PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA

EVALUACIÓN DE DIVERSAS METODOLOGÍAS PARA RECOMENDACIÓN DE LIBROS

JORGE LUIS SCHELLMAN SEPÚLVEDA

Tesis para optar al grado de
Magíster en Ciencias de la Ingeniería

Profesor Supervisor:
DENIS PARRA SANTANDER

Santiago de Chile, Enero 2019

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA

EVALUACIÓN DE DIVERSAS METODOLOGÍAS PARA RECOMENDACIÓN DE LIBROS

JORGE LUIS SCHELLMAN SEPÚLVEDA

Miembros del Comité:
DENIS PARRA SANTANDER
HANS LÖBEL DIAZ
ROBERTO GONZÁLEZ-IBÁÑEZ
JUAN CARLOS FERRER ORTIZ

Tesis para optar al grado de
Magíster en Ciencias de la Ingeniería

Santiago de Chile, Enero 2019

A mi familia y amigos, que me apoyaron.

AGRADECIMIENTOS

Me gustaría agradecer a mi familia y a mis amigos por el apoyo que me han dado.

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

En el presente trabajo de tesis se exploraron diferentes tipos de sistemas recomendadores en el dominio de recomendación de libros con el objetivo de averiguar qué algoritmos son los adecuados para este escenario genérico, qué tipos de datos son los convenientes para usar y de qué forma conviene representar ítems y usuarios para obtener los mejores resultados posibles. Para ello se compararon varias estrategias dentro de las tres principales familias de recomendadores: aquellos basados en contenido, basados en filtrado colaborativo e híbridos. Para entrenar algoritmos de filtrado colaborativo se usaron distintos tipos de fuentes de datos como feedback explícito de parte del usuario en forma de ratings hacia los libros (los que son una muestra directa sobre los gustos del usuario), feedback implícito, e información sobre el contexto de las interacciones usuario-libro, como la fecha del consumo y los autores del libro. Adicionalmente se analizó qué metadatos de los libros son los más relevantes al momento de ver similitudes entre ellos con el fin de entregar buenas recomendaciones. Se encontró en estudios offline y online que los métodos basados en contenido entregan recomendaciones más relevantes y diversas que los métodos colaborativos. Estos últimos por su parte mejoran sus resultados al incluir los nombres de los autores como información contextual. También mejoran cuando se alimentan con feedback implícito en vez de explícito. Este estudio provee nuevos resultados al área en el dominio de libros que pueden ayudar a desarrollar subsecuentes sistemas recomendadores para ser usados en producción y plantea nuevas interrogantes que pueden guiar el camino de futuras investigaciones.

Palabras Claves: sistemas recomendadores, recomendación de libros, recuperación de información, métodos basados en contenido, estudio de usuarios

ABSTRACT

In this thesis work we explored several types of recommender systems, specifically on the domain of books in order to find out which recommender algorithms are best suited for this generic scenario, which types of data are suitable to use and in which way it is convenient to represent the items and the users to obtain the best possible results. For this aim, we compare several recommender strategies: content-based, collaborative filtering-based and hybrid systems. To train collaborative filtering algorithms, different sources of data were used such as explicit feedback from the user in the form of ratings towards books (which are a direct sample of the user's tastes), implicit feedback, and information about the context of user-book interactions, such as the consumption timestamp and the books' authors. Additionally, we analyze metadata from books to figure out the incidence that each one has over the task of seeking similar books, and use that knowledge to give better book recommendations. We found in offline and online studies that content-based methods provide more relevant and diverse recommendations than collaborative methods. The latter in turn improve their results by including the names of the authors as contextual information. They also improve when they are fed with implicit rather than explicit feedback. This study provides new results to the area in the domain of books that can help develop subsequent recommender systems to be used in production and raises new questions that can guide the way of future research.

Keywords: recommender systems, book recommenders, information retrieval, content-based, user studies

CAPÍTULO 1. INTRODUCCIÓN

1.1. Motivación

En el 2012, se estimaba que la tasa de producción diaria de datos era de 2.5 *exabytes* ($2{,}5 \times 10^{18}$B) (*What is big data? – Bringing big data to the enterprise*, 2018), lo que significó poder decir que en el 2013 el $90\,\%$ del total de los datos en la Web fue producido en los últimos 2 años (Jacobson, 2013). Un estudio predecía que la cantidad global de datos crecería de manera exponencial entre 4.4ZB ($4{,}4 \times 10^{21}$B) a 44ZB entre el 2013 y el 2020 (Hajirahimova y Aliyeva, 2017), un orden de magnitud. El último reporte *Data Never Sleeps* de Domo (2018) menciona que en el 2020 se generarían 1.7MB por segundo por cada persona del planeta (*Becoming A Data-Driven CEO*, 2018), lo que lleva a la estimación de una producción diaria mundial de 1.1ZB. Es por estas razones de volumen y velocidad de datos que se dice coloquialmente que vivimos en la era del *Big Data*.

Así usado el término, **Big Data** da la idea de ser una época que pareciera no tener fin, pues la cantidad de datos en la Web no dejará de crecer, o bien de simplemente denotar el hecho de que la información (término que se usará indistintamente del término 'dato') en la Web es cada vez más grande. No está claro quién fue el primero en acuñar la expresión (Lohr, 2013), pero uno de los que la popularizó fue John Mashey de Silicon Graphics a fines de la década de los 90 (Lohr, 2013; Mashey, 1998). Él se refería a la adopción de un nuevo enfoque con el que miramos los datos. No puede ser lo mismo analizar una cantidad baja y fija de información del mismo tipo que un flujo grueso y constante de datos de tipo variable. Tampoco puede ser igual la manera de ir almacenando 2MB/s desde y hacia un lugar centralizado que 2TB/s desde y hacia distintos nodos de almacenaje. El tratamiento de los datos, sea cual sea el fin, debe ser diferenciado

cuando son masivos, diversos, complejos y distribuidos a que cuando no lo son. Es este nuevo *paradigma* desde el que se abordará esta problemática.

Con este foco se miran dos grandes temas: (i) el de almacenar y procesar grandes cantidades de datos y (ii) el de entender, en el sentido de lenguaje de máquinas, esos datos. El consenso está en que alrededor de un 20 % de los datos en la Web corresponden a datos estructurados (Grimes, 2008; Muse, 2017), tales como aquellos generados por formularios, GPSs, Web logs, etc. y que están típicamente almacenados en bases de datos relacionales, por lo que al tratar con grandes volúmenes de información existe un beneficio cuando ésta es estructurada, pues el computador la puede entender nativamente. El otro 80 % por tanto son datos no estructurados, como videos, *emails*, imágenes, PDFs, publicaciones en redes sociales, etc. Información no organizada y compleja, por lo que el computador no la puede entender.

Acerca del punto 1 se han desarrollado diversas tecnologías de *High performance computing* (HPC) como *clustering*, sistemas distribuidos y paralelización de tareas. Es particularmente notable el ecosistema Hadoop como ejemplo de entorno de trabajo que integra las técnicas mencionadas. Sobre el segundo tema, la meta es entender los datos no estructurados, que son los que componen el mayor porcentaje del total. El área de aprendizaje de máquina o *machine learning* (ML) nos proporciona herramientas importantes para este fin.

Sistemas de recuperación de información, tales como los populares buscadores de la Web tipo Google, Yahoo y Bing hacen uso de estas herramientas de ML para ayudar a usuarios a acceder a ítems de interés de manera oportuna. Lo que hace falta si es que tomamos en cuenta los intereses y preferencias del usuario es la priorización y personalización de la información recuperada.

A propósito de aplicaciones que se alimentan de muchos datos y que ayudan a darles sentido, están los sistemas recomendadores (Parra y Sahebi, 2013). Éstos

no sólo asisten a los usuarios a entender la información, sino que también los ayudan a encontrar información relevante para ellos en medio de una sobrecarga de información tanto relevante como irrelevante dependiendo del usuario. El propósito de esta tesis es la **creación y evaluación de estos sistemas**.

1.2. Sistemas Recomendadores

Los sistemas recomendadores (RS por sus siglas en inglés, en singular; RSs en plural) son un conjunto de técnicas y herramientas que proveen sugerencias de ítems para serles de utilidad a usuarios (Burke, 2007; Mahmood y Ricci, 2009; Ricci, Rokach, Shapira, y Kantor, 2010). Algunos términos importantes usados a lo largo de esta tesis y con el cuales el lector debería familiarizarse son:

1. 'Ítem' es el término con el que denotaremos genéricamente al objeto de recomendación. Éste puede ser un sitio Web, un periódico, una noticia, una canción, un CD, un libro, un artículo de belleza o cualquier producto que pueda ser adquirido a través de una aplicación de *e-commerce*, por ejemplo.
2. El 'usuario' es el sujeto que utiliza un RS a través de una aplicación que la integre. Ellos pueden tener diversos objetivos y características.
3. 'Uso' puede ser múltiple: escuchar, ver, comprar, leer, etc. Diremos que si el usuario aceptó la recomendación es porque ésta le pareció relevante a sus fines e intereses, y que por tanto le dió un uso al ítem, lo aceptó o que lo 'consumió'. A lo mismo a veces se le denomina 'interacción' del usuario con el ítem, dependiendo del contexto.

Un RS tiene el fin de ayudar a un usuario de un sistema a elegir ítems personalizadamente en un espacio poblado de ítems de acuerdo al comportamiento observado del usuario con respecto a los ítems del espacio (McNee, Kapoor, y

Konstan, 2006). Los RSs son beneficiosos tanto para los proveedores de servicios que los integran como para sus usuarios (Pu, Chen, y Hu, 2011), puesto que reducen los costos de transacción de encontrar y seleccionar artículos en un entorno de compras en línea (R. Hu y Pu, 2009), incrementan la satisfacción de los usuarios (Ricci et al., 2010), mejoran el proceso de toma de decisiones (B. Pathak, Garfinkel, Gopal, Venkatesan, y Yin, 2010) y hacen aumentar los ingresos en aplicaciones de e-commerce, ya que son medios efectivos para vender más productos (Pu et al., 2011).

A fin de que puedan generar recomendaciones personalizadas, los RSs aprovechan información que los usuarios dejan en su interacción con el sistema. Esta información puede estar estructurada (o no) de varias maneras, y la selección de qué información utilizar dependerá de la técnica de recomendación.

Feedback Explícito

Hay aplicaciones en las que el usuario puede proveer *ratings* (voz inglesa que significa una valuación en una escala numérica) a los ítems consumidos a través de su interfaz para que el RS pueda construir y mejorar su desempeño. Esto es muy común en comunidades virtuales de catalogación tales como Last.fm, Goodreads, Internet Movie Database o CiteULike, en donde los usuarios califican según una escala de ratings los libros leídos, videos vistos, etc. Estos ratings pueden tener distintas formas, por ejemplo ratings numéricos (de 1 a 5), ordinales (’de acuerdo’, ’neutral’, ’en desacuerdo’) o binarios (0 o 1, ’me gusta’ o ’no me gusta’). La exactitud de las recomendaciones dependen de la cantidad de ratings proporcionados por el usuario. El defecto está en que se requiere un esfuerzo por parte de los usuarios, y ellos no siempre están dispuestos a suplir suficiente información. Sin embargo estos son datos confiables sobre las preferencias de los usuarios, ya que no requieren extraer preferencias a partir de acciones. Los RS que se alimentan de este tipo de datos suelen ser percibidos como más confiables,

más transparentes y con recomendaciones de mejor calidad en general (Buder y Schwind, 2012).

Feedback Implícito

Un medio de aliviar la carga sobre los usuarios de dar ratings es infiriendo sus preferencias analizando su comportamiento con la aplicación. El sistema automáticamente lo hace monitoreando las diversas acciones del usuario como por ejemplo su historial de compras, historial de navegación, tiempo pasado en ciertas páginas Web, cantidad de clicks en ciertos elementos, o indicaciones de si es que el usuario observó o consumió el ítem, y si es que se dio el esfuerzo de calificarlo, pues de no haber interacción puede tomarse como feedback negativo. Si bien el método requiere menos esfuerzo por parte del usuario, son menos precisos, ya que no obtienen información directa sobre las preferencias de los usuarios. Por otro lado, se ha argumentado de que las preferencias implícitas pueden ser más objetivas ya que no hay sesgo por usuarios actuando de una forma socialmente deseable (Buder y Schwind, 2012), y no hay necesidad alguna de mantener una imagen para otros (Gadanho y Lhuillier, 2007).

1.2.1. Técnicas de Recomendación

Para que un RS pueda implementar su función principal, el sistema debe predecir la utilidad de un conjunto de ítems, o al menos comparar la utilidad entre ellos, para decidir si es que vale la pena recomendar algún ítem.

El Problema de Recomendación

Podemos modelar el grado de utilidad del ítem i para el usuario u como una función real $R(u, i)$ (Adomavicius y Tuzhilin, 2005). Para cada usuario $u \in U$

queremos escoger un item $i' \in I$ que maximice la utilidad del usuario:

$$\forall u \in U, i'_u = \text{argmax}_{i \in I} R(u, i) \tag{1.1}$$

Cada elemento del espacio de ítems I puede ser definido por un set de características propias, como su ID de ítem, y dependiendo del dominio, puede ser géneros, director, autores, año de lanzamiento. dirección, duración, etc. Análogamente, cada elemento del espacio de usuarios U puede ser definido con un perfil de usuario, tal que incluya características propias de cada uno como edad, género y/o simplemente ID de usuario.

El problema principal de los RS es que R no está definido para todo el espacio $U \times I$, sólo para un subset observado de éste (llámese el input de entrenamiento del sistema). Esto significa que R tiene que ser extrapolado para cubrir todo el espacio $U \times I$. El modo en que se aproxime R a una función $\hat{R}$ que cubra todo el espacio depende del tipo de técnica de recomendación que se utilice.

Existen múltiples categorizaciones en las que se pueden clasificar los diversos métodos para crear sistemas recomendadores (Adomavicius y Tuzhilin, 2005; Isinkaye, Folajimi, y Ojokoh, 2015; Parra y Sahebi, 2013; Ricci et al., 2010), mas en este trabajo, a parte de hacer mención a otros tipos de técnicas, la clasificación será en base principalmente a tres de los más importantes en la categorización de Parra y Sahebi (2013), pues son los que se investigan: sistemas basados en filtrado colaborativo, sistemas basados en contenido y sistemas híbridos.

Métodos Basados en Filtrado Colaborativo

En los RS basados en filtrado colaborativo (CF por sus siglas en inglés) al usuario se le recomiendan ítems que usuarios similares al usuario objetivo en cuanto a gustos y preferencias han consumido en el pasado.

En los sistemas CF la utilidad es el rating. La tarea fundamental de los RS de filtrado colaborativo es computar el rating $\hat{R}(u,i)$ para los ítems $i' \in I$, i.e., $\hat{R}(u,i_1),\ldots,\hat{R}(u,i_N)$, y luego el sistema recomendará aquellos ítems $i_{j_1},\ldots,i_{j_n}$ $(n \leq N)$ con la utilidad predicha más grande, donde $N = |I|$.

Los procedimientos para predecir ratings dependen de dos subclases de algoritmos: aquellos basados en memoria (*memory-based*) y aquellos basados en modelos (*model-based*). Los algoritmos basados en memoria son heurísticas que hacen de la predicción de rating sobre el ítem no observado un trabajo de agregación de los ratings observados de los usuarios similares al usuario objetivo:

$$r_{u,i} = \mathrm{A}_{u' \in \hat{U}}\, r_{u',i} \tag{1.2}$$

donde $\hat{U}$ es el set de los N_u vecinos más cercanos a u que han dado rating al ítem i (razón por la cual se les suele denominar a esta subclase de recomendadores como basados en vecinos cercanos, *neighborhood-based* o modelos *kNN*). Este cálculo puede a veces incluir a los ratings del usuario u, normalmente en forma del promedio de sus ratings $\bar{r_u}$. El operador $\mathrm{A}_{u' \in \hat{U}}$ puede ser un simple promedio o puede ser, como es más común, una suma ponderada de ratings con similaridades $sim(u,u')$ como factores de ponderación. Así, $r_{u',i}$ tendrá más peso si la similaridad entre u y u' es mayor.

Los algoritmos basados en modelos usan la colección de ratings para entrenar un modelo, el cual es usado para hacer predicción de ratings. El proceso de construcción del modelo puede ser hecho con técnicas de aprendizaje de máquinas o de minería de datos. Un beneficio que tienen estas técnicas es que pueden recomendar rápidamente ítems por el hecho de usar modelos pre-computados, y han demostrado tener tan buen desempeño como aquellos basados en memoria (Isinkaye et al., 2015). Algunas familias conocidas de modelos son los modelos de *clustering* y de redes Bayesianas (Adomavicius y Tuzhilin, 2005; Ricci et al., 2010), que pertenecen más que nada a un enfoque probabilístico. También están

las que pertenecen a modelos de factores latentes, como *singular value decomposition* (SVD) y algunas de sus variantes: *funkSVD*, *SVD++*, etc.

Como los sistemas colaborativos usan los ratings de los usuarios, pueden recomendar ítems de cualquier tipo de contenido. En general se percibe que las recomendaciones de sistemas colaborativos son más diversas que con respecto a sistemas basados en otras técnicas. Sin embargo aun poseen importantes limitaciones.

El RS primero debe aprender las preferencias del usuario a partir de sus ratings para que pueda hacer buenas recomendaciones, y mejor serán en cuanto el usuario provea más ratings. El **problema del nuevo usuario** ocurre cuando el usuario no ha suministrado suficientes ratings al sistema para que éste pueda generar recomendaciones precisas. A parte, nuevos ítems son añadidos regularmente a la aplicación. El sistema no podrá recomendar un nuevo ítem en cuanto no haya sido calificado por una cantidad considerable de usuarios, por lo que se le llama **problema del nuevo ítem**. Dependiendo del contexto, a uno o a ambos de estos problemas se les denomina **arranque en frío** o *cold start problem*.

Las matrices de ratings normalmente son muy poco densas (o que tienen una *sparsity* muy alta). En el dominio de películas, algunas películas pueden ser calificadas por pocos usuarios, por lo que éstas rara vez serán recomendadas, Para los usuarios con gustos inusuales, por definición no habrá muchos usuarios con gustos similares, lo que llevará a malas recomendaciones. Una manera de atacar esto es con la descomposición SVD, en la que reducimos la dimensionalidad $|U| \times |I|$ de matrices *sparse* de ratings a una matriz aproximada de menor rango. Resulta importante por tanto predecir ratings efectivamente a partir de un tamaño pequeño de muestras.

Métodos Basados en Contenido

En los RS basados en contenido (CB por sus siglas en inglés) al usuario se le recomiendan ítems similares a los que el usuario prefirió en el pasado. Las similaridades de los ítems son calculadas basados en ciertas características (o *features*) seleccionadas cuidadosamente para compararlos. Por ejemplo, si se sabe que la utilidad entre un usuario y un libro de cierto género es alta, el sistema puede aprender a recomendar libros similares del mismo género. En otras palabras, la utilidad $R(u, i')$ se estimará basada en las utilidades $R(u, i_k)$ asignadas por el usuario u a los ítems $i_k \in I$ que son similares al ítem i'.

Debido al significante progreso hecho por las comunidades de recuperación y filtración de información y debido a la importancia de varias aplicaciones basadas en texto, muchos sistemas CB se centran en recomendar ítems que contienen información textual, como documentos, sitios Web (URLs), etc., y son el tipo de recomendadores que investigamos en este trabajo.

Sea un set de atributos que caracterizan a un ítem i el *perfil de ítem* $\mathrm{IP}(i)$. Como el contenido del ítem estará dado por texto, nos referiremos a la unidad básica que compone al ítem como *término*. La importancia o descriptividad de un término t_i con respecto a un documento d_j es determinado por un peso $w_{i,j}$.

Una de las maneras más conocidas de especificar los pesos w es con la medida *frecuencia de términos/frecuencia invertida de documentos* (TF-IDF) (Salton, 1989). Sea N el número total de documentos que pueden ser recomendados, y digamos que el término t_i aparece en n_i documentos. Sea $f_{i,j}$ el número de veces que el término t_i aparece en el documento d_j. Definimos el *term frequency* $\mathrm{TF}_{i,j}(f_{i,j})$ de un término t_i en un documento d_j como una función de su frecuencia de ocurrencias, la cual puede tomar varias formas: la frecuencia misma de términos, la frecuencia normalizada por la suma de las frecuencias de los otros

términos en el mismo documento u otro tipo de normalización. Definimos el *inverse document frequency* IDF de un término t_i como:

$$\mathrm{IDF}_i = \log \frac{N}{n_i} \tag{1.3}$$

la cual también puede tomar varias formas, pero es generalmente definida así. Luego, el peso del término t_i en el documento d_j se define como:

$$w_{i,j} = \mathrm{TF}_{i,j} \times \mathrm{IDF}_i \tag{1.4}$$

en donde el contenido de d_j es $\mathrm{IP}(d_j) = (w_{1,j}, \ldots, w_{|T|,j})$, siendo $|T|$ la cardinalidad del set términos T llamado a veces *vocabulario* o *diccionario*.

Sea $UP(u)$ el perfil del usuario u aquél que describe sus gustos y preferencias. Estos perfiles se obtienen de analizar el contenido de los ítems vistos y calificados por el usuario. Dependiendo del contexto, al perfil se le pueden agregar -o pueden estar compuestos sólo de- información personal como su edad, sexo, locación geográfica, entre otros, y que puede ser obtenida explícitamente de los usuarios mediante cuestionarios o formularios, u obtenida implícitamente al aprender de su comportamiento a través del tiempo. $UP(u)$ puede estar definida también como un set de pesos $(w_{1,u}, \ldots, w_{|T|,u})$, donde cada peso $w_{i,u}$ sea la importancia del término t_i para el usuario u. Éste vector de pesos puede ser computado como un promedio de los vectores de pesos de los ítems preferidos por el usuario (Balabanović y Shoham, 1997), o como probabilidades de que cierto documento vaya a ser preferido (Pazzani y Billsus, 1997), entre otros enfoques.

La función de utilidad, en sistemas CB, la podemos definir como:

$$R(u, i) = score(UP(u), IP(i)) \tag{1.5}$$

Si al usuario y al ítem los podemos representar como vectores TF-IDF de pesos de términos $\vec{w}_u$ y $\vec{w}_i$, el *score* se suele representar con alguna heurística de

puntuación como la medida de similaridad coseno (Baeza-Yates, Ribeiro, et al., 2011; Salton, 1989):

$$R(u,i) = \cos\left(\vec{w}_u, \vec{w}_i\right) = \frac{\vec{w}_u \cdot \vec{w}_i}{||\vec{w}_u|| \times ||\vec{w}_i||} \tag{1.6}$$

Por ejemplo, si el usuario u lee artículos de psicología, el vector $\vec{w}_u$ tendrá altos valores en pesos relacionados a términos usados en psicología (como 'personalidad', 'psicoanálisis' o 'psicología'), por lo que los vectores $\vec{w}_i$ más cercanos a $\vec{w}_u$, según métricas de similaridad como ésta, tendrán valores altos en las entradas de ese tipo de términos, por lo que serán esos ítems los recomendados.

Al igual que SVD en sistemas CF, podemos hacer un análisis de factores latentes usando técnicas como *latent semantic analysis* (LSA) o *latent dirichlet allocation* (LDA) (Steyvers y Griffiths, 2007). Una matriz de co-ocurrencia de términos C (como ejemplo la tabla 1.1) podemos descomponerla en matrices de menor dimensionalidad. En la figura 1.1, cada una de las columnas de la matriz Φ (o filas de Θ) se les llama *tópicos* pues, si bien son vectores de factores latentes, y por tanto no son características observables que se les entreguen al modelo a modo de selección de *features* para entrenamiento, en inspección visual se les suele atribuir ciertas características observables derivado del hecho que estas descomposiciones se hacen de matrices de términos, y son los términos que en conjunto describen los ítems. Así, el sistema aprende automáticamente factores que el humano atribuye como tópicos.

Las filas de Φ son los vectores latentes de los ítems en los que cada entrada son *scores*, usualmente probabilidades, que indican la pertenencia del término a cierto tópico. Análogamente para los vectores de los documentos. Al usuario por lo tanto podemos representarlo como una agregación de vectores probabilísticos de términos, aquellos términos de los documentos preferidos por el usuario, y entonces $\mathrm{UP}(u)$, el perfil del usuario u, describirá qué tópicos son los preferidos

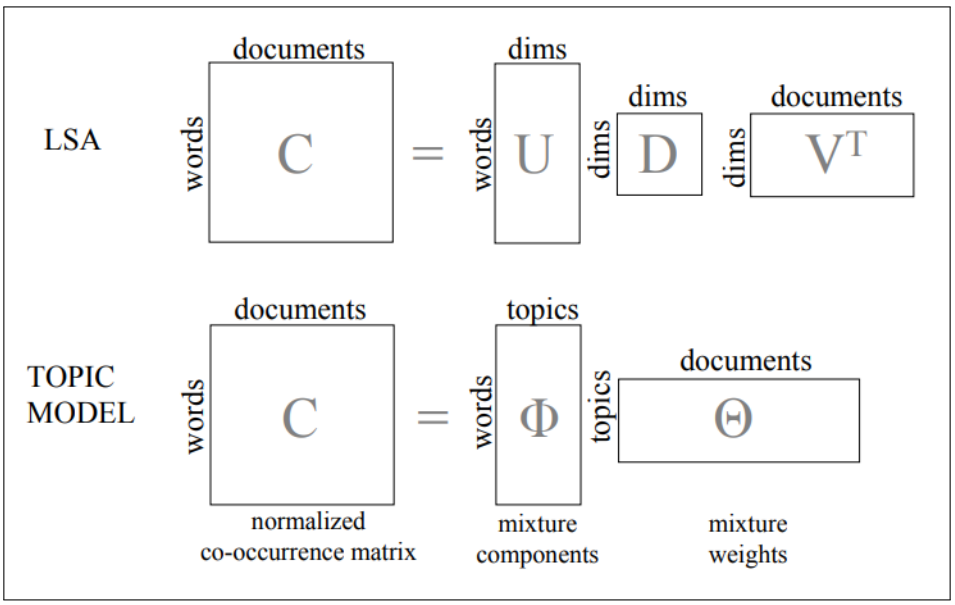

Figura 1.1. Topic modeling visualmente (Steyvers y Griffiths, 2007).

Cuadro 1.1. Ejemplo de matriz de co-ocurrencias. Las columnas corresponden a los vectores de pesos de los términos de los documentos. Las filas son los términos cuyos pesos pueden ser distintos por documentos. Las entradas no corresponden a algún cálculo en particular.

	d_1	d_2	d_3
t_1	8	41	35
t_2	10	4	2
t_3	1	2	15
t_4	29	18	17

por el usuario. Finalmente, la recomendación al usuario será de aquellos ítems no observados previamente por el usuario pertenecientes a sus tópicos preferidos.

Como estas técnicas dependen de representar los perfiles de usuario e ítem en forma de vector en donde cada entrada está asociada a un término del diccionario, a estos métodos se les suelen llamar modelos de *bolsa de palabras* (o BoW por sus siglas en inglés). Aun en el caso de modelamiento de tópicos, puesto que la descomposición de matrices se hace a partir de una matriz de co-ocurrencias. Un paradigma distinto es el de *word embeddings*. Las técnicas de word embeddings

aprenden de una manera no supervisada una representación de términos (o palabras) en espacio de vectores de baja dimensionalidad analizando su uso en corpus grandes de documentos. Éstas son usadas para aprender regularidades lingüísticas e información semántica de grandes datasets de texto y han recibido cada vez más atención debido a su buen desempeño en distintas tareas de procesamiento de lenguaje natural (NLP) (Musto, Semeraro, De Gemmis, y Lops, 2015).

Entre las ventajas de los sistemas CB, está el que la recomendación al usuario objetivo es independiente de los otros usuarios, a diferencia de los sistemas CF en donde no sólo el usuario objetivo debe suplir ratings, sino que también los demás usuarios de la aplicación. Las explicaciones del porqué de las recomendaciones pueden darse fácilmente listando los *features* o descripciones que hicieron que el ítem esté en la lista de recomendación, lo que hace aumentar la transparencia del sistema. Adicionalmente los sistemas basados en contenido no sufren del problema del nuevo ítem, puesto que se valen de su contenido y no de la cantidad de calificaciones que los pares le hayan hecho.

De todos modos ningún recomendador existe sin defectos, y algunos de ellos para los recomendadores CB incluyen el que se haya que hacer análisis de contenido limitado. Las técnicas CB están limitadas por los *features* de los ítems, tanto en cantidad (número de features) como de calidad (qué feature seleccionar). Los features pueden ser dados manualmente, pero esto no es práctico debido a la limitación en recursos. Podrían extraerse automáticamente, pero esto es difícil cuando tratamos con datos multimedia. Otro problema de esto es que si dos documentos están representados por el mismo set de features, son indistinguibles, por lo que el sistema no puede distinguir si es que un documento está escrito de una manera atractiva para el usuario objetivo o si es que no. Por otro lado, estos recomendadores tienen la tendencia a recomendar ítems similares a los ya consumidos seguidamente a través del tiempo, sin un método propio para que el usuario

encuentre un ítem bueno y a la vez inesperado. En otras palabras, tienen un problema de sobre-especialización, que impide aumentar la serendipia percibida por los usuarios con las recomendaciones. Al igual que los métodos CF, tienen el problema del nuevo usuario, puesto que necesitan analizar los gustos y preferencias de los usuarios. Si el usuario no ha consumido ni ha expresado su opinión sobre los ítems, el sistema no será capaz de proporcionar recomendaciones confiables.

