de Robinson *si S puede ser ordenado linealmente de tal forma que*

$$\forall i < j < k, \quad M(i,j) \geq M(i,k) \quad \wedge \quad M(j,k) \geq M(i,k).$$

Por otro lado, se definen los conceptos de *permutación* y *matriz Robinsoniana*, lo que servirá para la identificación y construcción de matrices Robinson. Una *permutación* corresponde a una variación del orden o posición de los elementos de un conjunto ordenado o tupla, y en este caso, también de una matriz.

Definición 4. *(**Matriz Robinsoniana**). Una matriz M simétrica de orden $n \times n$ se dirá **Robinsoniana** si existe una permutación $\boldsymbol{\pi}$ tal que $\Pi M \Pi^T = M^{\boldsymbol{\pi}} := (M_{\boldsymbol{\pi}(i)\boldsymbol{\pi}(j)})_{i,j}$ es una matriz de similaridad de Robinson.*

Por otro lado, en caso de existir dicha permutación $\boldsymbol{\pi}$ que permita reordenar la matriz M de tal forma que esta sea una matriz Robinson, entonces, llamaremos a dicho orden $\boldsymbol{\pi}$, ***orden de Robinson***.

Ejemplo 7. *En la Figura 2.9 presentamos un ejemplo de una matriz M simétrica de orden 5×5 Robinsoniana, que al aplicar una permutación de filas y columnas es transformada en una matriz Robinson. Para este caso el orden de Robinson será $\boldsymbol{\pi} = [1, 5, 2, 3, 4]$.*
Observemos que la matriz Robinsoniana de la Figura 2.9 coincide con la matriz de pesos aumentada definida en el Ejemplo 6, por lo que podemos intuir que a través del modelamiento de problemas con grafos, podemos llegar a trabajar con matrices Robinsonianas al tratar de ordenar nuestro conjunto

de vértices pensando en posicionar más cerca aquellos más similares entre sí. Así, si inicialmente consideramos el orden de los vértices v_1, v_2, v_3, v_4, v_5 para construir la matriz de pesos aumentada, entonces el orden v_1, v_5, v_2, v_3, v_4, nos entrega directamente una matriz de pesos aumentada que es una matriz Robinson, y dicho orden de los vértices es un orden de Robinson, que posiciona más cerca entre sí a los vértices más similares entre sí.

Observación 2. *Note que, de las definiciones antes mencionadas, una matriz de pesos aumentadas puede coincidir con una matriz Robinsoniana, por lo cual, a partir de un grafo ponderado, definimos una matriz de pesos aumentada que puede ser una matriz Robinsoniana. En el caso de que esto ocurra, diremos que el grafo de dicha matriz es un* ***grafo de Robinson****.*

2.3. Cortes y conductancia

Ahora, presentamos conceptos aplicados tanto a grafos como digrafos, que son necesarios para definir e implementar medidas que permitirán evaluar los distintos algoritmos de recomendación.
Dado $S, T \subseteq V(G)$, se denota $[S, T]$ al conjunto de aristas con un extremo en el conjunto S y otro en el conjunto T. Dado S es un conjunto propio no vacío de $V(G)$ y $\bar{S} = V(G) - S$, el ***conjunto de corte*** dado por S corresponde a $[S, \bar{S}]$.

Ejemplo 8. *Para comprender mejor el concepto de conjunto de corte, consideramos el grafo de la Definición 2 y realizamos una partición del conjunto de vértices $V(G)$. Consideramos $S = \{v_2, v_4\}$ y $\bar{S} = \{v_1, v_3, v_5\}$. En la Figura 2.10, la recta de color rojo indica la separación del conjunto de vértices S y $\bar{S}$ que hemos definido. Luego, el conjunto de corte corresponderán a $[S, \bar{S}] = \{v_3v_2, v_5v_2, v_5v_4, v_3v_4\}$; en la misma figura podemos ver como estas aristas cruzan la recta roja que está diviendo nuestro conjunto de vértices, mientras que las aristas que están en el conjunto de corte no la cruzan.*

A continuación, definimos el concepto de *conductancia* tal como se define en [12]. Esta mide qué tan entrelazados (o enlazados) se encuentran los vértices de un grafo.

Definición 5. *Sea G un grafo y $S \subset V(G)$ un conjunto que define un corte del grafo. La* ***conductancia*** *de S es definida por:*

$$\phi(S) = \frac{||\partial S||}{||S||}$$

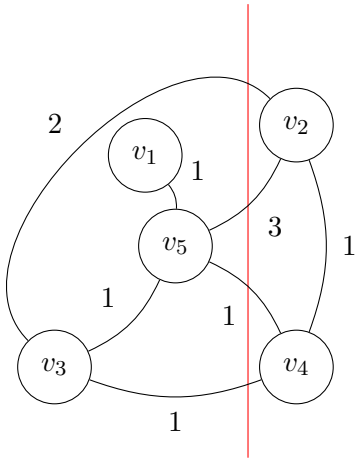


Figura 2.10: La figura muestra un dibujo del grafo definido en el Ejemplo 5 con una recta roja demarcando el corte del grafo dado por los conjuntos de vértices $S = \{v_2, v_4\}$ y $\bar{S} = \{v_1, v_3, v_5\}$. Además, para este corte se tiene: $||\partial S|| = 7$, $||S|| = 8$, $\langle\langle S \rangle\rangle = 1$, $\phi(S) = 0{,}875$ y $\phi'(S) = 7$.

donde $||\partial S||$ *corresponde a la suma de los pesos del conjunto de corte* $[S, \bar{S}]$*, también representado por la siguiente igualdad:*

$$||\partial S|| = \sum_{v_i \in S, v_j \in \bar{S}} \omega(v_i v_j),$$

y $||S||$ *es la suma de los pesos de las aristas con un extremo en* S*:*

$$||S|| = \sum_{v_i \in S} \sum_{v_j \in V(G)} \omega(v_i v_j).$$

Además, introducimos una nueva medida asociada a un conjunto de corte definido por $S \subset V(G)$, la cual llamaremos *expansión de* S. Esta medida es similar a la conductancia de la Definición 5, sin embargo, difieren ya que no consideramos la suma total de los pesos de las aristas con un extremo en S, sino que restringimos dicha suma sólo a las aristas albergadas en el subgrafo inducido por S.

Definición 6. *Sea un grafo* G *y* $S \subset V(G)$ *que define un corte del grafo. La* ***expansión de*** S *es definida por:*

$$\phi'(S) = \frac{||\partial S||}{\langle\langle S \rangle\rangle}$$

donde $||\partial S||$ corresponde a la suma de los pesos de las aristas del conjunto de corte $[S, \bar{S}]$, explicitado en la Definición 5, y con $\langle\langle S \rangle\rangle$ la suma de los pesos de todas las aristas con sus dos extremos en S, esto es:

$$\langle\langle S \rangle\rangle = \sum_{v_i \in S} \sum_{v_j \in S} \omega(v_i v_j).$$

Observamos que esta nueva medida entrega una proporción entre los pesos de las aristas en el conjunto de corte respecto al peso que suman las aristas del grafo inducido por S.

También definimos una tercera métrica para un corte del grafo llamada *corte normalizado*. Esta nos entregará una proporción de la suma de los pesos de las aristas en el corte sobre la suma total de los pesos de las aristas del grafo.

Definición 7. *Sea un grafo G y $S \subset V(G)$ que genera un corte del grafo. El corte normalizado S es definido por:*

$$\phi''(S) = \frac{||\partial S||}{||G||}$$

donde $||\partial S||$ corresponde a la suma de los pesos de las aristas del conjunto de corte $[S, \bar{S}]$, y $||G||$ es la suma de los pesos de todas las aristas del grafo, esto es:

$$||G|| = \sum_{v_i \in V(G)} \sum_{v_j \in V(G)} \omega(v_i v_j).$$

Además, si representamos un grafo o digrafo ponderado G a través de una matriz, reescribimos las métricas antes definidas para un conjunto de corte dado por los conjuntos $S \subset V(G)$ y $\bar{S} = V(G) - S$ en función de las entradas de la matriz de pesos $W(G)$ como sigue:

$$\phi(S) = \frac{\sum_{v_i \in S} \sum_{v_j \in \bar{S}} w_{i,j}}{\sum_{v_k \in S} \sum_{v_l \in V} w_{k,l}} \tag{2.1}$$

$$\phi'(S) = \frac{\sum_{v_i \in S} \sum_{v_j \in \bar{S}} w_{i,j}}{\sum_{v_k \in S} \sum_{v_l \in S} w_{k,l}} \tag{2.2}$$

$$\phi''(S) = \frac{\sum_{v_i \in S} \sum_{v_j \in \bar{S}} w_{i,j}}{\sum_{v_k \in V} \sum_{v_l \in V} w_{k,l}} \tag{2.3}$$

2.4. Red social y recomendaciones

Una *red social* será representada a través de un conjunto $\mathbf{X}$, el que contiene todos los perfiles de la red. Un usuario particular de la red social es caracterizado por $\mathbf{V} \subseteq \mathbf{X}$ (subconjunto de $\mathbf{X}$), donde $\mathbf{V}$ corresponde a los perfiles de *amigos* en la red social de dicho usuario. Además, se denota $\mathcal{P}(\mathbf{X})$ al conjunto potencia (o conjunto de las partes) de $\mathbf{X}$. Luego, se define la función $d : \mathcal{P}(\mathbf{X}) \times \mathcal{P}(\mathbf{X}) \to \mathbb{Z}$, que a cada par de usuarios $\mathbf{U}$ y $\mathbf{V}$ entrega el valor de similitud dado por $d(\mathbf{U}, \mathbf{V}) = |\mathbf{U} \cap \mathbf{V}|$, es decir, el tamaño de la intersección entre $\mathbf{U}$ y $\mathbf{V}$, lo que representa la cantidad de perfiles en común que siguen ambos usuarios.

Observación 3. *Notemos que para todo par de usuarios* $\mathbf{U}$ *y* $\mathbf{V}$*, esta función cumple* $d(\mathbf{U}, \mathbf{V}) \leq \text{mín}\{|\mathbf{U}|, |\mathbf{V}|\}$*, y además* $d(\mathbf{V}, \mathbf{V}) = |\mathbf{V}|$ *para todo usuario* $\mathbf{V}$*. Por lo tanto, la similitud de un usuario es máxima cuando es comparado consigo mismo.*

Dado un conjunto de usuarios $\mathcal{W} \subseteq \mathcal{P}(\mathbf{X})$ de tamaño m de la red social, y un orden de $\mathcal{W}$, se define la matriz simétrica $S(\mathcal{W})$ de tamaño $m \times m$ tal que la entrada $S_{\mathbf{UV}}$, correspondiente a la fila indexada por $\mathbf{U}$ y la columna indexada por $\mathbf{V}$, es igual a $d(\mathbf{U}, \mathbf{V})$. Además, se define la matriz de similitud normalizada $B(\mathcal{W})$ del conjunto $\mathcal{W}$, como la matriz cuya fila indexada por $\mathbf{U}$ y columna indexada por $\mathbf{V}$ es igual a $S_{\mathbf{UV}}/d(\mathbf{U}, \mathbf{U})$. Cabe mencionar que esta última matriz no es necesariamente simétrica.

Por último, definiremos los conceptos *recomendación* y *algoritmo de recomendación*, los cuales son el objetivo a encontrar en este trabajo y los algoritmos utilizados para ello, respectivamente.

Definición 8. *Dado un conjunto de usuarios $\mathcal{W}$, una* ***recomendación*** *para un usuario $\mathbf{V}$ de $\mathcal{W}$ es un subconjunto de usuarios $R_{\mathbf{V}} \subseteq \mathcal{W}$.*

Definición 9. *Un* ***algoritmo de recomendación*** *es un método o criterio que, dado un conjunto de m usuarios $\mathcal{W}$, asigna una recomendación $R_{\mathbf{V}}$ a cada usuario $\mathbf{V} \in \mathcal{W}$.*

Para finalizar, relacionamos las definiciones vistas en las secciones anteriores y con lo visto en esta sección. Por una parte, imaginamos nuestra red social como un grafo, donde cada vértice del grafo representa a un usuario de la red social. También, consideramos que todos los vértices están unidos con los otros vértices del grafo mediante una arista, y que a cada arista le asignamos un peso a través de la función d que correspondería a la función de asignación de pesos del grafo ponderado de la Definión 1. Esta representación de la red social a través de un grafo se muestra en la Figura 2.11.
Además, la matriz $S(\mathcal{W})$ corresponde a la matriz de pesos aumentada del grafo de la red social restringiendonos al conjunto de usuarios $\mathcal{W}$ y con función de pesos igual a d, en cambio, la matriz $B(\mathcal{W})$ se obtiene al considerar el conjunto de usuarios como vértices de un digrafo, donde cada arista que los une tiene pesos dados por la función de asignación de pesos $d(\mathbf{U}, \mathbf{V})/d(\mathbf{U}, \mathbf{U})$. Algo importante de mencionar es que en la literatura existen definiciones en donde los grafos tienen *bucles* (cf. [29]), el cual corresponde a una arista que une a un vértice con el mismo, es decir, dado un vértice $v_i \in V$, un bucle es una arista del tipo $v_i v_i$. Cuando trabajamos con un grafo con bucles, la definición de matriz de pesos aumentada debería cambiar, ya que los elementos de la diagonal tendrán entradas no nulas. Así, podemos abordar nuestro problema desde distintas miradas, una de ellas es tener como variable de entrada el grafo que representa la red social sin bucles, y luego definir la matriz de pesos aumentada; por otra parte, podemos considerar como input un grafo o digrafo ponderado con bucles; o bien, considerar directamente una matriz de similaridad como la variable de entrada del problema. Nuestro objetivo será generar algoritmos de recomendación. Una de las formas que utilizaremos para ello es obtener, si es posible, un orden de Robinson ya que de esta forma podremos reordenar los vértices o filas, para así tener un orden de nuestro conjunto de datos en que los usuarios más similares están más próximos entre sí.

