ISSN 1688-2784

Universidad de la República
Facultad de Ingeniería

CorrG-RS: Sistemas de Recomendación basados en Redes Neuronales sobre Grafos de Correlación

Tesis presentada a la Facultad de Ingeniería de la Universidad de la República por

Andrés Gomez Caram

en cumplimiento parcial de los requerimientos para la obtención del título de Magíster en Ingeniería Eléctrica.

Directores de Tesis

Dr. Ing. Federico Larroca ..... Universidad de la República
Dr. Ing. Germán Capdehourat Longres .. Universidad de la República

Tribunal

Dr. Ing. Lorena Etcheverry ..... Universidad de la República
Dr. María Inés Fariello..... Universidad de la República
Dr. Ing. Federico Lecumberry ..... Universidad de la República

Director Académico

Dr. Ing. Germán Capdehourat Longres .. Universidad de la República

Montevideo
miércoles 26 octubre, 2022

CorrG-RS: Sistemas de Recomendación basados en Redes Neuronales sobre Grafos de Correlación, Andrés Gomez Caram.

ISSN 1688-2784

Esta tesis fue preparada en LaTeX usando la clase iietesis (v1.1).
Contiene un total de 96 páginas.
Compilada el miércoles 26 octubre, 2022.
http://iie.fing.edu.uy/

Agradecimientos

A la Facultad de Ingeniería de la Universidad de la República, por darnos la posibilidad de acceder a educación de calidad. A Plan Ceibal por permitirme trabajar con datos reales, valiosos y de algo que amo y en lo que creo, como son los libros. A los doctores Federico y Germán por sus siempre valiosos comentarios y la confianza depositada en mí. A Cogniflow AI, a Marce y a Walde, por hacerme aprender tanto y darme la oportunidad de poder dedicarle tiempo a la tesis (y por esa GPU salvadora). A mis familiares y amigos por el constante apoyo recibido.

Esta página ha sido intencionalmente dejada en blanco.

Para G.E.M.A (Artigas, 1926 - Annecy, 2022)

Esta página ha sido intencionalmente dejada en blanco.

Resumen

Los Sistemas de Recomendación (RS por sus siglas en inglés) están cada vez más presentes en la vida diaria de las personas. Redes sociales, plataformas de *e-commerce* o de *streaming* son solo algunas de las organizaciones que dependen de estos sistemas para recomendar contenido y productos a sus usuarios. De esta manera pueden mejorar la experiencia en línea del usuario así como también aumentar sus utilidades o inducir a los usuarios a generar y consumir más contenido mediante la captación de su atención. Al igual que en otras áreas del Aprendizaje Automático (i.e. Machine Learning, ML), como el Procesamiento de Lenguaje Natural o la Visión Artificial, los RS se vieron revolucionados por las técnicas de Aprendizaje Profundo (i.e. Deep Learning, DL), pasando de basarse en métodos clásicos como la factorización de matrices a modelos basados en redes neuronales profundas. El ML para datos en grafos también se vio sacudido por el DL, desarrollándose así las Redes Neuronales sobre Grafos (GNN por sus siglas en inglés). Estas redes se basan en los mismos principios del DL pero sus arquitecturas se deben adaptar a la forma en que se representan los datos en un grafo. El objetivo principal de este trabajo es explorar la mayor capacidad de las GNN a la hora de implementar RS para datos en grafos. Para ello se trabajó con datos de la Biblioteca Ceibal, una biblioteca digital pública para beneficiarios de Ceibal y la población en general. En particular se utilizaron los datos de los usuarios adultos con el fin de implementar RS basados en GNN sobre grafos de correlación, denominado aquí como el modelo CorrG-RS. Se formuló el problema como uno de clasificación binaria para decidir si a un usuario le gustará determinado ítem o no y a partir de ello realizar recomendaciones. Esta forma de modelar el problema resulta de interés por lo novedoso respecto a la literatura consultada y a la mayor variedad de métricas disponibles a la hora de evaluar los RS. Con ese fin se implementaron las aquí llamadas métricas tradicionales y métricas alternativas. Las primeras capaces de evaluar aspectos más bien objetivos de los resultados mientras que las segundas hacen lo propio en rasgos más bien subjetivos. Luego de entrenados, estos modelos fueron comparados con otros basados en métodos clásicos y en GNN sobre grafos de conocimiento. Se encontró que CorrG-RS es capaz de competir con ambas clases de métodos, teniendo como ventaja un buen desempeño computacional comparado con su par basado en grafos de conocimiento. También este modelo demostró las mejoras que puede introducir la fácil incorporación de atributos de los ítems a los entrenamientos, sin necesidad de llevar a cabo grandes procesamientos manuales de estos. Ambas características de CorrG-RS son prometedoras de cara a un futuro donde cada vez se tengan más fuentes de datos y la necesidad de integrarlos en un mismo sistema de recomendación.

Esta página ha sido intencionalmente dejada en blanco.

Tabla de contenidos

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 1

Introducción

Los Sistemas de Recomendación (RS por sus siglas en inglés) se han transformado en pieza clave de dominios tales como la web y las plataformas digitales debido a la abrumadora cantidad de contenido disponible. Su rol principal es el de facilitar al usuario el acceso a la información que necesita, desea o que considera más útil, ya sean libros, noticias, películas, servicios turísticos o bienes raíces (i.e. *ítems* en el marco de los RS). El principal interés del proveedor de la recomendación es aumentar la utilidad [28, pp. 3], ya sea en concepto monetario (e.g. una plataforma de *e-commerce*) o no-monetario, donde se desea llegar al usuario aunque no se persigan fines de lucro, como es, por ejemplo el caso de una biblioteca digital pública. Además de aumentar las utilidades, también existen otros aspectos relevantes que los RS pueden solucionar. Mejorar la experiencia de usuario [28, pp. 4] es una de las dimensiones en donde los RS son capaces de aportar: que a los usuarios se les recomiende ítems que consideren relevantes y satisfagan sus necesidades y deseos; que la web de turismo le recomiende el hotel que quería, pero desconocía, o que luego de la compra de un libro se le sugieran otros que podrían ser de su interés.

Los RS son clave además para conquistar la *atención* del usuario. Este es el caso de las redes sociales, donde se suele sugerir contenido mediante motores de recomendación que aprenden en base al historial del usuario, el tipo de contenido que consume y el comportamiento en línea de otros usuarios con los que interactúa. Captar la atención de los usuarios es clave en estas plataformas, no solo para vender publicidad, sino también para realimentar sus propios RS y a su vez generar más datos. La atención del usuario, consumidor y productor a la vez, dispara la generación de más contenido. Por lo tanto no es de extrañar que la privacidad sea uno de los asuntos delicados que suelen aparecer en el área de la web y las plataformas digitales, y los RS no están exentos de cuestionamientos [28, pp. 25]. ¿Qué datos usa el RS? ¿Cómo los usa? ¿Cómo y dónde los obtiene? ¿Cuánto sabe el usuario del uso de sus datos? ¿La web utiliza datos de terceros? ¿Recomienda ítems inapropiados? También se deben tener en cuenta sesgos de género, etnia, clase, entre otros. La presencia de RS y motores de recomendación ya es tan ubicua, y puede llegar a ser tan delicada, que algunos países han considerado regularlos. Tal es el caso de China, que mediante la Cyberspace Administration of China (CAC) ha hecho público un primer borrador [5] de las regulaciones.

¿Pero qué es un RS? Se trata de un sistema capaz de recomendar *ítems* a *usuarios* en base a las características de estos y de su historial de *interacciones* [28, pp. 1]. Podría tratarse de un RS que recomiende ítems de forma aleatoria, por popularidad (i.e. qué tan consumido es un ítem) o según características conocidas de un usuario (e.g. su edad), entre otras. En los casos anteriores se tienen RS de sencilla implementación,

pero rígidos y sin inteligencia alguna. Por lo tanto se desea que los RS sean modelos que hayan *aprendido* de los datos (i.e. usuarios, ítems e interacciones) y en función de ello realizar *recomendaciones* (i.e. recomendar ítems a los usuarios) de interés. Es por ello que en las últimas décadas han surgido técnicas de aprendizaje automático específicas para resolver el problema de hacer recomendaciones y, como era de esperar, el Deep Learning (DL) también impactó de forma significativa en esta área.

En los últimos diez años los avances logrados en el campo del Deep Learning (DL) han revolucionado tareas vinculadas al Aprendizaje Automático y el Procesamiento de Señales. Estos avances se dan en distintos dominios, ya sea lenguaje natural, imagen, audio o vídeo, entre otros. Según el tipo de datos y el problema a resolver, distintas arquitecturas de redes neuronales profundas se han destacado sobre el resto. A modo de ejemplo, las Convolutional Neural Networks (CNN) [23] y sus variantes han sobresalido en el mundo de la Visión Artificial y el audio, mientras que las Recurrent Neural Networks [3] (y sus variantes) han hecho lo propio en el campo del Procesamiento del Lenguaje Natural (PLN). Estas distintas arquitecturas son propicias cuando se tiene a disposición lo que se suelen llamar datos *euclideanos* (i.e. que se encuentran en espacios euclideanos), como vectores (e.g. audio), matrices (e.g. una imagen de un solo canal) o tensores (e.g. una imagen de varios canales). La representación de datos en forma de grafos introduce un nuevo desafío a la hora de aplicar técnicas de DL y Machine Learning (ML) en general: son tipo de datos *no-euclideanos* (i.e. también llamados *geométricos*) [29]. La topología de datos representados mediante grafos requiere la existencia de nuevas arquitecturas de redes neuronales. Estas redes deben ser capaces de aprender de los datos utilizando los mismos principios generales y estrategias de entrenamiento del DL, pero adaptándose a los grafos. Esto provocó el surgimiento de las Graph Neural Networks (GNN) [32, pp. 47]. En el presente trabajo fue de interés saber cómo, al ser los grafos capaces de representar relaciones más abstractas entre los datos, a partir de éstos se podrían implementar mejores RS.

Los sistemas de recomendación clásicos (o no-profundos) se basan principalmente en técnicas de factorización de matrices o en modelos de aprendizaje automático. Estos modelos pueden ser clasificadores (e.g. cuando se quiere predecir si a un usuario le gustará determinado ítem) o regresores (e.g. se desea predecir la calificación que el usuario dará a un ítem). Como datos se tendrá (como mínimo) una matriz de interacciones con información del consumo que los usuarios hicieron de los ítems. En la Tabla 1.1 se puede ver un ejemplo con calificaciones de 1 a 5 dadas por los usuarios $[U_1, U_6]$ a seis películas. En esta matriz se tienen en las filas los usuarios, en las columnas los ítems y en las celdas las interacciones. Como es de esperar, los usuarios suelen haber interactuado con un número limitado de ítems, por lo que esta matriz tendrá muchas celdas vacías. Por lo tanto el problema a resolver será el de compleción de esta matriz (i.e. estimar los valores de las celdas vacías). De esta manera se predice si a un usuario le gustará o no un ítem con el que aún no ha interactuado en base a qué ítems consumieron otros usuarios. Este problema básico se puede solucionar con RS basados en técnicas de factorización de matrices. También, como es cada vez más usual, se tiene además información sobre tanto ítems como usuarios. Esta información extra pueden ser atributos de los ítems (e.g. en el caso de películas, quién es el director, los protagonistas, el país de origen, etc) o información de los usuarios (e.g. demográfica, geográfica, etc). Este tipo de información adicional puede ser mejor aprovechada por RS basados en modelos [28, pp. 71], ya sea usando los atributos extra de los ítems o los usuarios, o ambos.

Estas relaciones entre usuarios e ítems también pueden ser representadas en un grafo. Podría tratarse de un grafo homogéneo donde los nodos son ítems y las aristas las correlaciones entre los ítems calculadas en base a las calificaciones que los usuarios dieron a estos ítems. Podría ensamblarse un grafo heterogéneo y bipartito como el

	Casino	Platoon	Heat	Goodfellas	Scarface	Spartacus
U1	1			5		2
U2		5			4	
U3	5	3		1		
U4			3			4
U5				3	5	
U6	5		4			

Tabla 1.1: Matriz de interacciones con calificaciones de 1 a 5 dadas por los usuarios $[U_1, U_6]$.

de la Figura 1.1, donde hay nodos películas y nodos usuarios y las aristas son las calificaciones. Además, en un grafo heterogéneo se podrían incorporar atributos a nivel de ítems y usuarios de forma sencilla, donde se tendrían, además de los nodos ítems y usuarios, nodos correspondientes a atributos y en consecuencia nuevas aristas que indiquen las relaciones entre estos. A este tipo de grafo se lo conoce como grafo de conocimiento, o Knowledge Graph (KG por sus siglas en inglés) [9]. Sobre estos grafos se pueden entrenar GNN capaces de aprender éstos con el mismo fin de generar recomendaciones a los usuarios (i.e. predecir si a un usuario le gustará o no un ítem). En este trabajo se experimentó con modelos de terceros basados en GNN y KG y se implementó un modelo basado en grafos de correlación y GNN denominado CorrG-RS a modo de comparar con métodos tradicionales para RS basados en factorización de matrices.

1.1. La Biblioteca Ceibal

Para el presente trabajo se tuvo a disposición datos de la Biblioteca Ceibal [17], una biblioteca digital de uso público, creada y administrada por Plan Ceibal [16], donde tanto beneficiarios del programa como población en general pueden pedir libros electrónicos en préstamo. Al momento de comenzar este trabajo la Biblioteca Ceibal no contaba con un RS integrado a su plataforma. Los contenidos se dividían y desplegaban según los perfiles de usuario, como los beneficiarios del plan (ya fueran de Primaria, Secundaria, Media, Docentes o Ibirapitá) y los usuarios de la Biblioteca País, un catálogo destinado a la población general. Es para este último perfil (y en particular para la audiencia adulta) de usuario que se centró el presente trabajo.

Los datos disponibles corresponden al período comprendido entre 1 de mayo de 2019 y el 31 de enero 2021. Se consideraron únicamente los recursos (i.e. libros) no educativos (i.e. que se excluyeron los libros de texto para estudio) y aquellos que no habían aún caducado (i.e. que seguían disponibles en el catálogo de la biblioteca).