Métodos Híbridos

Estos métodos comúnmente combinan métodos CB y CF, aunque no salen de esta categoría aquellos que combinan otros tipos de algoritmos fuera de CB y CF. Como son las hibridaciones de sistemas CB con CF las investigadas en este trabajo es que nos centramos particularmente en ésas.

Los recomendadores híbridos tienen el objetivo se aliviar las carencias de ambos, sistemas CB y CF, con las fortalezas del otro, al usar varias fuentes de información y combinar ambos métodos (Burke, 2002; Parra y Sahebi, 2013; Suriati, Dwiastuti, y Tulus, 2017). Usan tanto datos del uso de los usuarios como datos del contenido de los ítems. Más aun, los métodos híbridos pueden alimentarse de feedback tanto explícito como implícito para minimizar sus debilidades. Esto se puede lograr usando datos implícitos para verificar ratings explícitos, o permitiendo al usuario dar un feedback explícito sólo cuando estime conveniente, y mientras no se tengan suficientes datos explícitos, usar aquellos implícitos. Normalmente los sistemas híbridos logran mejores resultados que sus algoritmos constitutivos (Alharthi et al., 2018; Ricci et al., 2010).

A continuación una descripción breve por cada uno de los esquemas típicos por los que se puede hibridar algoritmos de recomendación (Isinkaye et al., 2015). Por conveniencia los nombres estarán en el idioma de la denominación original, pero se explicarán en el mismo idioma de este documento:

- *Weighted hybridization*: se combinan los resultados de los diferentes recomendadores al integrar *scores* otorgados a cada recomendador, produciendo una lista única de recomendación. El beneficio es que todas las fortalezas de los recomendadores constitutivos están presentes durante el proceso de recomendación.
- *Switching hybridization*: el sistema intercambia entre recomendadores dependiendo de la situación según una heurística que refleje la habilidad del recomendador para producir buenas recomendaciones. La ventaja de esto es que el sistema es sensible a las fortalezas y carencias de los algoritmos constitutivos. La desventaja es que se introduce complejidad en el proceso por los criterios de intercambio, lo que hace incrementar el número de parámetros que el recomendador debe determinar.
- *Mixed hybridization*: las recomendaciones de varios métodos son presentados al mismo tiempo. Este esquema evita el *cold start* de nuevos ítems, ya que los ítems pueden ser recomendados sólo en base a su contenido, aun si no han sido calificados por los usuarios, aunque no evita el *cold start* de nuevos usuarios, ya que ambos métodos CB y CF necesitan datos sobre las preferencias del usuario para partir. A parte, recomendar usualmente requiere jerarquizar los ítems, y esto no es abordado por híbridos mixtos.
- *Feature combination hybridization*: los *features* provenientes de diferentes fuentes de datos usados para entrenar distintos algoritmos son combinados para ser usados en un solo algoritmo. Por ejemplo, combinar los features de un ítem de un sistema CF, que típicamente son los ratings, con los *features* de un vector latente de ítems proveniente de una factorización LDA, las probabilidades de pertenecer a ciertos tópicos, para producir una representación combinada del ítem a partir de diversos tipos de feedback. Lo bueno de esto es que no depende exclusivamente de datos colaborativos.

- *Cascade hybridization*: un sistema recibe como input las recomendaciones producidas por otro sistema, lo que hace que el sistema sea tolerante a recomendaciones ruidosas. Debido a que el segundo paso se enfoca sólo en aquellos ítems para los que se requiere discriminación adicional, es más eficiente que un híbrido *weighted* que aplica todas sus técnicas a todos los ítems.
- *Feature augmentation hybridization*: se incorpora en el procesamiento de la recomendación final el rating o clasificación producido por otro algoritmo. Por ejemplo, Mooney y Roy (2000) crean un recomendador CB de libros basados en datos de Amazon.com, y de entre los features de texto de los ítems se incluyen 'autores relacionados' y 'títulos relacionados', datos que Amazon genera internamente a partir de sus sistemas colaborativos. Híbridos de *augmentation* son atractivos ya que ofrecen una manera de mejorar el desempeño del recomendador de base.
- *Meta-level hybridization*: El modelo generado por una técnica de recomendación es usado como input para otro. El beneficio de los híbridos *meta-level* CB/CF es que el modelo aprendido puede ser una representación comprimida de los intereses del usuario, y un mecanismo colaborativo que le siga puede operar sobre este espacio más chico y más denso de forma más fácil que sobre datos *sparse* de ratings en bruto.

A parte de las categorías anteriores también son populares los enfoques **demográficos**, que se basan en el perfil demográfico del usuario y que recomiendan según diferentes nichos demográficos. Por ejemplo, usuarios que son redireccionados a distintos sitios Web dependiendo de su país o idioma (Mahmood y Ricci, 2007). También están los **basados en conocimiento** (o *knowledge-based)*, que recomiendan basados en conocimiento específico que el sistema tiene sobre el dominio, lo que infieren a través de patrones en los datos y en las interacciones del usuario (Bridge, Göker, McGinty, y Smyth, 2005). Por ejemplo, un usuario de Facebook interesado en noticias debe tener más peso para posts populares

compartidos, mientras que usuarios más bien interesados en las acciones de sus amigos tendrá bajo peso en lo anterior (Ahde, 2015). Por otro lado están los recomendadores **basados en comunidad** (o *community-based*), los que modelan y adquieren información sobre las relaciones sociales de los usuarios y en las preferencias de los amigos del usuario (Arazy, Kumar, y Shapira, 2009). La recomendación puede estar basada por ejemplo en los ratings de los amigos del usuario. Estos RSs siguen el ascenso de las redes sociales online y permiten una adquisición simple y comprensiva de los datos sobre las relaciones sociales de los usuarios (Ricci et al., 2010).

1.3. Sistemas Recomendadores en el Dominio de Libros

Los RSs pueden traer beneficios importantes en escenarios de comercio electrónico, como explicado en la sección 1.2, pero también pueden hacerlo en otros contextos. Por ejemplo, las recomendaciones pueden ser útiles en bibliotecas, escuelas y portales de educación en línea (*e-learning*). Los números dan la idea que la lectura de libros está siendo cada vez más extendida: la proliferación de *e-books*, que dan alternativas económicas y de poco esfuerzo a lecturas; el que Google cuente cerca de 130 millones de libros disponibles en el 2010 (Skipworth, 2010); y el que cerca de un millón de nuevos títulos hayan sido publicados en el Reino Unido, China y EE. UU. sólo en el 2013 (Flood, 2014). Sin embargo, se ha argumentado que la lectura por placer va en descenso, sobretodo entre los jóvenes, como señala Flood (2015).

Varios estudios han demostrado los beneficios de la lectura. En uno compararon, entre adultos, lectores regulares con no regulares. Los primeros se han encontrado que son significativamente más propensos a reportar mejor salud física, mental y mayor satisfacción con la vida (Hill, 2013). En otros, vieron que la exposición a la ficción está correlacionada con mejores habilidades empáticas

(Mar, Oatley, y Peterson, 2009) y con estimular comunicaciones sociales profundas (Mar y Oatley, 2008). Por otro lado, vale la pena invertir en desarrollar sistemas de recomendación automática y personalizada, puesto que estudios como el de Chen (2008) han mostrado que los usuarios confían (*trust*) en los RSs, puesto que estaban más interesados en libros marcados como 'usuarios que compraron este libro también compraron' que aquellos marcados como 'recomendados por personal de la librería', lo que adscribe a recomendaciones no personalizadas.

CAPÍTULO 2. TRABAJOS RELACIONADOS

En esta sección describimos brevemente la historia sobre cómo se ha abordado el problema de recomendación de libros en investigaciones académicas y las técnicas empleadas para este fin.

Los algoritmos CF han sido usados para varios recomendadores (Givon y Lavrenko, 2009; D. Pathak, Matharia, y Murthy, 2013; Rajpurkar et al., 2015; Rana y Jain, 2012; Sase, Varun, Rathod, y Patil, 2015; Vaz, Martins de Matos, y Martins, 2012; Vaz, Martins de Matos, Martins, y Calado, 2012). En el caso de los CB, típicamente el contenido de los ítems lo componen el título, reseña, géneros, autores, año de publicación, editorial, número de página o incluso el texto completo (Alharthi et al., 2018). Éstos son ampliamente usados para recomendación de libros (Alharthi, Inkpen, y Szpakowicz, 2017; Crespo et al., 2011; Garrido, Pera, y Ilarri, 2014; Givon y Lavrenko, 2009; Kapusuzoglu y Öguducu, 2011; Koberstein y Ng, 2006; Mooney y Roy, 2000; D. Pathak et al., 2013; Pera, Condie, y Ng, 2010; Pera y Ng, 2011, 2013, 2014a, 2014b, 2015; Priyanka, Tewari, y Barman, 2015; Sase et al., 2015; Sohail, Siddiqui, y Ali, 2013; Vaz, Martins de Matos, y Martins, 2012; Zhang, Chow, y Wu, 2016; Zhou, 2010). Otros enfoques han sido los recomendadores basados en información demográfica y en comunidades. Zhou (2010) usó la confianza entre usuarios, mientras que Givon y Lavrenko (2009) y Pera y Ng (2011) incorporaron *tags* (metadatos en forma de términos claves que ayudan a describir un ítem) para hacer las recomendaciones. Avances con aquellos basados en conocimiento también se han hecho. Al encontrar patrones en transacciones de librerías, el sistema deduce la compra siguiente (Rajpurkar et al., 2015; Zhu y Wang, 2007). Reglas de asociación y técnicas de *clustering* son muy comunes en un escenario de préstamos de biblioteca, puesto que éstas cuentan con datos demográficos de los miembros, de sus actividades de lectura y con sistemas de clasificación de libros. En éstos se recomienda libros a los clientes según la demografía a la que pertenezcan y a sus hábitos de lectura

(Maneewongvatana y Maneewongvatana, 2010; Tsuji et al., 2014; Yang, Zeng, y Huang, 2009).

Un factor interesante para filtrar libros es el adoptado en Pera y Ng (2013, 2014a, 2014b, 2015), en donde recomiendan a partir de la legibilidad del libro, filtrando así aquellas lecturas lejos del nivel de pericia del usuario. Estos trabajos hacen análisis de tipo *estilométrico* para descubrir el estilo de escritura de los autores preferidos. Usan más que nada técnicas provenientes del área CB, por ejemplo basados en frecuencias de palabras (Vaz, Ribeiro, y de Matos, 2013a), o basados en modelamiento de tópicos (Vaz, Martins de Matos, y Martins, 2012) para extraer features estilométricos.

Los e-books como Kubo o Kindle han abierto puertas para nuevas fuentes de datos sobre el comportamiento de los usuarios, como la duración y repeticiones de sesiones de lectura, permiten también analizar el texto completo del libro y su *metadata*. Investigaciones que hacen uso de esta información son por ejemplo los de Crespo et al. (2011) y Zhang et al. (2016), en donde recomiendan autores y libros electrónicos.

Otra fuente de información importante son las críticas o reseñas (en inglés *reviews*) de los libros hechas por la comunidad online, ya que por una parte permiten a los investigadores examinar los factores que hacen a un libro atractivo a ojos de los lectores; por otra, no hay necesidad de tener el contenido completo del libro, que no siempre es asequible (y aun con él la complejidad del análisis es grande), ya que el libro se puede representar a partir de la perspectiva de sus lectores. Las recomendaciones luego se hacen jerarquizando los libros por la similaridad de sus *reviews* y en la correspondencia de palabras clave. Trabajos de este tipo suelen apoyarse en métodos de análisis de sentimiento con técnicas NLP. Por ejemplo, Priyanka et al. (2015) evalúa el sentimiento de las *reviews* contando las ocurrencias de palabras definidas como 'negativas' y como 'positivas', Sohail et al. (2013) se basa en minería de opiniones para descubrir los gustos de los

usuarios. Por otro lado se pueden aplicar nuevamente técnicas de BoW para calcular similaridades entre reseñas (Koberstein y Ng, 2006). Lugares típicos para extraer estos datos son sitios de comercio electrónico como Amazon.com, o de comunidades de catalogación virtual como Goodreads o LibraryThing.

Otra forma de datos generados por usuario es el contenido de medios sociales, como las anteriormente mencionadas o incluso como Facebook o Twitter. Las publicaciones del usuario y sus amigos pueden ser una fuente para enriquecer los perfiles de usuario y por tanto pueden ayudar a recomendar personalizadamente. Incluso si no tenemos feedback del usuario sobre sus gustos, o si las publicaciones del usuario no hablan acerca de libros o algún otro tipo de productos, usar este tipo de información puede reflejar los intereses del usuario sobre ciertos tópicos, aun en situaciones de *cold start*. El sistema de Pera et al. (2010) mide la similaridad entre libros calificados por el usuario objetivo y los libros de sus conexiones; el de Givon y Lavrenko (2009) usa vectores TF-IDF de etiquetas extraídas de LibraryThing y del texto completo del libro; el de Alharthi et al. (2017) usa embeddings para representar libros de Goodreads y perfiles de usuarios a partir de sus publicaciones en Twitter. Investigaciones en recomendación de libros basado en medios sociales parece prometedor, puesto que las redes sociales son un gran repositorio de datos sociodemográficos, opiniones de los usuarios, e información psicológica.

Datasets

En la tabla 2.1 mostramos algunos de los datasets de dominio público comúnmente usado en investigaciones sobre RS de libros.

Adicionalmente, en Project Gutenberg se puede encontrar el texto completo de más de 50,000 libros fuera de copyright, publicados hace al menos 50 años. La aplicación Goodreads tiene registrados 55 millones de usuarios, 1.5 billones

Cuadro 2.1. Características de algunos datasets usados para recomendación de libros. Versión modificada de la mostrada en Alharthi et al. (2018). Acá incluimos la cantidad de usuarios e ítems en cada dataset y algunas publicaciones que hacen uso de ellas.

	Book-Crossing	LitRec	LibraryThing	INEX
Usuarios	278,858	1,927	7,564	94,000
Libros	271,379	3,710	39,515	2,800,000
Datos demográficos	ubicación, edad	ubicación		
Metadata	título, autores, año, editorial, imagen de portada	título, autores		título, autores, año, editorial
Sinopsis				✓
Texto completo		✓		
Fechas de inicio y término de lectura		✓		
Tags generados por usuario			✓ (2,415,517)	✓
Semántica (mapeo a DBpedia)			✓	
Críticas textuales				✓
Solicitud del usuario por recomendaciones				✓
Investigaciones	Ahn (2008); Garrido et al. (2014); Zeng, Zeng, Shang, y Zhang (2013); Ziegler, McNee, Konstan, y Lausen (2005)	Vaz, Martins de Matos, y Martins (2012); Vaz, Martins de Matos, Martins, y Calado (2012); Vaz, Ribeiro, y de Matos (2013b)	Noia, Ostuni, Tomeo, y Sciascio (2016)	Benkoussas y Bellot (2013)

de libros y 50 millones de reviews. Está disponible un dataset para uso público con alrededor de 226 millones de interacciones en Goodreads [1].

Oportunidades de Investigación

Las técnicas de NLP han sido ampliamente usadas para recomendación de libros, sin embargo son pocos los estudios que utilizan técnicas de word embedding. Pareciera una idea apropiada, dado que los libros están compuestos de grandes cantidades de texto que métodos de NLP pueden aprovechar para su análisis. Si bien esto puede presentar un problema dado el costo de tiempo y poder computacional, sería un despropósito no aprovechar todos los datos posibles si

[1] https://cseweb.ucsd.edu/~jmcauley/datasets.html#goodreads

eso incide en tener mejores recomendaciones, aún más si tratamos con recomendadores basados en modelos, los que son entrenados periódicamente y no cada vez que el sistema deba generar una nueva recomendación. Incluso si no contamos con todo el contenido del libro, en las aplicaciones de catalogación virtual se cuenta con información importante que habla acerca del libro, como los metadatos, citas textuales, etc. El estudio de Alharthi et al. (2017) ha hecho avances en ésta área en particular, pero no profundiza sobre el uso de otros modelos más allá del modelo word2vec de Google news[2] para representar libros de Goodreads y tuits de los usuarios de Goodreads. Una hipótesis de este trabajo es que podría ser más apropiado usar un modelo entrenado con texto de Twitter para representar los tuits de los usuarios. Por otro lado se carece de estudios que corroboren o aprovechen estudios psicológicos sobre la motivación de los lectores con el fin de crear RS. Según Ross (1998), los cuatro elementos más importantes que influyen en la elección de un libro son las siguientes, en orden: el estado anímico, fuentes de serendipia (sorpresivamente encontrar un libro relevante y novedoso), autor y géneros preferidos, y costo físico e intelectual de obtener o leer el libro. Tampoco hay estudios que comparen un amplio rango de recomendadores de estado del arte para los datasets que utilizan en recomendación de libros, son pocos los que integran distintas fuentes de datos con el fin de evaluar el impacto que tiene cada uno sobre las recomendaciones finales, al igual que la cantidad de estudios que evalúan sus sistemas en un estudio con usuarios (Alharthi et al., 2018).

[2] `https://code.google.com/archive/p/word2vec/`

CAPÍTULO 3. PREGUNTAS DE INVESTIGACIÓN Y METAS

3.1. Hipótesis

Dada la gran cantidad de investigación que se ha hecho en el área, supondremos que un sistema que genere recomendaciones automáticas y personalizadas de libros para los usuarios que la utilicen es posible y que puede tener un buen desempeño. Supondremos a su vez que tanto técnicas colaborativas, como basadas en contenido y como sus hibridaciones pueden rendir buenos resultados.

En cuanto a los datos entregados por los usuarios, asumiremos que si un usuario dio un rating a un libro es porque lo leyó de principio a fin. Si el rating es bajo es porque el contenido del libro no refleja los gustos e intereses del usuario; a la inversa si el rating es alto. También asumiremos que los ratings y todo tipo de feedback entregado son confiables como verdadero reflejo de los intereses del usuario, y que por tanto no tratamos con ratings falsos dados por motivos maliciosos que dificulten el buen funcionamiento del sistema de un modo significativo. Finalmente, dado que en el dominio de los libros se da el caso que un mismo libro pueda tener diversas versiones (distintas editoriales, ediciones, etc.), asumiremos que el consumo (y por tanto rating) de un usuario será hacia únicamente una de las múltiples versiones que un libro pueda tener, y que si interacciona con un libro es porque no hay interacciones en el pasado, y no las habrá en el futuro, con otras ediciones del mismo libro.

En cuanto a la psicología y hábitos de lectura del usuario, asumiremos que todos los usuarios cuyos datos de interacciones tratemos son lectores regulares de libros, cuyos modelos psicológicos de lectura a los que adscriban son irrelevantes para efectos de percepción en las recomendaciones (Ross, 1998), pues no hacemos alcance alguno en este trabajo sobre cómo las formas de leer de los usuarios afectan los resultados.

En cuanto a las fuentes de los datos para la creación de perfiles de ítems y de usuarios, asumiremos que las fuentes son confiables, y que la información que se da sobre el libro desde la interfaz en donde se encuentre no viene de una motivación irónica: que las citas textuales sean verdaderamente las que aparecen en el libro, que los autores mostrados sean los mismos escritores, que la reseña describa el libro en cuestión y no otro, etc.

3.2. Preguntas de Investigación

(RQ. 1) **¿Qué enfoque es preferible tener sobre el feedback del usuario para crear un sistema recomendador que produzca las mejores recomendaciones de libros?**

El feedback del usuario sobre un ítem puede tener varias formas, como ratings, comentarios o simplemente saber si es que el usuario consumió el ítem, y podemos transformarlo en otras, por ejemplo mapeando una escala de rating de 1 a 5, a un feedback implícito que según un umbral de rating decidamos si es que al usuario le pareció o no relevante el ítem consumido. Queremos saber qué tipo de feedback usado con qué tipo de sistema recomendador produce los mejores resultados.

(RQ. 2) **¿Cuál es la manera de representar libros y perfiles de usuarios para recomendar libros que produzca los mejores resultados?**

Si consideramos que 'libro' es una entidad compuesta de texto que puede estar en formato impreso, digital o en otro soporte, queremos saber de qué manera codificar y estructurar esa información, y qué información propia del usuario utilizar para producir recomendaciones personalizadas de libros para ellos que más los satisfagan.

(RQ. 3) **¿Existe un sistema híbrido que produzca mejores resultados que los algoritmos investigados en (RQ. 1) y (RQ. 2)? Si es así, ¿qué tipo de hibridación es la mejor?**

En función de buscar el sistema recomendador que de los mejores resultados en esta situación, no descartamos investigar estrategias de hibridación, puesto que apuntan a suplir las carencias de sus sistemas componentes y a aprovechar todas las distintas fuentes de datos posibles. En general estos sistemas muestran mejores resultados al predecir los intereses de los usuarios (Jannach, Zanker, Felfernig, y Friedrich, 2010; Parra y Sahebi, 2013).

3.3. Objetivos y Aportes de la Investigación

El objetivo de este trabajo es responder las preguntas (RQ. 1), (RQ. 2) y (RQ. 3). Para ello evaluamos una amplia gama de sistemas basados en filtrado colaborativo, basados en contenido e híbridos, con datos contextuales de interacciones de usuarios con libros y metadatos, y analizamos el impacto que tiene el usar distintos tipos de feedback y fuentes de datos sobre los resultados. Las evaluaciones son tanto offline como online según métricas de evaluación descritas en las secciones 5.2 y 5.3.

Los aportes de esta investigación al área son las siguientes: hallazgos con respecto al tipo de contenido usado para generar recomendaciones de libros; nuevos resultados de varios recomendadores estado del arte y el efecto que en ellos tienen sus parámetros en el contexto de recomendación de libros; evaluación de los recomendadores de libros tanto offline como con estudio de usuarios y análisis sobre los gustos y preferencias de los usuarios en el estudio online; nuevas directrices que pueden encaminar hacia subsiguientes investigaciones en el área de recomendación de libros.

CAPÍTULO 4. SOLUCIÓN PROPUESTA

Con el fin de responder las preguntas de la sección 3.2, proponemos una evaluación de distintos tipos de algoritmos estado del arte aplicados al dominio de libros, usando una variedad de diferentes fuentes de datos, con experimentos offline y online.

Los algoritmos que proponemos los detallamos a continuación, y pertenecen a tres de las más populares clases de sistemas recomendadores. Por simplicidad, asumamos que contamos con todos los datos necesarios para entrenar y evaluar los algoritmos que se presentan a continuación. Tanto información de interacciones de usuarios con ítems (para métodos CF) como información sobre el contenido de libros (para métodos CB).

4.1. Métodos de Filtrado Colaborativo

Como mencionado en la sección 1.2.1 de la introducción, dos grandes enfoques de las técnicas CF de recomendación son los basados en memoria y los basados en modelo. Explicaremos brevemente los primeros.

Sea I_{xy} el set de todos los ítems co-calificados por ambos usuarios x e y. El enfoque más popularmente usado para calcular la similaridad entre dos usuarios $sim(x, y)$ es mediante un cálculo de una covarianza normalizada llamada el coeficiente de correlación de Pearson:

$$sim(x,y) = \frac{\sum_{i \in I_{xy}} (r_{x,i} - \bar{r_x})(r_{y,i} - \bar{r_y})}{\sqrt{\sum_{i \in I_{xy}} (r_{x,i} - \bar{r_x})^2 \sum_{i \in I_{xy}} (r_{y,i} - \bar{r_y})^2}} \tag{4.1}$$

Una estrategia común es pre-calcular todas las similaridades $sim(x, y)$ y luego re-calcularlas cada cierto tiempo, debido a que normalmente no hay cambios drásticos en la red de pares en tan poco tiempo (Adomavicius y Tuzhilin, 2005).

Los recomendadores CF que investigamos principalmente en este trabajo son aquellos de la familia de modelos de factores latentes, perteneciente al enfoque basado en modelo.

4.1.1. Factorización Matricial

El enfoque de modelos de factores latentes tiene la meta de descubrir factores latentes que puedan explicar los ratings observados. El ejemplo paradigmático es la aplicación del método de descomposición espectral de matrices llamado *descomposición de valores singulares* (*singular value decomposition* o SVD) a la matriz de ratings de usuarios-ítems (en el que vemos un ejemplo en la tabla 4.1) que se produce por la interacción de usuarios con un sistema CF. Los sistemas CF basados en factorización matricial (MF por sus siglas en inglés) son populares por sus buenos resultados y escalabilidad (Ricci et al., 2010). El problema es que SVD no está definida para matrices con valores faltantes, como es el caso en situaciones reales. Por lo general las matrices de ítems son muy poco densas, o bien llamadas *sparse*. Abordar sólo las pocas entradas hace que sea muy propenso sobreajustar el modelo. Esfuerzos previos han investigado formas de rellenar los espacios vacíos de la matriz (Kim y Yum, 2005; Sarwar, Karypis, Konstan, y Riedl, 2000). El problema es que esto puede ser excesivamente costoso puesto que el número de datos crece significativamente. Además, fuese el caso que llegue un usuario nuevo al sistema, la matriz se debería calcular nuevamente. Otra forma de hacer SVD es trabajar sólo con los ratings observados, evitando el sobreajuste mediante un modelo de regularización adecuado (Bell, Koren, y Volinsky, 2007; Canny, 2002; Takács, Pilászy, Németh, y Tikk, 2007).

Bajo el paradigma de factores latentes de usuarios e ítems, los ratings se definen de forma general como

$$r_{u,i} = q_i^T \cdot p_u \tag{4.2}$$

Cuadro 4.1. Ejemplo de matriz de ratings. Las entradas representan ratings en una escala de 1-5. En este ejemplo los 0 corresponden a entradas no observadas y por tanto indefinidas.

	u_1	u_2	u_3
i_1	2	3	5
i_2	1	0	5
i_3	0	2	4
i_4	2	1	5

donde q_i es el vector de factores del ítem i en el espacio latente $\mathbb{R}^r$, y p_u el vector de factores del usuario u en el mismo espacio.

Sea $A \in \mathbb{R}^{m \times n}$ la matriz observada de ratings. Como la descomposición en valores singulares está definida sólo para matrices cuadradas, podemos hacer:

$$A = \sum_{i=1}^{r} \sigma_i u_i v_i^T = U \tilde{S} V^T, \tilde{S} = \begin{pmatrix} S & 0 \\ 0 & 0 \end{pmatrix} \tag{4.3}$$

donde $U \in \mathbb{R}^{m \times m}$ y $V \in \mathbb{R}^{n \times n}$ son cuadradas, y $S = diag(\sigma_1, \ldots, \sigma_r)$ contiene los valores singulares de la descomposición de A. Luego, podemos reducir la dimensionalidad de $\tilde{S}$ al descomponerla en otras 2 matrices:

$$A_{m \times n} = U_{m \times m} \sqrt{\tilde{S}'}_{m \times r} \cdot \sqrt{\tilde{S}}_{r \times n} V^T_{n \times n} \tag{4.4}$$

Si en A, similar a la tabla 4.1, las filas son vectores de ratings por cada uno de los m ítems y las columnas los n vectores de ratings de usuarios, entonces de las filas de la matriz $U\sqrt{\tilde{S}'}$ obtenemos los vectores latentes de los ítems y de las columnas de $\sqrt{\tilde{S}}V^T$ los vectores latentes de los usuarios.

Una gran ventaja que tiene SVD es que puede manejar características adicionales de los datos, incluyendo feedback implícito, información contextual como marca temporal (o *timestamp*) o información del usuario.

Ahora bien, no podemos hacer el tratamiento teórico anterior si la matriz A no está definida para todas sus entradas, como es el caso real. En cambio, los

vectores q_i y p_u se suelen obtener minimizando una función de pérdida como la siguiente:

$$min_{q*,p*} \sum_{(u,i)\in K} (r_{u,i} - q_i^T \cdot p_u)^2 + \lambda(||q_i||^2 + ||p_u||^2) \tag{4.5}$$

donde K es el conjunto de pares (u,i) para los que $r_{u,i}$ es conocido y λ es un parámetro de regularización que controla el sobreajuste. Incrementar λ resulta en menos sobreajuste pero mayor *bias*.

Una manera de optimizar la ecuación 4.5 es usando descenso de gradiente estocástico (SGD) (Funk, 2006) en donde el algoritmo itera sobre todos los ratings del set de entrenamiento. Para cada caso observado $r_{u,i}$, el sistema predice $q_i^T \cdot p_u$ y calcula el error de predicción

$$e_{u,i} = r_{u,i} - q_i^T \cdot p_u \tag{4.6}$$

Luego se actualizan los parámetros según una magnitud proporcional a una tasa de entrenamiento γ en la dirección opuesta al gradiente:

$$\begin{aligned} q_i &= q_i + \gamma(e_{u,i} \cdot p_u - \lambda \cdot q_i) \\ p_u &= p_u + \gamma(e_{u,i} \cdot q_i - \lambda \cdot p_u) \end{aligned} \tag{4.7}$$

Esto se detiene según un número máximo de iteraciones alcanzadas sin convergencia.

Esta solución combina facilidad de implementación con un tiempo de ejecución relativamente bajo (Koren, Bell, y Volinsky, 2009). SVD así entrenado recibe el nombre de *funkSVD*. Una implementación de esto es la creada por Sepulveda, Dominguez, y Parra (2017), y que es uno de los modelos que evaluamos.