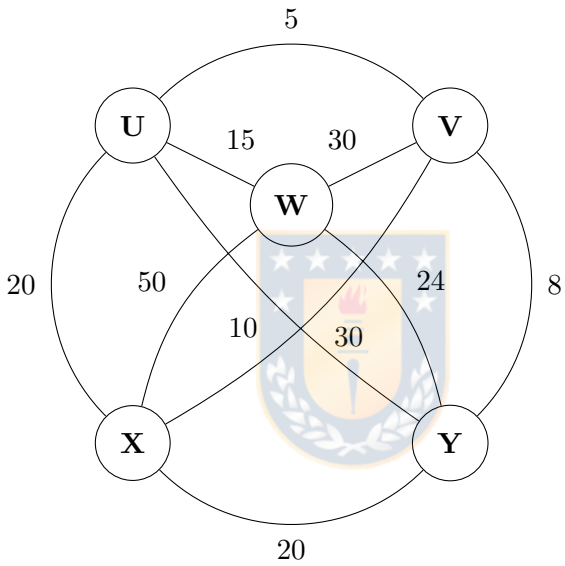


Figura 2.11: El grafo de esta figura representa las relaciones entre un conjunto de 5 usuarios de una red social, donde cada peso sobre las aristas indica el total de *amigos* en común entre los respectivos usuarios.

2.5. Medidas para algoritmos de recomendación

Basándonos en el hecho de que podemos modelar nuestra red social a través de un grafo, aprovecharemos las medidas vistas para grafos en la Sección 2.3 para medir la calidad de las recomendaciones entregadas por un algoritmo de recomendación. En lo que sigue, se reescribirán las medidas antes vistas en la notación que hemos adoptado para el problema de redes sociales, sin olvidar que ambas medidas siguen teniendo el mismo sentido, por lo que nos aportarán una medida de que tan entrelazados están los usuarios de nuestra red, y que tanta similitud existe entre los usuarios que permanecen en el corte que está definiendo la recomendación respecto a los que están fuera de la recomendación. En particular, usamos las igualdades 2.1 y 2.2 para definir las medidas a partir de la matriz normalizada $B(\mathcal{W})$.
Para un algoritmo de recomendación $\mathcal{A}$ y su conjunto de recomendaciones $\{R_{\mathbf{V}}(\mathcal{A})\}_{\mathbf{V}\in\mathcal{W}}$, definimos:

Definición 10. ***Promedio de las conductancias.*** *Se denota por $C(\mathcal{A})$, el promedio de las conductancias de los cortes definidos por cada recomenación de $\mathcal{A}$ para los m usuarios, lo que matemáticamente se expresa como sigue:*

$$\begin{aligned} C(\mathcal{A}) &= \frac{1}{m}\cdot\sum_{\mathbf{V}\in\mathcal{W}}\phi(\{R_{\mathbf{V}}\cup\{\mathbf{V}\}\}) \\ &= \frac{1}{m}\cdot\sum_{\mathbf{V}\in\mathcal{W}}\frac{\sum_{\mathbf{U}'\in\{R_{\mathbf{V}}\cup\{\mathbf{V}\}\}}\sum_{\mathbf{V}'\in\mathcal{W}\setminus\{R_{\mathbf{V}}\cup\{\mathbf{V}\}\}}B_{\mathbf{U}'\mathbf{V}'}}{\sum_{\mathbf{U}'\in\{R_{\mathbf{V}}\cup\{\mathbf{V}\}\}}\sum_{\mathbf{V}'\in\mathcal{W}}B_{\mathbf{U}'\mathbf{V}'}}. \end{aligned}$$

Definición 11. ***Promedio de la expansión de*** $R_{\mathbf{V}}\cup\{\mathbf{V}\}$. *Se denota por $C'(\mathcal{A})$, calcula el promedio de la expansión de cada recomenación de $\mathcal{A}$, lo que matemáticamente se expresa como sigue:*

$$\begin{aligned} C'(\mathcal{A}) &= \frac{1}{m}\cdot\sum_{\mathbf{V}\in\mathcal{W}}\phi'(\{R_{\mathbf{V}}\cup\{\mathbf{V}\}\}) \\ &= \frac{1}{m}\cdot\sum_{\mathbf{V}\in\mathcal{W}}\frac{\sum_{\mathbf{U}'\in\{R_{\mathbf{V}}\cup\{\mathbf{V}\}\}}\sum_{\mathbf{V}'\in\mathcal{W}\setminus\{R_{\mathbf{V}}\cup\{\mathbf{V}\}\}}B_{\mathbf{U}'\mathbf{V}'}}{\sum_{\mathbf{U}'\in\{R_{\mathbf{V}}\cup\{\mathbf{V}\}\}}\sum_{\mathbf{V}'\in\{R_{\mathbf{V}}\cup\{\mathbf{V}\}\}}B_{\mathbf{U}'\mathbf{V}'}}. \end{aligned}$$

En esta segunda medida, determinamos para cada algoritmo y sus recomendaciones, el promedio de la expansión del conjunto que genera el corte vista en la Definición 6, donde el conjunto equivalente a S está dado por la recomendación entregada a cada usuario y el usuario mismo, esto es $R_{\mathbf{V}}\cup\{\mathbf{V}\}$.

Por último, definimos el promedio del corte normalizado, medida que nos entregará la proporción entre la suma de los pesos de las aristas en el conjunto de corte definido por las recomendaciones y el usuario al realizamos la recomendación, y la suma de los pesos de las aristas de todo el grafo que representa al conjunto de muestra de la red social $\mathcal{W} \subseteq \mathcal{P}(\mathbf{X})$.

Definición 12. ***Promedio del corte normalizado*** *Denotamos por $C''(\mathcal{A})$, el promedio de los cortes normalizados definidos por cada recomenación de $\mathcal{A}$ para los m usuarios, lo que matemáticamente se expresa como sigue:*

$$\begin{aligned} C''(\mathcal{A}) &= \frac{1}{m} \cdot \sum_{\mathbf{V} \in \mathcal{W}} \phi''(\{R_{\mathbf{V}} \cup \{\mathbf{V}\}\}) \\ &= \frac{1}{m} \cdot \sum_{\mathbf{V} \in \mathcal{W}} \frac{\sum_{\mathbf{U}' \in \{R_{\mathbf{V}} \cup \{\mathbf{V}\}\}} \sum_{\mathbf{V}' \in \mathcal{W} \setminus \{R_{\mathbf{V}} \cup \{\mathbf{V}\}\}} B_{\mathbf{U}'\mathbf{V}'}}{\sum_{\mathbf{U}' \in \mathcal{W}} \sum_{\mathbf{V}' \in \mathcal{W}} B_{\mathbf{U}'\mathbf{V}'}}. \end{aligned}$$

Intuitivamente, esperamos que mientras más bajas sean $C(\mathcal{A})$, $C'(\mathcal{A})$ y $C''(\mathcal{A})$ para algún algoritmo dado $\mathcal{A}$ entonces mejor será $\mathcal{A}$, sin embargo, se reitera que no es objetivo de este proyecto minimizar dichas medidas.

Capítulo 3

Estado del Arte

En este capítulo, revisamos el *estado del arte* de los conceptos más novedosos e importantes de este trabajo. En primer lugar, veremos los inicios de las matrices de Robinson, y en particular los algoritmos que se han generado para reconocer una matriz de Robinson. Luego, repasamos brevemente la historia de la conductancia y sus aplicaciones en distintas áreas de la investigación matemática. Después, comentamos algunos estudios en el ámbito de los algoritmos de recomendación, mayormente se citan investigación que han recopilado información de este tema. Por último, nos centramos en *k-means*, el cual es uno de los algoritmos de recomendación más populares y utilizado desde la década de los sesentas hasta la actualidad.

3.1. Matrices de Robinson

En la Definición 3 presentamos las matrices de Robinson. Dichas matrices poseen diversas aplicaciones en ordenamiento y clasificación. En 1951, Robinson definió las matrices Robinsonianas en [21] para el estudio de cómo ordenar cronológicamente depósitos arqueológicos. En este mismo trabajo, Robinson presenta el *problema de ordenamiento*, que busca decidir si una matriz de similaridad asociada a un conjunto de datos es Robinsoniana y además, escribirla como una matriz Robinson si es posible. En relación a esto, se identifica un desafío en el reconocimiento de las matrices de Robinson, problema que ha sido abordado por distintos autores a lo largo de los años. En la década de los noventas, Chepoi y Fichet en [4] presentan un algoritmo de orden $O(n^3)$ basado en la técnica de "divide y conquista", para reconocer matrices Robinsonianas de tamaño $n \times n$. Posteriormente, Préa y Fortin en [18] presentan un algoritmo óptimo de orden $O(n^2)$ para reconocer matrices

Robinsonianas usando *PQ-trees* (o PQ-árboles).
Ahora, presentamos la definición de grafos de intervalo unitario entregada por Gardi en [5]: *un grafo de intervalo unitario es un grafo no dirigido* $G = (V, E)$ *donde cada vértice* $v \in V$ *es asociado a un intervalo cerrado* $I_v = [l_v, r_v]$*, todos de igual longitud, y tal que cada par de vértices distintos* $u, v \in V$ *son adyacentes si* $I_v \cap I_u \neq \emptyset$. Roberts en [20] presenta una relación entre las matrices Robinsonianas y los grafos de intervalo unitario, la cual indica que una matriz A de tamaño $n \times n$ y entradas 0 y 1 será una similaridad Robinsoniana si sólo si A es la matriz de adyacencia de un grafo de intervalo unitario G. Esta propiedad ha sido relevante para varios autores ([1],[13] y [18]) y en particular para Laurent y Seminaroti en [13], donde configuran un algoritmo de reconocimiento de matrices Robinsonianas utilizando dicha relación y el algoritmo de búsqueda de amplitud lexicográfica (o Lex-BFS por sus siglas en inglés); el algoritmo Lex-BFS fue inicialmente desarrollado por Donal J. Rose et al. en [22]. El algoritmo presentado por Laurent es de orden polinomial $O(L(n + m))$ cuando se aplica a una matriz simétrica de tamaño $n \times n$, donde m es la cantidad de entradas distintas de cero y L el número de valores distintos en la matriz. Luego, en el año 2017, los mismos autores presentan un nuevo algoritmo llamado *similarity first search* (SFS) [14], que extiende el algoritmo Lex-BFS a grafos ponderados (o con pesos) y cuyo tiempo de ejecución es de orden $O(n^2 + nm \log(n))$. Es importante destacar que el software R implementa los últimos algoritmos mencionados (Lex-BFS y SFS) en el paquete SFS, de la cual hemos utilizado funciones con las que implementamos los algoritmos de recomendación en este trabajo.

3.2. Conductancia en teoría grafos

La conductancia es una propiedad eléctrica de la cual se ha adoptado el nombre en la teoría de grafos gracias a la utilización de grafos en teoría de circuitos, impulsada por el trabajo de G. Kirchhoff en [10], el cual modeló circuitos eléctricos con líneas y puntos, es decir, trabajó con el grafo subyacente a un circuito eléctrico, transformando así el álgebra de grafos en una importante herramienta para el análisis y modelamiento de circuitos eléctricos. Por otro lado, la conductancia es también conocida como constante de Cheeger (o número isoperimétrico) para grafos, esta es definida por Mohar en [17], trabajo en el que se discuten distintas propiedades de esta medida. Sin embargo, la constante de Cheeger es definida y utilizada en la geometría espectral (o geometría de Riemman), por lo que el término de conductancia a permitido diferenciar los términos más claramente entre estos campos de

estudio.
La conductancia mide qué tan bien entralazado se encuentra el grafo, y en el estudio de caminos aleatorios en un grafo, controla qué tan rápido estos converge a su distribución estacionaria. Esta medida ha tenido aplicaciones en distintas áreas, entre ellas el estudio de cadenas de Markov en [24] de Sinclair y Jerrum, el estudio de cortes normalizados y su aplicación en la segmentación de imágenes en [23], evaluar el tiempo de convergencia en un proceso de difusión de información como en los trabajos de Kiwi y Thraves en [11] y de Giakkoupis en [6], y además, ha permitido evaluar la calidad espectral de un cluster, como David F. Gleich y C. Seshadhri en [7], quienes generan vecindarios a partir de dos propiedades y utilizan la conductancia para evaluar su calidad y comparar dicho método con otros algoritmos populares como el corte de Fiedler y un método de detección de comunidad personalizado de PageRank (marca creada por Google a fines de los años noventas). Esta última aplicación será la más relevante para el presente trabajo ya que nos permitirá, a través de la conductancia, evaluar los cortes generados en el grafo por las recomendaciones obtenidas de cada algoritmo desarrollado, es decir, evaluamos la calidad del cluster a través de la conductancia.
Por otra parte, Kannan et al. en [9] analizan algunos inconvenietes generados al aplicar distintas métricas para evaluar la calidad de clusters como el *diámetro mínimo*, *k-centro*, *k-media* y *suma mínima*. Uno de los inconvenientes detectados es la baja calidad en los clusters, pues algunos agrupan puntos que debiesen estar separados. Los autores plantean que esta baja calidad es producida al minimizar la disimilaridad máxima entre los puntos de un cluster, por lo que se sugiere modelar los problemas a través de un grafo con similaridades en vez de disimilaridades. Además, en el mismo trabajo, Kannan et al. presentan una medida de bicriterio para evaluar la calidad de clusters, medida que consiste en optimizar dos parámetros , α y ϵ, que representan la *conductancia mínima* del cluster, y la proporción entre el peso de las aristas entre los clusters y el peso total de todas las aristas, respectivamente. Posteriormente, Krishnan y Goldberg en [12] presenta un algoritmo llamado *minimum conductance dissimilarity cut* (MCDC) que aproxima la solución al difícil problema de optimización multiobjetivo que implica el incrementar la novedad, diversidad y calificación esperada de una recomendación.
Por último, cabe recalcar que encontrar la conductancia mínima de un grafo es un problema NP-Duro (ver [9], [12]), es decir, es difícil de resolver, por lo que comunmente se realizarán aproximaciones de la misma o no se busca encontrar el mínimo exacto.