Se contó con datos correspondientes a los libros de la biblioteca, también referidos como *recursos* (i.e. ítems en un RS) y datos de interacciones usuario-ítem (i.e. el *feedback* que un usuario brinda, sea ya explícito o implícito). El *feedback* explícito corresponde a los casos en que un usuario calificó de forma voluntaria un recurso con un puntaje de 0 a 5. Por su parte, el *feedback* implícito, como su nombre lo indica, es recolectado de forma involuntaria. Se trata de información que el usuario genera de forma automática y que se considera valiosa a la hora de evaluar sus preferencias futuras, es decir, poder utilizar esta información para entrenar RS capaces de recomendar ítems de interés para el usuario. Este tipo de *feedback*, más allá de lo útil que pueda ser en cuanto a la cantidad y calidad de información que aporta, es ventajoso además por ser usualmente más abundante y proveniente de diversas fuentes. No siempre los usuarios están dispuestos a calificar un ítem, pero su comportamiento en línea puede

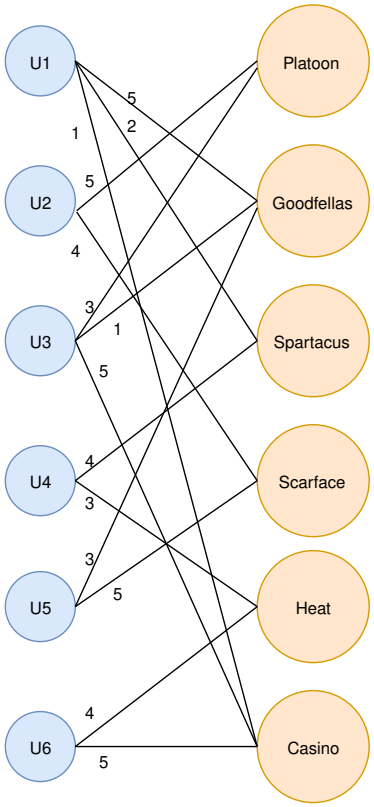

Figura 1.1: Grafo bipartito usuario-ítem asociado a la Tabla 1.1. En las aristas se indica la calificación que un usuario en $[U_1, U_6]$ le dio a cada película.

ser monitoreado y almacenado. Ese fue el caso de la Biblioteca Ceibal, por lo que se tuvo a disposición cuatro tablas generadas con interacciones usuario-ítem, siendo tres de ellas de tipo implícitas y una explícita. El *feedback* explícito corresponde a calificaciones entre 1 y 5 dadas por los usuarios a los ítems de forma voluntaria, mientras que las tres tablas de *feedback* implícito corresponden a porcentaje de avance de lectura, la cantidad de préstamos y el tiempo total de lectura.

Además de los datos correspondientes a las interacciones usuario-ítem, se contó con información de los ítems. Cada recurso tiene asignados veintiséis atributos descriptivos, de entre los cuales se consideraron algunos más relevantes (e.g. título, autor, idioma, editor, audiencia, etc). Una de las preguntas surgidas en el presente trabajo fue si la incorporación de estos atributos a los entrenamientos (i.e. RS basados en GNN) sería capaz de enriquecer el desempeño de los RS implementados. Se encontró, como se detalla más adelante, que la incorporación de estos atributos aporta a los modelos entrenados, aunque por motivos de recursos computacionales disponibles solo se pudo experimentar con grafos más pequeños.

1.2. Resumen del Trabajo

Explorados y procesados los datos, el objetivo consistió en entrenar distintos RS con los datos de la biblioteca a modo de comparar técnicas clásicas contra sus pares

basadas en datos en grafos. Para ello el primer paso consistió en revisar la bibliografía y explorar los métodos más clásicos para RS, pertenecientes al ML no-profundo, también llamado *shallow ML*. En particular se hizo centro en algoritmos desarrollados para *feedback* implícito. Con algunos de estos métodos, como Alternating Least Squares (ALS) [42] y Bayesian Personalized Ranking (BPR) [38], se entrenaron RS para la Biblioteca Ceibal, con el fin de utilizarlos como puntos de partida y referencia para futuras implementaciones basadas en redes neuronales sobre grafos. Además de entrenar los RS clásicos de base mencionados anteriormente, se exploraron distintas métricas para evaluarlos, dividiéndolas en dos grandes grupos: *tradicionales* y *alternativas*. Estas métricas evalúan distintos aspectos de un RS, como la precisión, la recuperación, la capacidad del sistema de recomendar ítems novedosos, diversos, o de qué porcentaje del catálogo de ítems es capaz de cubrir (i.e. cobertura), entre otras. En el Capítulo 2 se detallan las distintas aproximaciones a estos métodos llamados aquí clásicos, las implementaciones realizadas y las métricas exploradas con las que se los evaluó.

En el Capítulo 3 se realiza un análisis exploratorio de datos (EDA por sus siglas en inglés) para tener un conocimiento más en profundidad de la información disponible de la Biblioteca Ceibal. Además se evaluó cuál (y qué tipo) de datos de *feedback* disponibles era más conveniente utilizar, eligiéndose la tabla de porcentaje de avance de lectura luego de considerar la mayor cantidad de interacciones disponibles así como también su mayor confiabilidad a la hora de decidir si a un usuario le gustó el recurso o no.

En el Capítulo 4 se desarrolla el concepto de GNN a partir de dos ópticas distintas: desde el modelo de Neural Message Passing (NMP) y con herramientas del mundo del Graph Signal Proccessing (GSP). La primera se consideró de relevancia por su popularidad, mayor intuición y su uso en algunos de los modelos de terceros estudiados, mientras que el segundo se tuvo en cuenta por su formulación más formal, basada en convoluciones sobre señales en grafos. Estas dos formas de modelar las GNN resultan necesarias de detallar debido a que se aplicaron en los modelos implementados, tanto propios como de terceros.

Previo a implementar RS basados en GNN, se exploró el estado del arte en el área y en particular entre los modelos aplicados al mundo de los RS. Se estudió con especial profundidad el modelo KGAT [39] para luego entrenarlo con los datos de la Biblioteca Ceibal. Este modelo se basa en Knowledge Graphs y el mecanismo de atención. Finalmente se decidió explorar la implementación de un RS desde la óptica del GSP, desarrollándose el modelo CorrG-RS, basado en grafos de correlación y arquitecturas de GNN de la biblioteca Alegnn [1]. Para ambos experimentos se entrenó con los datos disponibles de la Biblioteca Ceibal, tanto de los recursos (i.e. libros) como de la información disponible de interacciones. Para ver más en detalle lo realizado, se sugiere ir al Capítulo 5.

En el Capítulo 6 se analizan los resultados obtenidos para todos los RS implementados, tanto propios como de terceros. Se comparan distintos entrenamientos de cada modelo así como distintos modelos entre sí, para finalmente en el Capítulo 7 exponer las conclusiones obtenidas y posibles trabajos a futuro.

CorrG-RS resulta de interés dado lo novedoso de su formulación. Este modelo considera el problema de implementar un RS para la Biblioteca Ceibal como uno de clasificación binaria, para todos los usuarios e ítems con interacciones y usando las arquitecturas de GNN de Alegnn, a diferencia de los trabajos revisados basados en esta biblioteca, que formulan el problema como una regresión con el fin de estimar la calificación sobre un solo ítem. Además, también a modo de contrastar con otros trabajos consultados, con CorrG-RS se experimentó con la incorporación a los entrenamientos de atributos a nivel de ítem, encontrándose que, para grafos pequeños, estos lograban algunas mejoras.

El modelo CorrG-RS desarrollado en este trabajo compite con modelos de terceros basados en GNN y KG como KGAT, obteniendo resultados similiares y superiores en algunas métricas a un menor costo computacional y tiempos de preparación de los datos y entrenamiento. Comparado con métodos clásicos como BPR o ALS, logra superar ampliamente al primero, mientras que se mantiene aún por debajo del segundo, pero alcanzando valores cercanos. Lo anterior demuestra el potencial de CorrG-RS considerando lo probado y maduro de métodos como ALS y la posibilidad de incorporar más atributos y relaciones a los grafos.

Capítulo 2

Sistemas de Recomendación

El objetivo de un RS es recomendar al usuario ítems, ya sean estos productos a la venta, libros de una biblioteca digital pública o contenido en general (e.g. fotos, vídeos, películas, *tweets*). Podría tratarse de un RS básico que recomiende ítems al azar, por popularidad o por perfil de usuario, en el caso de que estos estén agrupados según cierto criterio (e.g. edad). Pero lo que interesa es que los RS *aprendan* de forma automática a partir de los datos que se tienen de los ítems y de los usuarios, de su historial y comportamiento en línea. Estos RS son modelos (o un conjunto de) entrenados para generar recomendaciones relevantes (i.e. qué tan de interés es para un usuario un ítem). Los entrenamientos se llevan a cabo con técnicas de aprendizaje automático: a partir de un conjunto de datos se generan los sub-conjuntos de entrenamiento, validación y prueba. Según el tipo de RS a entrenar se tienen distintas formas de particionar el conjunto de datos. Es posible particionar por usuarios (i.e. dejar un conjunto de usuarios y sus interacciones para prueba) o a nivel de interacciones (i.e. usar todos los usuarios, pero particionando sus interacciones). Los distintos métodos se ven más adelante en el Capítulo 6. Luego de entrenado el RS, se evalúa con el conjunto de prueba, encontrándose a disposición numerosas métricas encargadas de medir diversos aspectos del modelo, ya que la *relevancia* de un ítem no es la única dimensión a tener en cuenta por un RS. También existen otros aspectos de importancia, como:

1. *Novedad*. Que el RS sea capaz de recomendar ítems novedosos para el usuario y así mejorar su experiencia.
2. *Diversidad*. La capacidad del RS de brindar recomendaciones con diversidad de contenidos, evitando así que solo se recomienden ítems similares.
3. *Cobertura*. Es el porcentaje del catálogo de ítems que es recomendada a todos los usuarios, en promedio. Puede ser que el RS solo sea capaz de recomendar un subconjunto pequeño de ítems, o lo contrario, según cómo se implemente y qué se desee.
4. *Serendipity*. Se trata de un concepto difícil de traducir desde el inglés, pero que significa una especie de sorpresa, o descubrimiento, gratificante; "no conocía al autor de este libro que me recomiendan, pero parece interesantísimo" [28, pp. 3].

Una vez entrenado y evaluado el RS, es posible realizar recomendaciones a los usuarios. Las recomendaciones son generadas por el modelo entrenado, el cual decide qué ítems serán relevantes para determinado usuario. Así se generan *listas de recomendación* de largo K. Según el tipo de modelo puede importar el orden o no. Estos

	Gladiator	Godfather	Ben Hur	Goodfellas	Scarface	Spartacus
U1	1			5		2
U2		5			4	
U3	5	3		1		
U4			3			4
U5				3	5	
U6	5		4			

	Gladiator	Godfather	Ben Hur	Goodfellas	Scarface	Spartacus
U1	1			1		1
U2		1			1	
U3	1	1		1		
U4			1			1
U5				1	1	
U6	1		1			

Figura 2.1: A la izquierda un ejemplo de *feedback* explícito. A la derecha, *feedback* implícito.

K ítems son, por ejemplo, los que se expondrán ordenados de arriba a abajo en una plataforma de *e-commerce*, *tweets* en la *timeline* de un usuario de Twitter o libros en una biblioteca digital.

En este capítulo se presentan de forma general algunos los métodos de RS más conocidos y extendidos, referidos como métodos clásicos. En particular se detallan métodos de filtrado colaborativo resueltos mediante técnicas de factorización de matrices. Posteriormente se describen algunas de las métricas más usadas a la hora de evaluar los RS, divididas en las aquí llamadas métricas tradicionales y métricas alternativas. Por último se exponen algunos modelos clásicos de filtrado colaborativo entrenados con los datos de la Biblioteca Ceibal y que son utilizados como modelos de base para comparar con implementaciones basadas en GNN.

2.1. RS Clásicos

En su forma más general, los RS funcionan con los siguientes datos [28, pp. 8]:

1. Las interacciones usuario-ítem, como pueden ser las calificaciones que dio un usuario (i.e. *feedback* explícito) o sus patrones de comportamiento, ya sean *clicks*, búsquedas, tiempo que se detiene en un ítem (i.e. *feedback* implícito). En la Figura 2.1 se muestran ejemplos de ambos tipos de *feedbacks*, para seis usuarios $[U_1...U_6]$ y seis películas. En el caso del *feedback* implícito, un 1 indica que el usuario interactuó con la película, mientras que en el caso de *feedback* explícito, como se indicó, los usuarios dan una calificación entre 1 y 5.
2. Atributos tanto de los usuarios como de los ítems, como los datos demográficos o geográficos de los primeros, o atributos y palabras clave en las descripciones de los últimos.

Los métodos que hacen uso principalmente de la información del primer punto son los llamados *collaborative filtering methods* (CFM), o métodos colaborativos, mientras que los del segundo punto son los *content-based filtering methods* (CBFM), o métodos basados en contenido. También existe la familia de los *knowledge-based RS* (KBRS), que se basan en especificaciones del usuario indicadas por éste de forma explícita. La familia de RS que nuclea a todas las anteriores y trata de usar lo mejor de cada mundo es la de los sistemas híbridos (HS).

2.1.1. Filtrado Colaborativo

Los filtros colaborativos se apoyan en las calificaciones hechas por múltiples usuarios para realizar las recomendaciones. Las calificaciones dadas por los usuarios a los ítems se almacenan en una matriz llamada *ratings matrix* (RM). Por lo general, estas matrices son *dispersas*: los usuarios solo han calificado a un bajo número de ítems, lo que representa un desafío a la hora de que el RS haga una recomendación de calidad, así como también a nivel de cómputo (i.e. las matrices grandes y dispersas pueden

consumir demasiados recursos computacionales, aunque para la dispersión es posible usar herramientas que las representen de forma más eficiente, como las provistas por la biblioteca SciPy [22]).

Un RS de tipo CFM hará uso de los las interacciones realizadas en el pasado para generar nuevas recomendaciones, basándose en el hecho de que existe correlación entre usuarios e ítems para estas observaciones. De modo intuitivo: si dos usuarios tienen gustos y preferencias similares, entonces es probable que si uno de ellos calificó de determinada manera a un ítem, el otro usuario también lo hará de forma similar con el mismo ítem.