Otra manera de optimizar 4.5 es con mínimos cuadrados alternantes (o *Alternating Least Squares*, ALS), lo que es posible cuando se fija el vector de factores de usuario o el de factores de ítems. De esta forma la ecuación se vuelve cuadrática y puede ser resuelta óptimamente alternando entre fijar p_u y q_i, resolviendo problemas de mínimos cuadrados hasta la convergencia. Un caso de uso favorable de esto es cuando se usa en sistemas centrados en datos implícitos. Dado que el set de train no es *sparse*, como en el caso de feedback explícito, no sería práctico iterar por cada par observado (u, i), como lo hace SGD. En ALS el sistema calcula cada q_i independientemente de los otros. Lo mismo para los p_u, por lo que estas tareas se pueden paralelizar (Y. Hu, Koren, y Volinsky, 2008).

Para adaptarnos al caso implícito se debe mapear ratings ordinales $r_{u,i}$ a ratings binarios $p_{u,i}$, donde 0 es feedback negativo y 1 feedback positivo. Esto lo hacemos según el promedio de ratings del usuario de sus interacciones en el set de entrenamiento:

$$p_{u,i} = \begin{cases} 1 & r_{u,i} \geq \bar{r_u} \\ 0 & r_{u,i} < \bar{r_u} \end{cases} \tag{4.8}$$

Notar que tanto los ratings $r_{u,i} < \bar{r_u}$ como aquellos no observados son iguales a 0, ya que el supuesto es que los ítems no observados por el usuario son feedback negativo.

Una librería que implementa esto es la llamada *implicit* por Frederickson (2018), que es con la que trabajamos este método.

Una manera distinta de trabajar con feedback implícito es con *Bayesian Personalized Ranking* (BPR) (Rendle, Freudenthaler, Gantner, y Schmidt-Thieme, 2012). Éste es un método que optimiza sus parámetros para ranking de ítems en una lista de recomendación, lo cual se hace entrenando el modelo con tuplas de la forma (u, i, j), en donde el usuario u prefiere el ítem i sobre el ítem j. Esto

permite el uso de valores tanto positivos como faltantes (casos no observados) en el entrenamiento, al igual que antes.

En nuestro caso esta preferencia la decidimos por diferencia de rating. Por ejemplo, si el usuario dio ratings 5, 4 y 3 a los ítems i_A, i_B e i_C respectivamente, se formarán las siguientes tuplas:

$$(u, i_A, i_B),\ (u, i_A, i_C),\ (u, i_B, i_C)$$

Para abordar el problema de los casos no observados, muestreamos aleatoriamente de entre los ítems no observados una cantidad igual a aquellos observados por cada usuario. A todos los ítems no observados les otorgamos el menor rating posible, con tal de representar el que cualquiera de los ítems no observados sea menos preferido que alguno de los observados.

Una librería de uso público que implementa éste método con factorización de matrices es *fastFM* (Bayer, 2015).

En suma, queremos saber si cada valor de rating es realmente representativo de los gustos del usuario, si es que son representativos de una preferencia binaria o si es que más bien los ratings son reflejo de una mentalidad de gustos asociativa.

4.1.2. Máquinas de Factorización

Una forma de generalizar diversos métodos de factorización matricial es usando máquinas de factorización (FM) (Rendle, 2010). Inspiradas en *Support Vector Machines* (SVM), permiten agregar un número arbitrario de features *(usuario, ítem, contexto)*. Un ejemplo lo vemos en la figura 4.1. Acá se ve que podemos incluir información sobre el contexto de las interacciones, las que se modelan como interacciones entre variables en una regresión polinomial, cuyo objetivo es y, que pueden ser los ratings. Se puede demostrar que cuando no hay contexto, éste método es equivalente a SVD (Rendle, 2010).

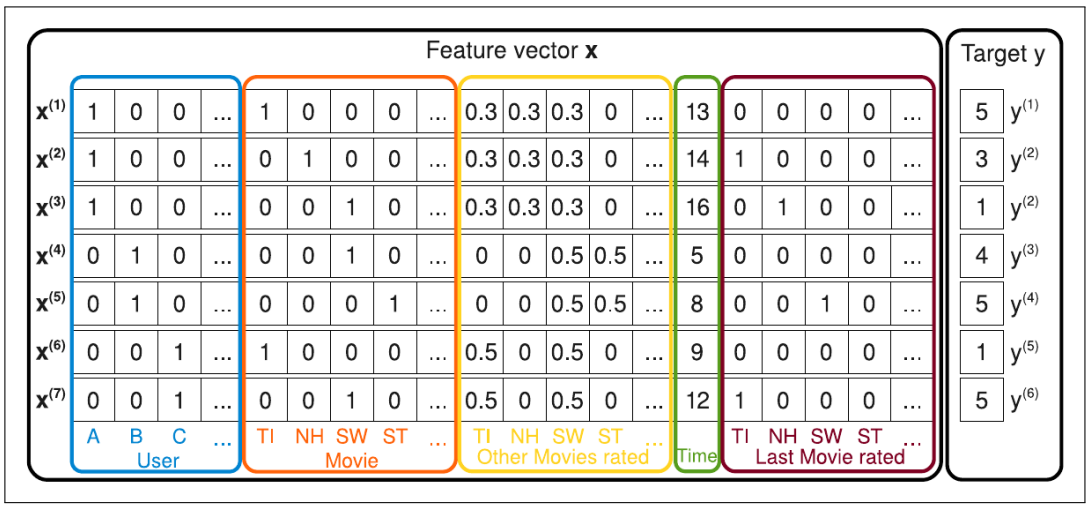

Figura 4.1. Ejemplo de una matriz en donde las interacciones son con películas. La información sobre el contexto de las interacciones que hay corresponde a otras películas vistas, el tiempo y última película calificada. El objetivo son las entradas de la columna y: los ratings. Cada fila $x^{(i)}$ corresponde a una interacción, normalmente entre 1 usuario (representado como sólo 1 valor en las columnas de la sección azul) y 1 ítem (1 solo valor en las columnas naranjas) (Rendle, 2010).

Esto por lo tanto nos permite agregar información sobre el contexto de las interacciones entre usuarios y libros.

Dado que es normal encontrar diferencias en implementación, daremos la libertad de comparar distintas implementaciones de la misma técnica. Librerías que implementan FM son pyFM (Lynch, 2018) y fastFM (Bayer, 2015), este último siendo una implementación en Python de libFM (Rendle, 2012), paquete creado por Rendle (2010).

Típicamente el contexto que ayuda a mejorar los resultados, sin tomar en cuenta el dominio, es el *timestamp* de la interacción. Vaz et al. (2013b) reporta que su recomendador CF de libros obtuvo mayores errores de predicción al sólo

usar ratings recientes e ignorando los más viejos. Por otro lado, el recomendador de libros de Vaz, Martins de Matos, Martins, y Calado (2012) dio mejores resultados al incluir a autores en comparación a no hacerlo. Guiados por estos resultados, evaluaremos tanto el caso de sin contexto, el de timestamp por sí solo, el de incluir autor del libro por sí solo, como también ambas fuentes de contexto combinadas. Por restricciones de tiempo, nos limitaremos a agregar contexto al modelo de la librería que mejores resultados rinda por sí solo sin contexto.

4.2. Métodos Basados en Contenido

Para encontrar la mejor manera de representar libros y de armar perfiles de usuario para recomendar libros es que probaremos dos grandes formas de hacer recomendación basada en contenido. Una es con modelos BoW y otra con modelos de *word embedding*.

4.2.1. Modelos *Bag-of-words*

Como explicado en la sección 1.2.1, podemos asignar pesos $w_{i,j}$ a cada palabra de un documento de un corpus (ecuación 1.4). Dado que documentos que tienen más ocurrencias de un término dado reciben un *score* más alto, y pensando en que podamos tener documentos con muy alta diferencia en número de términos, intentaremos normalizar esta situación penalizando la puntuación de aquellos documentos que contengan mucho de un mismo término, definiendo TF como:

$$\mathrm{TF}_{i,j} = \sqrt{f_{i,j}} \tag{4.9}$$

donde $f_{i,j}$ es la frecuencia de aparición del término i en el documento j, e IDF como:

$$\mathrm{IDF}_i = \log\left(\frac{N}{n_i + 1}\right) \tag{4.10}$$

en donde el denominador $n_i + 1$ da cuenta de los casos en que el término i no aparezca en documento alguno.

Una implementación de esto la podemos encontrar en la clase de similaridad de Apache Lucene para consultas en búsquedas de documentos. A parte de TF-IDF, el buscador de Lucene tiene incluidas otras funcionalidades para personalizar de mejor manera la recuperación de documentos, tales como búsqueda por *fields*, *boosts* agregables a los campos, etc[1]. Como es el estándar, usamos similaridad coseno para calcular distancias entre ítems (ecuación 1.5).

El componente `MoreLikeThis` (MLT) del entorno *Solr* (*MoreLikeThis*, 2017) nos permite hacer uso de esta herramienta, sólo para los documentos que tengamos en en índice de Solr. Debido a esto, creamos un índice en Solr con los documentos que vendrían a representar a los ítems para poder hacer las comparaciones. Estos documentos están estructurados de forma que cada campo represente una característica o metadato del libro para investigar la influencia que cada metadato tiene sobre los resultados.

Por simplicidad, llamaremos a este método *TF-IDF+*.

4.2.2. *Embeddings*

El uso de técnicas de *word embeddings* ya ha sido evaluado en otros estudios con buenos resultados (Alharthi et al., 2017). Sin embargo más investigación falta en el área.

Los modelos de *embeddings* se entrenan con grandes corpus al codificar las co-ocurrencias entre los términos y aprenden regularidades lingüísticas y sutilezas semánticas. Esto pues toman en cuenta la *hipótesis distribucional* (Harris, 1954): "elementos lingüísticos con distribuciones similares tienen significados similares".

[1] http://lucene.apache.org/core/3_6_1/api/core/org/apache/lucene/search/Similarity.html

De este modo el sistema aprende, por ejemplo, que 'perro' y 'gato' son similares en cuanto aparezcan en frases similares: 'mi mascota es un perro', 'mi mascota es un gato', etc.[2] Esta noción de similaridad entre términos en un espacio latente es única para las técnicas de word embedding y de topic modeling, con la diferencia que los modelos de tópicos se entrenan supervisadamente.

Se les suele llamar a estas técnicas modelos de *word vectors*, dado que existe una función (el modelo) que mapea la palabra textual a un vector en un espacio de baja dimensionalidad. En este espacio se pueden definir métricas de distancia entre palabras. De este modo el perfil del ítem $\mathrm{IP}(i) \in \mathbb{R}^{|V|}$ queda representado como una ponderación de los embeddings $\vec{v}_t$ de los términos que lo componen, donde el tamaño del espacio $|V|$ se da como un parámetro al modelo. Esta ponderación puede ser una suma:

$$\mathrm{IP}(i) = \sum_{t \in T_i} \vec{v}_t \tag{4.11}$$

con T_i la lista de términos que describen al ítem i; o una función de *max pooling* para cada una de las dimensiones $k = 1, \ldots, |V|$:

$$\mathrm{IP}(i)_k = \mathrm{maxpool}(v_{1,k}, \ldots v_{|T_i|,k}) \tag{4.12}$$

Luego, siguiendo los procedimientos anteriores, $\mathrm{UP}(u)$ se puede calcular como ponderaciones de los términos de los ítems observados, y la recomendación al usuario u se hará de los ítems i no observados para los que la utilidad $score(\mathrm{UP}(u), \mathrm{IP}(i))$ sea mayor.

Como se sugirió en la sección 1.2.1, la manera en que representaremos un ítem será obteniendo los *word-vectors* de todos los términos que componen el documento. Luego aplicando un procedimiento de *max pooling* a cada dimensión obtendremos un vector-ítem como ponderación de los términos de su contenido, luego de un proceso de remoción de *stop-words*.

[2]Siempre que este tipo de frases pertenezcan al corpus de entrenamiento del modelo

Word2Vec (w2v) es una de las primeras técnicas, propuesta por Mikolov, Sutskever, Chen, Corrado, y Dean (2013). El enfoque aprende una representación en espacio de vectores de términos al aprovechar una red neuronal de dos capas. En la primera, los pesos de la red están aleatoriamente distribuidas. Luego la red es entrenada usando la metodología *Skip-gram* o *Continuous Bag-of-Words* (CBoW) para modelar regularidades sutiles en el uso de las palabras como en el lenguaje corriente. En cada paso, los pesos son actualizados usando SGD, y una representación en el espacio de vectores de cada término es obtenida al extraer los pesos de la red al final del entrenamiento (Musto, Semeraro, de Gemmis, y Lops, 2016).

GloVe, por otro lado, aprende construyendo una matriz de co-ocurrencias que básicamente cuenta qué tan frecuentemente aparece una palabra en un *contexto*. Luego esta matriz se factoriza para obtener una representación de más baja dimensionalidad.

FastText es una extensión de w2v propuesta por Facebook en 2016 (Bojanowski, Grave, Joulin, y Mikolov, 2017). En vez de alimentar términos individuales a una red neuronal, el algoritmo FastText crea varios n-gramas en base a cada palabra. El *word-vector* resultante de cada término es la suma de todos sus n-gramas. Al entrenar la red neuronal, se tienen *embeddings* para todos los n-gramas del corpus de entrenamiento. Así, palabras poco comunes pueden ser representadas adecuadamente dado que es bien probable que algunos de sus n-gramas ya hayan aparecido en otras palabras.

Una popular librería que integra todas estas funcionalidades es la de Gensim para Python (Řehůřek y Sojka, 2010). Haremos uso de 3 modelos pre-entrenados disponibles en `Gensim-data`[3]. Uno es un modelo w2v entrenado con datos de Google News[4]. El dataset de Google News consta cerca de 100 billones de

[3]https://github.com/RaRe-Technologies/gensim-data
[4]https://code.google.com/archive/p/word2vec/

términos. El modelo contiene 3 millones de vectores, uno para cada término y frase, en un espacio de 300 dimensiones. Las frases fueron obtenidas usando un enfoque orientado a datos, básicamente encontrando patrones de palabras muy comúnmente encontradas juntas (como 'New York') (Mikolov, Sutskever, et al., 2013). A este modelo por simplicidad lo llamaremos WE G*.

Un segundo modelo pre-entrenado que usaremos será un modelo FastText entrenado con Wikipedia del 2017, el corpus UMBC webbase y el dataset statmt.org News en conjunto. En total 16 billones de *tokens*. Consta de 1 millón de vectores de dimensionalidad 300. Lo llamaremos modelo WE W* por simplicidad.

Un tercer modelo pre-entrenado es un modelo GloVe entrenado con 2B de tuits, lo que resulta en 27B de tokens, y 1.2M de vectores de dimensionalidad 200. Lo llamaremos WE T* por simplicidad.

La métrica por la que haremos las comparaciones es distancia coseno y distancia euclidiana, dado el estudio de Kapusuzoglu y Öguducu (2011) en el que reportan obtener mejores resultados usando distancia euclidiana para sus experimentos.

Debido a que éstos son modelos pre-entrenados, los parámetros de cada uno están fijos, por lo que no tendremos un proceso de optimización de parámetros para los métodos de *word embedding*.

4.2.3. Modalidades de Recomendación para Métodos CB

Si el usuario estuviera representado por la representación de ítems con los que ha interactuado, entonces dese el caso que el usuario ha interactuado solamente con un ítem. Así, el perfil de este usuario sería equivalente al del ítem consumido, i.e., $UP(u) = IP(i_u)$. Sin embargo en el dominio de libros, el usuario puede haber interactuado con más de un ítem, dado que puede haber leído más de un

libro. Teniendo esto en mente proponemos dos formas diferentes de armar la lista de recomendación final presentada al usuario en nuestras evaluaciones CB:

Modalidad 1

Una alternativa es representar al usuario como sus ítems del set de train de manera disjunta, es decir, cada uno de estos ítems servírselos al modelo por separado. El modelo busca los k ítems más cercanos para cada uno. Luego, la recomendación final corresponde al set de todos los documentos encontrados anteriormente de manera ordenada.

Ordenación de la lista recomendada:

1. El 1er ítem más cercano al 'ítem 1' del set de train del usuario (escogido aleatoriamente de entre los ítems del set de train).
2. El 2do ítem más cercano al 'ítem 1'.
3. ...
4. El k-ésimo ítem más cercano al 'ítem 1'.
5. El 1er ítem más cercano al 'ítem 2'.
6. ...
7. El k-ésimo ítem más cercano al 'ítem X'.

y así hasta encontrar n ítems, con n el tamaño de la lista de recomendación.

Por simplicidad llamaremos a esta configuración 'modo 1' o 'm.1'.

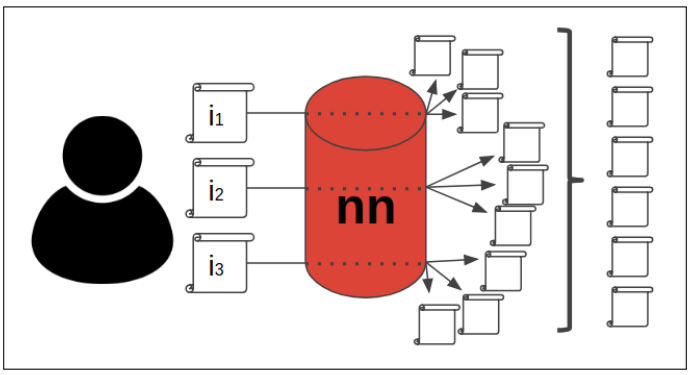

Figura 4.2. Gráfica que representa la Modalidad 1 de recomendación para métodos CB. El modelo busca los ítems más cercanos (los *nearest-neighbors* o *nn*) a cada uno de los ítems del usuario. La recomendación final resulta de una ordenación de los ítems recomendados de todos los ítems del usuario.

Modalidad 2

Representar al usuario como una agregación de los ítems en el set de train del usuario.

Para el caso TF-IDF+, la agregación depende de qué campos se utilicen. Sea un campo F un campo del esquema de estos documentos. Sea i.F el contenido del campo F del documento i. Si usáramos sólo el campo F para hacer la agregación de los documentos del usuario, el documento i_{1+2+3} sería la concatenación de todos los términos de los campos F de los documentos i_1, i_2 y i_3. Qué campo (o campos) F usemos para hacer el testing depende de los resultados del ajuste de parámetros, en donde los campos ideales para hacer las comparaciones (i.e. que produzcan mejores resultados) los tomaremos como un parámetro más del modelo.

Para el caso de *word embedding*, la agregación es el *max pooling* de todos los *word-vectors* de todos los documentos del set de training del usuario. Es decir,

primero armar un documento que contiene todos los términos de los ítems consumidos en el train set del usuario. Luego obtener el *embedding* de cada uno de los términos. Finalmente por procedimiento de *max pooling*, ese usuario queda representado como la agregación de todos esos *word-vectors*.

Para ambos casos la recomendación final resulta luego de darle el ítem agregado como input al modelo.

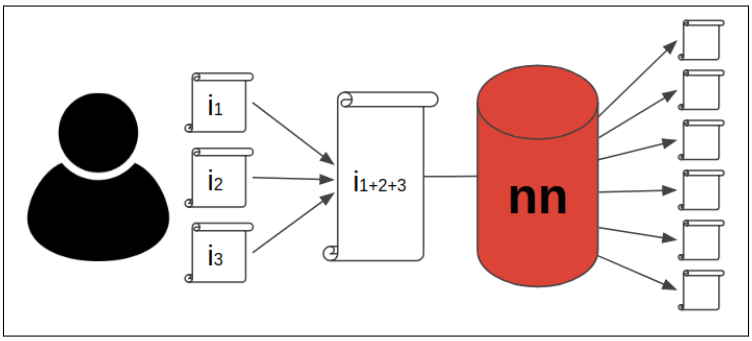

Figura 4.3. Gráfica que representa la Modalidad 2 de recomendación para métodos CB. La agregación depende del método CB usado.

Por simplicidad llamaremos a esta configuración 'modo 2' o 'm.2'. De esta manera cuando nos referiremos a los métodos CB individualmente diremos por ejemplo TF-IDF+ m.1, WE G* m.2, etc.

4.3. Estrategias de Hibridación

En el estudio de Vaz, Martins de Matos, y Martins (2012) se reportó que su sistema recomendador de libros híbrido CB + CF era superior a sus sistemas individuales CB y CF. Dado este y otros estudio es que vemos una posibilidad que un sistema híbrido logre mejor resultado que sus sistemas constituyentes.

Se propone la construcción de un sistema híbrido que integre recomendadores de distinto tipo para suplir sus desventajas. Dada la gran variedad de algoritmos

contemplados, se prefiere estructurar un sistema lo suficientemente genérico tal que pueda trabajar con cualquiera de los algoritmos ya usados sin necesidad de adaptar el sistema para cada caso, de ahí la razón por la que se haya decidido por un esquema de hibridación *weighted* en el que se agreguen ponderadamente las listas de recomendación de cada uno de los algoritmos que participan en la mixtura. Considerando que hay casos en los que las recomendaciones se hacen por vía de predicción de ratings y casos en los que se hacen por comparación de ítems (según distintas maneras de representarlos), independiente de que por otros mecanismos podamos últimamente predecir ratings en sistemas CB (Suriati et al., 2017), o inversamente crear y comparar representaciones de ítems en cuyos features se encuentren los usuarios que lo hayan consumido con los correspondientes ratings, para mantener el sistema lo suficientemente simple y flexible para integrar cualquier tipo de algoritmo, los pesos son usados no dentro de los mecanismos de los modelos que lo componen para recomendar ítems (pesos aplicados a ratings, a features, etc.) sino que fuera, a los resultados inmediatos de los algoritmos que son las listas de recomendación.

La manera en que se hace esto es dando *scores* a los ítems recomendados, los cuales dictarán el ránking en el que aparecerán en la lista fusionada (Bostandjiev, O'Donovan, y Höllerer, 2012; Parra y Brusilovsky, 2015). El score de un ítem recomendado está dado por:

$$score(rec_i) = [\sum_{m_j \in M} \frac{1}{rank_{rec_i,m_j}} \times W_{m_j}] \times |M_{rec_i}| \tag{4.13}$$

donde M es el set de todos los métodos disponibles para fusionar, $rank_{rec_i,m_j}$ es la posición en la lista del ítem recomendado rec_i usando el método m_j y $|M_{rec_i}|$ representa el número de métodos en los que rec_i resultó recomendado. En su formulación original, W_{m_j} corresponde al peso dado por el usuario al método m_j por medio de una interfaz usuario-controlable de Parra y Brusilovsky (2015). Acá lo trataremos como un parámetro del modelo híbrido sujeto a ajuste por proceso

de optimización de parámetros, con 5-fold cross-validation, tal como se explicó en el capítulo 5.

Las hibridaciones que haremos serán con los algoritmos que mejor resultados den de sus evaluaciones. Los parámetros W_{m_j}, al ser 2 los algoritmos que hibridaremos por vez, los llamaremos simplemente α y β.

CAPÍTULO 5. METODOLOGÍA EXPERIMENTAL Y DATOS

En esta sección describimos los datos usados y métodos seguidos para la obtención de resultados según las propuestas del capítulo anterior.

5.1. Dataset

Hacemos uso de un dataset de un corpus de 20,457 archivos recopilado por Zamani et al. (2015).

El objetivo del estudio de Zamani et al. (2015) es el de mejorar la evaluación de compromiso de usuario (o *user engagement*) con aplicaciones. Para eso reúnen datos de varias aplicaciones, cada una relacionada a un dominio distinto: IMDb para películas, YouTube para videos, Goodreads para libros y Pandora para música. Los datos con los que nos quedamos nosotros son aquellos provenientes de la aplicación Goodreads, desde las que extrajimos interacciones $(usuario, libro, rating, timestamp)$. Cada uno de estos servicios permite a sus usuarios expresar públicamente sus sentimientos sobre ítems al permitir compartir sus opiniones y ratings por medio de publicaciones en redes sociales. En particular, es común que al compartir el sentimiento en Twitter aparezca en el campo de texto un contenido predefinido por la aplicación, que indique la calificación del usuario y una descripción básica del ítem, como se ve en la figura 5.1. Todos estos tuits con contenido predefinido de estas diversas aplicaciones son parseados y compilados en el corpus de Zamani et al. (2015). Cada archivo contiene documentos estructurados en JSON los cuales son toda la información rescatable desde las APIs de Twitter. Las fechas en que se recolectaron los documentos son entre enero de 2010 y febrero de 2014.

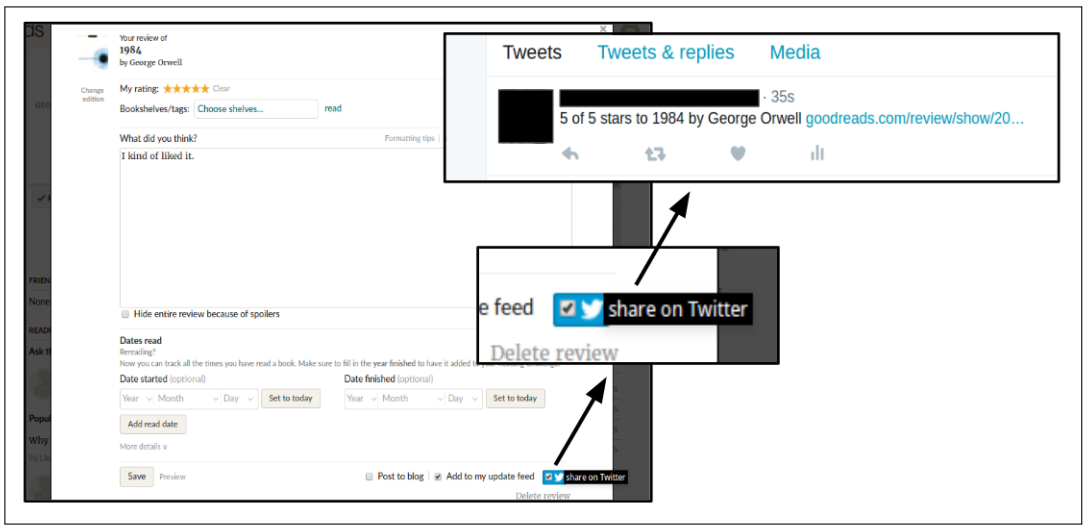

Figura 5.1. Ejemplo de cómo una aplicación sugiere al usuario compartir en Twitter su sentimiento sobre un ítem. En el caso de Goodreads, esto sólo es posible una vez marcado el libro como 'leído'. Es entonces cuando el usuario tiene la posibilidad de compartir su rating. Los usuarios de todas formas pueden dar ratings y escribir críticas sin necesidad de haber marcado el libro como 'leído'.

5.1.1. *Data Scraping* y *Parsing* de Libros y Tuits para Métodos CB

Para métodos basados en contenido necesitamos información sobre el contenido de libros. Para esto se recuperó de la misma aplicación Goodreads datos sobre libros y se compiló con ellos una base de datos con información estructurada de los libros. Se creó un esquema por el cual estructurar los datos de los libros. Estos dependen de la información disponible en Goodreads. El esquema con los principales campos se ve en la tabla 5.1. Algunos campos son obligatorios desde su comienzo. Los usuarios de la aplicación pueden añadir libros nuevos a la base de datos, cuyos campos requeridos son sólo el título y el autor. La aplicación luego se encarga de asignarle una identificación única y otros datos de inicialización (ej. cantidad de ratings y de *reviews* igual a 0).

Cuadro 5.1. Esquema resumido de los datos recopilados de Goodreads. Entre paréntesis la proporción de libros con respecto al total de libros recopilados que cuentan con esos campos. Los que reciben marcas de verificación no la tienen, pues es el caso trivial del 100 % de los documentos. En la tercera columna listamos el número de términos para cada campo promediado sólo para aquellos libros que los contengan.

Campos	Requerido	Prom. térms.	Descripción
URL canónica	✓	1.0	URL que apunta a la página de la -presunta- primera edición del libro
ID del libro	✓	1.0	ID único del ítem otorgado internamente por Goodreads
Sinopsis / Descripción	✗ (91.69 %)	141.5	Descripción textual en pocas palabras del contenido del libro
Título	✓	4.6	El título oficial del libro
Autores	✓	3.0	Las personas involucradas en la creación del libro. Pueden estar los escritores, algunos editores y/o traductores
Libros del mismo autor	✗ (95.97 %)	4.7	Sugerencias de Goodreads de otros libros del mismo autor
Biografía del autor principal	✗ (59.74 %)	169.5	Biografía textual del escritor del libro
Núm. de ratings	✓	1.0	Cantidad de ratings que acumula el libro
Rating promedio	✓	1.0	Rating promedio del libro otorgados por los usuarios
Número de críticas	✓	1.0	Cantidad de *reviews* que ha recibido el libro en su página
Géneros	✗ (81.68 %)	6.5	Un libro puede pertenecer a uno o más géneros, los que son asignados por los usuarios en la página del libro
Votos por género	✗ (81.68 %)	5.0	De los géneros ya asignados los usuarios pueden votar por qué géneros creen que se representa mejor el libro
Citas textuales	✗ (45.67 %)	69.3	Citas extraídas del contenido del libro
Votos por cita	✗ (45.67 %)	1.8	Los usuarios pueden votar por qué citas les gustan más o cuáles creen que mejor representan el contenido del libro
Formato del libro	✗ (73.19 %)	1.3	Formato físico del libro del que se representa en la página en cuestión, como tapa dura, encuadernación en tapa blanca, etc
Núm. de páginas	✗ (90.76 %)	1.0	Cantidad total de páginas del libro de la edición que se presenta en su página

El modo de *scraping* fue el de extraer de Goodreads el HTML de la ruta en Goodreads cada uno de los 50,864 libros que tenemos en el dataset CF. Adicionalmente se estimó un rango de número mínimo (1) y máximo (9,000,000) posible que un ID de libro de Goodreads pueda tener y se iteró sobre un generador de números aleatorio para incrementar esta colección. El número final de documentos una colección de 67,261 ítems.