3.3. Algoritmos de recomendación

Los algoritmos de recomendación o sistemas de recomendación forman parte de los sistemas de filtrado de información, estos permiten mejorar la experiencia de usuarios en la red, optimizando su tiempo de búsqueda de artículos o servicios. Ricci et al. en [19] realiza una introducción a los sistemas de recomendación, repasando algunos sitios web (o empresas) que implementan algoritmos de recomendación o filtrado, como Youtube, Netflix y Amazon.com. Además, hace mención a espacios académicos -talleres, conferencias y otros- que se han dedicado al estudio y desarrollo de los algoritmos de recomendación, en específico al *ACM Recommender Systems*, además de importantes periódicos académicos de este campo, como *IEEE Intelligent Systems* y *International Journal of Computer Science and Applications*, entre otros. En el mismo trabajo, Ricci, revisa técnicas, aplicaciones y métodos de evaluación utilizados en los sistemas de recomendación. Tatiya y Vaidya en [26] realizan una revisión de varios tipos de algoritmos de recomendación y la clasificación de los mismos, ya sea si son basado en la información o en el conocimiento que se tenga de usuarios y objetos, entre estos se encuentran los algoritmos de recomendación basados en el contenido (puede ser experiencia pasada del usuario), colaborativos (basado en las experiencias de otros usuarios similares), de conocimiento (en base a un perfil de usuario y características del item a recomendar), demográficos (de acuerdo a caracteríscas como edad, estrato social, educación, etc) o híbridos (combinaciones de dos o más técnicas de recomendación). Por otro lado, Ji Lu et al. en [15] entregan una revisión actualizada de los desarrollos en diversas aplicaciones de sistemas de recomendación, agrupandolas en ocho categorías principales: gobierno electrónico, negocios online, comercio online, librería online, aprendizaje electrónico, turismo electrónico, servicios de recursos electrónicos y actividades en grupo digitales.

Finalmente, destacamos un hito en el ámbito de los algoritmos de recomendación, el conocido *Netflix Price*, descrito y analizado por Bennett y Lanning en [2]. Fue convocado por primera vez en el año 2006, ocasión en la que la empresa Netflix dispuso una base de datos de 100 millones de evaluaciones anónimas a películas, desafiando a la comunidad a mejorar la exactitud de sus recomendaciones, entregando una atractiva suma de dinero a quien superara el desafío. Mejorar los algoritmos y crear nuevos es una constante entre las comunidades de la minería de datos, ciencias de la computación e inteligencia artificial, por lo que año a año podemos encontrar trabajos y estudios relacionados a este ámbito, incentivando el aprendizaje continuo y desafíos como el puesto por Netflix.

3.4. K-means

El popular algoritmo *K-means* o K-medias por su traducción al español, corresponde a un método de agrupamiento, que tiene como fin particionar un conjunto de n elementos en k grupos. El término *"k-means"* fue utilizado por primera vez por James McQueen en 1967 [16], donde lo describe como un método para particionar una población n-dimensional en k grupos. Sin embargo, esta idea se atribuye al trabajo de Hugo Steinhaus en [25] donde busca dividir un cuerpo (matemáticamente hablando) en K_i partes, a la vez, escoge n puntos que permitan minimizar el momento de inercia dado por estos puntos y cada partición.
Hoy en día, existen numerosos métodos de agrupamiento o *clustering*, parte de esto es analizado por Jain en [8]. Jain introduce el documento presentando una diferencia importante entre los métodos de agrupamiento de datos y clasificación de datos, y esta es esencialmente, la ausencia de información categórica en los primeros, mientras que los segundos sí la poseen. Esto también podemos distinguirlo como aprendizaje supervisado (cuando existen la información o etiquetas) y no supervisado (cuando no existe esta información). Además, hace una revisión de los más populares métodos de agrupamiento que se han presentado, pasados más de 50 años desde que *k-means* fue propuesto. Jain, también discute los principales desafíos y problemas de esta área. Cabe destacar, que a pesar de que tras k-means se hayan presentado numerosos métodos para agrupamiento, este sigue siendo muy utilizado, esto habla de la dificultad de diseñar algoritmos de agrupamiento, y la complejidad del problema de agrupamiento de datos.

3.5. Nuestras contribuciones

Los temas antes revisados son piezas esenciales de nuestro trabajo. Como primer paso, modelamos una red social a través de un grafo o digrafo con pesos, el cual representaremos computacionalmente a través de una matriz. Nuestro objetivo es obtener una recomendación de perfiles de usuarios para un individuo de la red social estudiada, para esto planteamos 5 formas distintas de generar una recomendación, basadas principalmente en el concepto de *similitud* entre dos usuarios de la red y la forma en que podemos escoger el tamaño de esta recomendación. Una de las formas en la que haremos esto es generando una matriz en la que relacionamos la *influencia* de un usuario sobre otro a partir de los *amigos en común* y del total de perfiles actuales

de los usuarios, y otra será obteniendo un orden de la lista de usuarios que se han seleccionado de la red social en el que los más similares a un usuario particular se encuentren posicionados más cerca de él. Para el segundo caso planteado, utilizaremos los algoritmos de reconocimiento de matrices Robinsonianas proporcionados por el software *R-studio*, el cual nos entrega un orden de los vértices en el que aquellos más similares se encuentran más cerca y los más disimilares están alejados. Usando este orden, seleccionamos los más similares y generamos una recomendación. Posteriormente, y aquí es donde entra en juego la *conductancia*, determinamos el valor del promedio de la conductancia, el valor del promedio del corte normalizado y el valor del promedio de la expansión, para cada recomendación, permitiendonos medir una calidad potencial de los algoritmos propuestos. Finalmente, para tener un referente de si nuestro método para generar recomendaciones fue bueno o malo, comparamos las medidas obtenidas para los algoritmos definidos con las medidas obtenidas al evaluar recomendaciones generadas usando *k-means*.
En síntesis, nuestras contribuciones son:

1. Modelar un problema de recomendación en redes sociales a través de grafos y digrafos.

2. Generar algoritmos de recomendación no supervisados que entregan recomendaciones a usuarios de una red social. Además, ciertos algoritmos propuestos tienen la característica especial de permitir escoger el tamaño de recomendación que se entregará al usuario de la red.

3. Proponer medidas para evaluar las recomendaciones entregadas por estos algoritmos.

4. Evaluar los algoritmos en una base de datos reales de la red social Twitter.

Capítulo 4

Algoritmos

En este Capítulo, presentamos los algoritmos utilizados más adelante para generar recomendaciones de perfiles de usuarios a otros usuarios de la red social. Cada algoritmo entrega una noción general del procedimiento que lleva a cabo para generar las recomendaciones a cada usuario.

4.1. Métodos basados en influencia entre usuarios

Generamos un método sencillo para dar recomendaciones en la red social, basándonos en la cantidad de perfiles en común entre dos usuarios. Acá utilizamos la matriz $B(\mathcal{W})$ definida en la Sección 2.4, pues, hacemos una distinción entre tener cierta cantidad de perfiles de *amigos en común* y la *razón* o *proporción* que estos representan sobre el total de perfiles que un usuario particular tiene en su lista de *amigos*. Si lo vemos de esta forma, un usuario i y un usuario j tienen la misma cantidad de perfiles en común, sin embargo, i puede tener más *amigos* en la red social que j, por lo que el generar una recomendación a partir de estos amigos en común tendrá mayor peso para j que tiene menos amigos, dado que la cantidad de amigos en común sobre el total actual de amigos es mayor para j que para i. Así diremos que i *influye* más sobre j que j sobre i, o también, i será más *relevante* para j que j para i. Cada algoritmo que presentamos recibe entre sus inputs, o variables de entrada, una matriz B de tamaño $m \times m$ que representa a la matriz $B(\mathcal{W})$ y cuyas entradas $b_{i,j}$ indica la influencia o relevancia del usuario j sobre i, además, cada fila i indica la influencia de los m usuarios sobre el usuario i-ésimo, esta fila caracterizará al usuario i en la matriz.
Por otro lado, hemos generado 3 formas distintas de determinar la cantidad

de usuarios que estarán en la recomendación de un usuario particular i. A continuación, explicamos estas formas de determinar el tamaño de la recomendación, junto con el detalle de los algoritmos.

4.1.1. Matriz influencia y recomendaciones determinados por umbral

El Algoritmo 1 recibe como variables de entrada la matriz B, un índice i que indica la fila en la que está indexado el usuario al que le realizaremos la recomendación y una constante umbral α que nos indicará sobre qué valor un usuario se considera relevante para el usuario al que se le quiere entregar la recomendación. El output, o variable de salida, es un conjunto v que contiene los usuarios más relevantes para el usuario al que se le quiere dar la recomendación segun el umbral entregado. Cabe destacar que las recomendaciones correspondientes a cada usuario i pueden ser de distintos tamaños.
En este algoritmo comparamos cada entrada de la fila i-ésima de la matriz con el umbral, si la entrada $b_{i,j}$ es mayor o igual al umbral α, entonces, el usuario j formará parte de la recomendación del usuario i. Para el usuario i (indexado en la fila i-ésima de matriz), almacenamos en una variable v los valores de j tales que $b_{i,j} \geq \alpha$. El conjunto v es la variable de salida.

Algoritmo 1: *Normalizada_umbral*(B,i, α)

input : Una matriz B de tamaño $m \times m$ con entradas $b_{i,j}$, un número natural i entre 1 y m, y una constante α.
output: Un conjunto v.

```
v ← ∅;
for j ← 1 to m do
    if b_{i,j} ≥ α then
        v = j ∪ v
    end
end
return v
```

4.1.2. Matriz influencia y recomendaciones de igual tamaño

El Algoritmo 2 recibe como variables de entrada la matriz B antes descrita, un número natural i entre 1 y m que indica la fila en la cual está indexado el usuario al que le realizaremos la recomendación y una constante k que indica

la cantidad de usuarios que formarán parte de la recomendación entregada a cada dicho usuario. El output, o variable de salida, es un conjunto v con los k usuarios más influyentes para el usuario indexado en la fila i-ésima de matriz de la red social. Para el usuario i se siguen los siguientes pasos:

1. Reordenar la fila i-ésima en orden decreciente
2. Guardar en un vector las posiciones que le correspondía en la fila original a cada entrada reordenada.
3. Escoger las primeras k entradas del vector y guardarlos en la variable v. Este será el conjunto de usuarios que se recomendará al usuario.

Algoritmo 2: *Normalizada_k*(B,i,k)

input : Una matriz B de tamaño $m \times m$, un número natural i entre 1 y m, y una constante k.
output: Un conjunto v.

```
1 v ← ∅ ;
2 Ordenar la fila i-ésima de B de mayor a menor.;
3 Guardar en pos las posiciones de la columna de la matriz en que se
    encontraba inicialmente cada valor ya ordenado.;
4 v ← pos[1 : k];
5 return v
```

4.1.3. Matriz influencia y tamaño de las recomendaciones a partir de un porcentaje de amigos actuales

En esta variante consideraramos información extra respecto a los usuarios de la red social. Para este caso se requiere conocer el total de usuarios en la lista de amigos actual de cada individuo indexado en la matriz y utilizaremos dicho dato para determinar una cantidad de recomendaciones personalizada para cada usuario. Por ejemplo, consideremos una proporción $\beta = 0{,}2$ y dos usuarios A y B, con un total 200 y 100 *amigos* en su red social, respectivamente, entonces, el tamaño del vecindario para A será $\beta \cdot 200 = 40$ y el de B será $\beta \cdot 100 = 20$; esto quiere decir que consideraremos el 20 % de los *amigos* que tiene un usuario como tamaño de referencia para generar un vecindario. Vale la pena considerar este tipo de detalles al generar una recomendación ya que un usuario con pocos perfiles de *amigos* puede ser

indicio de un usuario que no requiere numerosas recomendaciones de perfiles nuevos.

Luego, las variables de entrada del Algoritmo 3 son: la matriz B, un valor natural i que indica la fila en la que se encuentra indexado el usuario al que le realizaremos la recomendación, una constante t que contiene el total de perfiles que sigue el usuario indexado en la i-ésima fila de la matriz y una constante β que indica la proporción sobre el total de *amigos* que se considerará para generar el tamaño de la recomendación del usuario. El output es, nuevamente, un conjunto v que contiene las recomendaciones determinadas para el usuario i. El procedimiento es similar al realizado en el Algoritmo 2, sin embargo, acá k es distinto para cada usuarios, y está determinado por $\beta \cdot t$.

Primero, determinamos el tamaño de la recomendación dada por $k \leftarrow round(\beta \cdot t)$ (*round* redondea una constante). Luego, hacemos lo siguiente:

1. Reordenar la fila i-ésima en orden decreciente

2. Guardar en un vector denominado *pos* las posiciones que le correspondía en la fila original a cada entrada reordenada.

3. Escoger las primeras k entradas del vector *pos* y guardarlos en la variable v. Estos serán los perfiles recomendados al usuario i.

Algoritmo 3: *Normalizada_porcentaje*(B, i, t, β)

input : Una matriz B tamaño $m \times m$, un número natural i entre 1 y m ,una constante t y una constante β.

output: Un conjunto v.