Los CFM de tipo *neighborhood-based collaborative filtering* se apoyan en nociones de cercanía (vecindad) para predecir calificaciones. Existen variantes basadas en usuarios y en ítems. La primera entraría dentro del caso del ejemplo dado más arriba: si dos usuarios son similares, lo que haya calificado uno puede ser usado para recomendar al otro. La similaridad se mide con distintas funciones, usando como entradas las filas de la RM. En la segunda variedad, la de las vecindades ítem-ítem, para que el RS recomiende al usuario A un determinado ítem B, se identifica un conjunto de ítems similares a B que hayan sido calificados por A. A modo de ejemplo, si a A le gustan las películas de época y además ha calificado a cierta cantidad, B podría ser otra película del mismo género. En este caso la similaridad se mide entre las columnas de la RM. El mayor desafío para estos sistemas es que sufren en demasía la dispersión de la RM.

De entre los métodos de filtrado colaborativo se verá más en detalle los basados en técnicas de factorización de matrices para estimar la RM. A continuación se describe cómo se factoriza la RM en dos matrices: una para representar a los usuarios y otra para los ítems.

Factorización de Matrices

Estos métodos son básicamente modelos basados en *embeddings* [12] y tienen la ventaja de no necesitar de manualmente seleccionar y procesar atributos de forma previa. El RS recomendará ítems en base al historial del usuario y al historial de usuarios similares. Los vectores de *embedding* de tanto usuarios como ítems se representarán en el mismo espacio vectorial, siendo posible calcular las distancias entre uno y otro y así poder realizar recomendaciones. De esta manera se recomiendan los ítems más similares, o sea los de mayor producto interno. El problema se formula de la siguiente manera: dada la RM $\mathbf{A} \in \mathbb{R}^{m \times n}$ donde m es el número de usuarios y n el número de ítems, el modelo deberá aprender:

1. Una matriz de *embeddings* de usuarios $\mathbf{U} \in \mathbb{R}^{m \times d}$, donde la fila i corresponde al usuario U_i.
2. Una matriz de *embeddings* de ítems $\mathbf{V} \in \mathbb{R}^{n \times d}$, donde la fila j corresponde al ítem I_j.

Los *embeddings*, representaciones latentes de usuarios e ítems, se aprenden de forma tal que el producto matricial $\mathbf{UV^T}$ se aproxime lo más posible a la RM $\mathbf{A}$. Por lo tanto, la entrada (i, j) de $\mathbf{A}$ es el producto interno $\mathbf{U_i V_j^T}$. En la Figura 2.2 se muestra un ejemplo para cuatro usuarios ($m = 4$), tres ítems ($n = 3$) y representaciones vectoriales en un espacio de dimensión tres ($d = 3$). Para calcular las matrices $\mathbf{U}$ y $\mathbf{V}$ se debe elegir una función de pérdida y hallar los valores que la minimicen. A modo de ejemplo, si se eligió la siguiente función de pérdida:

$$\min_{\mathcal{U} \in \mathbb{R}^{mxd}, \mathcal{V} \in \mathbb{R}^{nxd}} \sum_{(i,j) \in obs} \omega_{i,j} (\mathbf{A}_{ij} - \mathbf{U}_i \mathbf{V}_j)^2 + \omega_0 \sum_{(i,j) \notin obs} (\mathbf{U}_i \mathbf{V}_j)^2. \tag{2.1}$$

en este caso se tiene una factorización de matrices ponderada. Esta función está descompuesta en dos términos sumados, uno que tiene en cuenta las interacciones

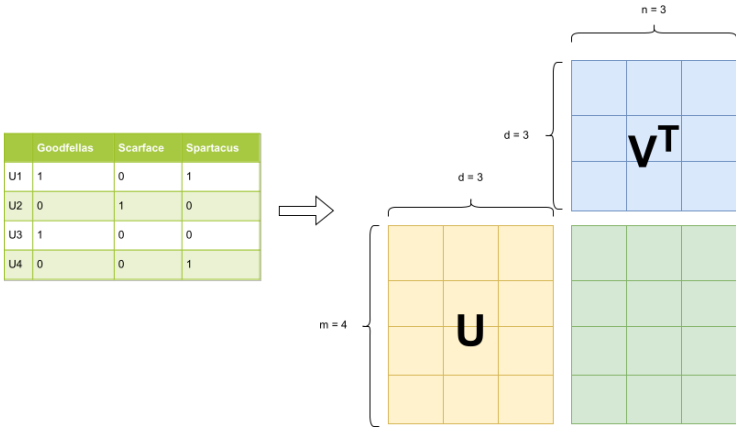

Figura 2.2: Factorización de matrices. Ejemplo para $m = 4$, $d = 3$ y $n = 3$. A la izquierda la RM, a la derecha la descomposición en las matrices **U** y **V**.

observadas en la RM (los 1) y otra para las no observadas (los 0). Cada término tiene a su vez coeficientes ω para ponderar el aporte de cada uno. De esta forma se quiere evitar que uno de los términos sea dominante. En el primer término dominarán tanto ítems como usuarios populares (altamente consumidos/consumidores) mientras que en el segundo se quiere evitar que hagan lo propio interacciones no existentes. Esto impactará más adelante a la hora de realizar recomendaciones, ya que de dominar el primer término se tendrán más bien recomendaciones populares, mientras que de ser el segundo término quien domine, las recomendaciones serán más bien poco relevantes. Luego se elige la función de pérdida a minimizar. Esta minimización de la función de pérdida se lleva a cabo mediante un método de optimización, como podría ser Stochastic Gradient Descent (SGD) o una variante más adecuada, según el caso (e.g. WALS: Weighted Alternating Least Squares [33]).

Ventajas y Desventajas del Filtrado Colaborativo

Como cualquier método a elegir, los basados en Filtrado Colaborativo tienen ventajas y desventajas [4].

Ventajas

1. No se necesita conocimiento del dominio porque los *embeddings* se generan a partir de la RM.
2. *Serendipity*. A partir de las interacciones de sus pares, un usuario puede descubrir nuevos ítems de su agrado.
3. Gracias al primer punto también se pueden considerar a estos sistemas como buenos puntos de partida, fáciles de implementar, para luego poder explorar otras vías.

Desventajas

1. Problemas al manejar nuevos ítems. Si un ítem es nuevo en el catálogo, o no fue tenido en cuenta en el entrenamiento, no se generará su *embedding* y por lo tanto no podrá ser recomendado. Este problema es usual en el mundo de los RS y se conoce como *cold-start problem*, o problema del arranque en frío. Este

problema también se presenta en caso de nuevos usuarios, de los cuales no se tendrá su historial de interacciones.

2. No incorpora atributos extra que se puedan tener de los ítems/usuarios y sean capaces de enriquecer la representación.

2.1.2. Otros RS

RS Basados en Bases de Conocimiento

Esta clase de RS son más bien usados en casos de productos y servicios que no se consumen de forma habitual y suelen ser únicos (e.g. bienes raíces), y por lo tanto no es posible contar con grandes conjuntos de datos [28, pp. 15]. Además se trata de productos donde los usuarios suelen ser extremadamente cuidadosos a la hora de elegir.

Estos productos o servicios complejos, costosos y de baja frecuencia de adquisición, generan que los usuarios sean muy detallistas en cuanto a las características deseadas. Por lo tanto, es usual que en estos RS se cuente con una interfaz gráfica para que el usuario ingrese sus preferencias de forma explícita. La similaridad se mide entre los requerimientos del usuario y la descripción del ítem. Para esta tarea son necesarias las bases de conocimiento [8].

En los RS Basados en Bases de Conocimiento también se tienen distintos sabores, como los basados en restricciones o los basados en casos. En los primeros, los usuarios especifican sus preferencias, las cuales restringirán la búsqueda, mientras que en los segundos, el problema sería algo como "quiero un producto similar a X".

El proceso de recomendación es iterativo: el usuario puede llegar a completar numerosas instancias de especificaciones para llegar al ítem deseado. Esta interactividad es lograda de diversas formas, como pueden ser los sistemas conversacionales, los basados en búsqueda o en navegación.

RS Demográficos

Estos RS incorporan información demográfica de los usuarios, como puede ser edad, género, domicilio, nivel socioeconómico, entre otras [28, pp. 19]. Esta información suele ser también combinada con información contextual, como podría ser la ubicación geográfica del usuario o la zona horaria.

Los RS demográficos suelen ser útiles cuando son parte de ensamblados híbridos, ya que representan otra fuente de información capaz de enriquecer el poder de recomendación del sistema en cuestión. A modo de ejemplo, podría no ser relevante recomendar un restaurante lejano o uno que ya se encuentre cerrado al momento de la recomendación, por más que sea del gusto del usuario.

RS Híbridos

Los RS híbridos apelan a lo mejor de cada mundo, utilizando distintas fuentes y tipos de datos, y a partir de ello, ensamblar RS que tengan un poco de las cuatro clases descritas más arriba [28, pp. 19].

RS Basados en Datos en Grafos

Los grafos representan de forma estructural las relaciones entre usuarios e ítems. Pueden ser construidos con los usuarios, los ítems, o ambos. En los últimos años han llamado particularmente la atención los Knowledge Graphs (KG). Los KG son grafos heterogéneos (i.e. los nodos pertenecen a distintas clases de entidades) donde

las aristas representan relaciones entre los nodos. En estos grafos se pueden integrar los ítems con sus atributos, así como los usuarios y sus datos contextuales (e.g. espacio y tiempo) [37].

Un KG es capaz de expresar relaciones de mayor nivel de abstracción y complejidad, pudiendo estas relaciones latentes ser capturadas por algoritmos para realizar recomendaciones. No solo los KG son capaces de lograr un mejor desempeño a la hora de recomendar, sino que también pueden alcanzar mayores niveles de *explicabilidad*, un aspecto valorado por los usuarios. La explicabilidad es la capacidad del RS de explicar porqué fue llevada a cabo la recomendación. A modo de ejemplo, explicar que la película A fue recomendada porque al usuario le gustó la película B, que es parecida, o sugerirle a X que siga a Y, porque ambos son amigos de Z en una red social.

Existen tres maneras de implementar RS basados en KG [37]:

1. **Embedding-based methods.** Mediante el uso de algoritmos se codifica el KG en vectores de baja dimensión.
2. **Path-based methods.** Se construye un grafo compuesto por usuarios e ítems y a partir de los patrones de conectividad entre las distintas entidades se realizan las recomendaciones. Estos métodos se asientan en la similaridad de conexiones entre usuarios e ítems.
3. **Unified methods.** Combinan lo mejor de los dos métodos descritos más arriba para alcanzar mayores niveles de desempeño.

En la sección 5.1 se desarrolla más en profundidad algunos de estos métodos. En particular el modelo KGAT (Knowledge Graph Attention Network for Recommendation) [39], perteneciente a los métodos unificados mencionados en el punto anterior.

2.1.3. Desafíos Clásicos de los RS

La diversidad tanto de los datos como los sistemas de recomendación a entrenar conllevan también desafíos que suelen aparecer a la hora de implementar estos sistemas. Estos desafíos están vinculados a aspectos de privacidad y seguridad, cómo aprovechar de mejor manera las distintas fuentes de datos disponibles o evaluar los RS. A continuación se describe de forma sucinta algunos de estos desafíos usuales en los RS con el fin de ejemplificar las complejidades que se pueden presentar.

Ataques al RS

La mejor forma de ejemplificar un ataque a un RS es con una plataforma de *e-commerce*. En estos sistemas los vendedores que publican sus productos en la plataforma pueden verse incentivados a manipular la salida del RS. Por ejemplo, aumentando artificialmente las calificaciones de los productos de interés, ya sea de forma manual o mediante *bots* desarrollados con tal fin. También se podría atacar por el frente de las reseñas, dejando opiniones positivas en sus productos, otra actividad factible mediante la utilización de *bots*, que podrían basarse en modelos de lenguaje entrenados con técnicas del campo del Procesamiento del Lenguaje Natural (PLN). Además podría, en vez de inflar sus propios productos, atacar a los de la competencia mediante malas calificaciones y reseñas negativas. De esa forma podría el atacante manipular el RS para que sus productos se posicionen mejor en cada salida, obteniendo de forma ilícita una ventaja competitiva. Estos ataques no solo perjudican directamente al RS, sino que además afectan la experiencia y satisfacción del usuario [28, pp. 24].

RS Grupales

Los RS grupales representan una extensión de los RS tradicionales. En estos sistemas se tiene como objetivo realizar recomendaciones a un grupo de usuarios en vez de a uno solo. Estos RS son efectivos en escenarios donde se puede encontrar un nivel considerable de homogeneidad en cada grupo, de lo contrario no tendrían demasiado valor [28, pp. 24].

RS Multi-criterio

En este escenario los usuarios no dan una única calificación global, sino que dan *feedback* a determinadas características del ítem a calificar. A modo de ejemplo, en un RS de películas, los usuarios podrían calificar aspectos tales como la trama, los efectos especiales, las actuaciones o la fotografía. Luego las distintas calificaciones se agregan a un vector donde se almacenarán para posterior uso por parte de los algoritmos [28, pp. 24].

Active Learning en los RS

Con el fin de enfrentar el problema de la dispersión de la RM (cuando la mayoría de los usuarios solo han calificado a una pequeña porción de los ítems), algunos RS toman una posición proactiva, generando incentivos para que los usuarios califiquen y den *feedback* sobre los ítems [28, pp. 25].

Privacidad

Los RS se basan en el *feedback* dado por los usuarios, ya sea de forma explícita o implícita. Por lo tanto, la recolección de estos datos puede resultar en una violación de la privacidad de las personas. No solo el hecho en sí de la recolección de *feedback* (de la que el usuario puede ni estar al tanto) es un asunto delicado, sino también lo que puede significar la información recabada. Se podrían realizar inferencias en cuanto a la orientación política o sexual de los usuarios, preferencias personales y aspectos que un individuo podría considerar privado [28, pp. 25].

2.2. Métricas para Sistemas de Recomendación

En esta sección se discuten distintas métricas empleadas en los RS, dividiéndose en dos grandes grupos: métricas tradicionales y métricas alternativas. Las primeras son métricas de carácter más bien objetivo, usadas en problemas de clasificación clásicos pero adaptadas a las necesidades de los RS. Las segundas son específicas a los RS y tienen características más bien subjetivas y dependientes del dominio, por lo que resultan de difícil evaluación en términos absolutos. Por esta razón las métricas alternativas serán útiles más bien para análisis comparativos entre distintos RS entrenados para el *mismo* problema (i.e. los mismos datos).