Los usuarios del dataset para ser usado en métodos CF, dada la forma en que Zamani et al. (2015) recopiló los datos, son tanto usuarios de Goodreads como de Twitter. Con la idea de armar un perfil de usuario a partir de sus tuits es que hicimos *scraping* de los tuits del *feed* de Twitter de los usuarios desde una cierta fecha hacia atrás (18 de enero de 2017). La fecha de publicación más antigua en nuestra recopilación depende de los usuarios, pero globalmente son aquellas publicadas en enero de 2010. La cantidad total de tuits con los que contamos asciende a 284,792, con promedio de tuits por usuario $M = 645{,}78$ y desviación estándar $D.E. = 265{,}20$.

Un histograma de la distribución de tuits por usuario lo podemos ver en la figura 5.2

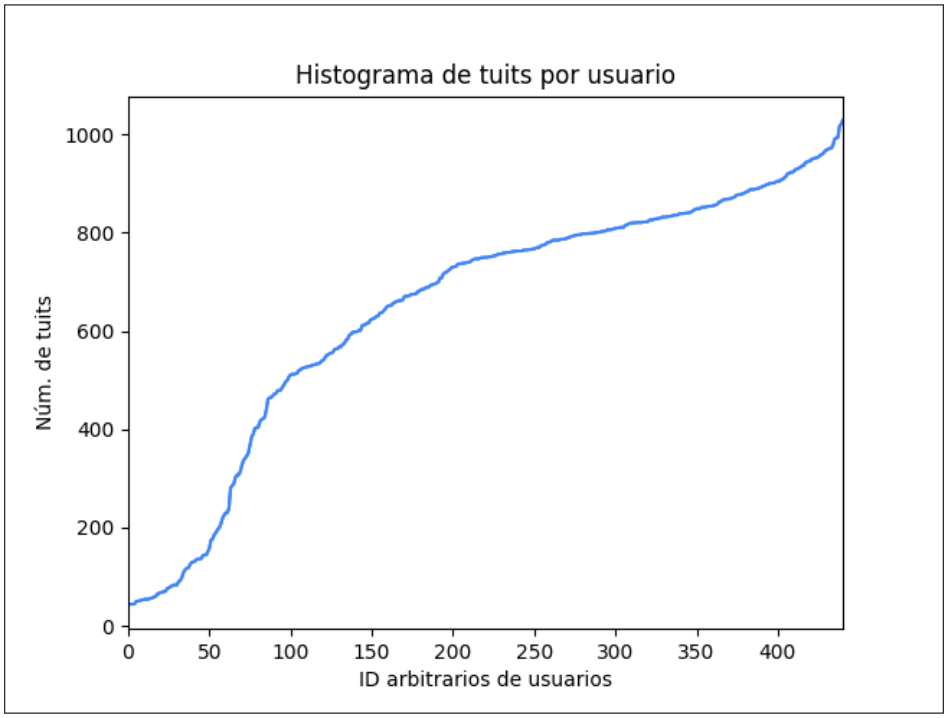

Figura 5.2. Cantidad de tuits que han publicado los usuarios en nuestra recopilación.

En estos tuits están incluidos aquellos con contenido predefinido, tanto de Zamani et al. (2015) como nuevos. Hashtags y otro tipo de contenido es encontrable. Notar que para este propósito no utilizamos las APIs públicas de Twitter, puesto que tienen ciertas restricción de *requests*[1] y nosotros queremos la mayor cantidad de texto posible, por lo que en vez de usar las APIs de Twitter, usamos la librería `selenium` para automatizar este proceso[2]. Notar también que el histograma de la figura 5.2 corresponde a aquellos usuarios con los que nos quedamos luego de la reducción del dataset según el protocolo explicado en la sección 5.2.1.

[1] `https://developer.twitter.com/en/docs/basics/rate-limiting.html`
[2] `https://selenium-python.readthedocs.io/`

5.1.2. Representación del Usuario en Métodos de *Word Embedding*

Dado que del usuario poseemos sus libros leídos como sus tuits publicados, tenemos por tanto más de una manera de representar al usuario. Esto lo aplicaremos a la modalidad 2 para generar recomendaciones de los modelos de word embedding (ver sección 4.2.3), sin embargo no consideraremos los tuits del usuario para recomendaciones en modo 1. Es decir, La razón es doble: (i) el número de tuits por usuario es muy elevado, lo que afectaría negativamente la eficiencia del sistema al recomendar para cada uno de estos tuits, dado que además (ii) cada uno de los tuits, por separado, no presenta contenido lo suficientemente significativo como para discernir acerca de los gustos del usuario, aunque pensamos que en conjunto sí.

Representación por Libros

El documento agregado, que vendría a representar el perfil de usuario en nuestra modalidad 2 de recomendación, es un vector en el *feature space* del modelo WE con el que se crea, que resulta de aplicar *max pooling* a todos los *word vectors* de los términos de todos los libros en el set de entrenamiento del usuario.

Representación por Tuits

El perfil del usuario queda representado por un vector como resultado del *max pooling* de todos los términos de todos los tuits del usuario. La recomendación es el resultado de pasar como input dicho vector de usuario al sistema.

Representación Mixta

Por último representaremos al usuario como la unión de los términos de sus libros y de sus tuits, siguiendo el mismo procedimiento anterior para el tratamiento de sus vectores de palabras. La idea de esta representación mixta es tener la

mayor cantidad posible de información para poder saber acerca de las preferencias y gustos del usuario. El problema de esto es que puede introducir sesgo, ya que los usuarios pueden no solamente hablar de sus gustos mediante tuits, sino de sucesos noticiosos, pueden expresar su opinión sobre un evento o idea, o incluso estar hablando de algo que no les gusta. Queremos evaluar si es que son los tuits, los libros o ambos la forma idónea en este escenario para poder recomendarle los mejores ítems posibles al usuario.

Sea M* alguno de los modelos WE anteriormente presentados. Por simplicidad, si representamos al usuario como una agregación de sus libros, llamaremos a esa configuración WE M* m.2b. Por sus tuits y mixta, respectivamente, m.2t y m.2mix.

5.2. Metodología de Evaluación *Offline*

A continuación precisaremos sobre cómo evaluaremos los sistemas presentados en el capítulo 4.

5.2.1. Protocolo de Evaluación y Partición del Dataset

Debido a la gran *sparsity* del dataset (como vemos en la columna 'Original' de la tabla 5.2) y debido a que queremos imitar lo más posible el escenario de un RS en producción es que adscribimos al método estandarizado para evaluación experimental de recomendaciones top-*N* de Said et al. (2013). Aquel trabajo muestra que el método que proponen pueden capturar de mejor manera la calidad del sistema recomendador que lo que lo hacen métodos de evaluación tradicionales, ya que 'no está afectada por las características de los datos (ej. tamaño, densidad, etc.)'. El modelo de evaluación que ellos proponen estipula que la evaluación tiene un valor óptimo alcanzable, y que la los valores de *accuracy* no están sesgados por el número de ítems con los que los usuarios han interactuado.

En marcos de evaluación tradicionales los splits de set de train y de test se hacen sobre el dataset completo, usualmente en proporción 80-20, con el riesgo de entrenar un modelo con pocos/ningún ítem en el train o test. La propuesta de Said et al. (2013) podemos resumirla de la siguiente forma:

- *Selección de ítems*. **Problema**: cuando en el test set el número de ítems relevantes por usuario es menor que el tamaño de las listas. Si #ítems-relevantes $< N$, la medida de $precision@N$ nunca tendrá valor igual a 1.0, o sea nunca se alcanzará un óptimo, incluso si en la lista recomendada se encuentran sólo ítems relevantes para el usuario. **Solución**: definir sets de testing con exactamente N ítems relevantes. Para esto, se define un *threshold* de relevancia para los ítems según las fuentes de feedback (ej. ratings, feedback implícito). **Consecuencia**: *precision* = *R-precision*, donde *R-precision* es la *precision* al nivel del *recall*.
- *Selección de usuarios*. **Problema**: a evaluaciones con bajo N, usuarios con muchos ítems en el test set tienden a lograr mayor *accuracy* que aquellos con baja cantidad de ítems. **Solución**: imponer restricción M de mínima cantidad de ítems calificados, $M > N$, para elegir qué usuarios estarán disponibles para las evaluaciones a distintos tamaños de lista N. **Consecuencia**: distintos sets de train y test para distintos N, dado que habrá desigual cantidad de usuarios. Para cada usuario: exactamente N ítems en test set, y al menos $M - N$ en train set.

Implementación de Said et al. (2013):

- Splits de training y testing personalizados para cada usuario: se entrena un RS para cada usuario con todo el dataset, salvo los ítems del set de test del usuario candidato.
- *Threshold* de relevancia es 'agnóstico del tiempo': sólo tomar en cuenta el rating para hacer las particiones, no el tiempo (i.e. el *timestamp* de las interacciones), dado que tienen el supuesto que los ratings de los usuarios

son 'aspiracionales en vez de ser reflejo de su actividad diaria' y porque, a pesar de hacer el split más realista al ordenarlos temporalmente, haría impráctica la comparación con un enfoque de evaluación tradicional.

- Límite de rating para considerar sobre éste ítems como relevantes:

$$Threshold = 0{,}5^q \times \sigma(u) + \mu(u) \tag{5.1}$$

  q se inicializa con 0, e incrementa hasta que se hayan hallado N ítems relevantes para el test set del usuario u. μ es el promedio de ratings de u y σ su desviación estándar.
- Selección aleatoria de los nuevos ítems relevantes encontrados si con el siguiente valor de q se encuentran más de N ítems relevantes.
- El usuario se descarta si no tiene al menos N ítems relevantes.
- $M = 2N$. Experimentos con tamaños de lista de recomendación $N = \{1, 5, 10, 20, 50, 100\}$

Nuestra implementación:

- Modelos se entrenan una sola vez, y este modelo vale para todos los usuarios. Es decir, no hay distintos sets de train y test para cada usuario, con el que se entrene un modelo personalizado para cada uno de ellos.
- Adaptación para cross-validation tuning:
  - Ordenar temporalmente todas las interacciones por usuario.
  - División de dataset en 5 folds: 4 para train/validation y 1 para test. En el fold de test están los N ítems relevantes más recientes.
  - Se van eligiendo ítems relevantes para el test set desde las interacciones más recientes según el límite de relevancia de la ecuación 5.1.
  - Las al menos $M - N$ interacciones que corresponden al train set (la agrupación de los 4 folds de train) son repartidas en partes iguales en los 4 folds de train
- Ajuste de parámetros de los modelos:

- Se fija el set de parámetros con valores por defecto dependiendo del algoritmo.
- Se varía el valor de un parámetro en un intervalo fijo de valores posibles. Se fija como valor óptimo el que da la métrica más favorable: más bajo RMSE para métodos CF, más alto nDCG para métodos CB e híbrido.
- Se repite para el resto de los parámetros.
- Al ir iterando en cada posible valor de parámetro, se entrena y evalúa 4 veces el modelo: por cada vez, se evalúa con uno de los 4 folds de train (el 'fold de validación') el modelo entrenado con la agregación de los otros 3 folds de train.
- La métrica de evaluación obtenida por cada set de parámetros corresponde a la media de los resultados obtenidos en las 4 pasadas de entrenamiento-validación hecha con los folds de training y validation.

- Métrica de evaluación final:
  - Se entrena el modelo con el set de parámetros óptimos encontrado, con los 4 folds de validación agregados (o llamado simplemente el 'set de entrenamiento').
  - El testing se hace con los datos del fold de test, los cuales son temporalmente posteriores al set de entrenamiento.
- Dataset:
  - Se creó un dataset con $N = 20$, $M = 2N = 40$, con el cual se hizo el tuning y el testing con métricas de evaluación $@\{5, 10, 15, 20\}$

Observaciones:

(i) Un mismo usuario nunca tendrá un fold de train/validation con más de 1 interacción con respecto a otro fold de train/validation.

(ii) No necesariamente hay ordenación temporal de las interacciones entre los folds de train

(iii) No necesariamente hay ordenación temporal para las interacciones dentro de un fold de train

Ejemplo de una situación $M = 2N$, $N = 10$: sea un usuario con 25 ítems en total, 10 de ellos relevantes:

train/val	train/val	train/val	train/val	test
4	3	4	4	10

Para este usuario habrán exactamente 10 interacciones con ítems relevantes en el fold de test y 15 distribuidos equitativamente entre los folds de train/validation.

Al aplicar esta partición al dataset que obtuvimos de Zamani et al. (2015), observamos que las estadísticas cambian desde la columna 'Original' a la de 'Protocolo' en la tabla 5.2

Cuadro 5.2. Comparación de las estadísticas del dataset de Zamani et al. (2015) con las estadísticas del dataset actual, producto de la reducción por el protocolo de Said et al. (2013) con $N = 20$, número de ítems en el fold de testing para todos los usuarios.

	Original (Zamani et al., 2015)	Protocolo (N=20)
Usuarios	3,968	441
Libros	50,864	23,716
Ratings	78,851	29,701
Rating prom.	3.87 ± 0.98	3.60 ± 0.96
Ratings por ítem	1.55 ± 2.13	1.25 ± 0.88
Ratings por usuario	20.16 ± 31.14	67.35 ± 40.29
Densidad	0.039 %	0.284 %

Vemos que el número de usuarios disminuye bastante, y que la densidad de la matriz de ratings aumenta considerablemente dado que hay más ratings por

usuarios. Esto alivia los tiempos de procesamiento para el entrenamiento y testing de los modelos CF.

Para que las evaluaciones CF fueran justas, se evaluaron estos sistemas con la cantidad de ítems del dataset cortado por protocolo (23,716 ítems), esto es, dejar en el espacio de ítems recomendables sólo aquellos que alguna vez consumió alguno de los 441 usuarios, y no con la totalidad de ítems de nuestra colección (67,261 ítems). Esto para no tener obvios problemas de *cold start* de ítems, lo que perjudicaría sin razón los resultados. Para las evaluaciones CB dejamos los 67,261 ítems para poner a prueba si los métodos CB tan frecuentemente usados en la literatura son los convenientes para un contexto de recomendación de libros y para asemejar al escenario real en que el número de ítems puede estar muy por encima del de los usuarios

5.2.2. Selección de Campos en Evaluaciones TF-IDF+

Dado que creemos importante analizar la relevancia que cada campo de los documentos (i.e. cada metadato de los libros) tiene sobre los resultados, es que daremos especial atención al parámetro de qué campos usar para las comparaciones TF-IDF+. Qué field de los documentos utilizar para hacer las comparaciones está regido por el parámetro `mlt.fl` del componente MLT. Éste es solo uno de los tantos parámetros que optimizaremos en los modelos TF-IDF+[3].

Los campos presentes en nuestro esquema se describen en la tabla 5.1, sin embargo usaremos sólo un subset de éstos, que son los que creemos por intuición más relevantes al momento de decidir qué libro consumir. Estos son: (i) descripción, (ii) título, (iii) géneros, (iv) autores, (v) citas textuales, (vi) autores + descripción, (vii) autores + descripción + título, (viii) autores + descripción + título + géneros + citas textuales. Nuestra suposición es que cuanto más texto el método

[3]Descripción de los otros parámetros en `https://wiki.apache.org/solr/MoreLikeThis`

tenga para hacer comparaciones mejor los resultados. Esto sólo para el modo 1 de recomendación, puesto que para el modo 2 tuvimos la limitación de sólo poder utilizar 1 campo para las comparaciones (descripción o género o autores, etc., y no autores + descripción, etc.). Para el modo 2 el set de valores sobre el que optimizar el parámetro `mlt.fl` fueron los siguientes: (i) descripción, (ii) título, (iii) géneros, (iv) autores, (v) citas textuales. Creemos que el campo de la descripción por sí solo obtendrá buenos resultados, dado que es un campo que describe al ítem con varios términos, y en éste se encuentran importantes palabras clave que hablan acerca del libro, incluidos algunos aspectos sobre personajes, autores, géneros, etc.

Como TF-IDF no es un método que haga predicciones de ratings, la forma en que ajustaremos cada parámetro es según la métrica de ránking nDCG.

5.2.3. Métricas de Evaluación

Dado el amplio espectro de algoritmos, daremos uso de las típicas métricas de evaluación usadas para medir algoritmos de recomendación. En general medimos todo con métricas de ránking, y además los RS de CF los mediremos con métricas de predicción de rating

Métricas de Predicción

Sea N el número total de ratings en el set de ítems. Sea $r_{u,i}$ el rating observado de u hacia i y $\hat{r}_{u,i}$ el rating predicho por el sistema. Dos de las métricas más populares son *MAE* (*Mean Absolute Error*) y *RMSE* (*Root Mean Squared Error*) para medir exactitud del sistema en las predicciones (Isinkaye et al., 2015):

$$MAE = \frac{1}{N} \sum_{u,i} |\hat{r}_{u,i} - r_{u,i}| \tag{5.2}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(\hat{r}_{u,i} - r_{u,i})^2} \tag{5.3}$$

RMSE pone más énfasis en errores absolutos grandes, y mientras más chico el RMSE sea, mejor la exactitud de la predicción.

Métricas de Ránking

Dado que nuestros sistemas recomiendan listas ordenadas de ítems, asumiendo que el ítem en el primer lugar de la lista es el estimado con mayor utilidad para el usuario dentro de los de la lista, y el último el con menor. Existen ciertas formas estandarizadas de evaluar la precisión de ránking en recomendaciones top-n, las que explicamos a continuación.

Sea $rel(i)$ la relevancia real (observada) del ítem en el puesto i. Se define *DCG* (*Discounted Cumulative Gain*) (Järvelin y Kekäläinen, 2002) como:

$$DCG = \sum_{i}^{p} \frac{2^{rel(i)-1}}{\log_2(1+i)} \tag{5.4}$$

y que mide qué tan efectivo es el método de recomendación al ubicar los ítems más relevantes más 'arriba' (en los primeros puestos de la lista) y los menos relevantes más abajo. Usualmente el *normalized DCG* (*nDCG*) se usa más frecuentemente, dado que permite comparar con el DCG de listas de diferente largo. Es calculada al normalizar el *DCG* de una lista ordenada de ítems recomendados por el orden ideal que tendrían los ítems en la lista si es que fueran rankeados perfectamente (*iDCG*):

$$nDCG = \frac{DCG}{iDCG} \tag{5.5}$$

Por otro lado, la precisión (*Precision* o simplemente *P*) es otra de las métricas más usadas en recuperación de información. La precisión es la fracción de ítems

recomendados que son relevantes, definida como:

$$Precision = \frac{|\text{Ítems relevantes recomendados}|}{|\text{Ítems en la lista}|} \tag{5.6}$$

El número de ítems en una lista recomendada puede ser muy alta dependiendo del contexto, y por tanto a veces no es factible que el usuario las vaya a evaluar todas. Por esto es que en escenarios de recomendación se presente más comúnmente el *Precision@n*, con n el tamaño de lista de recomendación, la que da la proporción de ítems relevantes en la lista de recomendación hasta el puesto n:

$$Precision@n = \frac{\sum_{i=1}^{n} Rel(i)}{n} \tag{5.7}$$

Lo que en realidad utilizamos proviene del *Average Precision* (*AP*), la que se calcula al promediar la precisión cada vez que encontramos un ítem relevante, y que por tanto describe la tasa promedio con el que salen ítems relevantes en la lista:

$$AP = \frac{\sum_{k \in K} P@k \cdot Rel(k)}{|\text{relevantes}|} \tag{5.8}$$

pero como tenemos listas de distinto tamaño, y como no siempre sabemos de antemano el número de relevantes (a pesar que en este caso sí, pero usaremos la siguiente métrica para fines comparativos), nos referiremos mejor al *Average Precision at n* (*AP@n*):

$$AP@n = \frac{\sum_{k \in K} P@k \cdot Rel(k)}{|\text{relevantes hasta } k|} = \frac{\sum_{k \in K} P@k \cdot Rel(k)}{\text{mín}\left(|\text{relevantes}|, n\right)} \tag{5.9}$$

el cual da la tasa promedio por el que se van encontrando ítems relevantes si consideramos que la lista es de tamaño n. Luego para obtener una sola métrica que dé cuenta de todos los *AP@n*, la *Mean Average Precision* (*MAP*) se obtiene de promediar todos los *AP@n* por todos los usuarios:

$$MAP@n = \frac{\sum_{u} AP@n_u}{|U|} \tag{5.10}$$

Además, como sabemos de antemano el número exacto de ítems relevantes por usuario en su set de test, podemos usar la métrica $R - precision$ (Manning, Raghavan, y Schütze, 2008):

$$R - prec = \frac{r}{|Rel|} \tag{5.11}$$

donde $|Rel|$ es el número de documentos relevantes para la consulta, en este caso el número de libros relevantes que sabemos que tiene el usuario de entre los ítems que podemos recomendar, y r el número de ítems relevantes que fueron recomendados en la lista. Said et al. (2013) argumenta que *R-precision* es un mejor estimador de la calidad del sistema que *Precision@k*, puesto que si el usuario tiene menos de L ítems, usar *Precision@L* puede tener efectos negativos, por lo que es más adecuado un $Precision@|Rel|$, que sería justamente la definición de *R-precision*.

MRR (*Mean Reciprocal Rank*) por su parte es una métrica más simple. MRR es igual a la inversa del puesto del primer ítem relevante:

$$MRR = \frac{1}{r} \tag{5.12}$$

Para efectos de decidir relevancia en ítems, todas las funciones sobre relevancia $rel(i)$ serán binarias: 0 si el ítem es relevante o 1 si no lo es. Como sólo hay ítems relevantes en el set de test (aunque claramente los ítems relevantes de un usuario no lo serán necesariamente para otro), decidimos que $rel(i) = 1$ si es que el usuario consumió el ítem recomendado, pues por construcción será relevante (ya que sólo hay ítems relevantes en el set de test), y $rel(i) = 0$ en el caso contrario.

Métricas de Diversidad

Usamos una métrica de diversidad entre-listas definida por Lathia et al. (2010) para examinar si es que hay diferencias significativas entre sets de ítems de las

listas generadas por los sistemas híbridos y sus constituyentes:

$$diversity(L_1, L_2, N) = \frac{|L_2 \setminus L_1|}{N} \tag{5.13}$$

donde $L_2 \setminus L_1$ es el set de todos los ítems presentes en la lista de recomendación L_2 no presentes en la lista L_1. Con el fin de averiguar posibles correlaciones entre la diversidad percibida en las listas por los usuarios en nuestro estudio online y la diversidad calculada es que proponemos un método basado en word embedding para cálculo de diversidades intra-lista. Según el modelo de word embedding con mejor resultados, representamos cada ítem de la lista como una ponderación de sus *word vectors* como explicado en la sección 4.2.2. Luego calculamos distancias coseno entre ellos y el promedio es la diversidad de la lista.

5.3. Metodología de Evaluación *Online*

Trabajamos con el supuesto de que los usuarios tendrán un comportamiento similar e tiempo real en comparación al evaluado offline (Alharthi et al., 2018), sin embargo mejoras pequeñas de predicción en los algoritmos no siempre se traducen en una mejor percepción de los usuarios (Konstan y Riedl, 2012). La precisión de los algoritmos es sólo uno de los factores que influencian la aceptación de las recomendaciones por parte de los usuarios. Debido a que no podemos asegurar que el resultado offline se replicará exactamente igual al implementar los sistemas evaluados con usuarios en un entorno online, resulta necesario hacer un estudio con usuarios.

Llevamos a cabo un estudio con usuarios regulares de la aplicación Goodreads, en donde les presentamos listas de recomendación generadas por los 3 mejores algoritmos bajo nombres falsos ('A', 'B' y 'C'). En caso que los métodos híbridos den mejor resultado evitaremos su evaluación con usuarios para no hacer las comparaciones entre métodos injustas.

Se obtuvo el consentimiento de los usuarios para que nos enviaran sus libros de Goodreads ingresando a sus cuentas. De Goodreads se puede generar un documento CSV con datos sobre el ID de los libros, título, autores, rating del usuario, rating promedio, año de publicación, entre otros metadatos[4]. En Goodreads los usuarios pueden ingresar libros a los estantes virtuales de sus cuentas como *leídos*, *leyendo* o *por-leer*, información incluída en estos documentos. Fueron todos los libros los que se tomaron como parte del input para generar recomendaciones a los usuarios. En caso que un libro de algún usuario no estuviera en nuestra colección de documentos, repetimos el procedimiento presentado en la sección 5.1.1 para descarga y estructuración de datos de libros desde Goodreads.

En cuanto al espacio de recomendación, no son lo mismo los de los métodos CF que de los métodos CB. Los métodos CF recomendarán sólo de entre los 23,716 ítems que componen la matriz de ratings (tabla 5.2); en cambio los CB, alguno de los 67,261 ítems de la colección. Tenemos presente este sesgo como limitación de este estudio en expensas de evaluar de manera offline los métodos CB según un escenario más realista.

Claramente para el caso online, los datos de entrenamiento de los modelos probados de forma online son los 5 folds del dataset descritos en la sección 5.2.1.

Procedimiento

A cada usuario se le presentan listas de 10 recomendaciones, cada una producida por los 3 mejores recomendadores menos los híbridos (en total 30 ítems recomendados). Se les pide que evalúen, para cada ítem:

- La **relevancia** que el ítem tiene para ellos (debido a la asunción típica de que dado un ítem relevante es bien probable que el usuario lo consuma (Ricci et al., 2010)) en una escala de 3 puntos:

[4]Botón *Export Library* en `https://www.goodreads.com/review/import`

    - *relevante*: libro relevante a sus intereses, algo que les interesaría leer.
    - *algo relevante*: libro medianamente relevante, algo que podrían considerar leer en el futuro pero que pueden no estar muy interesados.
    - *no relevante*: libro no relevante a sus intereses, algo que no estarían dispuestos a leer.
- La **novedad**, o qué tanto el usuario estaba consciente sobre su existencia, en una escala de 3 puntos:
    - *novedoso*: libro cuya existencia no sabían, o que no hubieran encontrado fácilmente.
    - *algo novedoso*: libro que no conocían pero podrían haberlo encontrado fácilmente.
    - *no novedoso*: libro en el que estaban conscientes de su existencia.
- Al término de cada lista les pedimos que nos den su apreciación sobre la misma según su **satisfacción**, en una escala de 2 puntos:
    - *satisfecho*: en general están conformes con las recomendaciones, utilizarían nuevamente el sistema para buscar libros.
    - *no satisfecho*: en general no están conforme con las recomendaciones, no volverían a utilizar el sistema.
- Al final les pedimos que nos digan justificadamente su recomendador preferido.

La interfaz mediante la cual se enviaron y evaluaron las recomendaciones fue por medio de un mensaje a cada usuario vía correo electrónico. La plantilla que se usó para enviarle las recomendaciones a los usuarios y pedirles su opinión se puede encontrar en el apéndice.

Las métricas usadas para evaluar estos resultados son las mismas que para el caso offline, salvo dos diferencias: (i) obviamos las métricas de predicción pues no pedimos que los usuarios den rating a los ítems recomendados y tampoco inferimos ratings a partir de sus opiniones, y (ii) en vez de *R-precision*, dado que no sabemos de antemano los ítems relevantes para los usuarios en el espacio

de ítems recomendables, usamos *Precision@10*, que es el tamaño de las listas de recomendación presentadas a los usuarios. Para calcular diversidad percibida proponemos un cálculo simple de tipo acumulativo similar a *Precision@k*:

$$Diversidad\ percibida@n = \frac{\sum_{i=1}^{n} Nov(i)}{n} \tag{5.14}$$

en donde $Nov(i)$ es 1 si al usuario el item i le parece novedoso y 0 si no según un cierto *threshold*, y n es el tamaño de la lista de recomendación. Este threshold será mismo que se aplicará a las otras métricas de ránking para discernir relevancia (nDCG, MAP, MRR y precision). Como las evaluaciones de los usuarios a los ítems recomendados tienen forma de 3 posibles valuaciones nominales que se pueden traducir en escalas de ratings de 1 a 3, los valores de threshold que se usarán será tanto de 2 como de 3, y compararemos esos resultados. Así, si el threshold de relevancia y novedad es igual a 3, el ítem calificado como 'relevante' por el usuario (que podemos traducir como un rating de '3'), el ítem será considerado como relevante para los efectos de cálculo de métricas de ránking, y si el mismo es calificado como 'algo novedoso' por el usuario (que podemos tratar como un rating de '2' en cuanto a novedad), el ítem será no novedoso para efectos del cálculo de la expresión 5.14.