1 $round(c)$ entrega el valor entero más próximo a la constante c ;
2 $v \leftarrow \emptyset$;
3 $k \leftarrow round(\beta \cdot t)$;
4 Ordenar la fila i-ésima de B de mayor a menor.;
5 Guardar en *pos* las posiciones de la columna de la matriz en que se encontraba inicialmente cada valor ya ordenado;
6 $v \leftarrow pos[1 : k]$;
7 **return** v

4.2. Métodos basados en SFS (orden de Robinson)

En los capítulos anteriores hemos abordado las matrices de Robinson, orden de Robinson y el problema del ordenamiento, entre otros conceptos afines. En esta sección, articulamos estos temas para formular y plantear un algoritmo de recomendación a partir de ellos.

Nuestro objetivo es entregar una recomendación a un usuario de la red social, es intuitivo pensar que si ordenamos nuestro conjunto de usuarios posicionando más cerca entre sí a aquellos más similares, entonces podríamos generar una recomendación para un usuario a partir de este orden. Por lo tanto, planteamos el problema del ordenamiento de nuestra red social y trabajamos con una matriz de similaridad entre usuarios de la red para identificar si esta es una matriz Robinson. A partir de la matriz simétrica $S(\mathcal{W})$ definida en la Sección 2.4, aplicamos el algoritmo *SFS* presentado en la Sección 3.1 para identificar si una matriz es Robinsoniana o no. Notemos que en el caso de que la matriz analizada sea Robinsoniana, se requiere determinar el orden de Robinson que nos permita transformar la matriz original a una matriz Robinson. Para nuestra tranquilidad, los avances en los estudios de matrices Robinsonianas han aportado algoritmos que no sólo identifican si la matriz puede o no ser transformada a una matriz Robinson, sino que además, nos entregan el orden de Robinson para dicha matriz. Además, cuando la matriz no es Robinsoniana, la librería *SFS* de *R-studio* entrega un orden que hace que la matriz sea lo más parecida a una matriz de Robinson.

En lo que sigue de esta Sección, presentamos dos métodos para determinar las recomendaciones de usuarios de una red social a partir de un orden entregado por la función SFS (ya sea si el orden es Robinson o no) al aplicarlo sobre una matriz de similitud simétrica de tamaño $m \times m$. Para este caso consideramos la matriz $S(\mathcal{W})$ definida en la Sección 2.4 y modificamos su diagonal por el valor de su entrada más alta (similar a lo hecho en la Sección 2 para la matriz de pesos aumentadas) con el fin de facilitar el desarrollo de la función SFS. Las dos algoritmos que presentamos difieren en que el primero nos entrega recomendaciones de tamaños iguales para cada usuario y el segundo entrega recomendaciones de tamaños distintos y personalizados en base a la lista actual del usuario.

4.2.1. Orden de Robinson y recomendaciones de igual tamaño

El Algoritmo 4 tiene como variables de entrada una matriz M simétrica de tamaño $m \times m$, un número natural i entre 1 y m, y una constante k mayor o igual a 0. Las entradas $m_{i,j}$ de la matriz M representan los perfiles en común entre el usuario i y el usuario j, el número i indicará la fila en la cual está indexado el usuario al que le realizaremos la recomendación y la variable k indica el tamaño de las recomendaciones a generar para cada usuario. La variable de salida será, un conjunto v que contiene los perfiles de usuarios que se recomiendan al usuario i indexado en la matriz. Con ayuda de la función *SFS* obtenemos el orden de la matriz M que la hace lo más parecida a una matriz Robinson. Luego, se determinan los k individuos de la red social más cercanos para el usuario i en el orden entregado por la función SFS. Para ello primero se identifica la posición en que está i en el orden entregado por la función SFS y luego se escogen los k más cercanos a dicha posición. Por ejemplo, si tenemos 5 usuarios a, b, c, d y e, que forman la matriz M en ese orden originalmente; luego, al aplicar la función SFS obtenemos el orden b d a c e, entonces los dos usuarios más cercanos a a serán d y c.

Algoritmo 4: *Robinson_k*(M, i, k)

input : Una matriz M de tamaño $m \times m$, un número natural i entre 1 y m, y una constante k.
output: Un conjunto v.

1 $diag(M)$ es una función que escoge los elementos en la diagonal de una matriz
2 $SFS(M)$ entrega, si existe, un vector ord que corresponde al orden de Robinson de las filas de M. Si no existe, entrega un orden que la hace lo más parecida a una matriz Robinson
3 $diag(M) \leftarrow$ máx(M);
4 $ord \leftarrow SFS(M)$;
5 $v \leftarrow \emptyset$;
6 Determinar los k usuarios más cercanos a i de acuerdo con el orden dado por ord y adjuntarlos en v;
7 **return** v

4.2.2. Orden de Robinson y recomendaciones de distintos tamaño

Para el Algoritmo 5 hemos considerado la técnica usada en el Algoritmo 3 para generar tamaños de recomendaciones a cada usuario de forma personalizada, considerando una proporción de la cantidad de perfiles de *amigos* que tienen actualmente. El algoritmo tiene como variables de entrada una matriz M simétrica de tamaño $m \times m$ donde sus entradas $m_{i,j}$ representan los perfiles en común entre el usuario i y el usuario j, un número natural i que nos indica la fila en la que está indexado el usuario al que le realizamos la recomendación, una constante t con el total de los perfiles de *amigos* actuales del usuario i indexado en la matriz y una proporción β. La variable de salida será el conjunto v que contiene las recomendaciones para el usuarios. De igual forma que para el método anterior, con la función *SFS* obtenemos el orden de la matriz M que la hace lo más parecida a una matriz Robinson. Definimos el vector $k = round(\beta \cdot t)$ que indica el tamaño de la recomendación para el usuario, y en el orden determinado con *SFS*, identificamos la posición en la que se encuentra el usuario i al que le realizamos la recomendación y luego escogemos los k usuarios más cercanos a esta posición en el orden entregado por la función SFS. **Ojo, i es la fila en el orden original de la matriz entregada al algoritmo y no en el orden que obtendremos con la función SFS.**

Algoritmo 5: *Robinson_porcentaje*(M, i, t , β)

input : Una matriz simétrica M de tamaño $m \times m$, un número natural i entre 1 y m, una constante t y una constante β.
output: Un conjunto v.

1 $diag(M)$ es una función que escoge los elementos en la diagonal de una matriz
2 $SFS(M)$ entrega, si existe, un vector ord que corresponde al orden de Robinson de las filas de M. Si no existe, entrega un orden que la hace lo más parecida a una matriz Robinson
3 $diag(M) \leftarrow$ máx(M);
4 $ord \leftarrow SFS(M)$;
5 $v \leftarrow \emptyset$;
6 $k \leftarrow round(\beta \cdot t)$;
7 Determinar los k usuarios más similares a i de acuerdo con el orden dado por ord y adjuntarlos a v;
8 **return** v

Algoritmo 6: *k-means_vecinos*(M, i, k)

input : Una matriz simétrica M de tamaño $m \times m$, un número natural i entre 1 y m y una constante k.
output: Un conjunto v.

1 $kmeans(M, k)$ entrega una lista con el número de cluster al que pertenece cada elemento i indexado a la matriz.
2 $v \leftarrow \emptyset$;
3 $Cluster \leftarrow kmeans(M, k)$;
4 Identificar en $Cluster$ los usuarios que pertenecen al mismo cluster que el usuario i;
5 Guardar las filas identificadas en v;
6 **return** v

4.3. Método basado en clusters de *K-means*

Como hemos mencionado anteriormente, *K-means* es un método que nos permite separar o dividir un conjunto en k grupos o *clusters*. Usando esta premisa sobre el algoritmo, nos aventuramos a determinar grupos para nuestra red social a partir de él. En este caso entregamos como variable de entrada la matriz simétrica M de tamaño $m \times m$, un número natural i que indica la fila en la cual está indexado el usuario al cual le realizamos la recomendación y una constante k que indica la cantidad de *clusters* o conjuntos en los que dividiremos nuestro conjunto de usuarios de la red social. La variable de salida será un conjunto v que contiene las recomendaciones determinadas para el usuario i. Para que la aplicación de este método tenga sentido en nuestro contexto, la matriz M será la matriz simétrica $S(\mathcal{W})$ que hemos definido previamente en la Sección 2.4. En la Subsección 5.2.2, explicamos como determinar el valor más óptimo de k, ya que a priori, no tenemos conocimiento de cuál es el valor más apropiado de grupos en que debemos dividir el conjunto de usuarios. Por último, la función *k-means* entrega una lista con el número de cluster al cual pertenese el i-ésimo elemento indexado en la matriz, en nuestro caso este corresponde al usuario de la red social, por lo que para generar la recomendación para dicho usuario, escogemos en esta lista todos los otros usuarios que están en el mismo cluster. Cabe mencionar que este proceso puede genera grupos de distintos tamaños, por lo que las recomendaciones no serán necesariamente de igual tamaño.

Capítulo 5

Experimentos y Resultados

En este capítulo presentamos los experimentos realizados sobre un conjunto de datos de perfiles de una red social, sobre el cual hemos aplicado los algoritmos descritos en el Capítulo 4. Además, presentamos los resultados obtenidos.

5.1. Base de datos

Una base de datos corresponde a un conjunto de datos de un determinado contexto que se encuentran organizados y almacenados, generalmente en un sistema electrónico, para su posterior uso. Para este trabajo, hemos buscado y utilizado una base de datos de *Twitter*, en la que se incorpora información sobre usuarios de la red y en particular, la lista de *amigos* de cada usuario, la cual es fundamental para aplicar los algoritmos propuestos.

5.1.1. Descripción de los datos

En esta ocasión hemos obtenido una base de datos de la red social *Twitter* a través del sitio web *Kaggle* [27]. Esta base de datos consiste en una tabla de 40000 filas y 10 columnas, donde cada fila corresponde a datos de un usuario de la red social y las columnas corresponden información en la red social como: nombre del usuario en Twitter, un *ID* o número de identificación, total de seguidores del usuario, lista de *amigos* de cada usuario identificado por el *ID*, *tag* (etiquetas en publicaciones), entre otros. La variables de interés en esta base de datos serán: el nombre de usuario, el ID, la lista de amigos y los *tags*. También es válido pensar que cada usuario puede estar caracterizado directamente por la lista de *amigos* que le corresponde. Esta lista será el dato

	id	screenName	tags	lang	lastSeen	tweetId	friends
1	1.969528e+09	LIngoMakeEmCum_	[#nationaldogday]	en	1.472272e+12	7.693107e+17	[1969574754;1969295556;1969284056;1969612214;197006...
2	5.187849e+07	_notmichelle	[#nationaldogday]	en	1.472271e+12	7.693095e+17	[60789485;2420931980;2899776756;127410795;38747286;1...
3	1.393409e+09	jesseayye	[#narcos]	en	1.472804e+12	7.716226e+17	[86868062;19697415;2998836604;456295047;74931377241...
4	2.328914e+08	MrBrianLloyd	[#gloryoutnow]	en	1.472269e+12	7.693081e+17	[361335082;1405248468;24626354;725675895965552640;4...
5	7.101304e+17	sarahdorat_16	[#nationaldogday]	en	1.472271e+12	7.693098e+17	[1571896093;768938323612008448;2548665930;33256647...
6	3.649470e+09	wanderlustregui	[#veranomtv2016]	en	1.472737e+12	7.713409e+17	[2401096388;707864762;4096348512;76385585993189785...

Figura 5.1: En la figura vemos un extracto de la base de datos utilizada en este estudio

principal que utilizaremos. Además, un dato importante de mencionar es que el total de perfiles únicos identificados entre todas las listas de amigos son alrededor de 13 millones, indicio de la gran cantidad de usuarios que posee esta red social. En la Figura 5.1 vemos un extracto de la tabla que contiene la base de datos de la red social.
Para trabajar con los datos y algoritmos presentados en el Capítulo 4 hemos utilizado el software *R-studio* y algunas de sus librerías disponibles. Para la preparación de los datos utilizamos la librería *stringr*.
Primero, para utilizar los algoritmos propuestos, es indispensable generar las matrices $S(\mathcal{W})$ y $B(\mathcal{W})$ definidas en el Capítulo 2, para lo cual debemos contabilizar la cantidad de amigos en común entre dos usuarios de la red social. La base de datos que disponemos almacena la lista completa de amigos de cada usuario en formato de texto (o *string*), formato que no permite al computador identificar cuáles usuarios tienen coincidencias en sus listas, por lo tanto, luego de extraer la columna de la tabla original que almacena las listas de amigos para cada usuario, pasamos a trabajar con distintas funciones de la librería *stringr* para individualizar cada perfil de la lista de amigos por usuario, generando variables de tipo *listas* que almacenan dicha información. Este proceso para los 40000 usuarios tardó $376,04$ segundos. Posteriormente, teniendo estos datos, comparamos las listas de amigos de dos usuarios, determinamos el total de amigos en común entre ellos y obtenemos el total de amigos de cada usuario, datos con los cuales generaremos las matrices $S(\mathcal{W})$ y $B(\mathcal{W})$ para cualquier conjunto de usuarios $\mathcal{W}$ de la red social.

5.2. Experimentos Realizados

En esta Sección presentamos las consideraciones realizadas al aplicar los algoritmos y describimos los procesos realizados. Importante considerar que estos procedimientos fueron programados en el software *R-studio* y ejecu-

tados en un computador portatil con procesador AMD A6-9225, RAM de 4GB.