2.2.1. Métricas Tradicionales

Las métricas llamadas aquí tradicionales son variantes adaptadas para RS de métricas utilizadas en otros dominios del aprendizaje automático, como la *Precision* o la *Recall*, entre otras. Además se presentan dos métricas específicas para estos sistemas (o problemas similares), como el *Hit Rate* y la NDCG (*Normalized Discounted Cumulative Gain*). En todos los casos se calculan a nivel de usuario y luego se promedia para todos, obteniendo un único valor a nivel de RS. Es decir que a cada métrica

se le calcula su media para recomendaciones de largo K (i.e. se recomienda de a K items para cada usuario). Por lo tanto se calcula una *MeanAveragePrecision@K* o una *MeanAverageHitRate@K*. Al haber sido adaptadas al caso de los RS, interesa tener en cuenta la relevancia de los ítems así como su orden en la lista de recomendaciones.

Hit Rate

El *Hit Rate* vale 1 si el usuario ya había interactuado con alguna de las recomendaciones (i.e. la interacción está en el conjunto de prueba) y 0 en caso contrario. Se trata de una métrica sencilla de implementar e interpretar. Se podría decir que es poco "ambiciosa", pero útil a un nivel intuitivo para evaluar lo mínimo que el RS puede dar. El *Hit Rate* vale entre 0 y 1, siendo un buen resultado cuanto más cerca de 1 se encuentre.

Normalized Discounted Cumulative Gain

La Normalized Discounted Cumulative Gain [15] (NDCG) mide la calidad del *rankeo* en las recomendaciones. La métrica tiene en cuenta el orden de los K ítems recomendados. La *cumulative gain* es la suma de todos los *scores* (i.e. coeficiente de confianza que vale entre 0 y 1) asociados a los ítems recomendados. ***Discounted** Cumulative Gain* tiene en cuenta el orden (posición) de las recomendaciones, descontando cada *score* por el logaritmo de su posición (índice en la lista). NDCG es la métrica anterior normalizada, para así tener en cuenta las variaciones en el número de recomendaciones. La *NDCG* vale entre 0 y 1, siendo un buen resultado cuanto más cerca de 1 se encuentre. Esta métrica solo se utiliza en los modelos de base KGAT, ALS y BPR debido a que es usada como referencia en el primero.

Precision@K

Es la proporción entre los K ítems recomendados que son relevantes. Se considera que un ítem es relevante si tuvo interacciones con el usuario. Vale entre 0 y 1, siendo un buen resultado cuanto más cerca de 1 se encuentre. Se calcula como:

$$Precision@K = \frac{|R_{items}|}{K}. \tag{2.2}$$

donde $|R_{items}|$ es la cantidad de ítems relevantes en la recomendación y K el largo de la lista de recomendación.

Recall@K

Es la proporción de ítems relevantes en las recomendaciones. Se considera que un ítem es relevante si tuvo interacciones con el usuario. Vale entre 0 y 1, siendo un buen resultado cuanto más cerca de 1 se encuentre. Se calcula como:

$$Recall@K = \frac{|R_{items}|}{|H_{items}|}. \tag{2.3}$$

donde $|R_{items}|$ es la cantidad de ítems relevantes en la recomendación y H_{items} es la cantidad de ítems en el historial del usuario.

2.2.2. Métricas Alternativas

Estas métricas son capaces de expresar aspectos de carácter subjetivo de los RS y por consiguiente resultan difíciles de medir y de evaluar. No especialmente por la dificultad de su implementación, sino por los cuidados a tener en cuenta a la hora de utilizarlas. Esa es una de las razones por las que la mayoría de estas métricas no tienen

una versión única, existiendo distintas variedades según autores y dominios del RS en cuestión.

Otra dificultad al trabajar con estas métricas es que no se suele contar con valores de referencia, dependiendo de la implementación, tipo y procesamiento de datos. Por lo tanto, su principal valor será poder comparar distintas versiones de RS entrenados, así como contra los RS Aleatorios y de Popularidad, los cuales resultan útiles debido a sus comportamientos de cierta manera predecibles.

Además de la medida *offline* de estas métricas alternativas, donde se miden de forma asíncrona, existen mediciones de carácter interactivo, donde se pide *feedback* extra a los usuarios. Pero por motivos de alcance del presente trabajo no se tendrán en cuenta.

Para computar estas métricas es necesario calcular distancias entre los ítems. Por esta razón es necesario representar a los ítems de forma vectorial, asunto en sí no trivial, que se enmarca dentro de un problema de ingeniería de atributos. Se deben seleccionar qué atributos usar de los ítems y cómo procesarlos. En el presente trabajo se probaron distintas formas, optando finalmente por elegir cuatro atributos, representar cada ítem con *one-hot encoding* y luego aplicar una técnica de reducción de dimensionalidad para obtener vectores más densos. Una vez se tiene un vector para cada ítem, se calcula la matriz de distancias coseno por única vez para hacer los cálculos de forma más eficiente.

Ítem Coverage

Cobertura a nivel de ítems [7]. Representa el porcentaje de ítems que el RS fue capaz de recomendar a los usuarios en el conjunto de prueba. Esta métrica resulta de interés para evaluar qué proporción del catálogo de ítems disponibles está siendo recomendada. Lo anterior puede ser clave en casos que sea de particular interés aumentar la cobertura o de lo contrario disminuirla para impulsar algunos ítems sobre otros. Se calcula como:

$$ItemCoverage = 100 \times \frac{n}{N}, \tag{2.4}$$

donde n es el número de ítems recomendados y N los ítems en el conjunto de entrenamiento (i.e. ítems que potencialmente se recomendarían).

User Coverage

Cobertura a nivel de usuario [7]. Representa el porcentaje de usuarios a los que el RS fue capaz de recomendar algún ítem. Tener una medida de cuántos usuarios reciben recomendaciones también puede ser clave para alcanzar las metas deseadas con el RS implementado. Se calcula como:

$$UserCoverage = 100 \times \frac{u}{U}, \tag{2.5}$$

donde u es el número de usuarios a los que se le recomendó y U el número total de usuarios. Esta métrica resulta de interés en casos reales, donde se tienen usuarios aún no tenidos en cuenta en los entrenamientos del RS. Para los casos en que se evalúa con el conjunto de prueba resultado de la partición del conjunto de datos previo al entrenamiento, esta métrica siempre devuelve 100 %, ya que se recomendará para todos los usuarios.

Personalization

Mide qué tan personalizadas son las recomendaciones para los usuarios. Se toman todas las listas de recomendación (lista de ítems recomendados a cada usuario), se arma

una matriz de dimensión (#test_users, #recommended_items) con 1 en el elemento u, i cuando al usuario u se le recomendó el ítem i y 0 cuando no. Luego, a partir de esta matriz se computan las similitudes coseno entre las filas y se calcula el promedio de la matriz triangular superior (avg_upper_triangle). Finalmente, la personalización se calcula como:

$$Personalization = 1 - avg\_upper\_triangle. \tag{2.6}$$

Esta métrica usa la noción de similitud o distancia para evaluar qué tan parecidas o no son las listas de ítems recomendados. Cuánto más similares sean las listas entre ellas, menos personalizadas serán las recomendaciones. La personalización expresa la capacidad del RS de generar recomendaciones personalizadas, lo que puede repercutir positivamente en la experiencia de usuario del sistema.

Diversity

Mide el grado de diversidad de los ítems recomendados a los usuarios. Se calcula para cada usuario y luego se promedia para todos. Existen distintas formas de medir la diversidad. Para los experimentos realizados se eligió implementar la *diversity* basada en la Intra-List Similarity (ILS) como figura en la biblioteca Recmetrics [20]. La ILS mide qué tan similares son los ítems recomendados a un usuario. Se pueden usar distintas medidas de distancia. Se eligió en este caso la similitud coseno.

Para calcular la ILS se suman las distancias entre los ítems recomendados y luego se divide entre el número de recomendaciones y se multiplica por 0.5. Cuanto menor sea la métrica, mayor la diversidad. Por último, para obtener la *diversity* total del RS, simplemente se calcula la media de todas las ILS.

$$DiversityILS(R_u) = \frac{1}{2}\frac{1}{|R_u|}\sum_{i \in R_u}\sum_{j \in R_u} cos\_sim(i,j), \tag{2.7}$$

donde R_u es la lista de recomendaciones para el usuario u y $cos\_sim(i,j)$ es la distancia coseno entre los ítems i y j.

Novelty

Mide qué tan desconocidas (novedosas) son las recomendaciones para los usuarios. Se calcula para cada usuario y luego se promedia para todo el RS. Cuanto más alta es la *novelty*, más populares (en promedio) son los ítems recomendados, y peor es la métrica. Esto es importante en casos donde se desee recomendar ítems poco consumidos (poco populares), los cuales usualmente generan utilidades mayores (e.g. para una web de *e-commerce*). También es un aspecto a considerar en los casos en que la novedad resulte importante para los usuarios. La implementación se basó en la versión de [43] y se calcula como:

$$Novelty(R_u) = \sum_{i \in R_u} \frac{log_2(pop(i))}{|R_u|}, \tag{2.8}$$

donde R_u es la lista de recomendaciones para el usuario u y $pop(i)$ es la popularidad del ítem i (i.e. el número de interacciones en el conjunto de entrenamiento).

Serendipity

Serendipity no es un concepto que tenga una traducción literal en el idioma español ni una definición exacta en el inglés. Se entiende como una "novedad gratamente sorpresiva". A modo de ejemplo, supongamos un usuario habitual de Amazon, que compra numerosos libros al año y suele fijarse en las recomendaciones que la web

le brinda. Un día, comprando un libro, o simplemente explorando el catálogo de la plataforma de *e-commerce*, observa que en el área destinada a las recomendaciones aparece un libro que desconocía pero que, por ejemplo, es de una temática que le apasiona y además fue escrito por un autor que suele leer. El anterior sería un caso de *serendipity* elevada. Esta característica en un RS puede ser clave para mejorar la fidelidad y la experiencia de usuario, así como recomendar ítems poco populares pero rentables. Existen distintas implementaciones de esta métrica. En particular se eligió la siguiente [43]:

$$Serendipity(R_u) = \frac{1}{|H_u|} \sum_{i \in H_u} \sum_{j \in R_u} \frac{cos\_sim(i,j)}{|R_u|}, \tag{2.9}$$

donde R_u es la lista de recomendaciones para el usuario u, H_u es el historial del usuario u en el conjunto de prueba y $cos\_sim(i, j)$ es la distancia coseno entre los ítems i y j. En esta versión de *serendipity* se computan las distancias coseno entre los ítems recomendados y los ítems en el historial del usuario. Con esta implementación valores bajos representan buena *serendipity*. De esta manera se evalúa qué tan lejos están las recomendaciones de los ítems en el historial del usuario, ya que de estar muy cerca no se tendría el factor sorpresa en las recomendaciones (i.e. serían ítems muy similares).

2.3. RS Clásicos Implementados

En este capítulo se presentan RS llamados aquí *clásicos* (i.e. basados en técnicas de ML más bien tradicionales, sin uso de DL sobre grafos) implementados. Se implementaron para ser usados a modo de modelos de base (o *baselines*) para poder comparar luego con los implementados con GNN. Se describen dos modelos especializados en RS con *feedback* implícito, uno aleatorio (i.e. recomienda ítems al azar) y otro de popularidad (i.e. devuelve ítems populares nada más).

2.3.1. Filtrado Colaborativo con Feedback Implícito

El *feedback* implícito (FI) presenta ventajas respecto al *feedback* explícito (FE):

1. Es más fácil de recolectar información sobre interacciones, ya que no depende de que el usuario califique ítems por propia voluntad. Se almacenan acciones y comportamientos del usuario de manera automática.
2. Se pueden considerar diversas fuentes, como cantidad de *clicks* dados, tiempo de atención sobre un ítem o historial de compras, entre otras.

La principal desventaja del FI respecto al FE es que en el primero solo se tiene valoración positiva. Los usuarios no valoran de forma negativa y explícita, como pueden hacerlo al darle una mala calificación a un ítem en el marco de FE. Se puede entender que la asunción tomada sobre el FI es una forma de introducir ruido y sesgo en los datos, pero que es compensada con mayor cantidad de información capaz de ser recolectada.

Para las *baselines* se implementaron RS con los algoritmos Alternating Least Squares (ALS) [42] y Bayesian Personalized Ranking (BPR) [38]. Ambos métodos están disponibles en la biblioteca de Python Implicit [6].

2.3.2. Alternating Least Squares

ALS se encuentra entre los RS basados en métodos de factorización de matrices (MF, por sus siglas en inglés). Los métodos de MF tiene como fin atacar la alta dimensionalidad y dispersión de las matrices de *feedback* (interacciones) usuario-ítem.

Para ello, se descompone a la matriz de interacciones en el producto de otras dos matrices, una para los usuarios y otra para los ítems. En cada una de estas matrices, cada usuario (o ítem) estará representado por una fila, que se puede ver como un vector de *latent features* (i.e. atributos/*features* que no representan ninguna característica del mundo real, como podría ser el género de una película). De esta manera, al representar tanto usuarios e ítems como vectores, resulta sencillo calcular similitudes mediante el uso de alguna métrica de distancia, como puede ser la euclideana o la coseno, entre otras.

Se representa a la matriz de interacciones como $\mathbf{R}^{u \times i}$, donde u es el número de usuarios e i el de de ítems. La matriz $\mathbf{R}$ será factorizada en las matrices $\mathbf{U^{u \times f}}$ y $\mathbf{V^{i \times f}}$, donde $\mathbf{f}$ es el número de *latent features*, tal que $\mathbf{R} \approx \mathbf{U} \times \mathbf{V}$. Las matrices $\mathbf{U}$ y $\mathbf{V}$ serán las que se deban aprender con el algoritmo ALS.