Si bien a un usuario una lista le puede parecer completamente no novedosa, siendo por ejemplo sólo libros que el usuario ya había leído, aún puede pensar que la lista es diversa pues, de entre esos ítems no novedosos, puede parecerle todos bien distintos entre sí. Nuestra definición arbitraria de 'diversidad' tiene más bien el fin de poder interpretarse como diversidad sesgada a favor de libros nuevos con respecto a la percepción del usuario. Según la ecuación 5.14, entre más alto sea este resultado mayor percepción de novedad en general tendrá el usuario con la lista de recomendación. Estamos suponiendo que si al usuario le parecen novedosos todos los ítems, es porque no tenía forma de encontrarlos por sí mismo (en el caso extremo), y que si el usuario marca muy novedosos un par de libros de la lista, es porque, a partir del conocimiento sobre uno (o sobre cualquier

otro libro de la lista), el usuario no creía tener manera de encontrar el otro ítem muy novedoso. Por ejemplo, dese el caso que a un usuario le pueda parecer muy novedoso un libro de una serie y muy relevante, pero muy poco novedoso y muy relevante otro libro de la misma seríe. Estamos suponiendo que esto es siempre porque aquel usuario, bajo sus propias vías, a partir del libro novedoso-relevante, sabrá de la existencia de libros de la misma serie, que justamente por ser de la misma serie serán fácilmente encontrables, y que serán por este motivo no-novedosos para el usuario. Esta definición de 'diversidad' toma en cuenta ese tipo se situaciones. Claro que no debe interpretarse altos resultados de esta cantidad como 'buenas' o 'malas' por sí solo, sino en acompañamiento de las métricas de relevancia que estamos calculando.

CAPÍTULO 6. RESULTADOS

Presentamos los resultados de los experimentos *offline* y *online*. Por fines de simplicidad verbal, diremos que un algoritmo es 'mejor' que otro si, de los valores de las métricas producidas al evaluar ambos algoritmos, tanto de ránking como de predicción de rating, las del uno son mayores que las del otro (razonamiento inverso para definir la expresión 'peor'). Sobre qué métricas nos estamos refiriendo, cuántos valores tomamos en cuenta y de cómo las agregamos para realizar estas comparaciones dependerá del contexto en el que estemos hablando.

En las tablas 6.1, 6.2 y 6.3 se resumen los resultados de los experimentos offline. Cada valor numérico corresponde al valor de nDCG@10 obtenido por cada método. El algoritmo aleatorio obtiene según esta métrica un valor de 0.00201. Como es de esperarse es un resultado bajo. Sin embargo resultados tan o más bajos que éste son los que obtuvieron algunos métodos basados en contenido, particularmente los métodos con word embedding modo 2 con distancia euclidiana (salvo el modelo FastText con corpus de entrenamiento de Wikipedia, representación por libros y mixta) y aquellos con representación del usuario a base de sus tuits. De los algoritmos basados en contenido, los que más alto nDCG@10 obtienen son TF-IDF+ (m.1 sobre m.2), el modelo word2vec con corpus de Google (distancia euclidiana sobre distancia coseno) y el GloVe con corpus de Twitter (distancia euclidiana m.1). El peor CF (fastFM sin contexto) es cercano al random, y el mejor CF (implicit) obtiene a penas la mitad de lo que obtiene el más alto de los CB (TF-IDF+ m.1). Éste último es recién comparable con el que se queda en séptimo lugar, según la métrica en cuestión, de los CB (WE W* distancia coseno m.1), y son 8 en total los recomendadores CF probados, lo que quiere decir que en buena parte los mejores recomendadores CB obtienen resultados superiores a los CF. Por otro lado, es sorpresivo lo bajo de algunos resultados de métodos basados en contenido. Se esperaba que usando un modelo de word embedding entrenado con tuits (WE T*), al hacer corresponder a los usuarios,

Cuadro 6.1. Resumen de resultados offline para algoritmos CF según método de entrenamiento y contexto. En negrita las 2 combinaciones con mejores resultados.

Entrenamiento	Contexto	funkSVD	fastFM	pyFM	implicit
SGD	s/c	.00611	.00211	.03021	
	timestamp (1)			.04472	
	autores (2)			**.05833**	
	(1) + (2)			.02908	
BPR	s/c		.00760		
ALS	s/c				**.08646**

representados por sus tuits, con los libros, hubiera una correlación favorecedora, sin embargo esta combinación es tan mala o peor que la recomendación aleatoria. De hecho la representación por tuits, independiente del modelo, es lo que peor funciona, al punto que produce 0 recomendaciones relevantes en el top 10 de las listas a todos los usuarios con similaridad euclidiana. Por el lado de los métodos CF, fue de esperar que al ir agregando contexto los resultados fueran mejorando, pero lo que mejor funciona es considerar los ratings como feedback implícito. En cuanto a los sistemas híbridos, si bien obtienen altos resultados, relativos al resto de los resultados, por lo visto no alcanzan a ser mayores que el más alto, aquel producido por el CB componente del híbrido. Si bien se esperaba que el híbrido lograra mejores resultados en métricas de ránking, las diferencias son mínimas. En cuanto al estudio online, los usuarios percibieron los ítems como relevantes bastante más frecuentemente que en los cálculos del estudio offline. El algoritmo en el estudio offline que mejor resultados obtuvo mantuvo su jerarquía en el estudio con usuarios. No fue así con los otros dos algoritmos que se evaluaron (como se ve en la tabla 6.4), presumiblemente por una mayor percepción de diversidad en las listas, bajo la asunción de que mayor diversidad implica mayor satisfacción de los usuarios en las listas recomendadas. El algoritmo preferido por los usuarios es TF-IDF+ m.1, que es el que recibe la mayor parte de los votos de satisfacción.

A continuación, detalles de estos resultados son dados para los métodos evaluados, con métricas de ránking a distintos tamaños de lista de recomendación.

Cuadro 6.2. Resumen de resultados offline de métodos CB, desagregados por métrica de distancia, modo de recomendación y por manera de representar el perfil de usuario.

Distancia	Modo	Representación	TF-IDF+	WE G*	WE W*	WE T*
coseno		m.1	**.19070**	**.12079**	.08889	.09488
	m.2	libros	**.10502**	.01558	.01430	.01054
		tuits		.00354	.00137	.00227
		mix		.01202	.00406	.00620
euclidiana		m.1		**.12103**	.09606	**.10323**
	m.2	libros		.00068	.01345	.00302
		tuits		.00143	.0	.00098
		mix		.00068	.01034	.00283

Cuadro 6.3. Resumen de resultados offline de las hibridaciones. Método CB en la horizontal, métodos CF en la vertical. Los cruces dan los resultados reportados.

CF \ CB	TF-IDF+ m.1
implicit	.18995
pyFM auth	.18326

Cuadro 6.4. Resumen de los resultados del estudio con usuarios. Se comparan los métodos usados en el caso online con el caso offline según nDCG@10 y diversidad, esta última calculada de distintas formas en ambos casos en búsqueda de posibles correlaciones.

	Offline		Online	
	nDCG	Diversidad calculada	nDCG	Diversidad percibida
TF-IDF+ m.1	.18853	.04031	.72353	.46364
implicit	.08646	.03221	.34682	.46818
WE G* m.2b cos	.01558	.01847	.64109	.60909

Por conveniencia, todos los valores están aproximados hasta el quinto decimal. En las siguientes tablas n se refiere al tamaño de las listas recomendadas. Así, si $n = 10$, la lista generada por cierto método consta de 10 ítems.

6.1. Evaluación *Offline*

El mínimo punto de referencia lo muestra la tabla 6.5, que detalla las métricas usando la forma aleatoria de recomendación. Según esto, si vamos recuperando aleatoriamente ítems del índice para cada usuario, en promedio el primer ítem relevante aparecería en la recuperación número 826 aproximadamente (MRR de 0.00121).

Cuadro 6.5. Método random.

n	nDCG	MAP	MRR	R-prec
5	.00201	.00218	.00121	.00091
10	.00201	.00218	.00121	.00045
15	.00201	.00218	.00121	.00030
20	.00360	.00218	.00158	.00057

En las tablas 6.6 hasta la 6.21 se muestran los resultados de los métodos basados en contenido. Los que logran mejores resultados son los de TF-IDF+, ambos modos, y los de word embedding en modo 1[1]. Estos alcanzan a obtener valores del orden de las décimas en nDCG en listas top 10, como visto en la tabla 6.2. El mejor de ellos, TF-IDF+ m.1, alcanza también valores del orden de las décimas en otras métricas, como se ve en la tabla 6.6. Estos altos valores de MRR significan que, a $n = 20$, teóricamente el primer ítem relevante se encontrará en el lugar 7.14. Un MAP@10 de aproximadamente 0.12 significa que la tasa promedio por el que se van encontrando ítems relevantes en una lista de largo 10 es de 1.2. Es decir, cada 10 ítems vamos encontrando en promedio 1 a 2 ítems relevantes (con mayor frecuencia 1). Lo mismo para largo 15 y 20. Queda propuesto como trabajo futuro experimentos con TF-IDF+ con representación de usuario a parte de sus libros leídos.

[1]Modo 1: consultas por cada uno de los documentos en el set de entrenamiento del usuario $\xrightarrow{\text{lista recomendada}}$ selección de los libros recomendados por cada libro del usuario . Modo 2: concatenación de todos los documentos en el set de entrenamiento del usuario en uno solo $\xrightarrow{\text{lista recomendada}}$ resultado de la consulta por este documento.

Cuadro 6.6. TF-IDF+ modo 1.

n	nDCG	MAP	MRR	R-prec
5	.14718	.12582	.11478	.06032
10	.19070	.11889	.13314	.05805
15	.21571	.11889	.14086	.05639
20	.22529	.11889	.14302	.05238

Cuadro 6.7. TF-IDF+ modo 2. Representación de usuarios por sus libros.

n	nDCG	MAP	MRR	R-prec
5	.08824	.07746	.07422	.02948
10	.10502	.07685	.08097	.02358
15	.11493	.07685	.08416	.02071
20	.12044	.07685	.08567	.01848

Como pensamos que es importante analizar en particular los resultados del *tuning* para estos métodos según el parámetro de los campos con el fin de ver la relevancia de cada campo sobre los resultados, es que mostramos estos resultados en las tablas 6.8 y 6.9.

Cuadro 6.8. TF-IDF+ modo 1. Resultados del ajuste del parámetro mlt.fl.

Campos	nDCG
descripción + autores + título + géneros + citas textuales	.11742
descripción + autores + título	.10935
descripción + autores	.10843
descripción	.10721
autores	.02871
géneros	.01179
título	.01039
citas textuales	.01028

Cuadro 6.9. TF-IDF+ modo 2. Resultados del ajuste del parámetro mlt.fl.

Campos	nDCG
descripción	.06396
autores	.04555
título	.03432
géneros	.02463
citas textuales	.01199

Dejamos ordenadas las filas de estas tablas según el valor de los resultados. Claramente si disponemos de más información sobre los libros, es mejor incluirla en el modelo, y entre más información entregada mejor será el resultado para este caso. Es consistente la ordenación de los campos, salvo que en el modo 2 comparar los documentos por títulos es mejor que por géneros, y al contrario en el modo 1, sin embargo la diferencia es pequeña.

Notar que esto es sólo una parte de los resultados del ajuste de los parámetros de ambos métodos TF-IDF+. El resultado del proceso total del ajuste se encuentra en el apéndice.

Son en general los métodos con word embedding en modo 2 los que obtienen peores resultados. El modo 1 de recomendación es notoriamente superior para todos los tipos de representación de usuario. Si bien se puede esperar que de entre los métodos probados con modo 2, aquel con representación de usuario por sus tuits pueda obtener bajos (o muy bajos) resultados debido a diferencias léxicas en los textos encontrados tanto en los posts de Twitter como en la información de los libros encontrada en Goodreads (y que por tanto se acarree parte de esa carga en la representación mixta, lo que resulta también en bajos resultados para esta forma de representación de usuario), es sorpresivo que la representación por libros obtuviera tan malos resultados, puesto que los léxicos de ambas partes

(texto de representación de usuario y texto de información de libro) en este caso tienen un origen común.

Cuadro 6.10. WE G* con distancia coseno, modo 1.

	coseno			
n	nDCG	MAP	MRR	R-prec
5	.09880	.08381	.08076	.03492
10	.12079	.08381	.09019	.03061
15	.13040	.08381	.09366	.02630
20	.13528	.08381	.09479	.02313

Cuadro 6.11. WE G* con distancia coseno, modo 2 y diversas formas de representar al usuario.

	coseno											
	libros				tuits				mixto			
n	nDCG	MAP	MRR	R-prec	nDCG	MAP	MRR	R-prec	nDCG	MAP	MRR	R-prec
5	.01123	.01301	.00888	.00363	.00354	.00344	.00246	.00136	.00970	.01076	.00763	.00317
10	.01558	.01301	.01064	.00317	.00354	.00344	.00246	.00068	.01202	.01076	.00866	.00227
15	.01558	.01301	.01171	.00302	.00540	.00344	.00304	.00091	.01382	.01076	.00919	.00196
20	.02128	.01301	.01219	.00272	.00540	.00344	.00304	.00068	.01598	.01076	.00972	.00193

Cuadro 6.12. WE G* con distancia euclidiana, modo 1.

	euclidiana			
n	nDCG	MAP	MRR	R-prec
5	.09399	.08080	.07456	.03401
10	.12103	.08034	.08633	.03175
15	.13001	.08034	.08955	.02661
20	.13439	.08034	.09070	.02392

Cuadro 6.13. WE G* con distancia euclidiana, modo 2 y diversas formas de representar al usuario.

	euclidiana											
	libros				tuits				mixto			
n	nDCG	MAP	MRR	R-prec	nDCG	MAP	MRR	R-prec	nDCG	MAP	MRR	R-prec
5	.0	.00118	.0	.0	.00143	.00178	.00113	.00045	.0	.00103	.0	.0
10	.00068	.00118	.00025	.00023	.00143	.00178	.00113	.00023	.00068	.00103	.00025	.00023
15	.00126	.00118	.00041	.00030	.00143	.00178	.00113	.00015	.00068	.00103	.00025	.00015
20	.00182	.00118	.00055	.00034	.00143	.00178	.00113	.00011	.00176	.00103	.00025	.00034

Cuadro 6.14. WE W* con distancia coseno, modo 1.

	coseno			
n	nDCG	MAP	MRR	R-prec
5	.06980	.06121	.05593	.02494
10	.08889	.06080	.06418	.02290
15	.10181	.06080	.06807	.02147
20	.10752	.06080	.06952	.01871

Cuadro 6.15. WE W* con distancia coseno, modo 2 y diversas formas de representar al usuario.

	coseno											
	libros				tuits				mixto			
n	nDCG	MAP	MRR	R-prec	nDCG	MAP	MRR	R-prec	nDCG	MAP	MRR	R-prec
5	.00571	.00967	.00461	.00181	.0	.00134	.0	.0	.00185	.00325	.00102	.00091
10	.01430	.00967	.00804	.00363	.00137	.00134	.00051	.00045	.00406	.00325	.00193	.00113
15	.01729	.00967	.00892	.00317	.00262	.00134	.00090	.00060	.00469	.00325	.00214	.00091
20	.01838	.00967	.00919	.00261	.00315	.00134	.00103	.00057	.00795	.00325	.00293	.00136

Cuadro 6.16. WE W* con distancia euclidiana, modo 1.

	euclidiana			
n	nDCG	MAP	MRR	R-prec
5	.07250	.06682	.05967	.02494
10	.09606	.06642	.06985	.02426
15	.10764	.06642	.07344	.02237
20	.11513	.06642	.07529	.01995

Cuadro 6.17. WE W* con distancia euclidiana, modo 2 y diversas formas de representar al usuario.

	euclidiana											
	libros				tuits				mixto			
n	nDCG	MAP	MRR	R-prec	nDCG	MAP	MRR	R-prec	nDCG	MAP	MRR	R-prec
5	.00823	.01141	.00718	.00227	.0	.00053	.0	.0	.00678	.00802	.00529	.00227
10	.01345	.01141	.00938	.00227	.0	.00053	.0	.0	.01034	.00802	.00669	.00227
15	.01521	.01141	.00988	.00227	.0	.00053	.0	.0	.01149	.00802	.00700	.00181
20	.01627	.01141	.01013	.00193	.00055	.00053	.00014	.00011	.01201	.00802	.00712	.00147

Cuadro 6.18. WE T* con distancia coseno, modo 1.

	coseno			
n	nDCG	MAP	MRR	R-prec
5	.07541	.06801	.06119	.02630
10	.09488	.06801	.06938	.02268
15	.10443	.06801	.07228	.01935
20	.10997	.06801	.07379	.01712

Cuadro 6.19. WE T* con distancia coseno, modo 2 y diversas formas de representar al usuario.

	coseno											
	libros				tuits				mixto			
n	nDCG	MAP	MRR	R-prec	nDCG	MAP	MRR	R-prec	nDCG	MAP	MRR	R-prec
5	.01054	.01150	.00877	.00317	.00227	.00308	.00227	.00045	.00324	.00545	.00283	.00091
10	.01054	.01150	.00981	.00227	.00227	.00308	.00227	.00023	.00620	.00545	.00407	.00136
15	.01519	.01150	.01046	.00212	.00405	.00308	.00279	.00060	.00740	.00545	.00443	.00121
20	.01628	.01150	.01072	.00181	.00457	.00308	.00291	.00057	.00848	.00545	.00469	.00113

Cuadro 6.20. WE T* con distancia euclidiana, modo 1.

	euclidiana			
n	nDCG	MAP	MRR	R-prec
5	.07955	.07020	.06292	.02993
10	.10323	.07020	.07319	.02585
15	.11317	.07020	.07634	.02192
20	.11954	.07020	.07789	.01939

Cuadro 6.21. WE T* con distancia euclidiana, modo 2 y diversas formas de representar al usuario.

	euclidiana											
	libros				tuits				mixto			
n	nDCG	MAP	MRR	R-prec	nDCG	MAP	MRR	R-prec	nDCG	MAP	MRR	R-prec
5	.00231	.00263	.00159	.00091	.00098	.00106	.00057	.00045	.00143	.00246	.00113	.00045
10	.00302	.00263	.00187	.00068	.00098	.00106	.00057	.00023	.00283	.00246	.00167	.00068
15	.00359	.00263	.00202	.00060	.00154	.00106	.00072	.00030	.00342	.00246	.00184	.00060
20	.00520	.00263	.00241	.00079	.00154	.00106	.00072	.00023	.00502	.00246	.00222	.00079

En las tablas 6.22, 6.24, 6.26, 6.27, 6.28 y 6.30 se muestran los valores de las métricas de ránking para los algoritmos de factorización matricial y de máquinas de factorización evaluados. El peor de ellos, el obtenido con la librería fastFM entrenado con SGD, es comparable al método random. Se especulaba que el método

BPR pudiera tener buenos resultados, pero aun en $n = 20$ el primer ítem relevante se encuentra en promedio en la posición 247.

Cuadro 6.22. Métricas de ránking obtenidas con método de factorización matricial funkSVD.

n	nDCG	MAP	MRR	R-prec
5	.00328	.00793	.00215	.00136
10	.00611	.00793	.00326	.00159
15	.01275	.00793	.00526	.00272
20	.01602	.00793	.00600	.00283

Cuadro 6.23. Métricas de predicción de rating de funkSVD.

RMSE	MAE
1.14421	1.02760

Dado que los modelos de máquinas de factorización, al no ser provistos de contexto más allá del trío (usuario, ítem, rating), son básicamente factorización matricial, los resultados de éstos debieran ser al menos similares, sin embargo difieren considerablemente. Esto puede deberse a diferencias en programación de los métodos, diversas formas de optimizar los entrenamientos, etc. Mayor análisis es requerido para entender mejor esta situación.

Cuadro 6.24. Métricas de ránking para fastFM, entrenado con SGD.

n	nDCG	MAP	MRR	R-prec
5	.00143	.00283	.00113	.00045
10	.00211	.00283	.00139	.00045
15	.00450	.00283	.00209	.00091
20	.00505	.00283	.00223	.00079

Cuadro 6.25. RMSE de fastFM, SGD.

RMSE
1.61337

Distintas librerías proveen diferentes modos para entrenar sistemas recomendadores basados en filtrado colaborativo, aunque no todos los algoritmos de estas librerías pueden ser entrenados de formas arbitrarias, algunos están fijos y no

Cuadro 6.26. Resultados de BPR.

n	nDCG	MAP	MRR	R-prec
5	.0	.00498	.0	.0
10	.00760	.00498	.00286	.00249
15	.00889	.00498	.00320	.00212
20	.01222	.00498	.00405	.00227

son cambiables como parámetros por decisión de los desarrolladores, así como tampoco es posible incluir contexto en cualquiera de los modelos de forma simple. Razones por las que algunas celdas de la tabla 6.1 están vacías. En los que sí es simple agregar contexto, el paquete pyFM es el que proveyó el sistema de máquina de factorización que mejor resultados dio sin dar mayor información contextual que el usuario, ítem y rating, por lo que se decidió por utilizar esta librería para agregar contexto al dataset de entrenamiento, como la fecha de consumo, los autores, y ambos. Estos resultados pueden verse en la tabla 6.28.

La inclusión de autores tiene que ver con el hallazgo del set de parámetros que óptimizan los resultados para el modo 2 del TF-IDF+[2], en donde se encontró que los autores pueden ser buen predictor de los gustos de los usuarios (mayores explicaciones se entregan en capítulo 7). En el momento de recolección de estos resultados, el promedio de autores por libro es de 1.37 y el máximo es de 51[3]. De hecho, de los 50,863 documentos en el índice en el momento de este cálculo, el 19.57 % son libros con 2 o más autores, y un 5.35 % con 3 o más. Dado esto, por cada interacción se consideraron hasta 3 autores del libro en cuestión como parte del contexto.

Al ir incluyendo contexto se van logrando mejores resultados, salvo cuando el contexto son ambos la fecha de consumo y los autores, lo cual es imprevisto. Cuando el contexto son los autores, se obtienen tanto uno de los dos mejores recomendadores en cuanto métricas de ránking (el otro siendo SVD con dataset

[2]Ver apéndice para el detalle de los parámetros.

[3]*50 Great American Short Stories*, `https://www.goodreads.com/book/show/580387`

implícito), como uno de los mejores en predicción de rating (ver tablas 6.29, 6.25, 6.23). Esto significa que para hacer un sistema de filtrado colaborativo en un contexto como éste, considerar los ratings como una expresión de feedback implícito puede ser lo indicado. Aun así, este algoritmo es parecido en resultados recién con el séptimo mejor de los algoritmos basados en contenido, lo que puede insinuar que son estos últimos los que mejor se adaptan a este problema.

Cuadro 6.27. Métricas de ránking de máquina de factorización del paquete pyFM, sin contexto.

n	nDCG	MAP	MRR	R-prec
5	.02182	.02709	.01697	.00726
10	.03021	.02709	.02020	.00635
15	.04066	.02709	.02340	.00710
20	.04739	.02709	.02490	.00714

Cuadro 6.28. Métricas de ránking de máquina de factorización del paquete pyFM, con distintos niveles de contexto: por sí solo las fechas de consumo, por sí solo los -a lo más- 3 autores, y ambos. Marcado se encuentra la métrica nDCG con valor más alto con $n = 10$.

	timestamp				autores				timestamp + autores			
n	nDCG	MAP	MRR	R-prec	nDCG	MAP	MRR	R-prec	nDCG	MAP	MRR	R-prec
5	.02663	.03476	.02271	.00771	.04381	.03905	.03243	.01633	.01824	.02405	.01379	.00635
10	.04472	.03476	.02991	.00975	**.05833**	.03905	.03852	.01292	.02908	.02405	.01818	.00658
15	.05524	.03476	.03297	.00952	.06672	.03905	.04104	.01134	.03719	.02405	.02042	.00695
20	.06189	.03476	.03453	.00918	.07218	.03905	.04245	.01043	.04249	.02405	.02165	.00635

Cuadro 6.29. Métricas de predicción de rating de máquina de factorización del paquete pyFM, sin y con contexto.

contexto	RMSE
sin contexto	1.11905
timestamp	1.21242
autores	1.09096
timestamp + autores	1.17878

Dado que el mejor sistema CB (TF-IDF+ m.1) es notoriamente superior al segundo mejor (WE G* m.1, distancia coseno) con una diferencia de 0.070 en nDCG@10, y dado que el desbalance entre los 2 mejores algoritmos CF no es

Cuadro 6.30. Resultados de factorización matricial con feedback implícito de la librería implicit.

n	nDCG	MAP	MRR	R-prec
5	.07007	.05853	.06107	.01995
10	.08646	.05853	.06787	.01655
15	.09400	.05853	.07058	.01497
20	.10288	.05853	.07295	.01474

muy grande, con una diferencia de 0.028 en nDCG@10, habrán dos hibridaciones. Cada uno corresponde a un esquema ponderado del recomendador CB con uno de los recomendadores CF. Los resultados están en la tabla 6.31.

Aun si los recomendadores híbridos obtienen buenos resultados, no son más altos, con $n = 10$, que el mejor de los CB (tabla 6.6). Sin embargo se considera que la diferencia es muy pequeña como para decir con seguridad que este método CB es notoriamente mejor que el mejor de los híbridos, incluso si comparamos con la misma métrica en listas de recomendación más largas. La situación se revierte a $n = 20$ cuando uno de los híbridos ligeramente supera al sistema CB que lo compone. Si bien se esperaba que el híbrido lograra resultados marcadamente mejores en métricas de ránking, las diferencias son mínimas. Mayor análisis es requerido para revisar si este sistema suple efectivamente las faltas de sus componentes en esta situación, como el problema de *cold start* de uno, la sobre-especialización de otro, etc.

Consistente con que, por sí solos, los resultados de la evaluación del sistema CF con feedback implícito sean mejores que los de FM con contexto, el sistema híbrido que incluye al CF implícito es mejor que aquel con pyFM con contexto.

Se obtienen mejores resultados cuando las recomendaciones del híbrido son mucho más parecidas a las del CB que a los CF. Sin embargo tenemos el beneficio de que les damos un poco más de variedad a las recomendaciones con respecto al sistema CB puro dada la incorporación del CF (ver tabla 6.32), lo que podría

explicar la leve ventaja que tiene el híbrido Hyb-1 con respecto al método CB simple.

Como es esperable debido a los parámetros encontrados, la lista híbrida con la lista CB que la compone difieren con una diversidad de 0.1, i.e., cerca de una décima parte de la lista híbrida se compone de recomendaciones no otorgadas por la parte CB, mientras que el resto sí. Calculada separadamente, la diversidad de la lista Hyb-1 con la lista CF (SVD implícito) es cercana a 0.9, lo que quiere decir que esta décima parte la componen ítems dados por el sistema CF, lo cual también es esperable. Las listas CB y CF difieren prácticamente en su totalidad

Cuadro 6.31. Resultados de los sistemas híbridos. El recomendador Hyb-1 corresponde a TF-IDF+ m.1 y SVD implícito. Hyb-2, el mismo CB y FM con autores de libros como contexto.

	Hyb-1				Hyb-2			
n	nDCG	MAP	MRR	R-prec	nDCG	MAP	MRR	R-prec
5	.14479	.11738	.11326	.06168	.14186	.11675	.11251	.05805
10	.18995	.10323	.13222	.05601	.18326	.10308	.12987	.05215
15	.21403	.10245	.13997	.05200	.20840	.10231	.13858	.05110
20	.22736	.10245	.14316	.05102	.22441	.10231	.14209	.04921

Cuadro 6.32. Diversidad entre-lista para cada par de algoritmos calculada como el promedio de las diversidades de las listas @20 recomendadas a todos los usuarios según fórmula de Lathia et al. (2010).

L1	L2	Diversidad (Lathia et al., 2010)
Hyb-1	TF-IDF+ m.1	.10419
Hyb-1	SVD implícito	.89308
TF-IDF+ m.1	SVD implícito	.99728

En resumen, de los algoritmos evaluados el mejor recomendador CF es el SVD con feedback implícito, seguido cercanamente del recomendador con máquina de factorización de la librería pyFM con autores de contexto. El mejor recomendador CB es notoriamente el TF-IDF+ en modo 1. Los mejores resultados de

la hibridación se obtienen combinando el mejor CB con el mejor CF, el que a su vez es el recomendador superior en general.

6.2. Evaluación *Online*

En esta sección presentamos los resultados del estudio con usuarios. Todas las métricas corresponden a aquellas calculadas en listas de recomendación de largo 10. Los participantes en este estudio fueron 22 usuarios activos de Goodreads, todos latinoamericanos cuya lengua materna es el español y que cuentan con un uso competente del idioma inglés. Los usuarios fueron reclutados son aquellos que mostraron interés en el estudio luego de responder un correo electrónico enviado masivamente a miembros del Departamento de Ciencia de la Computación de la Pontificia Universidad Católica de Chile y aquellos que aceptaron participar luego de responder publicaciones hechas en algunos foros de Goodreads de lectores de habla hispana.

El promedio de libros por usuario es $M = 92$, con desviación estándar $D.E. = 134$. No todos los libros que contiene un usuario en su perfil son libros que ha leído. Por simplicidad diremos que todos los libros que han sido calificados en una escala de 1 a 5 en Goodreads son libros leídos, a pesar de que (i) en Goodreads los usuarios pueden dar ratings a libros aun si no los han leído completamente (y esto se muestra en el archivo generado del perfil de usuario de Goodreads) y (ii) los usuarios no están forzados a dar ratings si es que marcan el libro como leído. A pesar de esto, por revisión de las lecturas enviadas por los usuarios, se cumple que la mayoría de los libros leídos tiene rating dado por el usuario. Con esta simplificación, podemos obtener una razón de libros leídos con respecto a los no leídos ('por leer' o 'leyendo'): en promedio se han leído 64 libros, con una desviación estándar de 106. El mínimo es de 7 libros, y el máximo, 464. Esto quiere decir que en promedio, los libros no leídos (o, estrictamente, no calificados) es de 28, por lo que la proporción en promedio, de todos los libros

en el perfil, es de casi 1 libro no leído por cada 2 leídos. En cuanto al promedio de ratings, éste es de 4.1, con una baja desviación estándar de 0.4. Número más bien elevado si se considera la escala de 1 a 5.