5.2.1. Construcción de las matrices

Como mencionamos antes, un punto clave de este procedimiento es construir a partir de los datos adjuntos en la base de datos las matrices $S(\mathcal{W})$ y $B(\mathcal{W})$. Para ello, dada la capacidad computacional limitada de nuestro computador, decidimos generar 20 sub-redes aleatorias de usuarios de la red social, a cada una de ellas las llamamos $\mathcal{W}$, y sobre dicho conjunto construimos las matrices requeridas.
Para hacer más eficiente el procedimiento, se ha obtenido en primera instancia la matriz $S(\mathcal{W})$. Dado que esta matriz es simétrica, entonces la entrada $S_{\mathbf{UV}}$ que indica la cantidad de amigos en común entre el usuario $\mathbf{U}$ y el usuario $\mathbf{V}$ es igual a la entrada $S_{\mathbf{VU}}$, por lo que sólo se ha calculado la sección sobre la diagonal de la matriz (matriz triangular superior) y luego reflejamos esos valores en la parte inferior de la matriz. Para escribir la matriz $B(\mathcal{W})$ utilizamos la matriz $S(\mathcal{W})$ y como cada fila contiene la información correspondiente a un usuario $\mathbf{U}$, y de acuerdo con la definición dada en la Sección 2.4 para nuestra matriz de interés, multiplicamos cada fila indexada por el usuario $\mathbf{U}$ en la matriz $S(\mathcal{W})$ por la constante $1/d(\mathbf{U}, \mathbf{U})$ y así obtenemos la matriz $B(\mathcal{W})$.
Una vez construida la matriz $B(\mathcal{W})$ para cada sub-red definida, obtuvimos su histograma. Para analizar nuestra red original, seleccionamos 4 gráficos de distintas sub-redes para observar la tendencia de los valores en la matriz (ver figuras 5.2, 5.3, 5.4 y 5.5). En los gráficos, apreciamos una característica que se repite en estos y los gráficos de las sub-redes que omitimos, esta indica que de los 2,25 millones de entradas que tiene la matriz, entre 2,1 y 2,2 millones corresponden a valores entre 0 y 0,05, notemos que en todas estas matrices existirán por lo menos 1500 entradas iguales a 1, las cuales corresponden a los elementos de la diagonal, sin embargo, los histogramas muestran que existen valores cercanos pero distintos de 1, lo que indica que existen usuarios en las sub-redes que comparten gran parte de su lista de amigos con otro usuario estudiado en la misma. También vemos que los valores de la matriz se concentran entre 0 y $0,25$.

5.2.2. Aplicación de los algoritmos

Una vez generadas las matrices $S(\mathcal{W})$ y $B(\mathcal{W})$, siguiendo los algoritmos presentados en el Capítulo 4, escribimos computacionalmente dichos méto-

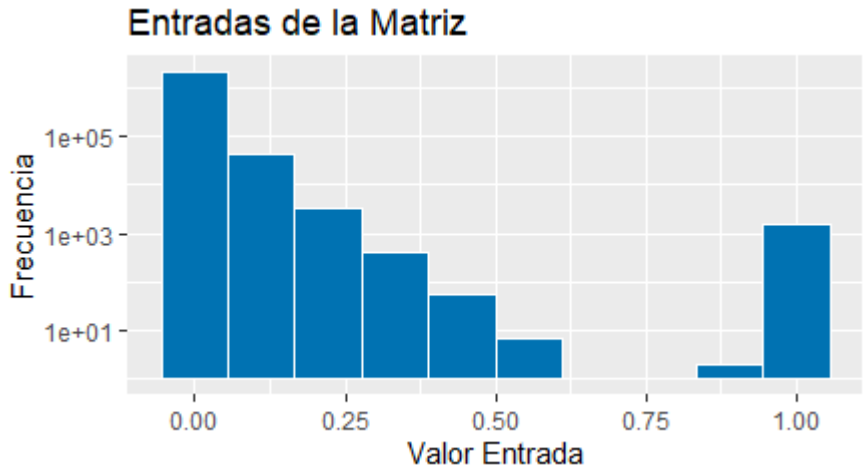


Figura 5.2: Gráfico de histograma de la matriz $B(\mathcal{W})$ de la sub-red 1.

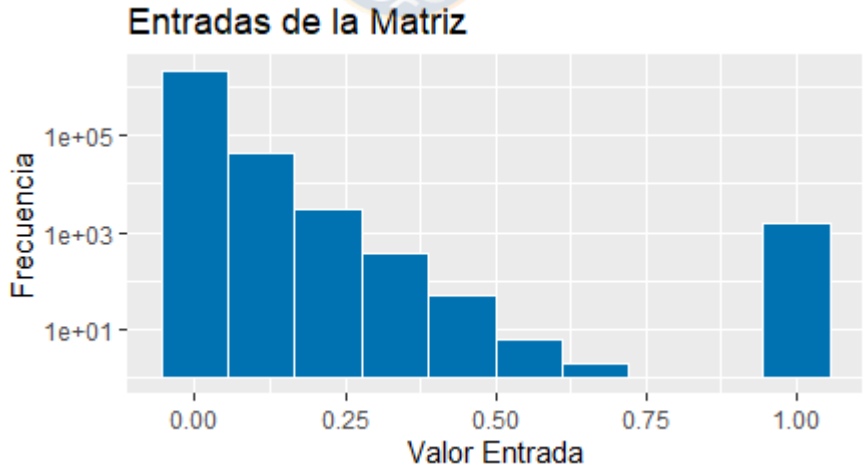


Figura 5.3: Gráfico de histograma de la matriz $B(\mathcal{W})$ de la sub-red 10.

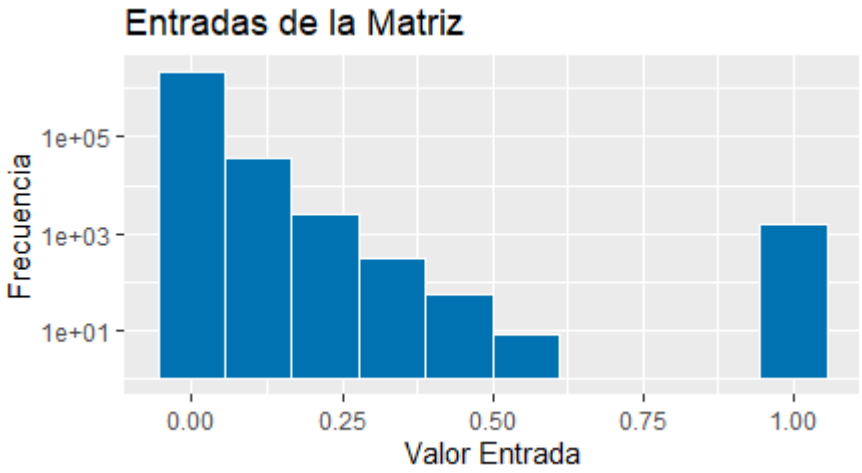


Figura 5.4: Gráfico de histograma de la matriz $B(\mathcal{W})$ de la sub-red 15.

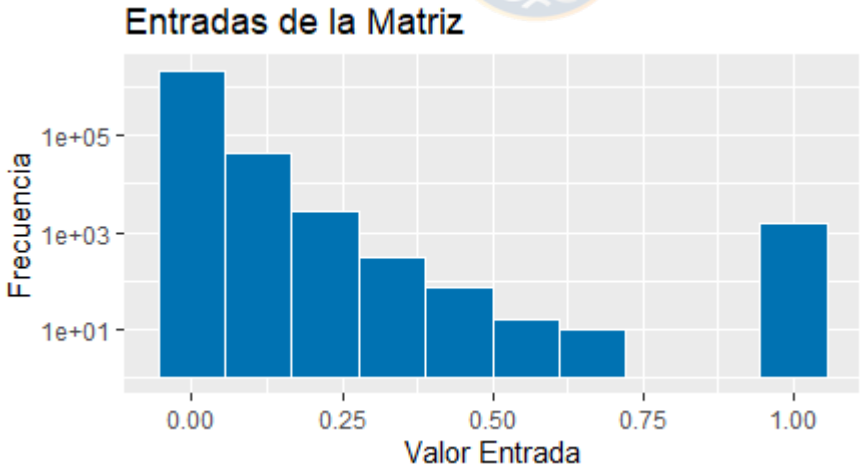


Figura 5.5: Gráfico de histograma de la matriz $B(\mathcal{W})$ de la sub-red 20.

dos. Para cada uno de ellos se consideraron ciertas especificaciones que se enumeran a continuación:

1. **Métodos con matriz de influencia**: comparamos las entradas de cada fila con el umbral entregado. Es clave la utilización de las funciones *which* para poder encontrar eficientemente las posiciones de los elementos en su orden original.

2. **Métodos con matrices Robinson**: acá utilizamos las funciones *SFS::read* y *SFS::sfs* de la librería SFS implementada por *R-studio* para obtener un orden de Robinson, o uno cercano a él, que hará que la matriz en estudio sea una matriz Robinson o lo más similar a una matriz Robinson. La función *SFS::read* recibe como entrada una matriz simétrica y la transforma en una lista de 3 elementos donde el primer valor indica una fila, el segundo la columna y el último el valor de dicha componente en la matriz. Esta función sólo considera las posiciones de los elementos con entradas no nulas en la matriz, lo cual servirá para optimizar el proceso realizado por la función *SFS::sfs* la cual nos entregará finalmente un vector que representa la permutación de filas y columnas que se debe hacer simultaneamente para obtener la matriz Robinson en caso de existir, o bien, la permutación que hace que la matriz sea lo más similar a una matriz Robinson.

3. **Método basado en clusters de *K-means***: para aplicar este método, primero se deben determinar los *cluster de K-means*. Utilizando la función *kmeans* de la librería *stats* podemos determinar sin mayor complicaciones los k clusters que genera dicho algoritmo para una matriz y valor de k dados. No obstante, como se ha anticipado en el Capítulo 4, se debe determinar empíricamente el valor óptimo de k. Para ello utilizaremos el conocido *método del codo*, el cual consiste en calcular los clusters para ciertos k dados y obtener el valor de la suma de cuadrados interna de los clusters (wcss) generados con cada valor de k y luego graficar k v/s wcss; al observar el gráfico con dichos valores, se espera que en la curva se genere "un codo"que muestra el punto en que los valores de wcss decrecen con rapidez al aumentar k y cuando se estabiliza este valor y no cambia notablemente al variar el valor de k. El punto donde se produce el codo será el valor óptimo de cluster que escogeremos para aplicar el algoritmo.

Cabe mencionar que ejecutamos cada algoritmo para 20 sub-redes conformada por una muestra de 1500 usuarios escogidos entre los 40000 usuarios

de la red social. Destacamos que cada sub-red es distinta de la otra y no repite usuarios.

5.2.3. Cálculo de las medidas

Las medidas presentadas en la Sección 2.5 fueron programadas en el mismo software usando herramientas básicas de cómputo como lo son: los ciclos *for*, accesos a elementos de la matriz, función suma, entre otras. Utilizando las funciones y la matriz $B(\mathcal{W})$, calculamos para cada sub-red los valores promedios del corte normalizado, promedio de las conductancias de los conjuntos de corte generados por las recomendaciones a cada usuario y el promedio de la expansión de $R_{\mathbf{V}} \cup \{\mathbf{V}\}$. Además, obtuvimos e identificamos características en las redes sociales que nos permitirán sacar conclusiones cualitativas de las recomendaciones generadas por los métodos, entre estas consideramos la cantidad de usuarios que no obtuvieron recomendación usando cada método, análisis de los tags de cada usuario recomendado y las coincidencias entre las recomendaciones entregadas a los usuarios por cada método.

5.3. Resultados

En esta sección presentamos los resultados obtenidos con el fin de comparar y determinar si los métodos propuestos son alternativas válidas a la hora de generar recomendaciones en redes sociales. En las tablas de resultados que presentamos más adelante, se utiliza como notación:

- Algoritmo 1 (M1-A): recomendaciones generardas a partir de matriz de influencia y umbral = 0.04, es decir, usuarios que presentan una proporción mayor o igual a 0.04 de amigos en común sobre el total de perfiles en la lista de amigos del usuario al que se le realiza la recomendación.

- Algoritmo 2 (M1-B): recomendaciones generadas a partir de matriz de influencia y k usuarios más influyentes, donde k es escogido como el 5 % del total de usuarios que evaluamos (n=1500). Los vecindarios serán de igual tamaño.

- Algoritmo 3 (M1-C): recomendaciones generadas a partir de matriz de influencia y k usuarios más influyentes, donde k es igual al 10 % del total de usuarios en la lista de amigos del individuo al que se le quiere dar la recomendación. Los vecindarios serán de distinto tamaño.

- Algoritmo 4 (M2-A): recomendaciones generadas a partir del orden de Robinson o cercano a orden de Robinson entregado por función SFS y k usuarios más cercanos en el orden obtenido, donde k es escogido como el 5 % del total de usuarios que evaluamos (n=1500). Las recomendaciones serán de igual tamaño.
- Algoritmo 5 (M2-B): recomendaciones generadas a partir del orden de Robinson o cercano a orden de Robinson entregado por función SFS y k usuarios más cercanos en el orden obtenido, donde k es igual al 10 % del total de usuarios en la lista de amigos del individuo al que se le quiere dar la recomendación. Las recomendaciones obtenidas serán de distintos tamaños.
- Algoritmo 6 (K-means): recomendaciones obtenidas a partir de los grupos generados al aplicar k-means.

5.3.1. Resultados promedio de conductancia

Utilizando la fórmula de promedio de conductancia presentada en la Definición 10, calculamos para cada sub-red el valor de esta medida, obteniendo los resultados presentados en la Tabla 5.1. En dicha tabla vemos que los valores obtenidos son similares en todos los métodos propuestos, siendo *k-means* el que presenta la más baja.
Como propusimos en capítulos anteriores, valores bajos de la conductancia indican que el algoritmo es bueno, pues la suma de pesos de las aristas en el corte es menor que la suma de todas las aristas que tienen un extremo en el conjunto de recomendaciones, es decir, las aristas de mayor peso, que en nuestro caso indican mayor similitud entre los usuarios, se concentran entre los usuarios recomendados y no fuera de la recomendación. Si escogemos un método basado sólo en esta métrica, escogeríamos *k-means*, por ser el que posee menor conductancia. Sin embargo, en los resultados que presentamos en la Subsección 5.3.7, veremos que *k-means* tiene la conductancia más baja, pues agrupó a la mayoría de los usuarios de las sub-redes en pocos clusters, por lo que la cantidad de aristas en los cortes generados por las recomendaciones son mucho menor a las que no son parte del corte, lo que hace que la proporción disminuya.