ALS se basa en aplicar el método de mínimos cuadrados alternando entre las matrices $\mathbf{U}$ y $\mathbf{V}$ durante el proceso de optimización. En el trabajo en [42] se toma en cuenta la preferencia p del usuario y la confianza c que se tiene sobre la primera. La preferencia de un usuario u hacia el ítem i, $p_{u,i}$, será 1 si hubo algún tipo de interacción ($r_{u,i} > 0$) y 0 en caso contrario ($r_{u,i} = 0$) . La confianza del usuario u hacia el ítem i será:

$$c_{u,i} = 1 + \alpha r_{u,i}, \tag{2.10}$$

donde α es un factor de escalamiento. El parámetro α controla qué tanto peso se le quiere dar a las interacciones observadas respecto a las no observadas. Para estas últimas se tiene que la confianza será de 1, mientras que para las interacciones observadas la confianza será mayor a medida que se aumente α. Para los vectores de cada usuario y cada ítem en el espacio de *latent features*, se optimiza la función:

$$\sum_{u,i} c_{ui}(p_{ui} - x_u^T y_i)^2 + \lambda(\sum_u \parallel x_u \parallel^2 + \sum_i \parallel y_u \parallel^2), \tag{2.11}$$

donde λ es el parámetro de regularización L_2.

Finalmente se cuenta con las representaciones vectoriales para usuarios e ítems en las matrices $\mathbf{U}$ y $\mathbf{V}$, resultando sencillo encontrar los ítems similares a $\mathbf{v_i}$ mediante:

$$SimilarItems = \mathbf{V} \cdot \mathbf{v_i^T}, \tag{2.12}$$

donde $\cdot$ es el producto matricial. También es sencillo realizar recomendaciones al usuario $\mathbf{u_i}$ mediante el producto interno:

$$Recommendations = \mathbf{V} \cdot \mathbf{u_i^T}. \tag{2.13}$$

Estos productos internos devolverán *scores* capaces de ser ordenados y así *rankear* la similitud de los ítems con respecto a otro o recomendar ítems a un usuario.

2.3.3. Bayesian Personalized Ranking

A diferencia de ALS, Bayesian Personalized Ranking (BPR) trata de fortalecer la capacidad de *rankeo* del RS mediante optimización. ALS *rankea* los ítems recomendados de acuerdo al *score* calculado, tomando como clase positiva (1) cuando hubo interacción, y como clase negativa (0) cuando no la hubo. En BPR, en vez de tomar cada interacción usuario-ítem, se considera para el conjunto de entrenamiento triplas de ítems (u, i, j), donde u es el usuario, i y j dos ítems, y la posición de i indica que es preferido sobre j. La optimización será basada en el *rankeo* de estos pares usuario-ítems.

El problema de encontrar el mejor *rankeo* personalizado para todos los ítems se formula de forma bayaesiana. Consiste en maximizar la siguiente probabilidad *a posteriori*:

$$p(\mathbf{\Theta}|>_u) \propto p(>_u|\mathbf{\Theta})p(\mathbf{\Theta}), \tag{2.14}$$

donde $\mathbf{\Theta}$ representa los parámetros del modelo y $>_u$ es la preferencia deseada pero latente del usuario u. $p(>_u|\mathbf{\Theta})$ es la función de verosimilitud y $p(\mathbf{\Theta})$ la densidad de probabilidad *a priori*. El objetivo es calcular los parámetros $\mathbf{\Theta}$ del modelo que mejor *rankee* según las preferencias del usuario u, apoyándose en una formulación *bayesiana* del problema. De esta manera, como se ve más adelante, en el proceso de optimización se maximiza la probabilidad *a posteriori* mediante una función de pérdida que tendrá en cuenta la función de verosimilitud y la función de densidad de probabilidad *a priori*. Por lo tanto se deben modelar primero estas dos últimas funciones.

Para modelar la función de verosimilitud se asume que todos los usuarios actúan de forma independiente y de igual manera es el ordenamiento de i y j para cada u, por lo tanto, la probabilidad de que un usuario u prefiera al ítem i sobre el j es:

$$p(i>_u j|\mathbf{\Theta}) := \sigma(\hat{x}_{uij}(\mathbf{\Theta})), \tag{2.15}$$

donde σ es la función sigmoide. $\hat{x}_{uij}(\mathbf{\Theta})$ es una función real que representa la relación entre el usuario u y los ítems i y j, calculada usando el modelo de factorización matricial. Para la función de probabilidad *a priori* $p(\mathbf{\Theta})$ se considera la distribución normal $\mathbf{N}(0, \mathbf{\Sigma}_\theta)$ con media cero y matriz de varianza-covarianza $\mathbf{\Sigma}_\theta$. Luego la optimización se llevará a cabo sobre la función BPR-OPT:

$$\begin{aligned} BPR-OPT &:= \ln p(\mathbf{\Theta}|>_u) \\ &= \ln p(>_u|\mathbf{\Theta})p(\mathbf{\Theta}) \\ &= \ln \prod_{(u,i,j)\in \mathbf{D}_S} \sigma(\hat{x}_{uij})p(\mathbf{\Theta}) \\ &= \sum_{(u,i,j)\in \mathbf{D}_S} \ln \sigma(\hat{x}_{uij}) + \ln p(\mathbf{\Theta}) \\ &= \sum_{(u,i,j)\in \mathbf{D}_S} \ln \sigma(\hat{x}_{uij}) - \lambda_\Theta \parallel \mathbf{\Theta} \parallel^2, \end{aligned} \tag{2.16}$$

donde λ_Θ son parámetros de regularización del modelo ($\mathbf{\Sigma}_\theta = \lambda_\Theta \mathbf{I}$) y $\mathbf{D}_S$ los datos de entrenamiento.

Para realizar recomendaciones se utilizan las representaciones vectoriales (*embeddings*) generadas para usuarios e ítems. En las matrices $\mathbf{U}$ y $\mathbf{V}$ se almacenan usuarios e ítems respectivamente, resultando sencillo encontrar tanto similitudes entre ítems como generar recomendaciones, al igual que para ALS:

$$SimilarItems = \mathbf{V} \cdot \mathbf{v_i^T}, \tag{2.17}$$

donde $\mathbf{v_i}$ es el vector del ítem i.

$$Recommendations = \mathbf{V} \cdot \mathbf{u_i^T}, \tag{2.18}$$

donde $\mathbf{u_i}$ es el vector del usuario i.

2.3.4. RS Aleatorio

Se implementó un RS Aleatorio para poder comparar con los entrenados. Este RS simplemente recomienda ítems de forma aleatoria. Se le pasa como parámetro K

para indicar la cantidad de recomendaciones a realizar (i.e. el largo de cada lista de recomendaciones).

Se espera que un RS Aleatorio no devuelva buenos resultados salvo en métricas como *novelty* o *diversity*. En cobertura alcanzaría buenos niveles, ya que es capaz de recomendar cualquier ítem del catálogo.

2.3.5. RS de Popularidad

Además del RS Aleatorio, se implementó un RS de Popularidad. Este RS recomienda solo ítems populares (i.e. ítems altamente consumidos), en la misma cantidad que el parámetro K que se le pase para indicar la cantidad de recomendaciones a realizar (i.e. el largo de cada lista de recomendaciones).

Se espera que este RS se desempeñe mal en todas las métricas, salvo en *Hit Rate*, ya que tratándose de ítems altamente consumidos (i.e. populares) es muy probable que entre los historiales de los usuarios del conjunto de prueba haya numerosas interacciones con ítems populares.

2.3.6. Resumen del Capítulo

En este capítulo se introdujeron los modelos más clásicos y probados de RS, sus principales fortalezas y debilidades, así como también los desafíos más comunes que presentan en aspectos como desempeño, arranque en frío o privacidad. Posteriormente se presentaron algunas métricas de interés para evaluar RS, dividiéndose en métricas aquí llamadas tradicionales y alternativas. Se vio cómo las primeras resultan en general análogas a métricas ya conocidas en otros problemas (e.g. clasificación) mientras que las segundas presentan desafíos, particularmente a la hora de representar a los ítems previo al cálculo de las métricas en sí. Por último, se presentan cuatro RS clásicos que formarán parte de los modelos de base. Dos de ellos, ALS y BPR, son métodos de factorización de matrices para *feedback* implícito, mientras que los RS Aleatorio y de Popularidad son implementaciones sencillas, de comportamiento predecible y por lo tanto valiosas a la hora de realizar comparaciones con otros modelos. En el próximo capítulo se describe el análisis exploratorio realizado sobre los datos disponibles de la Biblioteca Ceibal.

Capítulo 3

Análisis Exploratorio de Datos

En este capítulo se presenta el análisis exploratorio de los datos de la Biblioteca Ceibal. Se comienza con una descripción de los datos disponibles para luego explorar más a fondo las principales características de cada conjunto de datos y terminar con representaciones vectoriales de los recursos (i.e. libros) en el plano, a modo de verificar agrupamientos interesantes.

3.0.1. El Conjunto de Datos

Luego de firmado un NDA con Plan Ceibal, se tuvieron a disposición los datos de la biblioteca. Se contó con un *dataset* de ítems (libros, o recursos) disponibles con veintiséis atributos cada uno, de entre los cuales se destacan: tipo de recurso, título, autor, idioma, editor, año de publicación, género, audiencia (e.g. niños, adultos, etc) y materia (etiquetas del tesauro [25] de la biblioteca).

También quedaron a disposición los datos de interacciones usuario-ítem, organizados en cuatro tablas:

1. **Usuario vs calificación.** En esta tabla se guardan las calificaciones dadas por los usuarios a cada ítem. Tipo de *feedback*: explícito.

2. **Usuario vs porcentaje de avance de lectura.** Se almacena el porcentaje de avance de lectura de cada usuario para cada libro que haya pedido prestado. Que un usuario avance se considera positivo. Fue la tabla elegida para experimentar, sobre lo cual se expondrá más adelante. Tipo de *feedback*: implícito.

3. **Usuario vs tiempo de lectura total**. Indica tiempo de lectura total de cada usuario para cada recurso prestado. Tipo de *feedback*: implícito.

4. **Usuario vs cantidad de préstamos**. Se guarda la cantidad de préstamos realizados de cada recurso a cada usuario. Se considera que a mayor número de préstamos, más positiva es la valoración. Tipo de *feedback*: implícito.

Con el fin de conocer más a fondo los datos disponibles, en las secciones siguientes se muestra parte del análisis exploratorio llevado a cabo en los sub-conjuntos de datos (tablas) mencionados anteriormente.

Tabla	**#Interacciones**
Porcentaje de avance de lectura	356352
Cantidad de préstamos	453820
Calificaciones	25557
Tiempo de lectura	356352

Tabla 3.1: Cantidad de interacciones por tabla.

Porcentaje de avance sin filtrar

Dimensiones	Recursos únicos	Usuarios	Perfiles
356352 x 8	6289	133721	6

Porcentaje de avance luego de filtrado por perfil Adulto

Dimensiones	Recursos únicos	Usuarios	Perfiles
45561 x 8	5117	9134	Adulto

Tabla 3.2: Estadísticas generales de tabla de porcentaje de avance de lectura antes y después de filtrar por perfil Adulto.

3.1. Análisis exploratorio

3.1.1. Interacciones Usuario-Ítem

El primer paso consistió en extraer estadísticas de interés para cada una de las cuatro tablas con interacciones disponibles. En particular para la de porcentaje de avance de lectura, ya que, como se explica más adelante, será la utilizada para entrenar los RS. En la Tabla 3.1 se despliega el número de interacciones en cada tabla. Se puede verificar que la cantidad de calificaciones dadas es notoriamente inferior a las otras clases de interacciones, siendo este uno de los motivos a la hora de decidir explorar RS basados en *feedback* implícito. El otro motivo surgió de preguntarse qué tipo de interacciones aportaban más certezas en cuanto a qué tanto le gustó un recurso a un usuario. De esta manera, se descarta el tiempo de lectura, ya que el acumulado de horas no es una indicación clara de la opinión que un usuario tiene de un libro. Algo similar pasa con la cantidad de préstamos: un usuario puede pedir prestados muchos recursos pero no leerlos siquiera.

En la Tabla 3.2 se muestran las estadísticas generales para la tabla de interacción elegida (porcentaje de avance de lectura) antes y después del filtrado por perfil Adulto. Observar cómo la cantidad de usuarios se reduce de forma considerable. A pesar de que el presente trabajo se centrará en las interacciones provistas por la tabla de porcentaje de avance de lectura, de todas maneras se detalla información de interés surgida del análisis de los datos. Para cada tabla de interacciones se extraen estadísticas de interés (o curiosas), agrupando en cada caso por perfil de usuario, los cuales se dividen en Primaria, Media, Docente, Ibirapitá (adultos mayores), Adulto y Sin calificar. En la Tabla 3.3 se muestra el número de interacciones por perfil, siendo Primaria el más poblado. También se verifica que el tipo de recurso más pedido es el libro electrónico, mientras que los audiolibros son muy poco tomados en préstamo. Resulta interesante observar cómo Primaria (exceptuando Sin calificar) es quien en porcentaje consume más audiolibros, mientras que en caso contrario se encuentra el perfil Ibirapitá, que corresponde a usuarios de la tercera edad.

En la Tabla 3.4 se puede apreciar que la media de calificaciones por perfil es similar

Cantidad de préstamos				
Perfil	**#Interacciones**	**Tipo de recurso más pedido**	**Audios**	**Audios(%)**
Adulto	56127	Libro-e	2014	3,59
Docente	86173	Libro-e	4889	5,67
Ibirapitá	24831	Libro-e	462	1,86
Media	56332	Libro-e	1918	3,40
Primaria	228227	Libro-e	13651	5,98
Sin calificar	695	Libro-e	56	8,06

Tabla 3.3: Cantidad de préstamos. Número de interacciones por perfil

Calificaciones			
Perfil	**#interacciones**	**#interacciones - % del total**	**Calificación media**
Adulto	1355	5,31	4,07
Docente	2420	9,48	4,4
Ibirapitá	520	2,04	4,04
Media	2133	8,36	4,53
Primaria	19047	74,63	4,48
Sin calificar	48	0,19	4,68

Tabla 3.4: Calificaciones de 1 a 5 dadas por los lectores.

entre los grupos. Sí se observan diferencias interesantes a la hora de ver qué perfiles suelen calificar más de forma explícita. Otra vez se verifica que el perfil Ibirapitá es el que menos calificaciones tiene mientras que Primaria tiene casi el 75 %. Se puede concluir que para explorar RS basados en *feedback* explícito puede ser clave contar con datos de Primaria, ya que allí estarían los usuarios más propensos a calificar ítems.