Tanto en la parte offline como la parte online de la tabla 6.33, los resultados mostrados son de los métodos ahí indicados pero no con los parámetros que optimizan su desempeño en métricas de ránking y rating (a saber nDCG@20 para métodos CB y RMSE para métodos CF, explicado en la sección 5.2). Hay diferencia, pues en los ensayos del estudio se vio que a los usuarios del estudio se les recomendaba libros iguales pero de editoriales distintas, problema que no se había previsto. Lo intuitivo es que cada usuario, de leer un libro, lea uno sólo de un espacio de libros los cuales son todos iguales pero de editoriales distintas. Se consideró esto como hipótesis de este trabajo. Entiéndase por 'libros iguales' en este contexto a libros con el mismo contenido *canónico*, es decir, el texto escrito de un libro que al ser consumido se considera coloquialmente que está leído, independiente de la edición o editorial. En otras palabras, sean muchos libros de ediciones y editoriales distintas. Más allá de las diferencias en formatos físicos de los ejemplares, de las traducciones, notas del traductor y/o editor, prefacios escritos por distintos autores o etc., si el contenido grueso de la obra, sin importar el idioma, es el mismo, entonces lo consideraremos como distintas versiones de un mismo libro. Antes de seguir más adelante con el estudio se aplicó esta corrección inmediatamente, eliminando de las listas de recomendación libros equivalentes, dejando sólo una versión de cada uno, seleccionado aleatoriamente. Esto se pudo hacer debido a que en Goodreads, en el HTML de las rutas de cada libro se encuentra un elemento que contiene su URL canónica[4] (un elemento con `rel=canonical`), dato que está almacenado en los documentos de nuestro índice, pues es el valor de uno de los campos (más información sobre el diseño del esquema en la sección 5.1). En Goodreads, todos los 'libros iguales' pero de

[4] `https://yoast.com/rel-canonical/`

versiones distintas (y que por tanto tienen rutas distintas en la aplicación) comparten la misma URL canónica, la cual dirige hacia uno de esos libros[5]. Gracias a esto, se pudo agrupar a todos estos 'libros iguales', en el caso de que el sistema decida recomendar más de una versión del mismo libro, y recomendar sólo uno de ellos escogido aleatoriamente.

Cuadro 6.33. Métricas de evaluación para estudio online y su comparación con aquellas del estudio offline. Tamaños de lista de recomendación $n = 10$. 2 posibles umbrales de relevancia y diversidad.

Offline					
Algoritmo	nDCG	MAP	MRR	R-prec	Diversidad
TF-IDF+ m.1	.18853	.11888	.13470	.05465	.04031
SVD implícito	.08646	.05840	.06787	.01655	.03221
WE G* m.2b cos	.01558	.01301	.01064	.00317	.01847
Online. Threshold: 2					
Algoritmo	nDCG	MAP	MRR	P@10	Div. percibida
TF-IDF+ m.1	.91313	.81507	.93182	.70454	.75454
SVD implícito	.70620	.61480	.64773	.50909	.75000
WE G* m.2b cos	.83331	.70905	.85303	.56818	.93182
Online. Threshold: 3					
Algoritmo	nDCG	MAP	MRR	P@10	Div. percibida
TF-IDF+ m.1	.72353	.56851	.65844	.41818	.46364
SVD implícito	.34682	.25115	.26306	.15909	.46818
WE G* m.2b cos	.64109	.53115	.65000	.24091	.60909

Se siguió usando los mismos métodos con los mismos parámetros pero esta vez con esta corrección. Dado que la situación cambió, se procedió a hacer un nuevo proceso de búsqueda de parámetros para todos los métodos de manera offline, para luego correr los testings cuyos resultados son los que se mostraron en las tablas de la sección 6.1. Tanto la jerarquía de efectividad (medido por las métricas calculadas) de los algoritmos como los parámetros con que se evaluaron cambió con respecto a lo que se tenía antes de aplicar esta corrección. La tabla 6.33 muestra los algoritmos que generaron las listas de recomendación a

[5]No hay información disponible en la aplicación que clarifique a cuál de esos libros envía el enlace, pero la tendencia pareciera ser al más antiguo ingresado a la aplicación.

los usuarios del estudio con los parámetros encontrados antes de la corrección, sin embargo las listas presentadas a los usuarios si contienen esta corrección. Es decir, nunca los usuarios en el estudio online ven en una misma lista el mismo libro publicado en distintas ediciones.

A pesar que la lógica de selección de estos algoritmos está explicada en la sección 5.3, vemos a un algoritmo que no tiene los mejores resultados de entre los de word embedding (WE G* m.2b, distancia coseno). Esto debido a que antes de la corrección, era este algoritmo el mejor de los word embedding. Se tenía la presunción de que con la corrección los resultados no mostrarían una alta variabilidad. Esto fue cierto para los algoritmos de TF-IDF+ y los de CF, sin embargo los resultados que más alta variabilidad presentaron fueron los de word embedding, imponiéndose especialmente el modo 1 de recomendar ítems por sobre el modo 2, como se ve resumido en la tabla 6.2. Debido a esta presunción y por motivos de tiempo, antes de haber encontrado todos los nuevos parámetros óptimos para los algoritmos y luego revisar si hubo cambios en la estructura de los resultados, se prosiguió con el estudio de usuario. De ahí que los resultados acá presentados tengan su origen en las listas 'corregidas' de estos algoritmos, los cuales no todos son los mejores de su categoría respectiva. Por lo mismo, en la parte offline de la tabla 6.33, los valores reportados no se encuentran en algunas de las tablas anteriores, puesto que no son producidos por los algoritmos con sus parámetros óptimos, con la salvedad del método de word embedding ya que, como mencionado en la sección 5.2, estos modelos son pre-entrenados[6].

En cuanto al estudio, los usuarios percibieron los ítems como relevantes bastante más frecuentemente que en los cálculos del estudio offline. Si consideramos como límite de relevancia una puntuación de 2, el recomendador TF-IDF+ m.1 produce listas que en promedio los usuarios encuentran el primer ítem relevante en el primer puesto. Además, con el mismo umbral, en promedio la mayoría de

[6]El detalle de los parámetros de los algoritmos con que se obtuvieron los resultados de la tabla 6.33 se encuentran en el apéndice

los ítems recomendados los consideran relevantes. Esto es especialmente favorecedor para el algoritmo de word embedding, pues además se le percibe como ser altamente diverso. Esta situación decae al fijar un umbral más estricto de relevancia y de diversidad, no obstante el orden de qué algoritmo es mejor que otro se mantiene: TF-IDF+ m.1 sobre WE G* m.2b cos, ambos sobre implicit SVD. El algoritmo en el estudio offline que mejor resultados obtuvo mantuvo su jerarquía en el estudio con usuarios. No fue así con los otros dos algoritmos que se evaluaron, presumiblemente por una mayor percepción de diversidad en las listas, bajo el supuesto de que mayor diversidad implica mayor satisfacción de los usuarios en las listas recomendadas.

A cada usuario se le preguntó por su satisfacción con respecto a cada lista, así como también que nos dijera su recomendador favorito. Estos resultados están en la tabla 6.34. En cuanto a satisfacción, vemos que la mayor cantidad de votos se la lleva el mejor algoritmo de los 3 según métricas de ránking. Lo mismo con el segundo y el tercero. La situación cambia al considerar los favoritos, en donde en segundo y tercer lugar son respectivamente el de CF y el de word embedding.

Cuadro 6.34. Preferencia de los usuarios por los algoritmos evaluados. Los números son los votos de los usuarios. La suma de las columnas no son iguales pues cada usuario puede estar satisfecho con las 3 listas máximo, y mínimo con ninguna, en cambio éste está forzado a escoger uno sólo de favorito. De ahí que $11 + 6 + 5$ corresponda al total de participantes en el estudio.

	Satisfacción	Favoritos
TF-IDF+ m.1	14	11
SVD implícito	7	6
WE G* m.2b cos	10	5

CAPÍTULO 7. DISCUSIÓN Y CONCLUSIONES

En esta sección discutiremos los resultados obtenidos en este trabajo. En la sección de conclusiones responderemos las preguntas de investigación planteadas 3.2.

7.1. Discusión de los Resultados

TF-IDF+ modo 1

En cuanto al algoritmo TF-IDF+, vemos que el modo 1 obtiene mejores resultados que el modo 2. Que evaluaciones con el componente MoreLikeThis de Solr se obtengan drásticamente mejores resultados que con los otros métodos CB puede deberse a que, como se comparan ítems según un campo o un conjunto de ellos debido que la información del ítem está almacenada estructuradamente, se produce comparaciones más coherentes con respecto a comparar texto plano, y eso puede incidir en mejores resultados, siendo que además escoge -automáticamente los mejores- campos para comparación, lo que podría añadir un boost a los resultados.

La suposición era que el mejor campo (por sí solo) para hacer las comparaciones sería el de la descripción encontrada en Goodreads, dado que es un texto con importantes keywords acerca del ítem y porque contiene varios términos. Llevado por esa idea, se fueron agregando otros campos que se cree les siguieran en importancia. Contrario a lo que se pensaría, agregar fields al de la descripción empeora progresivamente los resultados, con la excepción de incluir todos los fields, que alcanza el mejor resultado. Sin embargo son diferencias muy bajas como para ser significativas. Esto puede deberse a que al considerar, a parte de la descripción, el autor, y luego el autor más el título, se añade incertidumbre al incluir mayor información. Por ejemplo: sea A el libro leído. Diérase el caso que B y C sean bien similares a A por descripción, y muy poco similares por autor.

Asumiendo que el usuario prefiere leer libros con descripciones similares y no -necesariamente- por autores similares, (i.e. 'es más importante para el preferir libros con descripciones similares a que preferir con autores similares'), es posible que comparar además por autor (comparar por descripción + autor) pueda bajar el score. Sin embargo, el que mejor resultados da es con todos los campos posibles. Esto puede tomarse como que lo mejor para comparar es un promedio de todas las maneras posibles (relevantes) de comparar libros de Goodreads, lo que puede hablar acerca de los usuarios viendo toda la información disponible de los libros a leer que consideren relevantes.

TF-IDF+ modo 2

En el caso del modo 2, se forma un documento agregado de todos los libros leídos por el usuario en un solo texto. Utilizando la funcionalidad *ContentStreams* de Solr, podemos mapear una cantidad arbitraria de documentos de nuestro índice en un archivo de texto plano que contiene todos los términos de éstos según un campo seleccionado. Por ejemplo si el campo es el de autores, el texto resultante estará hecho sólo de los nombres de los autores de los documentos de entrada.

Según resultados, la mejor manera de comparar ese documento-usuario con los documentos-libros es con la descripción. Esto significa que al tomar todo el contenido transmitido como 'descripción', y al compararlo con el campo de descripción de los documentos del índice, se obtienen las recomendaciones que mayor utilidad dan al usuario. Esto significa que, para las comparaciones TF-IDF+ entre documentos del índice con el documento-usuario, es mejor representar el documento-usuario como si fuera una 'descripción' de algún libro. Quizás porque el campo de la descripción es el que más términos contiene (8,727,675 sin contar filtros adicionales como stop-words), y dado que estamos comparando con un gran bloque de texto que es el documento-usuario hay más términos para calcular distancias, lo que podría llevar a comparaciones menos erróneas.

Viendo los resultados del tuning (tabla 6.9), el campo del autor es lo segundo que mejor optimiza el sistema para el parámetro `mlt.fl`. Una manera de expresar este resultado es que, en el documento-usuario (el contenido transmitido), el nombre de los autores es lo que más está presente, pues está incluido en el documento-usuario como texto. De hecho están incluidos todos los autores de los libros del train set del usuario. Al representar al documento-usuario como 'autor' se obtienen los scores más altos, lo que sólo puede ser causado gracias a que se han encontrado libros similares por el campo de autor (haberse encontrado 'autores similares'). Esto tiene sentido según la presunción de que un usuario, al haber consumido (o 'ser representado por..') varios libros de un mismo autor, es más 'proclive a leer libros del mismo autor', más que 'proclive a leer libros de géneros iguales sin importar el autor' (puede seguir siendo importante sólo el género, sin embargo usualmente un autor escribe libros en un set definido de géneros por los cuales se hace popular, por lo que libros pedidos con la consigna de 'mismo autor' y aquellos pedidos por 'mismo autor y mismo género' pueden resultar ser bien similares), o libros con citas similares, por ejemplo.

Cuando los libros son parte de una serie normalmente tienen títulos y descripciones similares. La serie es escrita por un mismo autor, quien escribe libros bajo un set de géneros definido (como quien asociaría a Stephen King con 'horror', 'drama' y 'acción') lo que puede explicar que sólo campos de 'título' y 'género' no tengan tan buenos resultados como el de autor. Esto tomando en cuenta el hecho que los usuarios de nuestro dataset han consumido muchos libros (al menos 40, según nuestro modelo de partición), lo que hace posible el que se trate de ávidos lectores buscando *series de libros*, más que libros individuales por leer.

Si seguimos la misma lógica, es posible que el campo de descripción sea más adecuado puesto que éste normalmente contiene palabras clave que describan el libro, tales como aquellas que digan sobre sus géneros, sus autores, personajes y contenido mismo del libro. Tiene sentido si vamos por esta línea, dado que

la jerarquía de resultados de los campos probados para modo 1 es la siguiente: descripción+autores+título+géneros+citas > descripción+autores+título > descripción+autores > descripción > autores > géneros > título > citas (tabla 6.8). Si consideramos que las diferencias de resultados entre título y géneros es cercano a 0.01 en ambos modos, la jerarquía es básicamente la misma, sin contar los resultados con campos compuestos (descr+autores, etc.).

Vemos que sólo tomando los campos compuestos, entre más campos usemos mejor serán los resultados en el modo 1. Sin embargo no podemos asegurar que entre más términos mejor. Viendo la columna de promedio de términos por campo en la tabla 5.1 y los resultados de las tablas 6.8 y 6.9, nos damos cuenta que no correlacionan la cantidad de términos con nDCG.

El campo de las citas se pensaría que pueda contener tanto o más texto que el de título, géneros o de autores, sin embargo éste está presente en menos de la mitad de los documentos en el índice (46 %, observado en la tabla 5.1), por lo que por sí solo la búsqueda de documentos similares sería aleatoria para más de la mitad de los casos.

Métodos WE

Comparamos los resultados de los distintos métodos de *word embedding* entre sí. Todas las comparaciones a continuación se hacen a base de promediar los nDCG por los distintos tamaños de listas de recomendación.

Comparación de métodos según distinto tipo de configuración. Todas las comparaciones se hacen con respecto a la mejor configuración:

- Modelo WE G*:
  - Coseno: m.1 > m.2b (-.10450) > m.2mix (-.10844) > m.2t (-.11685)
  - Euclidiana: m.1 > m.2t (-.11842) > m.2b (-.11891) > m.2mix (-.11907)

- Modelo WE W*:
  - Coseno: m.1 > m.2b (-.07808) > m.2mix (-.08737) > m.2t (-.09220)
  - Euclidiana: m.1 > m.2b (-.08454) > m.2mix (-.08768) > m.2t (-.09770)
- Modelo WE T*:
  - Coseno: m.1 > m.2b (-.08245) > m.2mix (-.89842) > m.2t (-.09288)
  - Euclidiana: m.1 > m.2b (-.10034) > m.2mix (-.10070) > m.2t (-.10261)

Vemos que indiferente de la distancia lo que da mejor resultados es modo 1 en comparación a las distintas configuraciones con modo 2.

Comparación de métodos según distintas métricas de distancia. Las comparaciones se hacen entre mismas modalidades de un mismo modelo (ej: m.1 coseno contra m.1 euclidiano en modelo WE G*). Para el caso de modo 2 se compara la representación de usuario que nDCG dé (ej. para WE W*, sería m.2b coseno contra m.2b euclidiana, a juzgar por las tablas 6.15 y 6.17). Todas las comparaciones se hacen con respecto a métrica coseno (si es positivo: es mejor distancia coseno).

- WE G*:
  - modo 1: +.00146
  - modo 2: +.01588
- WE W*:
  - modo 1: -.00583
  - modo 2: +.00063
- WE T*:
  - modo 1: -.00770
  - modo 2: +.01019

En modo 2 se compararon sólo representación por libros, dado que en general es la forma de representación de usuario que mejor resultados da.

Al usar modo 1 conviene buscar recomendaciones con distancia euclidiana, salvo con el modelo pre-entrenado con datos de Google WE G*; con modo 2, conviene distancia coseno.
Puede deberse a que en modo 2 el módulo promedio de los vectores-libros sea muy diferente al módulo promedio de los vectores-usuarios (que son producto de la agregación de varios libros), por lo que distancia coseno puede ser más adecuado. Distancia euclidiana puede dar muy lejos aún cuando vector-usuario y vector-libro estén muy cerca angularmente. Razón análoga, puede que por eso la distancia euclidiana funcione mejor entre vectores-libros (m.1), más que ser mejor (aunque son mejoras marginales), mejora en virtud del empeoramiento del uso de distancia angular para m.1.

Luego comparamos resultados con distintos modelos de word embedding y sets de entrenamiento (Google News word2vec, Wikipedia dump + UMBC webbase + statmt.org FastText, Twitter GloVe). Mejores escenarios según modelos:

- WE G*: m.1 coseno (G* m.1-cos)
- WE W*: m.1 euclidiana (W* m.1-euc)
- WE T*: m.1 euclidiana (T* m.1-euc)

Comparación entre modelos con respecto al mejor:

$$\text{G* m.1-cos} > \text{T* m.1-euc (-.01744)} > \text{W* m.1-euc (-.02348)}$$

Finalmente, que el modelo word2vec pre-entrenado con datos de Google News (G*) obtenga mejores resultados que aquél entrenado con tuits (T*), y éste a su vez que el entrenado con el Wikipedia dump del 2017, el corpus UMBC webbase y el dataset statmt.org News en conjunto (W*) puede deberse, entre otras razones, a que el primero tenga notoriamente más datos de entrenamiento (100B de tokens) que el segundo (27B) y que el tercero (16B), ya que esto puede ser un factor influyente en el rendimiento de los modelos (Mikolov, Chen, Corrado, y Dean, 2013).

Métodos Basados en Filtrado Colaborativo

En cuanto a los resultados de los métodos colaborativos, los resultados del método de factorización matricial con feedback implícito (tabla 6.30) muestran métricas comparables a las de otro método CB empleado (TF-IDF+ m.2, aunque TF-IDF+ m.1 sigue siendo superior) y a los m.1 de métodos WE.

Le agregamos información contextual al método de máquina de factorización (FM) que mejor resultados logró por sí solo (sin contexto), el cual fue el algoritmo que obtuvimos del paquete pyFM. Si bien según resultados TF-IDF+ fue la descripción el campo que logra mejores resultados por sí solo, intentar usarlo como contexto traería problemas adicionales: sería muy difícil encontrar 2 pares de descripciones que contengan exactamente los mismos términos. Incluir esta información podría verse como una tarea de planear una hibridación de datos del contenido del ítem en un sistema CF, tal como hacer *feature combination* (detalles en sección 1.2.1). Aquello cae fuera del alcance de este trabajo.

Tanto la inclusión de los timestamps como de los autores (por separado) a las interacciones base (user,item,rating) mejoraron prácticamente al doble los resultados en general, con la salvedad de que con timestamps y timestamps + autores empeora ligeramente el RMSE con respecto a resultados sin contexto.

El uso de ambas fuentes de información contextual por simultaneo empeora los resultados con respecto a usar una sola fuente. De los métodos con FM probados hasta ahora, el que prueba tener mejor rendimiento es pyFM con autores de los libros consumidos como contexto. De los métodos CF, este escenario es el que mejor resultados da según las métricas de evaluación empleadas, después de SVD con feedback implícito. Esto puede corroborar la hipótesis surgida de los experimentos CB con TF-IDF+ de que los autores de los libros puede ser un buen predictor para los siguientes ítems que el usuario consumirá.

Métodos Híbridos

Siguiendo la discusión de la sección de los resultados, fue esperable que los sistemas híbridos tuviesen buenos resultados en relación a nuestros sistemas evaluados.

Los parámetros de los dos sistemas híbridos evaluados, al igual que todos los otros sistemas, fueron ajustados por *5-fold cross-validation*. Éstos son los pesos que indican la injerencia de cada algoritmo constitutivo sobre la lista de recomendación final. Consistente con que por sí solo el método CB del híbrido (TF-IDF+ m.1) obtiene mejores resultados que los métodos CF del híbrido, los pesos se cargan para dar más importancia a la parte CB. Los resultados hasta este punto pueden dar evidencia sobre qué tipo de recomendador es más apropiado para este dominio.

Evaluación *Online*

Sobre la evaluación con usuarios, vemos que a pesar que la configuración WE G* m.2b-cos no sea el mejor modelo de word embedding, aún obtuvo mejores resultados online que uno que obtuvo mejores resultados que éste offline: SVD implícito, a juzgar por los resultados de las métricas que tienen que ver con relevancia en la tabla 6.33. El segundo y tercer mejor algoritmo son respectivamente el de word embedding y el de CF. Esto no refleja necesariamente la preferencia de los usuarios sobre los recomendadores, ni tampoco la cantidad de votos de satisfacción que consigue cada lista, dado que, según la cantidad de votos de favoritos, el algoritmo CF y el de word embedding están en puestos casi iguales. Mayor número de usuarios sería requerido para corroborar que efectivamente el método WE es preferido por sobre el CF, puesto que el algoritmo CF tiene más votos de favoritos, pero el de word embedding más de satisfacción. Incluso si este algoritmo satisface a varios usuarios (10 usuarios de 22), no significa que vaya a ser la favorita. De hecho con el algoritmo colaborativo pocos se satisficieron,

pero casi todos de ellos lo marcaron como favorito, en cambio la mitad de votos de 'satisfacción' del método WE se tradujeron en votos de 'favorito'. Parece ser condición necesaria que a un usuario le haya satisfecho la lista para que sea favorita, pero hubieron casos de usuarios a los cuales ninguna lista le satisfizo (5 usuarios). Dadas las condiciones del estudio no obstante los usuarios se vieron forzados a escoger un algoritmo favorito.

El método TF-IDF+ no es el que se percibe como más diverso, pero es el que produce ítems más relevantes con las mejores ordenaciones internas. Que el método WE G* m.2b-cos tenga malos resultados offline pero buenos online podría ser explicado por otros factores como diversidad, que aumenten la percepción de satisfacción del usuario con el algoritmo (Ricci et al., 2010). Aún cuando el espacio de ítems recomendables es bajo (67,261) en comparación al total de libros en Goodreads (2.3B de libros registrados[1]), se pudo obtener buenos resultados en términos de relevancia e incluso de diversidad percibida.

Al final de la encuesta a cada usuario, les pedimos que comenten con sus propias palabras acerca del porqué de su elección de algoritmo favorito. Las opiniones son variadas. A algunos usuarios les satisfizo cierto recomendador por ser los ítems de su lista novedosos y diversos entre sí, como pasó principalmente con la percepción de los usuarios sobre las lista de los recomendadores WE G* m.2b-cos y TF-IDF+ m.1. A otros lo contrario: a pesar de ser diversos, pudieron encontrar ítems relevantes en ellos, percepción mostrada en algunos casos sobre la lista CF. Sobre la lista del recomendador de word embedding se criticó que los libros recomendados pertenecían más que nada al mismo autor. Para algunos usuarios el idioma jugó un factor importante, comentando que les satisfacían menos las listas que recomendaban ítems en idioma distinto al español, lo cual sucedió para los tres algoritmos. Según los comentarios pareciera que la mayor percepción de relevancia la obtuvo la lista TF-IDF+, seguido por la de WE. La

[1] `https://www.goodreads.com/about/us`

mayor crítica hacia el algoritmo CF era su poca relevancia en las recomendaciones. Esto se correlaciona con los resultados de la tabla 6.33. También correlaciona la diversidad percibida, dado que la percepción de novedad por los usuarios sobre los ítems fue en general el mayor elogio que obtuvo el algoritmo WE.

Finalmente, son los sistemas basados en contenido los mayormente preferidos por los usuarios, lo que nuevamente puede dar indicios sobre el tipo de algoritmo adecuado para el dominio de libros.

7.2. Corrección por Libros con Múltiples Versiones

Ampliando la explicación dada en la sección 6.2, tuvimos que hacer una corrección debido a que observamos que en las listas recomendadas a los usuarios del estudio online[2] aparecían múltiples versiones de esencialmente el mismo libro (mismo título y mismo autor), y que estas múltiples versiones eran diversas editoriales que ocupaban distintas rutas en Goodreads. Acerca de este método de corrección, el razonamiento por el que se selecciona aleatoriamente uno de ellos es simplemente para no otorgar favoritismos a ciertas editoriales o formatos, sin embargo es posible que el usuario sí tenga favoritismos con respecto a este tipo de meta-contenido del libro. El problema de esta solución es que de entre los ítems desechados podría estar el ítem consumido por el usuario, y que por tanto éste no corresponda al seleccionado, asumiendo que todos estos ítems sean el mismo pero en otras versiones. En el caso que esto suceda, a pesar que el ítem seleccionado es un ítem relevante (por ser consumido), los resultados empeorarán. Un enfoque que ayudaría a inflar los resultados en el testing sería, en caso de que en la lista recomendada se hallen varios libros iguales en versiones diferentes, ver de antemano cuál de esas versiones fue la consumida por el usuario, de haber en la lista libros consumidos, y dejar sólo las versiones que el usuario consumió (en caso que no se hallen libros consumidos en la lista simplemente dejar cualquiera

[2]Notar que dicha corrección se aplicó antes de enviar las recomendaciones a los usuarios.

de los repetidos). El problema de esto es que no podemos hacerlo en la práctica, dado que no sabemos de antemano con certeza cuál de esas versiones será una de las preferidas por el usuario. Quizás el usuario sólo encontrará relevantes libros en ciertos idiomas, quizás tenga un traductor preferido, o puede que la página de Goodreads que muestra el libro específico recomendado no tenga información suficiente sobre éste, y que en otras versiones sí la tenga, y puede que tener o no esa información sea el factor decisivo de aceptación de la recomendación. Esto puede tener efectos incluso más allá de tener un papel en el discernimiento personal del usuario sobre la relevancia del libro. El llevar al usuario a un sitio cuyo contenido está en un idioma ininteligible para él, o que no tenga contenido sustantivo alguno, puede hacer disminuir la confianza que el usuario tiene por el recomendador.

Una alternativa sería seleccionar el libro correspondiente a aquél cuya URL sea la URL canónica que, por lo visto, llevan a páginas con buena cantidad de datos. El asunto con esto es que, aun si supiésemos de antemano los idiomas preferidos por el usuario dado su input (asumiendo que el idioma de las páginas de los libros que forman parte de su input es parte de los idiomas que el usuario maneja y prefiere), las URL canónicas nos pueden dirigir a páginas con texto en un idioma ininteligible. Por ejemplo, sea un sistema recomendador preparado para recomendar ítems del mismo idioma que los ítems del input del usuario; y sea, por simplicidad, todos los ítems del input de un mismo idioma, dado que es de la preferencia del usuario. Sea este idioma el español. De algún modo, de entre los ítems recomendados se encuentran aquellos con las siguientes URIs: `/book/show/15868.Harry_Potter_y_la_piedra_filosofal`, `/book/show/430569.Harry_Potter_l_cole_des_sorciers` y `/book/show/3.Harry_Potter_and_the_Sorcerer_s_Stone`[3]. La información encontrada en el primero, segundo y tercero, respectivamente, está

[3]Para reconstruir la ruta completa, anteponer `https://www.goodreads.com`

en español, francés e inglés. En los HTML de cada una de estas rutas encontramos la misma URL canónica: `/book/show/3.Harry_Potter_and_the_Sorcerer_s_Stone`. Fuera ésta la recomendada al usuario, es bien probable que él no se sienta a gusto con la recomendación. En este caso la URL canónica dirige a información del mismo idioma que del libro original, pero hay casos en donde dirige a información no del mismo idioma del libro original. Por ejemplo, `/book/show/46171.Rayuela`, que está en español, idioma tanto de la información de ahí como del libro original, y `/book/show/53413.Hopscotch` que está en inglés, son en realidad el mismo libro, es decir comparten la misma URL canónica, la cual es la con información en inglés: `/book/show/53413.Hopscotch`. Sumando a la complejidad, hay libros que parecen ser los mismos, por lo que sigue que deban tener la misma URL canónica, pero no es así. El sitio de 'El Hobbit' de J. R. R. Tolkien, en español, es `/book/show/42963399-el-hobbit`; y en inglés, `/book/show/5907.The_Hobbit`. Sin embargo la URL canónica del primero es `/book/show/42963399-el-hobbit`; y la del segundo, `/book/show/5907.The_Hobbit`, por lo que, llegara un sistema a generar una lista de recomendación con ambas URIs incluidas, no podría descartar ninguno de estos dos ítems que son aparentemente el mismo, y como efecto de recomendar ambos pueda bajar la confianza del usuario sobre éste, además de innecesariamente gastar un espacio en la lista que podría ocupar otro ítem, posiblemente relevante.