5.3.2. Resultados promedios de la expansión de la recomendación

La Tabla 5.2 contiene los resultados del promedio de la expansión de $R_{\mathbf{V}} \cup \{\mathbf{V}\}$ generado por las recomendaciones dadas a cada usuario de las 20 sub-

Sub-red	M1-A	M1-B	M1-C	M2-A	M2-B	K-means
1	0.66	0.72	0.75	0.78	0.77	0.38
2	0.67	0.72	0.74	0.81	0.80	0.38
3	0.66	0.72	0.74	0.82	0.81	0.57
4	0.67	0.72	0.74	0.80	0.79	0.50
5	0.67	0.73	0.75	0.79	0.78	0.61
6	0.66	0.72	0.74	0.81	0.79	0.57
7	0.67	0.72	0.74	0.83	0.80	0.64
8	0.67	0.72	0.74	0.80	0.79	0.61
9	0.69	0.73	0.75	0.82	0.80	0.60
10	0.67	0.72	0.74	0.81	0.80	0.62
11	0.67	0.72	0.74	0.80	0.79	0.63
12	0.67	0.72	0.74	0.83	0.81	0.64
13	0.68	0.73	0.75	0.82	0.80	0.62
14	0.67	0.72	0.75	0.79	0.79	0.62
15	0.68	0.72	0.74	0.79	0.78	0.59
16	0.67	0.73	0.75	0.84	0.82	0.68
17	0.67	0.72	0.74	0.78	0.77	0.62
18	0.67	0.71	0.73	0.79	0.77	0.52
19	0.68	0.73	0.75	0.79	0.78	0.55
20	0.67	0.72	0.75	0.82	0.80	0.64
Promedio	0.67	0.72	0.74	0.81	0.79	0.58

Tabla 5.1: Resultados del promedio de conductancia para cada sub-red generada.

redes. En promedio para estas 20 sub-redes, obtuvimos valores entre 1.6 y 5.46, siendo k-means el método que presenta la más baja y los método basado en orden de Robinson los que tienen los más altos.
Esta métrica indica la proporción de la suma de la función de similitud entre usuarios recomendados y los que no han sido recomendados, o suma de los pesos de aristas del corte generado por la recomendación, sobre la suma de la función de similitud entre los usuarios que sí fueron recomendados. Valores bajos de esta métrica, indican que no es necesario *expandir* nuestro conjunto de recomendación ya que los usuarios con mayores similitudes se encuentran en la recomendación. Si los valores son altos, entonces podríamos expandir nuestro conjunto de recomendación incorporando usuarios que quedaron fuera para lograr aumentar la suma de la función de similitud entre usuarios de la recomendación, y a la vez, disminuir el valor de la suma de la función de similitud en el corte. En adición a lo anterior, si el algoritmo presenta valor

Sub-red	M1-A	M1-B	M1-C	M2-A	M2-B	K-means
1	3.39	2.78	3.96	4.51	5.14	0.62
2	3.33	2.67	3.85	4.94	5.40	1.56
3	3.37	2.76	3.86	5.49	6.07	1.42
4	3.33	2.71	3.85	4.88	5.44	1.08
5	3.49	2.78	3.98	4.59	5.26	1.66
6	3.22	2.71	3.78	4.95	5.39	1.45
7	3.34	2.66	3.76	5.17	5.52	1.91
8	3.45	2.76	3.90	5.01	5.51	1.64
9	3.65	2.85	4.02	5.22	5.65	1.61
10	3.40	2.75	3.81	4.89	5.43	1.78
11	3.27	2.71	3.88	4.60	5.10	1.78
12	3.43	2.65	3.82	5.52	6.23	1.86
13	3.44	2.80	3.95	5.25	5.64	1.71
14	3.42	2.77	4.00	4.56	5.33	1.81
15	3.41	2.69	3.86	4.50	5.13	1.57
16	3.47	2.86	4.08	5.66	6.41	2.32
17	3.35	2.71	3.87	4.33	4.92	1.73
18	3.37	2.59	3.53	4.50	4.89	1.19
19	3.48	2.80	4.03	4.45	5.05	1.29
20	3.35	2.77	3.96	5.25	5.77	1.97
Promedio	3.40	2.74	3.89	4.91	5.46	1.60

Tabla 5.2: Promedios de la expansión de $R_{\mathbf{V}} \cup \{\mathbf{V}\}$ obtenidas para cada método y para las 20 sub-redes generadas.

bajo en esta métrica, también podría significar que se estén incorporando muchos usuarios en la recomendación, lo que se traduce en tener mayor suma de la función de similitud entre usuarios de la recomendación porque la mayoría de usuarios de la sub-red se encuentra en ella, lo que tampoco es deseable. Por lo que esta métrica por si sola no nos permitiría escoger el mejor método, pero sí entrega un criterio para complementar los resultados de la conductancia y las métricas que veremos en las subsecciones siguientes.

5.3.3. Resultados promedio del corte normalizado

En la Tabla 5.3 podemos ver que los promedios del corte normalizado obtenidos por los métodos propuestos son más bajos que el alcanzado por el método *k-means*. También se observa que los valores son más variables para este último método (k-means), mientras que los valores obtenidos en los al-

Sub-red	M1-A	M1-B	M1-C	M2-A	M2-B	K-means
1	0.03	0.05	0.04	0.04	0.04	0.26
2	0.03	0.06	0.04	0.04	0.04	0.20
3	0.03	0.05	0.04	0.04	0.04	0.22
4	0.03	0.05	0.04	0.04	0.04	0.26
5	0.04	0.05	0.04	0.04	0.04	0.22
6	0.03	0.05	0.04	0.04	0.04	0.21
7	0.03	0.05	0.04	0.04	0.04	0.18
8	0.03	0.05	0.04	0.04	0.04	0.22
9	0.03	0.06	0.04	0.04	0.04	0.21
10	0.03	0.05	0.04	0.04	0.04	0.19
11	0.03	0.05	0.04	0.04	0.04	0.20
12	0.04	0.05	0.04	0.04	0.04	0.20
13	0.03	0.06	0.04	0.04	0.04	0.20
14	0.03	0.05	0.04	0.04	0.04	0.21
15	0.03	0.05	0.04	0.04	0.04	0.21
16	0.03	0.05	0.04	0.04	0.04	0.16
17	0.03	0.05	0.04	0.04	0.04	0.20
18	0.03	0.05	0.04	0.04	0.04	0.24
19	0.03	0.05	0.04	0.04	0.04	0.24
20	0.03	0.05	0.04	0.04	0.04	0.20
Promedio	0.03	0.05	0.04	0.04	0.04	0.21

Tabla 5.3: Promedio de corte normalizado obtenido para cada método y las 20 sub-redes generadas.

goritmos 3, 4 y 5 fueron constantes para las 20 sub-redes, esto considerando que los valores indexados en la tabla están aproximados con 2 cifras significativas. El método que presenta menor promedio de corte normalizado es el Algoritmo 1, el que posee la más alta es k-means y es de un orden más alto que otros los 5 métodos propuestos.

Nuevamente, en esta métrica es deseable obtener valores bajos, ya que esto nos indica que la suma de la función de similitud entre los usuarios recomendados y los no recomendados, es más baja que respecto a las similitudes presentes entre todos los usuarios de la red. Es importante considerar que esta métrica presenta valores más bajos que la conductancia porque dividimos por la suma total de la red. Si utilizamos esta métrica, el Algoritmo 1 sería el más recomendable, sin embargo, como se verá en la Subsección 5.3.5, este método dejó cerca del 20 % de usuarios de sin recomendación, lo que se

traduce en tener menos términos en el numerador que aporten a la suma de similitud para este método, en comparación con los otros métodos. Así, los métodos mejor evaluados desde este criterio serían los basados en orden de Robinson. El hecho de que *k-means* tenga los valores más altos, nos indica que dejó usuarios fuera de la recomendación que tienen una similitud alta con los recomendados.
Notemos que esta métrica es más confiable para estudiar el valor la suma de los pesos en el corte que la conductancia, ya que esta no se ve afectada por el término del denominador, el cual es constante en todos los métodos.

5.3.4. Resultados tags usuarios recomendados

Adicional a las métricas definidas en el Capítulo 2 y considerando lo propuesto en el sitio *Kaggle* desde donde se ha obtenido la base de datos de *Twitter*, hemos considerado estudiar los tags utilizados por los usuarios recomendados a cada individuo de las sub-redes, obteniendo el porcentaje de usuarios en la lista de recomendación que han usado el mismo *tag* que el usuario al que se le ha realizado la recomandación. Considere que la base de datos presenta 345 tags distintos y en la mayoría de los casos, tenemos un tag por usuario. Estos resultados se muestran en la Tabla 5.4 y en la misma podemos ver que los porcentajes obtenidos de coincidencia en los tags superan el 50 % para todos los métodos, es decir, más de la mitad de los usuarios recomendados utilizan el mismo tag que el individuo al que son recomendados, *k-means* obtuvo en promedio el porcentaje de coincidencia más alto, alcanzando el 66 %, seguido del Algoritmo 3 con 64 %, el más bajo fue obtenido por el Algoritmo 1.
Adicional a lo anterior, hemos obtenido una recomendación de 73 perfiles (los cuales representan el promedio del tamaño de recomendación obtenido entre los cinco métodos que estamos analizando, sin considerar k-means) para 30.000 usuarios de la red social mediante muestreo aleatorio simple. Luego, comparamos la coincidencia de tags entre estas recomendaciones y cada uno de los 30.000 usuarios a los que se le recomendaron, con lo cual obtuvimos el valor esperado de esta proporción, la cual fue de 0,5872 y obtuvo una desviación de 0,295. De estos resultados, vemos que los porcentajes de coincidencia en tags obtenidos por los métodos presentados son mayores al valor esperado para esta base de datos.
Observemos en particular que para la sub-red 17 se han obtenido los porcentajes más altos de coincidencia, donde los métodos generados a partir del orden de Robinson alcanzaron la proporción más alta de coincidencia en tags, valores que se encuentran dentro del valor esperado y desviación

Sub-red	M1-A	M1-B	M1-C	M2-A	M2-B	K-means
1	0.55	0.64	0.65	0.63	0.64	0.61
2	0.53	0.65	0.65	0.61	0.62	0.66
3	0.55	0.64	0.64	0.62	0.64	0.68
4	0.57	0.66	0.67	0.62	0.63	0.67
5	0.56	0.61	0.62	0.63	0.64	0.65
6	0.52	0.62	0.62	0.61	0.62	0.67
7	0.55	0.64	0.64	0.59	0.60	0.67
8	0.53	0.62	0.62	0.62	0.63	0.65
9	0.53	0.60	0.61	0.58	0.60	0.65
10	0.52	0.60	0.61	0.60	0.60	0.65
11	0.53	0.64	0.64	0.63	0.63	0.65
12	0.53	0.60	0.61	0.57	0.58	0.64
13	0.55	0.63	0.64	0.60	0.61	0.68
14	0.57	0.67	0.68	0.63	0.64	0.68
15	0.54	0.63	0.64	0.64	0.65	0.68
16	0.54	0.61	0.61	0.59	0.59	0.68
17	0.56	0.66	0.66	0.70	0.71	0.69
18	0.57	0.65	0.65	0.63	0.65	0.67
19	0.55	0.65	0.65	0.66	0.67	0.68
20	0.52	0.63	0.63	0.61	0.62	0.69
Promedio	0.54	0.63	0.64	0.62	0.63	0.66

Tabla 5.4: Promedios de proporción de usuarios recomendados con igual tag que el usuario al que se le recomienda sobre la lista completa recomendada en las 20 sub-redes.

estándar obtenida. K-means alcanzó el porcentaje de coincidencia más alto, lo que es esperable, pues como se verá más adelante, este método agrupa en pocos clusters a la mayoría de los usuarios, y de acuerdo con el valor esperado de coincidencia, se tiene una alta probabilidad de coincidir con el tag de otro usuario en las sub-redes, más aún si la recomendación entregada abarca a muchos usuarios.

5.3.5. Resultados usuarios sin recomendación

Otro parámetro importante a considerar al generar las recomendaciones con estos métodos, es la capacidad que cada uno tuvo de dar una recomendación a todos los usuarios comprometidos en la sub-red, esto nos permitirá analizar su efectividad y relacionar los valores obtenidos en conductancia y

Sub-red	M1-A	M1-B	M1-C	M2-A	M2-B	K-means
1	290.00	0.00	0.00	0.00	0.00	17.00
2	328.00	0.00	0.00	0.00	0.00	22.00
3	279.00	0.00	0.00	0.00	0.00	22.00
4	293.00	0.00	0.00	0.00	0.00	23.00
5	250.00	0.00	0.00	0.00	0.00	20.00
6	299.00	0.00	0.00	0.00	0.00	29.00
7	281.00	0.00	0.00	0.00	0.00	38.00
8	285.00	0.00	0.00	0.00	0.00	30.00
9	274.00	0.00	0.00	0.00	0.00	36.00
10	275.00	0.00	0.00	0.00	0.00	32.00
11	331.00	0.00	0.00	0.00	0.00	38.00
12	258.00	0.00	0.00	0.00	0.00	32.00
13	265.00	0.00	0.00	0.00	0.00	43.00
14	292.00	0.00	0.00	0.00	0.00	33.00
15	289.00	0.00	0.00	0.00	0.00	41.00
16	289.00	0.00	0.00	0.00	0.00	32.00
17	312.00	0.00	0.00	0.00	0.00	32.00
18	280.00	0.00	0.00	0.00	0.00	35.00
19	295.00	0.00	0.00	0.00	0.00	40.00
20	299.00	0.00	0.00	0.00	0.00	36.00
Promedio	288.20	0.00	0.00	0.00	0.00	31.55

Tabla 5.5: Usuarios que no obtuvieron recomendación por método en cada sub-red.

expansión de $R_{\mathbf{V}} \cup \{\mathbf{V}\}$, ya que dichos parámetros se ven afectados por los conjuntos de cortes definidos a partir de vértices aislados, es decir, usuarios sin recomendación.