En la Tabla 3.5 se puede ver cómo el perfil Ibirapitá es el que tiene una mayor media de tiempo de lectura, mientras que Primaria es el de menor media. ¿Se debe a la velocidad de lectura de cada grupo? ¿Al largo medio de los textos de cada grupo? De contar con los datos necesarios, serían un par de interrogantes interesantes de resolver.

En la Tabla 3.6 se puede apreciar que la media de porcentaje de avance de lectura de Ibirapitá es la mayor, mientras que en Media es la menor. De todas maneras, para todos los grupos se verifica que la media está por debajo del umbral de 70 %, tomado como valor de referencia: por encima de éste se considera al recurso completado y por debajo no. Uno de los motivos de tener bajos niveles de porcentaje de avance de lectura son los préstamos tomados para una inspección rápida del recurso y luego decidir si leerlo o no. Esto genera que haya muchos ítems que con suerte apenas fueron comenzados. En cuanto al porcentaje de recursos completados por cada grupo (i.e. por encima del 70 %), nuevamente se verifica Ibirapitá tiene el máximo mientras que Media ostenta el mínimo. De forma adicional, en la Tabla 3.7 se muestran los títulos y autores con más libros completados. Observar cómo nombres de escritores muy leídos, como Roy Berocay e Isabel Allende, son prácticamente dominantes en los perfiles esperables.

Como fue mencionado más arriba, se eligió la tabla de porcentaje de avance de lectura como interacciones para entrenar RS basados en GNN. Las interacciones se almacenan en una tabla con los siguientes campos:

Tiempo de lectura total			
Perfil	**#interacciones**	**Tiempo medio de lectura**	**Tiempo acumulado de lectura (horas)**
Adulto	45561	166	125866
Docente	67670	48	54033
Ibirapitá	21339	238	84466
Media	42978	57	40571
Primaria	177081	28	81262
Sin calificar	562	24	226

Tabla 3.5: Tiempo total de lectura.

Porcentaje de avance de lectura						
Perfil	**#Interacciones**	**#Usuarios**	**#Recursos**	**media de %avance**	**leídos (umbral 70 %)**	**leidos en %**
Adulto	45561	9134	5117	47,54	18152	39,84
Docente	67670	22757	5068	38,43	18683	27,61
Ibirapitá	21339	2757	3912	56,68	10761	50,43
Media	42978	23128	3407	30,81	9101	21,18
Primaria	177081	75068	1257	37,56	47691	26,93
Sin calificar	562	214	430	40,17	169	30,07

Tabla 3.6: Porcentaje de avance de lectura. Estadísticas.

Porcentaje de avance de lectura		
Perfil	**Título más completado**	**Autor más completado**
Adulto	La voz ausente	Allende, Isabel
Docente	La taberna del loro en el hombro	Berocay, Roy
Ibirapitá	Más allá del invierno	Allende, Isabel
Media	El tesoro de cañada seca	Vázquez, Marcos
Primaria	La taberna del loro en el hombro	Berocay, Roy
Sin calificar	Terror en el campamento	Sánchez Vegara, M^{a} Isabel

Tabla 3.7: Porcentaje de avance de lectura. Autores.

1. **ID_LECTOR:** Identificador único del usuario (i.e. lector). Este campo fue anonimizado para salvaguardar la privacidad del usuario.
2. **ID_RESOURCE:** Identificador único del recurso consumido.
3. **TITULO:** Título del recurso.
4. **AUTOR:** Autor del recurso.
5. **TIPO_DE_RECURSO:** Tipo de recurso (e.g. libro, audiolibro, imagen, etc).
6. **max_porcentaje_av:** Porcentaje máximo de avance que hizo el usuario del recurso.
7. **perfil_indicadores:** Perfil del usuario (e.g. Primaria, Media, Docente, Ibirapitá, Adulto).

En la Figura 3.1 se despliega cómo se distribuyen los recursos en cuanto a cantidad de interacciones para el perfil Adulto. En este gráfico se representa la cantidad de veces que un ítem tuvo interacción con un usuario. Como es usual en los RS, suele haber un pequeño número de ítems que fueron altamente consumidos, mientras que el resto ha recibido poco *feedback*. Por ello se indica la *long-tail*, por la larga cola que se forma a la derecha de la curva, conformada por los ítems de baja frecuencia de interacción. Esta distribución suele ser habitual en los RS.

En la Tabla 3.8 se muestran algunos usuarios con su número de ítems consumidos y media de porcentaje de avance de lectura. En la Figura 3.2 se grafica la cantidad de interacciones versus la media de porcentaje de avance de lectura por usuario. Observar cómo la mayoría de los usuarios consumen pocos recursos y no suelen finalizar la lectura. La media de porcentaje de avance total es de 47,55 %, mientras que la media de interacciones total es de 4,98 ítems por usuario.

3.1.2. Recursos de la Biblioteca Ceibal

Los recursos disponibles en la Biblioteca Ceibal se almacenan en una tabla de veintiséis columnas (atributos). A continuación se describen los atributos considerados más relevantes:

1. **TIPO_DE_RECURSO:** Tipo de recurso (libro, audiolibro, imagen, etc).
2. **TITULO:** Título del recurso.

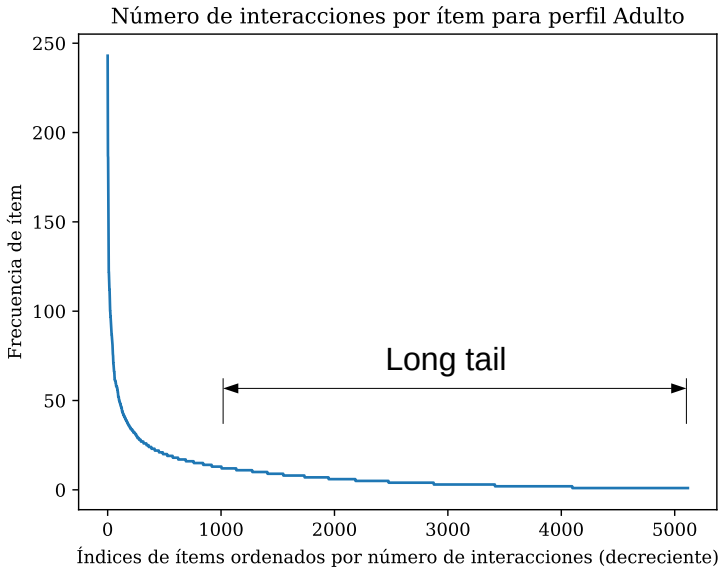

Figura 3.1: Distribución de número de interacciones por ítem para tabla de porcentaje de avance de lectura. De esta manera se representa la frecuencia de consumo de cada ítem para el perfil Adulto.

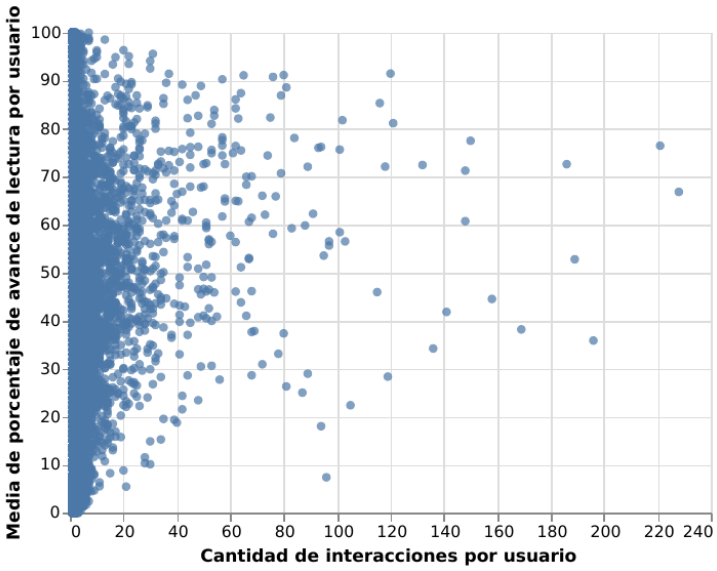

Figura 3.2: Cantidad de interacciones versus la media de porcentaje de avance de lectura por usuario.

ID_LECTOR	max_porcentaje_av count	max_porcentaje_av mean
0	3	20.00
1	2	39.50
2	2	5.00
3	8	71.12
4	2	59.50
...	...	...
5	15	46.66
6	10	68.60
7	1	58.00
8	1	25.00
9	12	38.41

Tabla 3.8: Usuarios con sus IDs (editados por razones de privacidad), número de ítems consumidos y media de porcentaje de avance de lectura.

Tabla de recursos			
#Recursos	tipo recurso	idioma	autor
7478	2	4	3302
editor	año publicación	género	materia
406	110	29	4342
audiencia			
20			

Tabla 3.9: Estadísticas generales de tabla de recursos. Se indican valores únicos de cada campo excepto el número de recursos total.

3. **AUTOR:** Autor del recurso.
4. **IDIOMA:** Idioma del recurso.
5. **EDITOR:** Editorial del recurso.
6. **ANO_DE_PUBLICACION:** Año de publicación del recurso.
7. **GENERO:** Género del recurso.
8. **AUDIENCIA:** Público objetivo del recurso.
9. **MATERIA:** Etiquetas del tesauro [25] de la biblioteca.
10. **Otros atributos:** ISBN, ISSN, restricciones de acceso, perfil de préstamo, tipo de licencia, etc.

En la Tabla 3.9 se despliegan estadísticas básicas para los atributos más relevantes de la tabla de recursos, como la cantidad de libros disponible y los distintos valores para cada uno de los campos indicados.

En la Tabla 3.10 se muestran los cinco valores más frecuentes para los atributos género, audiencia y tipo de recurso y en la Tabla 3.11 para los atributos autor, idioma y editor. Observando las columnas de porcentajes sobre el total de recursos (i.e. 7478) es posible afirmar que es una biblioteca de libros electrónicos en español, mayoritariamente de los géneros Narrativa y Ensayo, para audiencias de niños y adultos. Los cinco autores más presentes son clásicos de la narrativa de segunda mitad del s-XIX y principios del s-XX, oriundos de Europa (Chéjov, Christian Andersen) y Estados Unidos (Edgar Allan Poe, Jack London, Henry James), mientras que las principales editoriales son tres españolas (Alfaguara, Tusquets, Siruela), una multinacional (Penguin Random House) y el propio Plan Ceibal.

Género	**%**	**Audiencia**	**%**	**Tipo de recurso**	**%**
Narrativa;Audiolibro	1,90 %	Adultos;Jóvenes	2,33 %	Audio	5,55 %
Audiolibro;Narrativa	2,42 %	Jóvenes	7,15 %	Libro-e	94,45 %
Poesía y lírica	3,16 %	Jóvenes;Adultos	7,29 %		
Ensayo	15,43 %	Niños	14,15 %		
Narrativa	72,16 %	Adultos	64,24 %		

Tabla 3.10: Los cinco valores más frecuentes para los atributos GENERO, AUDIENCIA y TIPO_DE_RECURSO. Para cada valor se indica su porcentaje sobre el total de los recursos.

Autor	**%**	**Idioma**	**%**	**Editor**	**%**
London, Jack	0,49 %	eng	2,35 %	Penguin Random House	3,13 %
James, Henry	0,58 %	spa	97,29 %	Alfaguara	3,69 %
Allan Poe, Edgar	0,64 %			Tusquets	3,82 %
Chéjov, Antón	0,64 %			Siruela	5,00 %
Christian Andersen, Hans	1,22 %			Plan Ceibal	12,49 %

Tabla 3.11: Los cinco valores más frecuentes para los atributos AUTOR, IDIOMA y EDITOR. Para cada valor se indica su porcentaje sobre el total de los recursos.

En la Tabla 3.12 se muestran algunos títulos y sus medias de porcentaje de avance de lectura, ordenados ascendentemente por popularidad, mientras que en la Tabla 3.13 se los ordena en forma descendente.

En la Figura 3.3 se puede apreciar cómo la mayoría de los ítems tienen baja cantidad de interacciones (similar a lo visto en la Figura 3.1). La media de porcentaje de avance de lectura total es de 41,86 % y la media de interacciones total es de 8,97 por ítem. Se verifica otra vez que la mayoría de los ítems son poco consumidos y en general no terminados (leídos).

Representación de Recursos en el Plano

Usando PyTorch Geometric (PyG) [19], una biblioteca construida sobre PyTorch para el entrenamiento de GNN y el manejo de datos en grafos, se generaron *embeddings* de los recursos para ver cómo estos se agrupaban en el plano. Se usó como referencia lo detallado en [11], a saber, un tutorial básico de cómo leer grafos guardados en archivos CSV, generar *embeddings* usando modelos pre-entrenados y luego ensamblar un grafo heterogéneo capaz de ser entrada a futuros entrenamientos. Es necesario señalar la importancia del uso de modelos pre-entrenados (i.e. usar Transfer Learning) para generar representaciones vectoriales densas de calidad.

Con la tabla de recursos, que ya se tenía en formato CSV, se generan los *embeddings*

TITULO	max_porcentaje_av count	max_porcentaje_av mean
La cabeza del profesor Dowell	1	99.00
Lo que Maisie sabía	1	1.00
Lo que está y no se usa nos fulminará	1	99.00
El caracol y el rosal	1	100.00
El capote	1	98.00
...	...	
El amante japonés	187	54.55
La vida en tus manos: superando el síndrome de...	195	36.90
Más allá del invierno	199	49.29
Cuentos completos	207	21.39
La voz ausente	243	60.72

Tabla 3.12: Ítems ordenados por popularidad en orden ascendente. Se indica además la media de porcentaje de avance de lectura para cada uno.

TITULO	max_porcentaje_av mean
El figon de la reina patoja	0.0
Intrusos y huéspedes & Habitación doble	0.0
¿Tenemos suficiente inteligencia para entender...	0.0
Cómo ver el mundo: una nueva introducción a la...	0.0
El mismo mar	0.0
...	...
Todos eran mis hijos:	100.0
Las ruinas de Tiahuanaco	100.0
La casa de ceniza	100.0
El cazador de estilemas	100.0
Se vende mamá	102.0

Tabla 3.13: Ítems ordenados por promedio de porcentaje de avance de lectura, en orden ascendente. Se indica además la media de porcentaje de avance de lectura para cada uno.