Una solución que tome en cuenta todos estos posibles problemas sería primero detectar idiomas, traductores, ediciones y formatos preferidos del usuario a partir de su input, y al generar las recomendaciones, en caso que hayan múltiples versiones de los ítems, dejar aquellos que estén acorde a las preferencias del usuario sobre el libro más allá de su contenido (como las mencionadas anteriormente), tomando en cuenta la cantidad de información que en la página de Goodreads se muestre de ellos, contabilizada según un criterio conveniente producto de algún estudio paralelo. Puede que incluso estas características no sean

sólo cruciales en la aceptación del usuario sobre la recomendación, de presentarse el caso que el sistema deba elegir uno de entre un set de libros 'iguales', sino puede que sean también importantes en la aceptación aun si el sistema recomienda sólo ítems distintos, sin tener que haber pasado la lista previamente por una función de corrección como las descritas recién. Para evaluar la importancia de este meta-contenido habría que proceder parecido a como se hizo para buscar los parámetros óptimos de los modelos TF-IDF+, en donde se encontró, entre otros, los campos ideales de los documentos para hacer las comparaciones. Lamentablemente éstos no son campos obligatorios de los libros de Goodreads, i.e., no es condición necesaria que este tipo de información esté para que el libro forme parte de la base de datos de Goodreads (como detallado en la sección 5.1.1), por lo que sólo podría hacerse este experimento para aquellos libros que sí la posean. Se desconoce la proporción de libros que cuentan con estos datos con los que no, para cada uno de estos campos. Además, habría que evaluar en profundidad el efecto que tiene cada meta-contenido sobre cada usuario, lo que conllevaría a entrenar modelos para cada usuario como lo hace Said et al. (2013). La realización de cada uno de estos experimentos y el impacto que tienen sobre la satisfacción final de los usuarios queda planteado como trabajo futuro

7.3. Conclusiones

Nuestras preguntas de investigación eran (RQ. 1), (RQ. 2) y (RQ. 3). A modo de conclusión, en esta sección responderemos estas preguntas con el conocimiento adquirido por este trabajo.

(RQ. 1) **¿Qué enfoque es preferible tener sobre el feedback del usuario para crear un sistema recomendador que produzca las mejores recomendaciones de libros?**

Nuestro sistema colaborativo de SVD con feedback implícito por un amplio margen obtuvo mejores resultados que aquellos métodos de filtrado

colaborativo que explícitamente usan ratings. El sistema CF BPR (*Bayesian Personalized Ranking*), a pesar que también se alimenta de información implícita sobre los gustos del usuario, no dio resultados comparables con los del anterior. Esto puede deberse a que los gustos de los usuarios por libros no son en general asociativos, por lo que un tipo de recomendador que se entrene con ese tipo de datos no se adaptaría bien a la situación, mientras que considerar a los ratings en una escala 1-5 como representativos de un gusto binario ('me gusta'/'no me gusta') del usuario, discernible a partir del promedio de ratings del usuario, pareciera ser lo apto. Adicionalmente, agregamos evidencia a la literatura, a parte de los estudios de Vaz, Martins de Matos, Martins, y Calado (2012); Vaz et al. (2013a), que considerar el nombre de los autores como parte del feedback con el que se alimenta al sistema da mejores resultados en comparación a no agregar información contextual sobre las interacciones.

(RQ. 2) **¿Cuál es la manera de representar libros y perfiles de usuarios para recomendar libros que produzca los mejores resultados?**

Con respecto a la pregunta (RQ. 2) fuimos capaces de crear diferentes tipos de sistemas basados en contenido que recomiendan libros en base a distintos tipos de perfiles de libros y de usuarios. En una representación de documentos por Bag-of-Words, en donde los términos tienen básicamente pesos TF-IDF (con modificaciones añadidas por la interfaz de similaridad de Lucene), lo que dio mejor resultado fue usando toda la información posible del libro (descripción+autores+título+géneros+citas) en donde consideramos al usuario como una agrupación disjunta de los libros consumidos, y que por tanto la recomendación final surge producto de una ordenación de todas las recomendaciones de cada uno de los ítems del input (modalidad 1 de recomendación, descrito en la sección 4.2.3). De tener en cuenta sólo un campo, fue el campo de la descripción que rindió resultados más favorables, tanto en modo 1 como en modo 2. La

'modalidad 1' de recomendación también resultó ser notoriamente más favorable para los resultados que la modalidad 2 en cuanto a modelos de word embeddings se refiere. Sobre estos modelos, representar a los libros como un *max pooling* de sus vectores de palabras en un espacio latente de baja dimensionalidad y recomendar según la modalidad 1 dio resultados comparables al mejor de los métodos CB. Esto suma a la literatura hallazgos favorables para el uso de word embeddings en recomendaciones de este dominio (Alharthi et al., 2017).

(RQ. 3) **¿Existe un sistema híbrido que produzca mejores resultados que los algoritmos investigados en (RQ. 1) y (RQ. 2)? Si es así, ¿qué tipo de hibridación es la mejor?**

En relación a la pregunta (RQ. 3) logramos descubrir un sistema híbrido que produce mejores resultados que todos los otros algoritmos evaluados. La mejor hibridación se logró juntando los dos sistemas que mejor resultados lograban en un esquema ponderado: TF-IDF+ m.1 y SVD implícito, la que se compone en prácticamente un 90 % por el algoritmo CB. Esto nuevamente da señales sobre la adecuación de los algoritmos basados en contenido para este problema, incluso cuando penalizamos a los sistemas CB agrandando el espacio de ítems recomendables (67,261 en total) en relación al espacio de ítems recomendables para los sistemas CF (23,716 ítems, ninguno fuera de aquellos alguna vez consumidos por los 441 usuarios de nuestro dataset), para no lidiar este último con problemas de *cold start* .

7.4. Limitaciones

A continuación listamos algunas limitaciones de nuestro trabajo:

- Aun si diferenciamos los espacios de ítems recomendables para los sistemas evaluados por alguna razón, esto de todas formas introduce sesgo, inevitablemente penalizando a aquél con un espacio de ítems mayor, haciendo las comparaciones entre los resultados de distintos algoritmos desbalanceada. Tanto para el caso offline como para el online.
- Aunque el espacio de ítems recomendables es bajo para los algoritmos basados en contenido en comparación con el total de libros de Goodreads (2.3B), se pudo obtener de igual forma buenos resultados en términos de relevancia e incluso de diversidad percibida. Sin embargo en el caso real sería preferible contar con todos los libros posibles de la aplicación.
- La razón por la que en evaluaciones modo 2 del método TF-IDF+ usamos sólo 1 campo (ver tabla 6.9) y no campos compuestos como en modo 1 (tabla 6.8) fue por una limitación en el funcionamiento interno del componente MoreLikeThis del buscador de Lucene: para representar al usuario como una agregación de todos los documentos consumidos, usamos la funcionalidad `ContentStreams` de Solr[4] para transmitir todos los documentos consumidos en una plana de texto, el cual permitía, en versiones anteriores a Solr 7.4, usar sólo un campo para comparación (i.e. dar sólo un valor a `mlt.fl`). Sin embargo para nuestros experimentos usamos la versión 6.5.1.
- Según el ajuste de parámetros de los métodos TF-IDF+, el campo -no compuesto- que optimiza las recomendaciones es el de la descripción, seguido con un considerable margen por el de autores. Fue sin embargo los nombres de los autores lo que se usó (además de la fecha de las interacciones) para dar como información contextual al método CF que mejor resultados daba sin suplir información contextual. Esto guiados tanto por la intuición, como por los resultados del ajuste de parámetros del método TF-IDF+, como por las sugerencias que encontramos de la

[4] `https://wiki.apache.org/solr/MoreLikeThisHandler`

literatura (Vaz, Martins de Matos, Martins, y Calado, 2012; Vaz et al., 2013a; Zhang et al., 2016), pero también porque usar la descripción como contexto requeriría de un trabajo de modelamiento extra. Como sugerido anteriormente, esto podría verse como un problema de hibridación por combinación de features.

- A pesar que pudimos hacer un arreglo para corregir el problema de tener múltiples ediciones de un mismo libro, tenemos de todas maneras la incertidumbre de no saber exactamente qué libro de ese conjunto de libros de varias versiones el usuario va a consumir, en el caso de que alguno de ellos le parezca relevante. Vemos que este problema es único en el dominio de los libros.
- En el escenario con usuarios no usamos realmente los 3 mejores algoritmos no-híbridos según los resultados offline, dado que uno de ellos, WE G* m.2b-cos, no es mejor que WE G* m.1-euc como que ve en los resultados. Esto por el problema de libros con varias editoriales como explicado en 6.2. Esto podría ayudar a dar mayor evidencia de que los métodos WE pueden ser idóneos para la recomendación de libros, sin embargo aún con un método WE que dio resultados sub-óptimos en las evaluaciones offline se pudo obtener buenos resultados en la evaluación online gracias a la alta diversidad percibida por los usuarios sobre las listas de éste algoritmo.

CAPÍTULO 8. TRABAJO FUTURO

Este capítulo discute algunas directrices sobre posibilidades de investigación futuras en este dominio.

- Actualmente evaluamos el método TF-IDF+ con representación del usuario sólo por sus libros. Para complementar los métodos word embedding evaluados, se requeriría representar al usuario por sus tuits, y por sus libros y tuits, en los métodos TF-IDF+. Con el mismo argumento, se podría probar evaluar TF-IDF+ en ambos modos con distancia euclidiana al igual que en los métodos de word embedding.
- Dado que trabajamos con una versión de Solr previa a 7.4.0, no pudimos darle más de un valor al parámetro `mlt.fl`, que es el que rige qué campos se usarán para hacer la búsqueda de documentos similares. La versión actual de Solr es 7.5.0, y ahora que está implementado, sería buena idea componer campos en modo 2 tal como se hizo en modo 1 para buscar el valor óptimo de `mlt.fl`.
- En nuestros documentos-libros contamos con información importante que no aprovechamos. Entre ellos están los votos de los usuarios por los géneros y las citas textuales (como explicado en la tabla 5.1). Actualmente es posible dar distinto peso a distintos campos en la búsqueda de documentos similares usando el parámetro `mlt.qf`[1], pero no es posible, a menos que se implemente en el código base de Lucene, una manera de dar *boost* a ciertos valores de algunos campos (como aquellos en los campos de géneros y citas) basado en el valor de otro campo (como el de los votos de géneros y votos por citas). Como ejemplo consideremos el siguiente documento:

```
{
 "bookId": 1,
```

[1] `https://lucene.apache.org/solr/guide/6_6/morelikethis.html`

```
 "genres":["Nonfiction", "Science", "Skepticism", "Psychology"],
 "genresVotes":[8, 6, 3, 1]
}
```


La idea sería entonces, en el vector TF-IDF de este documento, darle más pesos a los términos dentro del campo 'genres', y de ellos, pesar todavía más 'Nonfiction', luego 'Science', etc., según una función que dependa de los valores del campo 'genresVotes'. Análogamente para el campo de las citas textuales y el campo de sus votos.

- Para los métodos CB, usamos como input todos los libros consumidos por el usuario: los que tienen tanto bajo como alto rating. Podría hacerse los mismos experimentos pero sólo tomando en cuenta aquellos libros considerados como relevantes por el usuario (discernido a partir de por ejemplo el promedio de sus ratings).
- Se podría hacer un mayor análisis del texto de los tuits del usuario en aquellos métodos métodos basados en contenido que los usen para representar al usuario, extrayendo por ejemplo palabras claves -como los hashtags- por las que se juzgue los gustos y preferencias del usuario.
- En el modo 1 del modelo BoW y del word embedding la ordenación de los ítems de salida para estructurar la lista final de recomendación está dada puramente por el orden por el que se van generando las recomendaciones para los ítems individuales de entrada (con más detalle en la sección 4.2.3). En otras palabras el ránking de los ítems de la lista de recomendación está dado por el orden de los ítems en el set de entrenamiento del usuario, lo cual puede ser completamente arbitrario. Una manera más controlada de generar la lista de salida sería priorizar aquellos ítems recomendados como resultado de dar como input al sistema aquellos ítems del set de entrenamiento con más alto rating, por ejemplo.
- Guíado por el estudio de Ross (1998), en el que reportan que el tercer factor más importante al momento de escoger libros nuevos por leer es el

de autores y géneros preferidos por el usuario, un experimento interesante sería incluir en el set de información contextual a los géneros de los libros.

- Éspecíficamente para los métodos de word embedding, en nuestros experimentos se mapeó el documento completo de cada ítem a vector, considerando los términos de todos sus campos. Un análisis adicional en esto sería ver la influencia de cada campo (y composiciones de éstos) sobre los resultados de los métodos de word embedding al mapear el ítem a vector sólo tomando en cuenta los términos de determinados campos, similar al procedimiento en el modelo BoW.
- Una métrica de distancia interesante de evaluar para los métodos de word embedding sería la *Word Mover's Distance* (WMD), definida en Kusner, Sun, Kolkin, y Weinberger (2015) como la medición de disimilaridad entre un par de documentos como la menor cantidad de distancia que las palabras embebidas de un documento tienen que 'viajar' para llegar a las palabras embebidas del otro documento, el cual es una métrica que muestra promisorios resultados para métodos con embeddings.
- Dado que el recomendador basado en filtrado colaborativo que mejor resultados dio es el SVD usando feedback implícito, un interesante cambio sería usar ese mismo feedback implícito con información contextual en métodos de máquinas de factorización, y ver los resultados en comparación a usar feedback explícito.
- Un recomendador de línea de base que se podría tener sería evaluar offline y online los recomendadores nativo de Goodreads, el cual tiene dos principales recomendaciones: una sección de recomendaciones colaborativas llamada *READERS ALSO ENJOYED*, y una sección de recomendaciones del mismo autor del tipo *BOOKS BY SAME AUTHOR*, aunque no todos los libros cuentan con estas recomendaciones. Por otro lado vemos en esto una posibilidad de hibridación con algunos de nuestros sistemas

evaluados del tipo *aumentación de features*, dado que en base al resultado de un algoritmo se alimentaría otro sistema para producir recomendaciones más refinadas.

- Los resultados y conclusiones de este trabajo son preliminares a aplicaciones en sistemas reales de recomendación de libros. Se podría estudiar si estos algoritmos son rentables para recomendar e-books, si valdría la pena aplicarlos a recomendación en tiempo real en librerías, o si sería significativo integrarlos en una aplicación de red social o a una comunidad virtual de catalogación. Mayores investigaciones y análisis son requeridos para evaluar el impacto que estos algoritmos tendrían en producción.

REFERENCIAS

Adomavicius, G., y Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge & Data Engineering*(6), 734–749.

Ahde, A. (2015, noviembre). *What are the differences between knowledge-based recommender systems and content-based recommender systems?* Descargado de `https://www.quora.com/What-are-the-differences-between-knowledge-based-recommender-systems-and-content-based-recommender-systems/answer/Ahti-Ahde-1` (Recuperado 2018-12-30)

Ahn, H. J. (2008, enero). A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem. *Inf. Sci.*, *178*(1), 37–51. Descargado de `https://doi.org/10.1016/j.ins.2007.07.024` doi: 10.1016/j.ins.2007.07.024

Alharthi, H., Inkpen, D., y Szpakowicz, S. (2017). Unsupervised topic modelling in a book recommender system for new users. En *Proceedings of the sigir 2017 workshop on ecommerce (ecom 17).*

Alharthi, H., Inkpen, D., y Szpakowicz, S. (2018). A survey of book recommender systems. *Journal of Intelligent Information Systems*, *51*(1), 139–160.

Arazy, O., Kumar, N., y Shapira, B. (2009). Improving social recommender systems. *IT professional*, *11*(4).

Baeza-Yates, R., Ribeiro, B. d. A. N., y cols. (2011). *Modern information retrieval*. New York: ACM Press; Harlow, England: Addison-Wesley,.

Balabanović, M., y Shoham, Y. (1997). Fab: content-based, collaborative recommendation. *Communications of the ACM*, *40*(3), 66–72.

Bayer, I. (2015). fastfm: A library for factorization machines. *CoRR*, *abs/1505.00641*. Descargado de `http://arxiv.org/abs/1505.00641`

Becoming a data-driven ceo. (2018). `https://www.domo.com/solution/data-never-sleeps-6`. (Recuperado 2018-11-19)

Bell, R., Koren, Y., y Volinsky, C. (2007). Modeling relationships at multiple scales to improve accuracy of large recommender systems. En *Proceedings of the 13th acm sigkdd international conference on knowledge discovery and data mining* (pp. 95–104).

Benkoussas, C., y Bellot, P. (2013, 06). Book recommendation based on social information..

Bojanowski, P., Grave, E., Joulin, A., y Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, *5*, 135–146.

Bostandjiev, S., O'Donovan, J., y Höllerer, T. (2012). Tasteweights: a visual interactive hybrid recommender system. En *Proceedings of the sixth acm conference on recommender systems* (pp. 35–42).

Bridge, D., Göker, M. H., McGinty, L., y Smyth, B. (2005). Case-based recommender systems. *The Knowledge Engineering Review*, *20*(3), 315–320.

Buder, J., y Schwind, C. (2012). Learning with personalized recommender systems: A psychological view. *Computers in Human Behavior*, *28*(1), 207–216.

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, *12*(4), 331–370.

Burke, R. (2007). Hybrid web recommender systems. *The Adaptive Web*, 377–408. doi: 10.1007/978-3-540-72079-9_12

Canny, J. (2002). Collaborative filtering with privacy. En *Security and privacy, 2002. proceedings. 2002 ieee symposium on* (pp. 45–57).

Chen, Y.-F. (2008). Herd behavior in purchasing books online. *Computers in Human Behavior*, *24*(5), 1977–1992.

Crespo, R. G., Martínez, O. S., Lovelle, J. M. C., García-Bustelo, B. C. P., Gayo, J. E. L., y Pablos, P. O. n. d. (2011, julio). Recommendation system based on user interaction data applied to intelligent electronic books. *Comput. Hum. Behav.*, *27*(4), 1445–1449. Descargado de http://dx.doi.org/10.1016/j.chb.2010.09.012 doi: 10.1016/j.chb.2010.09.012

Flood, A. (2014, Oct). *Uk publishes more books per capita than any other country, report shows.* Guardian News and Media. Descargado de https://www.theguardian.com/books/2014/oct/22/uk-publishes-more-books-per-capita-million-report (Recuperado 2018-12-30)

Flood, A. (2015, Jan). *Sharp decline in children reading for pleasure, survey finds.* Guardian News and Media. Descargado de https://www.theguardian.com/books/2015/jan/09/decline-children-reading-pleasure-survey (Recuperado 2018-12-30)

Frederickson, B. (2018). *Implicit.* https://github.com/benfred/implicit. GitHub.

Funk, S. (2006, Dec). https://www.ibm.com. (Recuperado 2019-01-03)

Gadanho, S. C., y Lhuillier, N. (2007). Addressing uncertainty in implicit preferences. En *Proceedings of the 2007 acm conference on recommender systems* (pp. 97–104).

Garrido, A. L., Pera, M. S., y Ilarri, S. (2014). Sole-r: A semantic and linguistic approach for book recommendations. En *2014 ieee 14th international conference on advanced learning technologies (icalt)* (pp. 524–528).

Givon, S., y Lavrenko, V. (2009). Predicting social-tags for cold start book recommendations. En *Proceedings of the third acm conference on recommender systems* (pp. 333–336).

Grimes, S. (2008). *Structured data.* `http://breakthroughanalysis.com/2008/08/01/unstructured-data-and-the-80-percent-rule/`. (Recuperado 2018-12-11)

Hajirahimova, M. S., y Aliyeva, A. S. (2017). About big data measurement methodologies and indicators. *International Journal of Modern Education and Computer Science*, *9*(10), 1.

Harris, Z. S. (1954). Distributional structure. *Word*, *10*(2-3), 146–162.

Hill, K. (2013). The arts and individual well-being in canada. *Statistical Insights on the Arts*, *11*(2), 1–35.

Hu, R., y Pu, P. (2009). Acceptance issues of personality-based recommender systems. En *Proceedings of the third acm conference on recommender systems* (pp. 221–224).

Hu, Y., Koren, Y., y Volinsky, C. (2008). Collaborative filtering for implicit feedback datasets. En *Data mining, 2008. icdm'08. eighth ieee international conference on* (pp. 263–272).

Isinkaye, F., Folajimi, Y., y Ojokoh, B. (2015). Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, *16*(3), 261–273.

Jacobson, R. (2013). *2.5 quintillion bytes of data created every day. how does cpg & retail manage it?* https://www.ibm.com/blogs/insights-on-business/consumer-products/2-5-quintillion-bytes-of-data-created-every-day-how-does-\cpg-retail-manage-it/. (Recuperado 2018-11-21)

Jannach, D., Zanker, M., Felfernig, A., y Friedrich, G. (2010). *Recommender systems: an introduction*. Cambridge University Press.

Järvelin, K., y Kekäläinen, J. (2002). Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, *20*(4), 422–446.

Kapusuzoglu, H., y Öguducu, S. G. (2011). A relational recommender system based on domain ontology. En *Emerging intelligent data and web technologies (eidwt), 2011 international conference on* (pp. 36–41).

Kim, D., y Yum, B.-J. (2005). Collaborative filtering based on iterative principal component analysis. *Expert Systems with Applications*, *28*(4), 823–830.

Koberstein, J., y Ng, Y.-K. (2006). Using word clusters to detect similar web documents. En *International conference on knowledge science, engineering and management* (pp. 215–228).

Konstan, J. A., y Riedl, J. (2012, 01 de Apr). Recommender systems: from algorithms to user experience. *User Modeling and User-Adapted Interaction*, *22*(1), 101–123. Descargado de https://doi.org/10.1007/s11257-011-9112-x doi: 10.1007/s11257-011-9112-x

Koren, Y., Bell, R., y Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*(8), 30–37.

Kusner, M., Sun, Y., Kolkin, N., y Weinberger, K. (2015). From word embeddings to document distances. En *International conference on machine learning* (pp. 957–966).

Lathia, N., Hailes, S., Capra, L., y Amatriain, X. (2010, 01). Temporal diversity in recommender systems. En (p. 210-217). doi: 10.1145/1835449.1835486

Lohr, S. (2013). *The origins of 'big data': An etymological detective story.* `https://bits.blogs.nytimes.com/2013/02/01/the-origins-of-big-data-an-etymological-detective-story/`. (Recuperado 2018-11-22)

Lynch, C. (2018). *Factorization machines in python.* `https://github.com/coreylynch/pyFM`. GitHub.

Mahmood, T., y Ricci, F. (2007). Towards learning user-adaptive state models in a conversational recommender system. En *Lwa* (pp. 373–378).

Mahmood, T., y Ricci, F. (2009). Improving recommender systems with adaptive conversational strategies. *Proceedings of the 20th ACM conference on Hypertext and hypermedia - HT 09*, 73–82. doi: 10.1145/1557914.1557930

Maneewongvatana, S., y Maneewongvatana, S. (2010). A recommendation model for personalized book lists. En *Communications and information technologies (iscit), 2010 international symposium on* (pp. 389–394).

Manning, C. D., Raghavan, P., y Schütze, H. (2008). *Introduction to information retrieval*. Cambridge University Press. doi: 10.1017/CBO9780511809071

Mar, R. A., y Oatley, K. (2008). The function of fiction is the abstraction and simulation of social experience. *Perspectives on psychological science*, *3*(3), 173–192.

Mar, R. A., Oatley, K., y Peterson, J. B. (2009). Exploring the link between reading fiction and empathy: Ruling out individual differences and examining outcomes. *Communications*, *34*(4), 407–428.

Mashey, J. R. (1998). *Big data ... and the next wave of infrastress.* Presentación de charla invitada: https://www.usenix.org/conference/1999-usenix-annual-technical-conference/big-data-and-next-wave-infrastress-problems. (Recuperado 2018-11-22)

McNee, S. M., Kapoor, N., y Konstan, J. A. (2006). Don't look stupid: Avoiding pitfalls when recommending research papers. En *Proceedings of the 2006 20th anniversary conference on computer supported cooperative work* (pp. 171–180). New York, NY, USA: ACM. Descargado de http://doi.acm.org/10.1145/1180875.1180903 doi: 10.1145/1180875.1180903

Mikolov, T., Chen, K., Corrado, G., y Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., y Dean, J. (2013). Distributed representations of words and phrases and their compositionality. En *Advances in neural information processing systems* (pp. 3111–3119).

Mooney, R. J., y Roy, L. (2000). Content-based book recommending using learning for text categorization. En *Proceedings of the fifth acm conference on digital libraries* (pp. 195–204).

Morelikethis. (2017, Jun). https://lucene.apache.org/solr/guide/6_6/morelikethis.html. (Recuperado 2019-01-02)

Muse, D. (2017). *Structured data.* https://www.datamation.com/big-data/structured-data.html. (Recuperado 2018-12-11)

Musto, C., Semeraro, G., De Gemmis, M., y Lops, P. (2015). Word embedding techniques for content-based recommender systems: An empirical evaluation. En *Recsys posters.*

Musto, C., Semeraro, G., de Gemmis, M., y Lops, P. (2016). Learning word embeddings from wikipedia for content-based recommender systems. En *European conference on information retrieval* (pp. 729–734).

Negative hypergeometric distribution. Encyclopedia of Mathematics. (2010). `https://www.encyclopediaofmath.org/index.php/Negative_hypergeometric_distribution`. (Recuperado 2018-12-24)

Noia, T. D., Ostuni, V. C., Tomeo, P., y Sciascio, E. D. (2016, septiembre). Sprank: Semantic path-based ranking for top-n recommendations using linked open data. *ACM Trans. Intell. Syst. Technol.*, *8*(1), 9:1–9:34. Descargado de `http://doi.acm.org/10.1145/2899005` doi: 10.1145/2899005

Parra, D., y Brusilovsky, P. (2015, junio). User-controllable personalization. *Int. J. Hum.-Comput. Stud.*, *78*(C), 43–67. Descargado de `http://dx.doi.org/10.1016/j.ijhcs.2015.01.007` doi: 10.1016/j.ijhcs.2015.01.007

Parra, D., y Sahebi, S. (2013). Recommender systems: Sources of knowledge and evaluation metrics. En *Advanced techniques in web intelligence-2* (pp. 149–175). Springer.

Pathak, B., Garfinkel, R., Gopal, R. D., Venkatesan, R., y Yin, F. (2010). Empirical analysis of the impact of recommender systems on sales. *Journal of Management Information Systems*, *27*(2), 159–188.

Pathak, D., Matharia, S., y Murthy, C. (2013). Nova: Hybrid book recommendation engine. En *Advance computing conference (iacc), 2013 ieee 3rd international* (pp. 977–982).

Pazzani, M., y Billsus, D. (1997). Learning and revising user profiles: The identification of interesting web sites. *Machine learning*, *27*(3), 313–331.

Pera, M. S., Condie, N., y Ng, Y.-K. (2010). Personalized book recommendations created by using social media data. En *International conference on web information systems engineering* (pp. 390–403).

Pera, M. S., y Ng, Y.-K. (2011). With a little help from my friends: generating personalized book recommendations using data extracted from a social website. En *Proceedings of the 2011 ieee/wic/acm international conferences on web intelligence and intelligent agent technology-volume 01* (pp. 96–99).

Pera, M. S., y Ng, Y.-K. (2013). What to read next?: making personalized book recommendations for k-12 users. En *Proceedings of the 7th acm conference on recommender systems* (pp. 113–120).

Pera, M. S., y Ng, Y.-K. (2014a). Automating readers' advisory to make book recommendations for k-12 readers. En *Proceedings of the 8th acm conference on recommender systems* (pp. 9–16).

Pera, M. S., y Ng, Y. K. (2014b). How can we help our k-12 teachers?: using a recommender to make personalized book suggestions. En *Proceedings of the 2014 ieee/wic/acm international joint conferences on web intelligence (wi) and intelligent agent technologies (iat)-volume 02* (pp. 335–342).

Pera, M. S., y Ng, Y.-K. (2015). Analyzing book-related features to recommend books for emergent readers. En *Proceedings of the 26th acm conference on hypertext & social media* (pp. 221–230).

Priyanka, K., Tewari, A. S., y Barman, A. G. (2015). Personalised book recommendation system based on opinion mining technique. En *2015 global conference on communication technologies (gcct)* (p. 285-289). Descargado de `https://app.dimensions.ai/details/publication/pub`

.1095323265 (Exported from https://app.dimensions.ai on 2018/12/30) doi: 10.1109/gcct.2015.7342668

Pu, P., Chen, L., y Hu, R. (2011). A user-centric evaluation framework for recommender systems. En *Proceedings of the fifth acm conference on recommender systems* (pp. 157–164).

Rajpurkar, S., Bhatt, D., Malhotra, P., Rajpurkar, M. S. S., Bhatt, M. D. R., y Malhotra, M. P. (2015). Book recommendation system. *International Journal For Innovative Research In Science And Technology, ISSN: 314–316*, *1*(11, 1).

Rana, C., y Jain, S. K. (2012). Building a book recommender system using time based content filtering. *WSEAS Transactions on Computers*, *11*(2), 2224–2872.

Řehůřek, R., y Sojka, P. (2010, 22 de mayo). Software Framework for Topic Modelling with Large Corpora. En *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks* (pp. 45–50). Valletta, Malta: ELRA. (http://is.muni.cz/publication/884893/en)

Rendle, S. (2010). Factorization machines. En *Data mining (icdm), 2010 ieee 10th international conference on* (pp. 995–1000).

Rendle, S. (2012). Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *3*(3), 57.

Rendle, S., Freudenthaler, C., Gantner, Z., y Schmidt-Thieme, L. (2012). BPR: bayesian personalized ranking from implicit feedback. *CoRR*, *abs/1205.2618*. Descargado de http://arxiv.org/abs/1205.2618

Ricci, F., Rokach, L., Shapira, B., y Kantor, P. B. (2010). *Recommender systems handbook* (1st ed.). Berlin, Heidelberg: Springer-Verlag.