En la Tabla 5.5 tabulamos el total de usuarios que no recibieron recomendación por método y sub-red, obteniendo además el promedio. Acá vemos que el Algoritmo 1 es el que presenta mayor cantidad de usuarios sin recomendación, dejando en promedio 288.2 usuarios sin recomendación sobre un total de 1500, lo que representa un 19,21 % del tamaño de la sub-red sin recomendación. Le sigue *K-means* que dejó en promedio a 31.55 usuarios sin recomendación, esto representa 2,1 % del tamaño de la sub-red. Vemos además que los algoritmos 2, 3, 4 y 5, siempre generan una recomendación, ya que el tamaño del conjunto que se recomienda es dado o será distinto de 0 con las condiciones escogidas.

5.3.6. Comparación entre recomendaciones obtenidas por los métodos

Ahora presentamos los resultados obtenidos al comparar las listas de recomendaciones entregadas por los 6 algoritmos. Para comparar las recomendaciones realizadas a los 30.000 usuarios (1500 por cada sub-red), hemos obtenido los diagramas de Venn para cada uno de ellos y luego calculamos el promedio de las proporciones que obtuvo cada conjunto. Para cada usuario obtuvimos: el diagrama de Venn de las recomendaciones dadas por los algoritmos 1, 2 y 3, también determinamos el diagrama dado para las recomendaciones de los algoritmos 4 y 5; por último, comparamos las recomendaciones realizadas por un representante de los algoritmos basados en la matriz de influencias (algoritmos 1,2 y 3), un representante de los algoritmos basados en el orden de Robinson (algoritmos 4 y 5) y las recomendaciones obtenidas usando *k-means*. Los representantes de los algoritmos basados en la matriz de influencia y orden de Robinson fueron escogidos seleccionando el que aporta mayor cantidad de recomendaciones entre los tres y dos algoritmos que se basan en dichos métodos, respectivamente.

a) **Comparación algoritmos basados en matriz de influencia:** En la Figura 5.6 vemos el promedio de coincidencias que tuvieron las recomendaciones realizadas por los algoritmos basados en la matriz de influencia. Vemos que, en promedio, el 16 % de las recomendaciones entregadas por estos métodos estuvieron en las tres listas. Los algoritmos 2 y 3 tuvieron el la mayor coincidencia de perfiles, alcanzando el 28 %. Por otro lado, dado que el Algoritmo 2 obtuvo el mayor porcentajes solo reocmendados por él y mayor coicidencia con los otros métodos, inferimos que las recomendaciones entragas por dicho método abarcan la mayor cantidad de los perfiles recomendados en su lista.

b) **Comparación algoritmos basados en orden de Robinson:** Para los métodos basados en orden de Robinson obtuvimos el diagrama graficado en la Figura 5.7. En el diagrama vemos que los métodos tienen cerca del 50 % de coincidencia en las recomendaciones. El Algoritmo 3 logra recomendar más del 85 % de los perfiles recomendados entre ambos algoritmos, de lo que inferimos que los k determinados para el Algoritmo 2 en su mayoría fueron más pequeños que el usado en el Algoritmo 1.

c) **Comparación representantes de cada algoritmo con k-means:** En la Figura 5.8 encontramos el promedio de los resultados obtenidos al comparar el método que más recomendaciones entregó entre los méto-

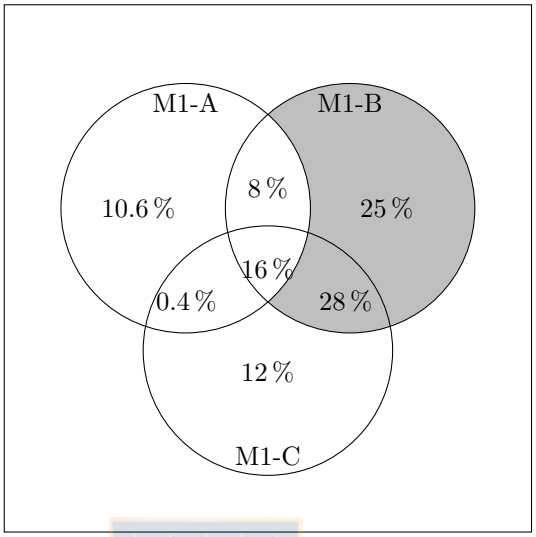


Figura 5.6: Diagrama de Venn para los 3 métodos basados en matriz de influencia

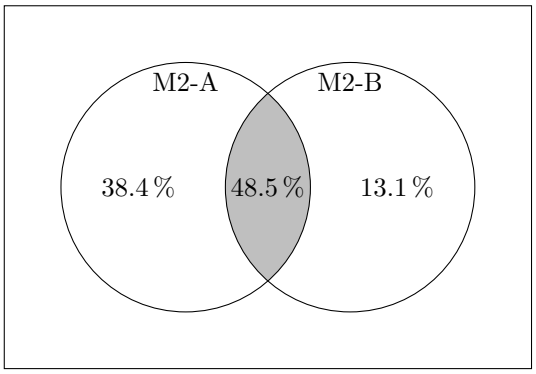


Figura 5.7: Diagrama de Venn para los 2 métodos basados en orden de Robinson

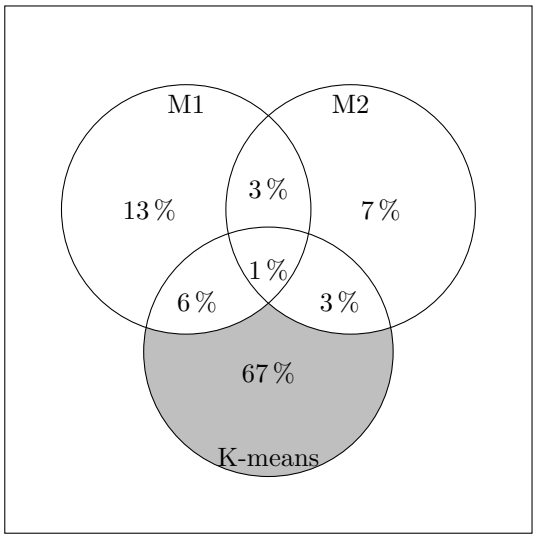


Figura 5.8: Diagrama de Venn comparación de los métodos basado en matriz influencia, orden de Robinson y *k-means*.

dos basados en matriz de influencia, el que entregó más recomendaciones entre los basados en orden de Robinson y *k-means*, para cada usuario. Del diagrama podemos concluir rápidamente que *k-means* es el método que incluye más perfiles en la lista de recomendación de los tres métodos. Sólo el 1 % de los perfiles recomendados se encuentran en las tres listas a la vez. También vemos que las recomendaciones realizadas con los algoritmos 4 y 5 alcanzan la misma cantidad de coincidencia tanto con los algoritmos basados en matriz de influencia como con *k-means*. Por último, se observa que *k-means* comparte más perfiles con los métodos obtenidos a partir de la matriz de influencia que con los métodos basados en orden de Robinson.

5.3.7. Otros resultados

Dentro del desarrollo de los algoritmos y resultados intermedios del procedimiento cabe destacar dos cosas extras a lo que se ha mencionado a lo largo de esta sección. Una de ellas es el tiempo de ejecución promedio obtenido para la función *SFS* de *Rstudio*, este fue de 7428.64 segundos de ejecución, es decir, alrededor de $2,06$ horas sólo para dicho algoritmo, sin embargo, los otros métodos fueron de ejecución mucho más rápida, sin superar los 5 minutos.

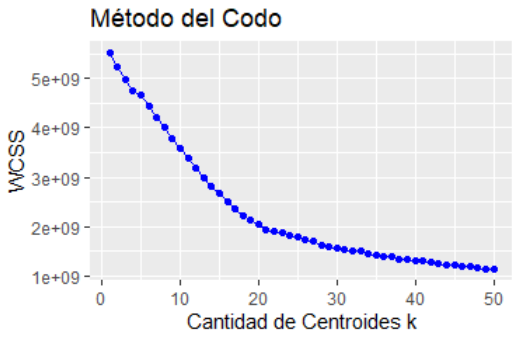


Figura 5.9: En el gráfico se muestran los valores obtenidos para *withinss* usando desde 1 a 50 clusters en k-means aplicado a la matriz de la sub-red 1.

Por otra parte, para determinar la cantidad de cluster en *k-means* a través del método del codo, obtuvimos gráficos como los mostrados en 5.9, 5.10, 5.11 y 5.12. De estos gráficos podemos notar que la suma de cuadrados interna (*withinss*) es del orden $e + 09$, por lo que el punto en que vemos el codo no indica directamente el valor de k donde deja de ser significativa la variación, por lo que para estos casos se tomaron valores de k entre 20 y 50, alcanzando un promedio de 40,25. Cabe mencionar, que por inspección de los datos, notamos que si aumentamos el valor de k para este método, aumenta la cantidad de puntos aislados, lo cual no es deseable para nuestro objetivo principal que es el generar recomendaciones a través de este método. Por otra parte, en la Tabla 5.1 y en la Tabla 5.2 vemos que la sub-red 1 tienen los valores más bajos en el método de k-means, y a la vez, el k escogido en la sub-red 1 fue $k_1 = 20$, el menor de todos los k, lo que indica cierta relación entre el k escogido para este método y los resultados obtenidos en las métricas.

Por último, consideremos la **novedad** que presentaron las recomendaciones entregadas por cada método. Consideraremos una recomendación es *novedosa* para un usuario cuando el perfil que recomendamos no está en la lista actual de amigos del usuario. Después de aplicar los métodos para cada sub-red y usuario en ella, obtuvimos el porcentaje de novedad que presentaba cada recomendación obtenida. Para ello, comparanos la lista de amigos actual del usuario con la lista de recomendaciones y se obtuvo sobre el 99 % de novedad para todos los métodos. Esto indica que prácticamente la totalidad de usuarios recomendados no formaban parte de la lista de amigos de los usuarios a los que fueron recomendados. Inferimos que esto ocurre por

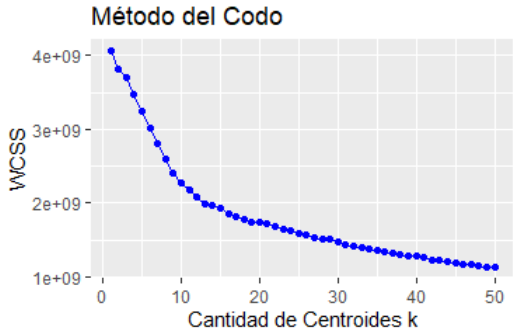


Figura 5.10: En el gráfico se muestran los valores obtenidos para *withinss* usando desde 1 a 50 clusters en k-means aplicado a la matriz de la sub-red 5.

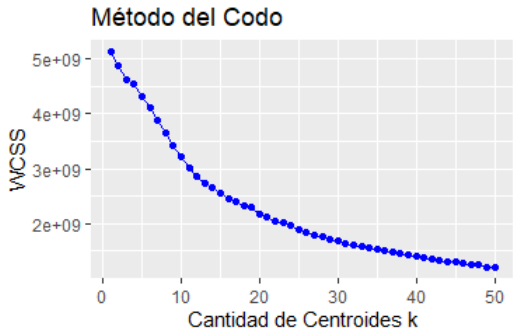


Figura 5.11: En el gráfico se muestran los valores obtenidos para *withinss* usando desde 1 a 50 clusters en k-means aplicado a la matriz de la sub-red 15.

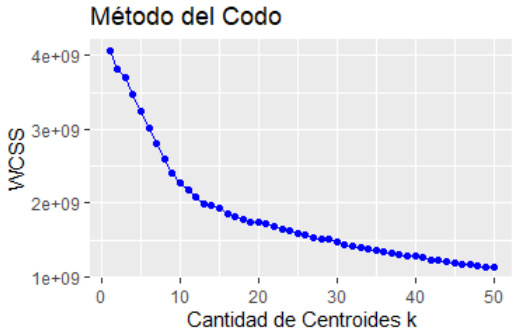


Figura 5.12: En el gráfico se muestran los valores obtenidos para *withinss* usando desde 1 a 50 clusters en k-means aplicado a la matriz de la sub-red 20.

características particulares de la base de datos que disponemos y no será un parámetro significativos para comparar los métodos. El detalle de los resultados del análisis de novedad se puede revisar en el Anexo 1.

Capítulo 6

Conclusiones

A lo largo de estas páginas hemos presentado y analizado conceptos de la teoría de grafos y matemáticas en general que nos permiten modelar problemas de la vida real, como lo son las redes sociales. Haciendo uso del *problema del ordenamiento* en el contexto de redes sociales, planteamos y testeamos nuevas técnicas para generar recomendaciones a usuarios de ella. Las herramientas más novedosas abarcadas en este texto son las *matrices de Robinson* y el uso de la *conductancia* para evaluar la calidad de los algoritmos, conceptos que se han revisado en el Capítulo 3, y cuya complejidad es NP-Duro.

En el Capítulo 4, hemos presentado 5 algoritmos articulados por nosotros para generar recomendaciones. Los primeros 3 se basan en la idea de influencia entre usuarios (o amigos en común) y los algoritmos 4 y 5 generan recomendaciones a partir de un orden de Robinson o un orden cercano a el orden de Robinson entregado por la función *SFS* del software *Rstudio*. A pesar de que los algoritmos entre sí tiene la misma base para el orden, estos se diferencian en la forma en que se escoge el tamaño de la recomendación de usuarios que se recomendará a cada individuo de la red, lo cual genera diferencias en los resultados. Para comparar los nuevos algoritmos con un método común o bien validado, generamos recomendaciones para la misma base de datos utilizando *k-means*, pues, dada las características de la base de datos, requerimos un método de recomendación o clustering no supervisado (no existen etiquetas en la base de datos).