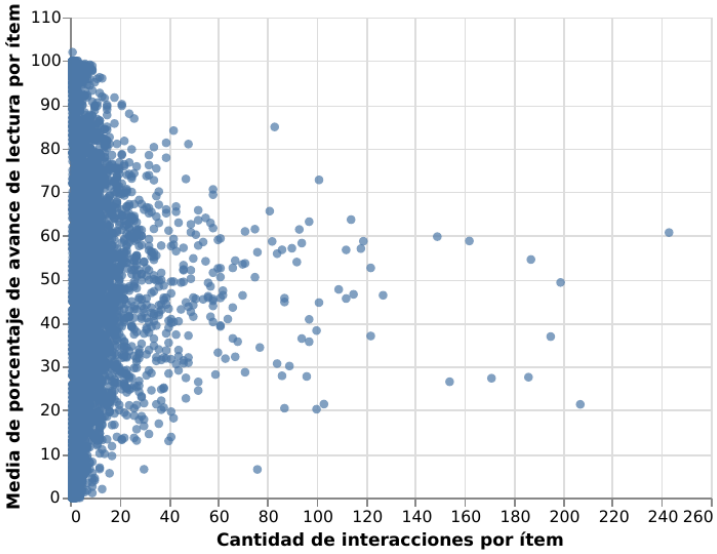

Figura 3.3: Cantidad de interacciones versus la media de porcentaje de avance de lectura por ítem.

usando *encoders* del *framework* Sentence Transformers [21]. Este *framework* es usado para generar *embeddings* de frases, imágenes y texto en general. Soporta más de cien idiomas, incluidos el español y el inglés, que son los presentes en la tabla de recursos. Al generar representaciones vectoriales densas de, por ejemplo, una frase, es posible calcular similaridades semánticas midiendo la distancia entre los vectores. Lo anterior resulta útil en tareas clásicas del PLN como búsqueda semántica o parafraseo, entre otras. Sentence Transformers está basado a su vez en los *frameworks* en PyTorch y Transformers [26] y ofrece variedad de modelos pre-entrenados y la posibilidad de realizar *fine-tuning* a partir de estos con datos propios.

Para generar los *embeddings* de los recursos se eligieron los campos TITULO y GENERO y el modelo pre-entrenado *distiluse-base-multilingual-cased-v1*. Este modelo es multi-lenguaje (i.e. es capaz de codificar múltiples lenguajes, incluidos el español y el inglés) y está disponible en la biblioteca. Este modelo generará vectores a partir de los dos campos mencionados más arriba. Para cada recurso se generará un vector que tenga en cuenta ambos campos. En el caso de TITULO es una frase (e.g. *El ingenioso hidalgo Don Quijote de La Mancha*) mientras que para GENERO puede ser tanto

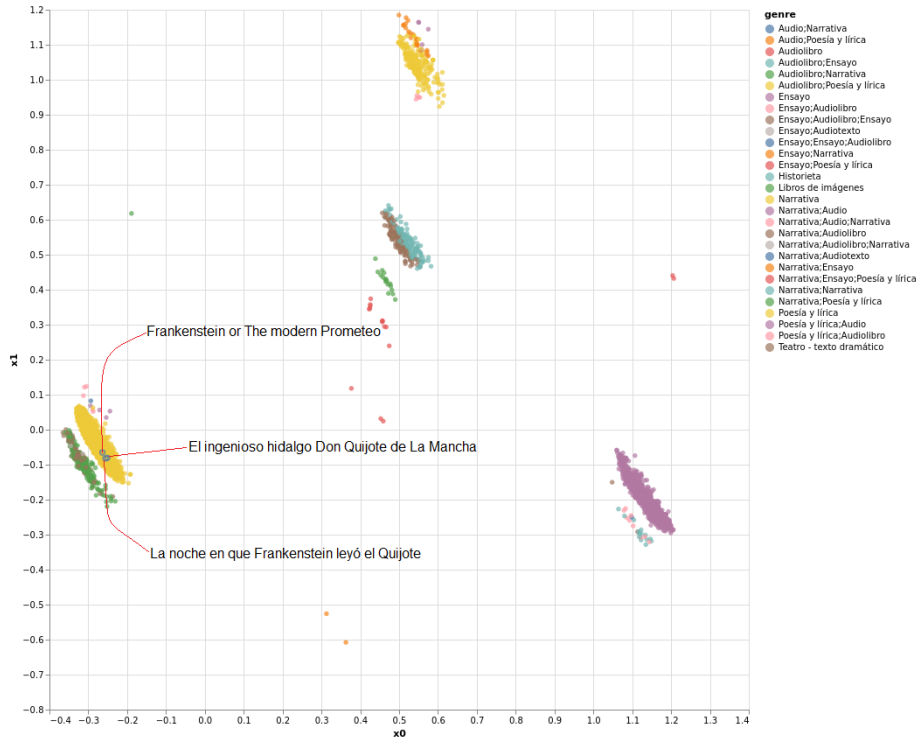

Figura 3.4: *Embeddings* de los recursos generados con PyG para los campos TITULO y GENERO. Se marcan con círculos azules los tres ítems elegidos para calcular las distancias entre ellos y se indica con texto sus títulos.

uno solo (e.g. Narrativa) como múltiples géneros separados por punto y coma (e.g. Narrativa;Ensayo).

Una vez se codifique cada recurso, se tendrá un tensor de dimensiones $(7478, 521)$ siendo 7478 el número de recursos y 521 da dimensión de cada vector de cada ítem. Luego, para poder representar los recursos en el plano, y previamente mapeando índices en el tensor con los títulos, se aplica la técnica de reducción de dimensionalidad PCA. Se obtiene así un tensor de dimensiones $(7478, 2)$, o sea 7478 puntos en el plano graficables, como se puede observar en la Figura 3.4.

En la Figura 3.4 es posible apreciar los agrupamientos de los recursos por géneros, los cuales se despliegan a la derecha. A su vez, cada recurso es representado con un punto en el plano, y se diferencian según qué palabras contengan sus títulos. Es de esperar que cuantos más similares sean dos títulos más cerca estarán en el plano y viceversa. Cada *cluster* corresponde a un género literario. Por ejemplo, la agrupación en amarillo, abajo a la izquierda de la figura, corresponde a recursos del género Narrativa, mientras que el *cluster* en violeta de abajo a la derecha, a ítems del género Ensayo.

A modo de ejemplo, se eligieron tres títulos dentro del género *Narrativa* para comparar distancias: (1) *El ingenioso hidalgo Don Quijote de La Mancha*, (2) *La noche en que Frankenstein leyó el Quijote* y (3) *Frankenstein or The modern Prometeo*. Estos ítems son los marcados con círculos azules y líneas rojas en la Figura 3.4. Esta elección fue hecha a propósito para ver el impacto de las palabras *Quijote* y *Frankenstein* a la hora de computar las distancias, en este caso eculideanas, aunque podría haber sido otra. La distancia entre los ítems 1 y 2 es de 0,94 mientras que entre los 1 y 3 es

1,11. En el primer caso se ve el efecto de cercanía provocado por la palabra *Quijote* contenida en el título, mientras que en el segundo caso, a pesar de también tener la palabra *Frankenstein* se verifica una mayor distancia en el plano debido a la ausencia de *Quijote*.

3.1.3. Resumen del Capítulo

En este capítulo se llevó a cabo un análisis exploratorio con el fin de presentar y conocer más en profundidad los datos. Se mostró cómo la mayoría de los ítems tienen pocas interacciones, el porqué de elegir *feedback* implícito como es el porcentaje de avance de lectura y se presentaron los atributos a nivel de recurso que *a priori* resultan más relevantes. Estos aspectos son relevantes de cara a los modelos que se entrenaron y que se detallan en los siguientes capítulos. De forma adicional, se usó el *framework* PyG para verificar agrupamientos entre los recursos de la biblioteca.

Introducidos ya los RS clásicos y el conjunto de datos, en el próximo capítulo se da entrada a los grafos y las GNN a modo de agregar la tercera pieza de los RS que se implementaron y se detallan en el Capítulo 5.

Capítulo 4

Graph Neural Networks

Los grafos son estructuras de datos capaces de describir sistemas complejos. Se componen de un conjunto de *nodos* interconectados por *aristas* [32, pp. 1]. Pueden ser homogéneos (i.e. una sola clase de nodos), heterogéneos (i.e. múltiples clases de nodos), dirigidos (i.e. las aristas tienen dirección) o no, de conocimiento (i.e. Knowledge Graphs [32, pp. 38]), entre otros.

A modo de ejemplo, un grafo puede resultar útil para representar una red social, donde los nodos son personas y las aristas, cuando existen, indican que dos personas son amigas, como se puede ver en la Figura 4.1. También suelen ser utilizados a la hora de representar moléculas complejas, como las proteínas. El poder de los grafos radica en las *relaciones* entre los datos (más que en las propiedades de éstos) así como también en su *generalidad* (i.e. la capacidad de representar una gran variedad de datos) [32, pp. 1].

La enorme producción de datos de las últimas décadas ha generado la posibilidad tanto de representar como de aprender de los grafos. Para esto último es posible aplicar técnicas tradicionales de aprendizaje automático no-profundo (i.e. *shallow machine learning*) así como métodos más recientes de DL, pero aplicado al caso particular de los grafos (i.e. las GNN). Los RS, como tarea de ML, no son la excepción, resultando de interés la representación de las interacciones usuario-ítem y los atributos tanto de usuarios como ítems en grafos, sean de la clase que sean.

Los grafos no escapan a los problemas que distintas técnicas de ML pueden resolver, ya sean tareas supervisadas, no supervisadas o semi-supervisadas [32, pp. 4]. De esta forma es posible resolver problemas típicos a nivel de grafo, como clasificación de nodos, predicción de relaciones, *clustering* y detección de comunidades; o a nivel de conjuntos de grafos, como la clasificación, regresión o *clustering* de grafos.

Para poder aplicar métodos tradicionales de ML sobre grafos se presenta de forma usual la necesidad de completar determinadas tareas que hacen más complejo el procesamiento de los datos, a la vez que impactan en el desempeño de los modelos entrenados. La extracción de *features* manuales es un caso paradigmático, resultando de forma habitual una tarea exhaustiva, capaz de consumir una parte importante del tiempo empleado en la resolución del problema. Al igual que para con otras estructuras de datos, como las imágenes o el audio, el DL también revolucionó la forma en que es posible aprender de datos en grafos. En vez de extraer *features* de forma manual, se busca aprender representaciones que codifiquen la información estructural del grafo [32, pp. 27]. Aquí es cuando hacen aparición las GNN.

En este capítulo se presentan dos formas distintas de modelar las redes neuronales sobre grafos: el modelo de *Neural Message Passing* y las GNN vistas como convolucio-

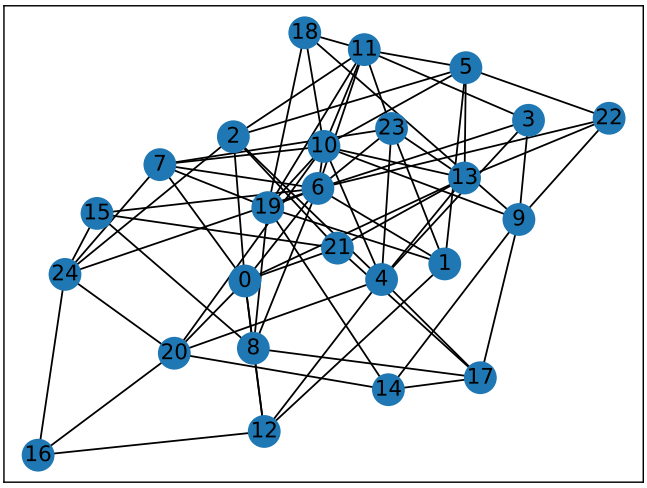

Figura 4.1: Se representan las relaciones de amistad en una red social. Una arista une a dos nodos (personas) si son amigas.

nes sobre grafos. La primera resulta más adecuada para introducir los modelos básicos por su mayor nivel intuitivo, mientras que la segunda resulta de interés por su mayor nivel de formalidad y su origen en el campo del *Graph Signal Processing* (GSP).

4.1. Ganando Intuición: el Modelo *Neural Message Passing*

El primer desafío a enfrentar para las GNN es que los métodos usuales de procesar los datos y entrenar modelos de DL no aplican al caso en que se cuenta con grafos [32, pp. 47]. No se tienen matrices de enteros (e.g. imágenes) o secuencias (e.g. texto). Por lo tanto se necesita de una nueva arquitectura de redes neuronales para lidiar con los datos en grafos.

Una forma conveniente, y de mayor intuición, de abordar el problema de las GNN es comenzar por el modelo conocido como Neural Message Passing (NMP). La clave para entender el modelo NMP es ver a las GNN como una arquitectura que usa una forma de pasaje de mensajes en la que mensajes en forma de vectores son intercambiados entre los nodos y actualizados usando redes neuronales [32, pp. 47].

En una GNN cada nodo $u \in \mathcal{V}$ tiene su correspondiente vector de *embedding oculto* $\mathbf{h}_u^{(k)}$ que es actualizado en cada iteración de acuerdo a la información agregada del vecindario de u, $\mathcal{N}(u)$. Lo anterior se puede expresar como:

$$\begin{aligned} \mathbf{h}_u^{(k+1)} = UPDATE^{(k)} \left(\mathbf{h}_u^{(k)}, AGGREGATE^{(k)}(\mathbf{h}_v^{(k)}, \forall v \in \mathcal{N}(u) \right) = \\ = UPDATE^{(k)} \left(\mathbf{h}_u^{(k)}, m_{\mathcal{N}(u)}^{(k)} \right). \end{aligned} \tag{4.1}$$

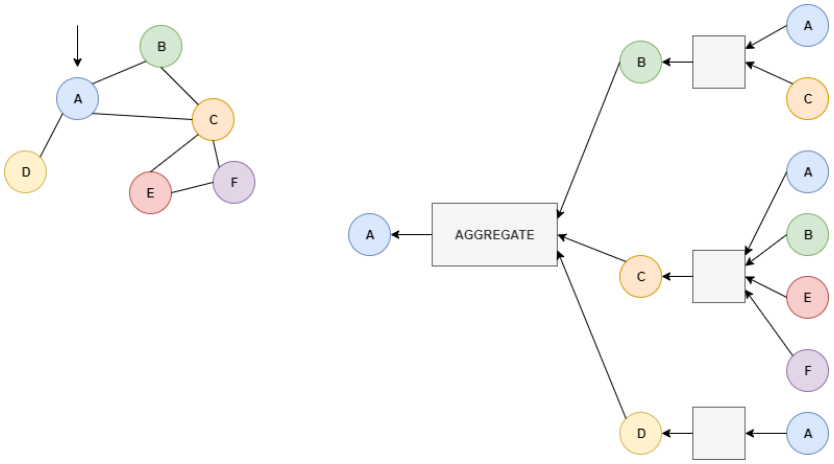

Figura 4.2: Ejemplo de operación de agregación sobre nodo A.