Ross, C. (1998, 06). Making choices: What readers say about choosing books to read for pleasure.. doi: 10.13140/2.1.1410.9129

Said, A., Bellogín, A., y De Vries, A. (2013). A top-n recommender system evaluation protocol inspired by deployed systems..

Salton, G. (1989). Automatic text processing: The transformation, analysis, and retrieval of. *Reading: Addison-Wesley*.

Sarwar, B., Karypis, G., Konstan, J., y Riedl, J. (2000). *Application of dimensionality reduction in recommender system-a case study* (Inf. Téc.). Minnesota Univ Minneapolis Dept of Computer Science.

Sase, A., Varun, K., Rathod, S., y Patil, D. (2015). A proposed book recommender system. *International Journal of Advanced Research in Computer and Communication Engineering*, *4*(2), 481–483.

Sepulveda, G., Dominguez, V., y Parra, D. (2017, August). pyreclab: A software library for quick prototyping of recommender systems..

Skipworth, H. (2010, Aug). *Google counts total number of books in the world.* Telegraph Media Group. Descargado de `https://www.telegraph.co.uk/technology/google/7930273/Google-counts-total-number-of-books-in-the-world.html` (Recuperado 2018-12-30)

Sohail, S. S., Siddiqui, J., y Ali, R. (2013, Aug). Book recommendation system using opinion mining technique. En *2013 international conference on advances in computing, communications and informatics (icacci)* (p. 1609-1614). doi: 10.1109/ICACCI.2013.6637421

Steyvers, M., y Griffiths, T. (2007). Probabilistic topic models. *Handbook of latent semantic analysis*, *427*(7), 424–440.

Suriati, S., Dwiastuti, M., y Tulus, T. (2017). Weighted hybrid technique for recommender system. En *Journal of physics: Conference series* (Vol. 930,

p. 012050).

Takács, G., Pilászy, I., Németh, B., y Tikk, D. (2007). Major components of the gravity recommendation system. *Acm Sigkdd Explorations Newsletter*, *9*(2), 80–83.

Tsuji, K., Takizawa, N., Sato, S., Ikeuchi, U., Ikeuchi, A., Yoshikane, F., y Itsumura, H. (2014). Book recommendation based on library loan records and bibliographic information. *Procedia-social and behavioral sciences*, *147*, 478–486.

Vaz, P. C., Martins de Matos, D., y Martins, B. (2012). Stylometric relevance-feedback towards a hybrid book recommendation algorithm. En *Proceedings of the fifth acm workshop on research advances in large digital book repositories and complementary media* (pp. 13–16).

Vaz, P. C., Martins de Matos, D., Martins, B., y Calado, P. (2012). Improving a hybrid literary book recommendation system through author ranking. En *Proceedings of the 12th acm/ieee-cs joint conference on digital libraries* (pp. 387–388).

Vaz, P. C., Ribeiro, R., y de Matos, D. M. (2013a). Book recommender prototype based on author's writing style. En *Proceedings of the 10th conference on open research areas in information retrieval* (pp. 227–228).

Vaz, P. C., Ribeiro, R., y de Matos, D. M. (2013b). Understanding temporal dynamics of ratings in the book recommendation scenario. En *Proceedings of the 2013 international conference on information systems and design of communication* (pp. 11–15). New York, NY, USA: ACM. Descargado de `http://doi.acm.org/10.1145/2503859.2503862` doi: 10.1145/2503859.2503862

Weisstein, E. W. (2013a). *Geometric distribution. From MathWorld–A Wolfram Web Resource.* `http://mathworld.wolfram.com/GeometricDistribution.html`. (Recuperado 2018-12-24)

Weisstein, E. W. (2013b). *Hypergeometric distribution. From MathWorld–A Wolfram Web Resource.* http://mathworld.wolfram.com/HypergeometricDistribution.html. (Recuperado 2018-12-24)

What is big data? – bringing big data to the enterprise. (2018). https://www.ibm.com. (Recuperado 2013-08-26)

Yang, X., Zeng, H., y Huang, Y. (2009, May). Artmap-based data mining approach and its application to library book recommendation. En *2009 international symposium on intelligent ubiquitous computing and education* (p. 26-29). doi: 10.1109/IUCE.2009.43

Zamani, H., Moradi, P., y Shakery, A. (2015). Adaptive user engagement evaluation via multi-task learning. En *Proceedings of the 38th international acm sigir conference on research and development in information retrieval* (pp. 1011–1014). New York, NY, USA: ACM. Descargado de http://doi.acm.org/10.1145/2766462.2767785 doi: 10.1145/2766462.2767785

Zeng, W., Zeng, A., Shang, M.-S., y Zhang, Y.-C. (2013, 11). Information filtering in sparse online systems: Recommendation via semi-local diffusion. *PLOS ONE*, *8*(11), 1-9. Descargado de https://doi.org/10.1371/journal.pone.0079354 doi: 10.1371/journal.pone.0079354

Zhang, H., Chow, T. W., y Wu, Q. J. (2016). Organizing books and authors by multilayer som. *IEEE transactions on neural networks and learning systems*, *27*(12), 2537–2550.

Zhou, M. (2010). Book recommendation based on web social network. En *Artificial intelligence and education (icaie), 2010 international conference on* (pp. 136–139).

Zhu, Z., y Wang, J.-Y. (2007). Book recommendation service by improved association rule mining algorithm. En *Machine learning and cybernetics, 2007*

international conference on (Vol. 7, pp. 3864–3869).

Ziegler, C.-N., McNee, S. M., Konstan, J. A., y Lausen, G. (2005). Improving recommendation lists through topic diversification. En *Proceedings of the 14th international conference on world wide web* (pp. 22–32). New York, NY, USA: ACM. Descargado de http://doi.acm.org/10.1145/1060745.1060754 doi: 10.1145/1060745.1060754

ANEXO

Comentario Sobre Resultados de Selección Aleatoria

Por cómo preparamos el dataset, tenemos en el set de testing 20 ítems relevantes para cada usuario. Sea A el número total de ítems que usaremos para el experimento. Teóricamente, la probabilidad de sacar aleatoriamente un ítem relevante, para un usuario, es $\frac{20}{A}$. Asumiendo distribución uniforme, y fijando $A = 67,261$ como el número total de ítems en nuestra base de datos[2], en promedio para todos los usuarios el primer ítem relevante lo sacaríamos en el intento 3,363. El valor experimentalmente obtenido, como visto en la sección 6.1, es de 826. Las razones de la diferencia con el valor experimental pueden ser varias.

En este cálculo no aplicamos la corrección por múltiples versiones de los libros detallada en la sección 6.2. Para poder hacerlo, debiéramos preguntarnos en cada selección aleatoria si es que el libro seleccionado es una versión distinta de algún otro libro seleccionado anteriormente o si es que es un libro completamente nuevo. Para saberlo, sin conocer de antemano qué ítem se seleccionó, habría que formular una distribución que a partir del -por ejemplo- ID de un libro 'canónico' (como definido en la sección 6.2), dé las ocurrencias de sus versiones, de modo que podamos aplicar el valor esperado de esta distribución a nuestro cálculo teórico.

En nuestra base de datos de 67,261 libros, la multiplicidad de libros que presenta el libro que más versiones tiene es de 17[3]. Si descontamos todas las distintas versiones de los libros y nos quedamos con uno de cada uno, el número total de ítems se vería reducido a 59,116. Esto produce un nuevo valor teórico del puesto del primer ítem relevante: 2,955. Cuando no aplicamos la corrección los resultados efectivamente bajan, como se puede ver en la tabla 8.3 del apéndice, lo que

[2] A tiene este mismo valor en el momento de evaluación de los algoritmos cuyos resultados están en la sección 6

[3] Gone Girl de Gillian Flynn: `https://www.goodreads.com/book/show/19288043`

incrementaría el valor experimental de selección del primer ítem relevante de 826 a 1,429, a $n = 10$.

Otra posible explicación es que estemos usando la distribución incorrecta para describir el problema.

Sea X_i, con $i = 1, \ldots, k, \ldots, n$, la variable aleatoria que es 1 si el $i-$ésimo ensayo es un éxito y 0 en el caso contrario, siendo n la cantidad total de ensayos. El número total de éxitos Y está dado por $Y = X_1 + \cdots + X_k + \cdots + X_n$.

Por el bien de la simplicidad, partiremos diciendo que la distribución binomial negativa es una distribución discreta sobre ensayos de Bernoulli[4], todos independientes e idénticamente distribuidos, que nos da la probabilidad de obtener una cantidad k de éxitos antes de una cantidad r de fracasos:

$$f(k; r) \equiv \Pr(X = k) = \binom{r + k - 1}{k} p^k (1 - p)^r \tag{8.1}$$

con $k = 0, 1, 2, \ldots$.

La distribución geométrica es un caso particular de la binomial negativa en donde $r = 1$, es decir, en la que tenemos la distribución de probabilidad de números de éxitos antes del primer fracaso, pudiendo ser reformulada para obtener la probabilidad del número de fracasos antes del primer éxito en el ensayo $k + 1$ (Weisstein, 2013a):

$$f(k) \equiv \Pr(Y = k) = (1 - p)^k p \tag{8.2}$$

El valor esperado de la distribución geométrica corresponde a

$$\mathrm{E}(Y) = \frac{(1 - p)}{p} \tag{8.3}$$

y nos da la cantidad promedio de fracasos antes del primer éxito.

[4]Aquel experimento aleatorio con sólo dos posibles resultados: 'fracaso' o 'éxito'

En primer lugar, el tratamiento que hicimos en el cálculo anterior fue considerando que Y distribuía geométricamente, mapeando el léxico de 'éxito' y 'fracaso' por el de 'relevante' y 'no relevante', y el de 'ensayo' por 'recuperación de ítem'. Segundo, en nuestro cálculo $p = \frac{20}{A}$, por lo que $\mathrm{E}(Y) = \frac{1-20/A}{20/A} = 3,363$. Tercero, la distribución geométrica, como subset de distribuciones binomiales, supone los casos en donde las muestras son extraídas con reemplazo de una población de tamaño N. Aun si fuera sin reemplazo, si N fuera lo suficientemente grande, la distribución geométrica sería una buena aproximación. Es por estas razones prácticas que las ecuaciones 8.1 y 8.2 no dependen de n ni de N. Sin embargo acá no consideraremos que la distribución geométrica sea una buena aproximación al problema, pues fuera $n = 5$ el tamaño mínimo de las muestras, como es el caso, el porcentaje del total sería un considerable $0.007\,\%$. Incluso así, la situación actual es que las muestras son extraídas sin reemplazo, pues de lo contrario un mismo ítem podría aparecer más de una vez en una lista de recomendación, si tomamos en cuenta que cada ítem en nuestra base de datos es único por cuanto cada uno tiene un ID distinto del resto.

El modelamiento debiese ser distinto, y es por eso que para este fin consideraremos la distribución hipergeométrica (HG) (Weisstein, 2013b), puesto que entre las selecciones las probabilidades son distintas y los ensayos son dependientes, ya que por cada selección la población va disminuyendo, cosa que no abordan las anteriores distribuciones. Ésta da la probabilidad de obtener k éxitos en n selecciones sin reemplazo de una población finita de tamaño N, que contiene exactamente K objetos con la característica por la cual cuya selección consideramos exitosa:

$$f(k; N, K, n) \equiv \Pr(Y = k) = \frac{\binom{K}{k}\binom{N-K}{n-k}}{\binom{N}{n}} \tag{8.4}$$

para $k = 0, 1, 2, \ldots, K$, con valor esperado de

$$\mathrm{E}(Y) = n\frac{K}{N} \tag{8.5}$$

lo cual denota la cantidad promedio de éxitos si el tamaño de muestra es n, dados K y N. Si en vez de extraer una cantidad fija de muestras n fuésemos extrayendo indefinidamente ítems hasta alcanzar una cantidad m de éxitos, la cantidad k de fracasos nos la da la distribución hipergeométrica negativa (NHG) (*Negative hypergeometric distribution. Encyclopedia of Mathematics*, 2010):

$$f(k; N, M, m) \equiv \Pr(X = k) = \frac{\binom{k+m-1}{k}\binom{N-m-k}{M-m}}{\binom{N}{K}} \tag{8.6}$$

donde M se define de la misma forma que definimos el K de antes. Su valor esperado es

$$\mathrm{E}(X) = m\frac{N-M}{M+1} \tag{8.7}$$

lo cual denota la cantidad promedio de fracasos antes de alcanzar m éxitos, dados M y N.

Por aplicar estas distribuciones (HG y NHG) al actual problema, consideremos lo siguiente:

- Aplicaremos la corrección por libros con múltiples ediciones de forma teórica como se hizo antes, esto es, reducir A de 67,261 a 59,116.
- La cantidad de ítems relevantes para cada usuario es exactamente 20, por razones del protocolo de evaluación definido en la sección 5.2.
- Para ambas distribuciones HG y NHG: $N = A$.
- Para la distribución HG: $K = 20$, $n = \{5, 10, 15, 20\}$ el tamaño de las listas de recomendación, y k, que fijaremos en 1, la variable aleatoria que distribuye hipergeométricamente, que representa la cantidad observada de ítems relevantes en un grupo de n ítems, dados K y N.
- Para la distribución NHG: $M = 20$, y k la variable aleatoria que distribuye negativa-hipergeométricamente, que representa la cantidad observada de ítems no relevantes antes de observar m ítems relevantes, que fijaremos en 1, dados M y N.

Cuadro 8.1. Cálculos de probabilidad con distribución HG, con parámetros $k = 1$, $K = 20$, $N = A$ y distintos n. Valores esperados sobre todos los k a distintos n.

n	HG(Y)	$E_{HG}(Y)$
5	.00169	.00169
10	.00337	.00338
15	.00505	.00507
20	.00672	.00677

Cuadro 8.2. Cálculos de probabilidad con distribución NHG, con parámetros $m = 1$, $M = 20$, $N = A$ y distintos k, ítems no relevantes antes de encontrar m relevantes. El valor esperado es uno solo por ser independiente de k.

k	NHG(Y)	$E_{NHG}(Y)$
50	.00033	2814.1
100	.00032	
500	.00029	
1000	.00024	

En las tablas 8.1 y 8.2 se muestran los cálculos que podemos obtener de estas distribuciones.

En la tabla 8.1 se muestra el resultado de la función 8.4 al ser evaluada con los parámetros dados y a distintos tamaños de lista. A mayor tamaño de muestra, vemos que lógicamente la probabilidad de obtener al menos 1 ítem relevante es cada vez más grande, al igual que la cantidad promedio de ítems relevantes encontrados por muestra, llegando a un valor de 0.00677 ítems por muestra en promedio. Si lo aproximamos a 0.007, podemos decir que se esperan encontrar 7 relevantes cada 100 ítems en una selección aleatoria. En cambio, si vamos indefinidamente seleccionando ítems hasta encontrar $m = 1$ ítems relevantes antes de $k = \{50, 100, 500, 1000\}$ no relevantes, el comportamiento es como se muestra en la tabla 8.2, en donde es cada vez menos probable encontrar 1 ítem relevante luego de una cantidad cada vez más grande de no relevantes. Esto es porque cada vez que incrementemos k en una unidad se agrega un nuevo factor $\frac{N-M-i}{N-i}$ al cálculo de la probabilidad total. Si la probabilidad de encontrar un ítem no

relevante es $\frac{N-M}{N}$, la probabilidad de encontrar consecutivamente k ítems no relevantes es

$$\frac{N-M}{N} \cdot \frac{N-M-1}{N-1} \cdots \frac{N-M-(k-1)}{N-(k-1)} = \frac{\binom{N-M}{N-M-k}}{\binom{N}{N-k}}$$

y dado que $N\ (= A)$ es muy cercano a $N - M\ (= A - 20)$, el descenso de esta probabilidad es lenta. Por su parte, las probabilidades son bastante bajas. Si k distribuye negativa-hipergeométricamente, hay una probabilidad aproximada de 0.0003 de encontrar un solo ítem relevante luego de 50 o incluso 500 no relevantes.

Si vamos a la ecuación 8.5 y despejamos para n, forzando $\mathrm{E} = 1$, se tiene que

$$n = \frac{N}{K} \tag{8.8}$$

lo que nos da el n necesario para obtener en promedio 1 ítem relevante. Si $N = A$ y $K = 20$:

$$n = \frac{A}{20}$$
$$n = 2955$$

Por otro lado, en la tabla 8.2 se da el promedio de ítems no relevantes antes de que aparezca el primer relevante. Cabe notar que estos valores provienen de definiciones distintas. $n = 2,955$ es el tamaño de la muestra necesario para que esperemos en ella al menos 1 ítem relevante, en cualquier puesto. $\mathrm{E}_{\mathrm{NHG}} = 2814$ es el número promedio de ítems no relevantes que se seleccionarán antes del 1 ítem relevante, el cual coincide con la definición de MRR. El que aún sea distinto del 826 obtenido experimentalmente puede deberse a que las repeticiones del mismo experimento sean insuficientes para alcanzar el valor teórico (considerando que estos valores teóricos se definen con un número de experimentos tendiendo a infinito). Sin embargo, por simplicidad, dejaremos los resultados de

este experimento como línea de base, aun cuando el experimento se haya hecho una sola vez.

Por último, de las ecuaciones 8.4, 8.5, 8.6 y 8.7 podemos entrever maneras alternativas de mirar el proceso de optimización de los parámetros de los sistemas recomendadores, donde la meta es obtener parámetros que generen la mayor cantidad de recomendaciones relevantes posible en un tamaño limitado n de recomendaciones en total por usuario, de un total de ítems para recomendar de N. En cuanto a la distribución NHG, si fijamos M, el objetivo sería maximizar m, a su vez minimizando k y $\mathrm{E}_{\mathrm{NHG}}(Y)$, dado que queremos esperar la menor cantidad posible de ítems no relevantes antes de encontrar algún relevante. Además, la probabilidad 8.4 debiese ser alta si $m > k$, y baja en el caso contrario. En cuanto a la distribución HG, se quiere que $\mathrm{E}_{\mathrm{HG}}(Y)$ sea lo mayor posible, ya que queremos el mayor número de ítems relevantes de los n recomendados, al igual que k, por la misma razón. Esto tiene sentido en cuanto sepamos el M (o K) de cada usuario, como es el caso dado el protocolo de evaluación que seguimos.

Plantilla de Encuesta a Usuarios

Lo siguiente es la plantilla de los correos enviados a los usuarios del estudio online. Presentamos con ella algunos ejemplos de ítems recomendados:

Hola X,

Gracias por participar en este experimento. Responder la siguiente encuesta te tomará entre 15 a 20 minutos. Tu opinión nos será muy valiosa en el desarrollo de este estudio, así que esperamos que te tomes un momento para completarla.

A continuación te presentamos 3 listas de 10 recomendaciones cada una, hechas a tu medida, producidas por distintos sistemas.

Necesitamos que evalúes cada recomendación usando 2 criterios, cada uno en una escala de 3 puntos:
1. Relevancia:
- RELEVANTE: libro relevante a tus intereses, algo que te interesaría leer.
- ALGO RELEVANTE: libro medianamente relevante, algo que podrías considerar leer en el futuro pero que no estás muy interesado.
- NO RELEVANTE: libro no relevante a tus intereses, algo que no estarías dispuesto a leer.

2. Novedad:
- NOVEDOSO: libro cuya existencia no sabías, o que no hubieras encontrado fácilmente.
- ALGO NOVEDOSO: libro que no conocías pero podrías haberlo encontrado fácilmente.
- NO NOVEDOSO: libro en el que estabas consciente de su existencia.

Al final de cada lista te pedimos que nos des tu apreciación sobre la misma según tu satisfacción en una escala de 2 puntos:
- SATISFECHO: en general estás conforme con las recomendaciones, utilizarías nuevamente el sistema para buscar libros.
- NO SATISFECHO: en general no estás conforme con las recomendaciones, no volverías a utilizar el sistema.

Finalmente te pediremos que nos digas cuál recomendador produjo la lista que más te gustó.

Para marcar las respuestas, sólo pon una X al frente de cada alternativa. Por ejemplo para una recomendación relevante y no novedosa:

Título: 'Título de Ejemplo'
Autor: Autor de Ejemplo
URL: http://www.example.com
- X RELEVANTE / ALGO RELEVANTE / NO RELEVANTE
- NOVEDOSO / ALGO NOVEDOSO / X NO NOVEDOSO

En cada recomendación te dejamos un enlace a Goodreads para que obtengas más información sobre el libro.

Para enviar tu encuesta, simplemente responde este mail con tus respuestas.
Te agradeceremos si pudieses enviarnos tus respuestas lo más pronto que puedas, ojalá durante la semana.

[RECOMENDACIONES]

RECOMENDADOR A

1
Título: Lagoon
Autor: Nnedi Okorafor
URL: https://www.goodreads.com/book/show/18753656-lagoon
- RELEVANTE / ALGO RELEVANTE / IRRELEVANTE
- NOVEDOSO / ALGO NOVEDOSO / NO NOVEDOSO

...

10
Título: Batman: A Lonely Place of Dying
Autor: Marv Wolfman
URL: https://www.goodreads.com/book/show/591355.Batman
- RELEVANTE / ALGO RELEVANTE / IRRELEVANTE
- NOVEDOSO / ALGO NOVEDOSO / NO NOVEDOSO

Respecto al recomendador A:
- SATISFECHO / NO SATISFECHO

RECOMENDADOR B

1

Título: NOS4A2

Autor: Joe Hill

URL: https://www.goodreads.com/book/show/15729539-nos4a2

- RELEVANTE / ALGO RELEVANTE / IRRELEVANTE

- NOVEDOSO / ALGO NOVEDOSO / NO NOVEDOSO

...

10

Título: What Was She Thinking? [Notes on a Scandal]

Autor: Zoë Heller

URL: https://www.goodreads.com/book/show/13258.What_Was_She_Thinking_Notes_on_a_Scandal_

- RELEVANTE / ALGO RELEVANTE / IRRELEVANTE

- NOVEDOSO / ALGO NOVEDOSO / NO NOVEDOSO

Respecto al recomendador B:

- SATISFECHO / NO SATISFECHO

RECOMENDADOR C

1

Título: The Duel

Autor: Anton Chekhov

URL: https://www.goodreads.com/book/show/312862.The_Duel

- RELEVANTE / ALGO RELEVANTE / IRRELEVANTE

- NOVEDOSO / ALGO NOVEDOSO / NO NOVEDOSO

...

10

Título: The Weed That Strings the Hangman's Bag

Autor: Alan Bradley

URL: https://www.goodreads.com/book/show/6777616-the-weed-that-strings-the-hangman-s-bag

- RELEVANTE / ALGO RELEVANTE / IRRELEVANTE

- NOVEDOSO / ALGO NOVEDOSO / NO NOVEDOSO

Respecto al recomendador C:

- SATISFECHO / NO SATISFECHO

La lista que más me gustó es la del recomendador:
A / B / C

Describe con tus propias palabras por qué te gustó más ese recomendador:

Gracias nuevamente por tu ayuda.
Saludos cordiales,

Jorge Schellman Sepúlveda
Estudiante de Magíster en Cs. de la Ingeniería, área Cs. de la Computación
Pontificia Universidad Católica de Chile

Otros Resultados

Cuadro 8.3. Método random sin corrección por libros con múltiples ediciones.

n	nDCG	MAP	MRR	R-prec
5	.00088	.00120	.00045	.00045
10	.00156	.00120	.00070	.00045
15	.00156	.00120	.00070	.00030
20	.00156	.00120	.00070	.00023

Cuadro 8.4. TF-IDF+ m.1 con la corrección, pero con los parámetros antiguos (aquellos que se ven en la columna 'modo 1 (online)' de la tabla 8.7.

n	nDCG	MAP	MRR	R-prec
5	.14663	.12363	.01172	.05896
10	.18854	.11888	.13470	.05465
15	.21454	.11888	.14256	.05185
20	.22988	.11888	.14600	.04921

Cuadro 8.5. TF-IDF+ m.1. Resultados del ajuste de parámetros.

Páram.	Valor	nDCG	Páram.	Valor	nDCG	Páram.	Valor	nDCG
fl	descripción	.1072	mindf	1	.1973	maxwl	0	.2035
	autores	.0287		2	.2022		2	.0
	géneros	.0118		3	.2006		4	.0822
	título	.0104		4	.1963		6	.1656
	citas	.0103		5	.1927		8	.1961
	descr+autor	.1084		6	.1896		10	.2029
	descr+autor+título	.1093		7	.1866		12	.2040
	descr+autor+título+genr+citas	.1174		8	.1846		14	.2036
boost	false	.1174		9	.1822		16	.2034
	true	.1078		10	.1789		18	.2034
mintf	1	.1927		11	.1775		20	.2032
	2	.1927		12	.1759		22	.2033
	3	.0718		13	.1748		24	.2033
	4	.0599		14	.1727	maxqt	1	.0597
	5	.0513		15	.1721		10	.1636
	6	.0437		16	.1692		20	.1925
	7	.0398		17	.1706		30	.2102
	8	.0329		18	.1692		40	.2080
	9	.0283		19	.1676		50	.2058
	10	.0259		20	.1654		60	.2101
	11	.0218	minwl	1	.2023		70	.2121
	12	.0180		2	.2029		80	.2120
	13	.0125		3	.2013		90	.2117
	14	.0111		4	.2035		100	.2115
	15	.0102		5	.1952	maxntp	500	.2121
	16	.0079		6	.1793		1000	.2121
	17	.0061		7	.1533		5000	.2121
	18	.0059		8	.1166		100000	.2121
	19	.0032		9	.0867		150000	.2121
	20	.0028		10	.0599	maxdf	13,238	.2027
							23,166	.2032
							33,095	.2035
							49,642	.2035
							59,571	.2035

Cuadro 8.6. TF-IDF+ m.2. Resultados del ajuste de parámetros.

Páram.	Valor	nDCG	Páram.	Valor	nDCG	Páram.	Valor	nDCG
fl	descripción	.0640	mindf	1	.0687	maxwl	0	.0840
	autores	.0455		2	.0684		2	.0
	géneros	.0246		3	.0675		4	.0519
	título	.0343		4	.0666		6	.0818
	citas	.0120		5	.0637		8	.0930
boost	false	.0642		6	.0620		10	.0949
	true	.0346		7	.0613		12	.0896
mintf	1	.0642		8	.0605		14	.0901
	2	.0639		9	.0579		16	.0901
	3	.0637		10	.0551		18	.0840
	4	.0637		11	.0549		20	.0840
	5	.0615		12	.0530		22	.0840
	6	.0566		13	.0508		24	.0840
	7	.0483		14	.0498	maxqt	1	.0110
	8	.0427		15	.0494		10	.0817
	9	.0403		16	.0487		20	.0921
	10	.0337		17	.0483		30	.0934
	11	.0328		18	.0462		40	.0102
	12	.0296		19	.0531		50	.1048
	13	.0280		20	.0523		60	.1044
	14	.0257	minwl	1	.0689		70	.1041
	15	.0250		2	.0832		80	.1041
	16	.0235		3	.0840		90	.1045
	17	.0222		4	.0786		100	.1070
	18	.0213		5	.0774	maxntp	500	.0753
	19	.0199		6	.0745		1000	.0980
	20	.0182		7	.0578		5000	.1070
maxdf	13,238	.0836		8	.0517		100000	.1070
	23,166	.0840		9	.0387		150000	.1070
	33,095	.0840		10	.0343			
	49,642	.0840						
	59,571	.0840						

Cuadro 8.7. Descripción de los parámetros usados en el método TF-IDF+. Antes, la cantidad total de documentos era menor, por eso el maxdf en el caso online es menor al maxdf en el modo 1 offline, ya que como explicado anteriormente, los parámetros usados para evaluar el algoritmo TF-IDF+ con usuarios acarrearon sus valores antes de la corrección por libros con múltiples ediciones, que fue cuando contábamos con menos libros en nuestro índice.

Parámetro	Descripción	modo 1 (offline)	modo 2 (offline)	modo 1 (online)
maxwl	Largo máximo de palabra sobre lo cual las palabras serán ignoradas.	12	10	8
minwl	Largo mínimo de palabra bajo lo cual las palabras serán ignoradas.	4	3	1
maxdf	La frecuencia por la cual las palabras serán ignoradas si aparecen en más de esta cantidad de documentos.	33095 (#docs $\times$ 0.5)	59909 (#docs $\times$ 0.75)	25431 (#docs $\times$ 0.5)
mindf	La frecuencia por la cual las palabras serán ignoradas si aparecen en menos de esta cantidad de documentos.	2	0	2
maxqt	Número máximo de términos de consulta que serán incluidos en la consulta generada.	70	100	90
boost	Especifica si la consulta será impulsada (*boost*) por relevancia de términos interesantes (*feature* de Solr).	false	false	false
fl	Los campos usados para similaridad.	descripción, título, géneros, autores, citas	description	descripción, título, géneros, autores, citas
mintf	La frecuencia por la cual los términos serán ignorados si ocurren bajo esta cantidad en el documento fuente	1	1	1
maxntp	Número máximo de *tokens* para analizar en cada campo del documento que no está almacenado con el soporte `TermVector` de Solr.	150000	150000	150000

Cuadro 8.8. Parámetros de los métodos híbridos. Hyb-1: TF-IDF+ m.1/SVD implícito. Hyb-2: TF-IDF+ m.1/FM con autores.

RS híbrido	α	β
Hyb-1	0.9	0.1
Hyb-2	0.9	0.1