Luego de aplicar los algoritmos con las consideraciones entregadas en las secciones 5.1 y 5.2, obtuvimos y analizamos distintas métricas y características que nos permiten comparar los algoritmos entre sí, para ello consideramos: el promedio del corte normalizado, el promedio de la conductancia obtenida

por cada método en las sub-redes generadas, el promedio de la expansión de $R_{\mathbf{V}} \cup \{\mathbf{V}\}$, revisión de los *tags* en común entre el usuario y la lista de usuarios que se le ha recomendado y la cantidad de usuarios que no obtuvieron recomendación con cada método. Presentamos nuestras conclusiones respecto a los resultados en lo que sigue:

1. **Sobre el promedio del corte normalizado:** esta métrica presenta el valor más bajo para el método propuesto en el Algoritmo 1 que utiliza la matriz de influencia y un umbral sobre el cual se escogen los usuarios más influyentes, y a la vez, k-means obtuvo el valor más alto para esta métrica. Uno podría pensar de esta forma que el primer método es mejor al minimizar el promedio del corte normalizado, sin embargo, si relacionamos lo obtenido con otras características, como la cantidad de usuarios que no reciben recomendación con este método, nos damos cuenta que este valor se minimiza pues el conjunto de corte que genera la recomendación son menos respecto a las de otros métodos.

2. **Sobre el promedio de conductancias:** los resultados entregan valores similares para los métodos propuestos. Considerando que nuestro referente es k-means, entonces concluimos que los algoritmos propuestos son válidos al tener un valor de conductancia cercano al de este algoritmo.

3. **Sobre el promedio de la expansión de $R_{\mathbf{V}} \cup \{\mathbf{V}\}$:** para esta métrica el método k-means tiene el valor más bajo entre los 6 métodos. Este valor se explica pues el método k-means se carecterizó por generar una cantidad pequeña de grupos que abarcaban a muchos usuarios, lo que hizo que la suma de los pesos de las aristas dentro del vecindario o cluster que contenía al usuario para el cual se calculaba la métrica, fuese mucho más alto en comparación a los que generaban otros métodos (como los generados a partir del orden de Robinson) para los cuales las recomendaciones tenían menor cantidad de individuos en él. Además, como los métodos basados en orden de Robinson obtuvieron los valores más altos de esta métrica, inferimos que es posible aumentar el tamaño de la recomendación en estos métodos, o dicho de otra forma, se puede expandir el conjunto de recomendación.

4. **Sobre las *tags***: utilizando este dato incorporado en la base de datos de *Twitter*, comparamos las etiquetas usadas por el usuario al que se le entrega la recomendación y la lista de usuarios recomendada. Los

promedios de proporciones de coincidencia en tags para los métodos fueron similares entre sí. Para los algoritmos 2, 3, 4, 5 y k-means la coincidencia en tags fue superior o igual a 62 %, mientras que para el Algoritmo 1 fue del 54 %. Es relevante ver que los métodos basados en orden de Robinson obtuvieron valores similares al de k-means a pesar de generar recomendaciones con menor cantidad de usuarios. Además, vemos que los porcentajes de coincidencia en tags obtenidos por los métodos que propusimos son mayores al valor esperado para la base de datos, cuyo valor es 58,72 % de coincidencia esperada.

5. **Sobre los usuarios sin recomendación:** los usuarios sin recomendación afecta directamente los valores obtenidos en las métricas, ya que estas dependen de la cantidad y pesos de las aristas en el conjunto de corte y de la cantidad y pesos de aristas en el subgrafo inducido por la recomendación. Por ejemplo, vemos que el Algoritmo 1 tiene el valor más bajo en el promedio de corte normalizado y es el método que dejó más usuarios sin recomendación. También vemos que el promedio de la expansión de $R_{\mathbf{V}} \cup \{\mathbf{V}\}$ fue más baja para *k-means*, método que tendió a agrupar varios usuarios en pocos cluster y dejar muchos usuarios aislados, en promedio dejó 31 usuarios sin recomendación; consideremos que en promedio el k escogidos para k-means fue igual a 40, esto significa que se generaron 9 cluster con más de un usuario y el resto eran usuarios sin recomendación, estos pocos clusters con muchos usuarios en él ayudaron a disminuir el valor de esta métrica en el algoritmo respecto a el valor obtenido para la misma por otros algoritmos cuyas recomendaciones tenían menos perfiles de usuarios.

6. **Sobre los gráficos del diagrama de Venn:** en la Subsección 5.3.6 usamos diagramas de Venn para comparar las recomendaciones dadas a dos usuarios por los distintos métodos. En ellos obtuvimos que *k-means* entrega recomendaciones con más perfiles en ella a diferencia de los otros métodos, también vemos que el porcentaje de coincidencia entre las listas no es muy alto, llama la atención que a pesar de esto, los métodos alcancen valores de conductancia bastante similares. También vemos que el Algoritmo 1 es el que entrega listas con menor cantidad de perfiles, esto se explica principalmente por el umbral escogido, pues las matrices presentan pocas entradas que sobrepasen dicho umbral. También vemos que en los métodos basados en orden de Robinson las listas obtenidas presentan alrededor del 50 % de coicidencia, y además, como el promedio de recomendaciones dadas sólo por el Algoritmo 4

llega al 38 %, deducimos que este método presentó las listas con mayor cantidad de usuarios en la mayor parte de los casos.

En síntesis, los métodos propuestos basados en matriz de influencia y orden de Robinson generan recomendaciones más particulares que k-means, pues el segundo agrupa a cientos de usuarios en un cluster y luego todos son recomendados entre sí, mientras que los propuestos a partir de influencia y orden de Robinson tiene un tamaño exacto asignado para la cantidad de perfiles que se recomendarán y estas se obtienen a partir de reordenar los usuario indexados en la matriz representante, es decir, el tamaño de la recomendación es moldeable en los algoritmos propuestos a diferencia de k-means. La desventaja principal de los algoritmos basados en el orden de Robinson es el alto costo computacional que tiene calcular un orden de Robinson (o uno cercano a él), lo que requiere mayor tiempo de ejecución. Además, si consideramos el análisis de los algoritmos que hemos presentado desde las métricas propuestas y comparamos con los valores que obtuvo k-means para estas métricas, diremos que los algoritmos propuestos son acertados pues obtuvieron valores similares a los de *k-means*, lo que se reitera en el análisis de los *tags*, sin embargo, para este último eran resultados esperados. Por otro lado, a pesar de que el Algoritmo 1 presenta valores pequeños en promedio de la conductancia y promedio del corte normalizado, no es un buen algoritmo, pues dejó una gran cantidad de usuarios sin recomendación; es necesario generar una técnica particular para este algoritmo que logre disminuir el 20 % de usuarios que no recibió recomendación, por ejemplo, probar con distintos valores para el umbral, o planificar un método que entregue un umbral personalizado a cada usuario de acuerdo a los valores de la función de similitud. De todo esto, finalmente podemos ver que los algoritmos propuestos, descartando o mejorando el Algoritmo 1, sí logran generar potenciales buenas recomendaciones, con ventajas sobre *k-means* como lo es personalizar el tamaño de la recomendación y siempre entregar una recomendación.
Finalmente, como trabajo futuro se propone estudiar los algoritmos con otras bases de datos de redes sociales. Adicional a lo planteado para determinar un valor óptimo para el umbral del Algoritmo 1, se podría estudiar en detalle la efectividad de k-means en este tipo de base de datos, y comparar también los métodos propuestos con otros algoritmos de recomendación no supervisado. También se pueden generar las recomendaciones a partir de otros criterios, como el obtener la intersección de perfiles en las listas de amigos de los usuarios seleccionados por los algoritmos presentados en este trabajo. Por último, proponemos aplicar estos métodos en otras bases de datos en los que se generen recomendaciones, por ejemplo, para datos de plataformas de

películas, música, videos, entre otras en las que se tenga información de una lista de artículos del gusto de cada usuario que permitirá finalmente hacer las comparaciones y definir una función de distancia o similitud.

Anexo 1

Sub-red	M1-A	M1-B	M1-C	M2-A	M2-B	K-means
1	1.00	1.00	1.00	1.00	1.00	1.00
2	1.00	1.00	1.00	1.00	1.00	1.00
3	1.00	1.00	1.00	1.00	1.00	1.00
4	1.00	1.00	1.00	1.00	1.00	1.00
5	1.00	1.00	1.00	1.00	1.00	1.00
6	1.00	1.00	1.00	1.00	1.00	1.00
7	1.00	1.00	1.00	1.00	1.00	1.00
8	1.00	1.00	1.00	1.00	1.00	1.00
9	1.00	1.00	1.00	1.00	1.00	1.00
10	1.00	1.00	1.00	1.00	1.00	1.00
11	1.00	1.00	1.00	1.00	1.00	1.00
12	1.00	1.00	1.00	1.00	1.00	1.00
13	1.00	1.00	1.00	1.00	1.00	1.00
14	1.00	1.00	1.00	1.00	1.00	1.00
15	1.00	1.00	1.00	1.00	1.00	1.00
16	1.00	1.00	1.00	1.00	1.00	1.00
17	1.00	1.00	1.00	1.00	1.00	1.00
18	1.00	1.00	1.00	1.00	1.00	1.00
19	1.00	1.00	1.00	1.00	1.00	1.00
20	1.00	1.00	1.00	1.00	1.00	1.00
Promedio	1.00	1.00	1.00	1.00	1.00	1.00

Tabla 6.1: Proporción novedad obtenida por cada método en las 20 sub-redes. Proporción aproximada a dos cifras significativas por el software R-studio.

Bibliografía

[1] Julio Aracena and Christopher Thraves Caro. The weighted sitting closer to friends than enemies problem in the line. *arXiv preprint arXiv: 1906.11812*, 2019.

[2] James Bennett, Stan Lanning, et al. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. Citeseer, 2007.

[3] Simon Kemp Ceo. Digital 2021 global overview report. Technical report, We Are Social and Hootsuite, 2021.

[4] Victor Chepoi and Bernard Fichet. Recognition of robinsonian dissimilarities. *Journal of Classification*, 14(2):311–325, 1997.

[5] Frédéric Gardi. The roberts characterization of proper and unit interval graphs. *Discrete Mathematics*, 307(22):2906–2908, 2007.

[6] George Giakkoupis. Tight bounds for rumor spreading in graphs of a given conductance. In *Symposium on Theoretical Aspects of Computer Science (STACS2011)*, volume 9, pages 57–68, 2011.

[7] David F Gleich and C Seshadhri. Vertex neighborhoods, low conductance cuts, and good seeds for local community methods. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 597–605, 2012.

[8] Anil K Jain. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666, 2010.

[9] Ravi Kannan, Santosh Vempala, and Adrian Vetta. On clusterings: Good, bad and spectral. *Journal of the ACM (JACM)*, 51(3):497–515, 2004.

[10] Gustav Kirchhoff. Ueber die auflösung der gleichungen, auf welche man bei der untersuchung der linearen vertheilung galvanischer ströme geführt wird. *Annalen der Physik*, 148(12):497–508, 1847.

[11] Marcos Kiwi and Christopher Thraves Caro. Fifo queues are bad for rumor spreading. *IEEE Transactions on Information Theory*, 63(2):1159–1166, 2016.

[12] Sanjay Krishnan and Ken Goldberg. The minimum conductance dissimilarity cut (mcdc) algorithm to increase novelty and diversity of recommendations. Technical report, Working paper, Industrial Engineering and Operations Research Department ..., 2015.

[13] Monique Laurent and Matteo Seminaroti. A lex-bfs-based recognition algorithm for robinsonian matrices. *Discrete Applied Mathematics*, 222:151–165, 2017.

[14] Monique Laurent and Matteo Seminaroti. Similarity-first search: a new algorithm with application to robinsonian matrix recognition. *SIAM Journal on Discrete Mathematics*, 31(3):1765–1800, 2017.

[15] Jie Lu, Dianshuang Wu, Mingsong Mao, Wei Wang, and Guangquan Zhang. Recommender system application developments: a survey. *Decision Support Systems*, 74:12–32, 2015.

[16] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[17] Bojan Mohar. Isoperimetric numbers of graphs. *Journal of combinatorial theory, Series B*, 47(3):274–291, 1989.

[18] Pascal Préa and Dominique Fortin. An optimal algorithm to recognize robinsonian dissimilarities. *Journal of Classification*, 31(3):351–385, 2014.

[19] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer, 2011.

[20] Fred S Roberts. Indifference graphs. proof techniques in graph theory. In *Proceedings of the Second Ann Arbor Graph Conference, Academic Press, New York*, 1969.

[21] William S Robinson. A method for chronologically ordering archaeological deposits. *American antiquity*, 16(4):293–301, 1951.

[22] Donald J Rose, R Endre Tarjan, and George S Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on computing*, 5(2):266–283, 1976.

[23] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[24] Alistair Sinclair and Mark Jerrum. Approximate counting, uniform generation and rapidly mixing markov chains. *Information and Computation*, 82(1):93–133, 1989.

[25] Hugo Steinhaus. Sur la division des corps matériels en parties. *Bull. Acad. Polon. Sci*, 1(804):801, 1956.

[26] Ruchita V Tatiya and Archana S Vaidya. A survey of recommendation algorithms. *IOSR Journal of Computer Engineering*, 16(6):16–19, 2014.

[27] Hubert Wassner. Twitter friends. `https://www.kaggle.com/hwassner/TwitterFriends`, 2016.

[28] Mirjam Wattenhofer, Roger Wattenhofer, and Zack Zhu. The youtube social network. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 6, 2012.

[29] Douglas Brent West et al. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, 2001.