Donde UPDATE y AGGREGATE son funciones diferenciables arbitrarias (i.e. redes neuronales) y $m_{\mathcal{N}(u)}$ es el mensaje que es agregado a partir del vecindario de u $\mathcal{N}(u)$. El uso de las k es para indicar las distintas capas, o iteraciones, de la GNN. En la Figura 4.2 se muestra cómo se agrega al nodo A su propio vecindario (B, C y D), mientras que sus vecinos a distancia 1 agregan a su vez los nodos a un salto de separación, por lo que en este caso se tienen dos iteraciones o capas en la GNN.

En cada capa k la función AGGREGATE toma como entrada los *embeddings* de los nodos vecinos de u y genera un mensaje $m^{(k)}_{\mathcal{N}(u)}$ en base a la información agregada de estos. Luego la función UPDATE combina el mensaje $m^{(k)}_{\mathcal{N}(u)}$ con el *embedding* de la capa anterior $\mathbf{h}^{(k-1)}_u$ del nodo u para generar su *embedding* actualizado $\mathbf{h}^{(k)}_u$. El *embedding* de la capa $k = 0$ es la entrada, o sea los datos, cumpliéndose así que $\mathbf{h}^{(0)}_u = \mathbf{x}_u, \forall \mathbf{u} \in \mathcal{V}$. Luego de K iteraciones (capas) del pasaje de mensajes se puede usar la salida de la última capa para definir el *embedding* final de cada nodo:

$$\mathbf{z}_u = \mathbf{h}^{(K)}_u, \forall \mathbf{u} \in \mathcal{V} \tag{4.2}$$

A medida que se itera en la red, cada nodo agrega información de los nodos vecinos. En la primera iteración ($k = 1$) se agrega información de los nodos a una distancia de 1, con $k = 2$ los que están a dos saltos y así sucesivamente hasta $k = L$. La información se codifica en los *embeddings* de cada nodo, ya sea esta información de carácter estructural (i.e. cómo se interconectan los nodos) o a nivel de *features* (i.e. información de cada nodo vecino en sí).

¿Pero qué tipo de funciones son las de UPDATE y AGGREGATE? Hasta el momento solo se ha dado una definición intuitiva y más abstracta del modelo NMP. Una forma básica de representar una GNN es como sigue:

$$\mathbf{h}^{(k)}_u = \sigma\left(\mathbf{W}^{(k)}_{self}\mathbf{h}^{(k-1)}_u + \mathbf{W}^{(k)}_{neigh}\sum_{v\in\mathcal{N}(u)}\mathbf{h}^{(k-1)}_v + b^{(k)}\right), \tag{4.3}$$

donde $\mathbf{W}^{(k)}_{self}, \mathbf{W}^{(k)}_{neigh} \in \mathbb{R}^{d^{(k)}\times d^{(k-1)}}$ son parámetros aprendibles, σ una función no-lineal (e.g. ReLU, LeakyReLU, etc.) y $\mathbf{b}^{(k)} \in \mathbb{R}^{d^{(k)}}$ el sesgo. Primero se suman los mensajes de los nodos vecinos y luego se combinan con el *embedding* del nodo en la iteración anterior. Por último se aplica a esta suma una no-linearidad.

La definición anterior es a nivel de nodo, con el fin de evidenciar de manera más simple las operaciones realizadas en cada nodo del grafo. También es posible dar una definición a nivel de grafo de la siguiente forma:

$$\mathbf{H}^{(k)} = \sigma\left(\mathbf{A}\mathbf{H}^{(k-1)}\mathbf{W}^{(k)}_{neigh} + \mathbf{H}^{(k-1)}\mathbf{W}^{(k)}_{self}\right), \tag{4.4}$$

donde $\mathbf{H}^{(k)} \in \mathbb{R}^{|V|\times d}$ es la matriz de representaciones de nodos en la capa k (cada nodo es una fila en la matriz) y $\mathbf{A}$ es la matriz de adyacencias del grafo.

A la hora de realizar la operación de agregación, puede ser importante considerar la normalización de los *embeddings* para evitar inestabilidades numéricas capaces de afectar el desempeño de la red y su entrenamiento [32, pp. 53]. También en la operación de agregación de vecindario es posible aplicar el mecanismo de atención, popularizado por las arquitecturas *transformers* en el área del PLN. La idea detrás del mecanismo de atención en las GNN es asignar a cada vecino cierto peso, el cual definirá la importancia del nodo en el proceso de agregación. Por lo tanto, se define el mensaje intercambiado entre u y su vecindario como:

$$\mathbf{m}_{\mathcal{N}(u)} = \sum_{v\in\mathcal{N}(u)} \alpha_{u,v}\mathbf{h}_v, \tag{4.5}$$

donde $\alpha_{u,v}$ denota la atención sobre el nodo $v \in \mathcal{N}(u)$ cuando se agrega información en el nodo u. Los coeficientes de atención son aprendibles y la forma básica de calcularlos es de la siguiente manera:

$$\alpha_{u,v} = \frac{exp(\mathbf{a}^T[\mathbf{W}\mathbf{h}_u \oplus \mathbf{W}\mathbf{h}_v])}{\sum_{v'\in\mathcal{N}(u)} exp(a^T[\mathbf{W}\mathbf{h}_u \oplus \mathbf{W}\mathbf{h}_{v'}])}, \tag{4.6}$$

donde $\mathbf{a}$, el vector de atención, y la matriz $\mathbf{W}$, son aprendibles y $\oplus$ es la operación de concatenación. Existen variantes de la anterior, como el modelo de atención bilineal o utilizando Multi-Layer Perceptron (MLP) con salidas escalares entre $\mathbf{h}_u$ y $\mathbf{h}_v$. Además es posible, al igual que en las arquitecturas *transformer*, usar múltiples *cabezas de atención*, donde se computan K pesos de atención $\alpha_{u,v,k}$ es capas de atención independientes. Agregar el mecanismo de atención a una GNN puede fortalecer su capacidad de representación.

Los modelos presentados anteriormente se ajustan a los casos en que se cuenta con grafos simples (i.e. homogéneos). Cuando los grafos son heterogéneos (e.g. un Knowledge Graph) se debe modificar la arquitectura de la GNN para tener en cuenta las múltiples relaciones y entidades presentes. Es el caso de la Relational Graph Convolutional Networks (RGCN). Ahora se tiene más de un tipo de relación entre nodos, para lo que se debe contar con una matriz de transformación para cada relación:

$$\mathbf{m}_{\mathcal{N}(u)} = \sum_{\tau\in\mathbb{R}} \sum_{v\in\mathcal{N}_\tau(u)} \frac{\mathbf{W}_\tau\mathbf{h}_v}{f_n(\mathcal{N}(u),\mathcal{N}(v))}, \tag{4.7}$$

donde f_n es una función de normalización que puede depender del vecindario de u y del de v. La información se agrega para cada tipo de relación.

4.2. Grafos y Convoluciones

Los grafos son utilizados como descripciones matemáticas de topologías de red, mientras que los datos pueden ser vistos como señales sobre estos grafos. El procesamiento de los datos teniendo en cuenta la topología de la red ha sido el objetivo del campo del GSP. Esta área del procesamiento de señales ha extendido a los grafos los

conceptos de Transformada de Fourier, convoluciones y filtrado de señales, teniendo en cuenta siempre la topología del grafo subyacente [31]. El advenimiento del DL también ha revolucionado el mundo del GSP.

Las *Graph Convolutional Neural Networks* (GCNN), un tipo particular de GNN, se construyen en base a convoluciones sobre un grafo, para así de forma efectiva incorporar la estructura de este al proceso de aprendizaje. Las GCNN consisten en concatenaciones de capas, en donde en cada capa se aplica una convolución seguida de una función no lineal. Estas redes tienen propiedades interesantes, como la equivarianza a permutaciones o la estabilidad frente a perturbaciones. La primera es capaz de explotar simetrías topológicas, mientras que la segunda expresa la robustez frente a pequeños cambios en la estructura del grafo. Estas propiedades permiten a las GCNN escalar a grafos más grandes y transferir parte de lo aprendido a otros (similares) escenarios [31].

Se define un grafo $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, donde $\mathcal{V} = \{1, ..., N\}$ denota el conjunto de nodos, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ el conjunto de aristas que conectan los nodos y $\mathcal{W} : \mathcal{E} \longrightarrow \mathbb{R}^+$ una función que representa los pesos en cada arista. El vecindario del nodo $i \in \mathcal{V}$ es el conjunto de nodos conectados por una arista y se define como $\mathcal{N}_i = \{j \in \mathcal{V} : (j, i) \in \mathcal{E}\}$. Además se define una matriz simétrica $\mathbf{S}$ de dimensiones $N \times N$ conocida como *Graph Shift Operator* (GSO). $\mathbf{S}$ satisface la condición:

$$[\mathbf{S}]_{ij} = s_{ij} = 0 \; si \; (j, i) \notin \mathcal{E} \; para \; j \neq i \tag{4.8}$$

(i.e. el GSO tiene un elemento 0 si dos nodos no están conectados). Los GSO más usuales son la matriz de Adyacencias y el Laplaciano.

Los datos son representados como señales $\mathbf{x} \in \mathbb{R}^N$ sobre el grafo $\mathcal{G}$, donde la entrada $[x]_i = x_i$ es el dato del nodo i. La señal $\mathbf{x}$ puede ser propagada sobre los nodos del grafo usando $\mathbf{S}$ tal que la entrada i de $\mathbf{Sx}$ es:

$$[\mathbf{Sx}]_i = \sum_{j=1}^{N} [S]_{ij}[x]_j = \sum_{j \in N_i} s_{ij} x_j. \tag{4.9}$$

La salida $\mathbf{Sx}$ es otra señal sobre el grafo donde el valor en cada nodo es la suma de los valores de $\mathbf{x}$ en los nodos vecinos. Teniendo en cuenta lo anterior, se define la convolución sobre grafos como una operación de propagación y suma. Dado un conjunto de parámetros $\mathbf{h} = (h_0, ..., h_k)$, se calcula la convolución sobre un grafo como:

$$\mathbf{H(S)x} = \sum_{k=0}^{K} h_k \mathbf{S}^k \mathbf{x}. \tag{4.10}$$

La operación anterior combina de forma lineal la información contenida en diferentes vecindarios. La señal $\mathbf{S}^k\mathbf{x}$ resume la información de un vecindario a k saltos, a la vez que h_k la pondera. La convolución sobre el grafo filtra la señal $\mathbf{x}$ con un filtro de tipo Finite Impulse Response (FIR) $\mathbf{H(S)}$, siendo h_k los pesos del filtro o *filter-taps*.

Graph Neural Networks

Para aprender de datos en grafos se debe calcular un mapa de representaciones $\Phi(\cdot)$ entre los datos $\mathbf{x}$ y una representación objetivo $y = \Phi(x; S)$ que aprovecha la estructura de la red. La imagen de Φ es un espacio de representaciones, de entre las cuales se deberá elegir la que se considere mejor dados $\mathbf{S}$ y $\mathbf{x}$. Un ejemplo de un mapa de representaciones es la convolución sobre un grafo como $\Phi(x; S, \mathcal{H}) = \mathbf{H(S)x}$, donde $\mathcal{H} = \{h\}$ es el conjunto de los coeficientes de los filtros que caracterizan su espacio de representaciones. Para aprender una representación, es necesario definir una función de costo $J(\cdot)$ y un conjunto de entrenamiento $\mathcal{T} = \{x_1, ..., x_T\}$ de $|\mathcal{T}|$ muestras. La representación aprendida es entonces $\Phi(x; S, H^*)$, siendo:

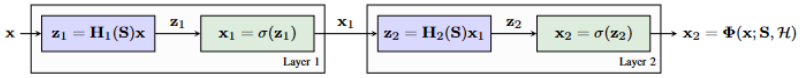

Figura 4.3: Una GNN de dos capas. En violeta los filtros lineales, en verde la operación no-lineal. Imagen tomada de [31].

$$H^* = \arg\min_H \frac{1}{|\mathcal{T}|} \sum_{x \in \mathcal{T}} J(\Phi(x; S, H)). \tag{4.11}$$

La función de costo puede ser la Mean-Squared Error (MSE) en el caso de tratarse de un problema de regresión o la Cross-Entropy Loss en caso de clasificación. El problema consiste en encontrar el conjunto de *filter-taps* que mejor se ajuste a los datos $\mathbf{x}$ con respecto a $J(\cdot)$. K es un hiperparámetro a definir (y eventualmente ajustar) para el entrenamiento.

Es necesario tener en cuenta que la convolución solamente es capaz de encontrar mapeos lineales. Para incrementar la *expresividad* del modelo entrenado (i.e. la capacidad de encontrar relaciones no lineales entre las $\mathbf{x}$ y las $\mathbf{y}$), es necesario introducir una no-linearidad. Lo anterior lleva al concepto de *graph perceptron*, el cual aplica una no-linearidad $\sigma(\cdot)$ a la salida de una convolución sobre el grafo $\mathbf{H(S)x}$:

$$\Phi(x; S, \mathcal{H}) = \sigma(\mathbf{H}(\mathbf{S})\mathbf{x}), \tag{4.12}$$

donde $\mathcal{H} = \{h\}$ contiene los coeficientes de los filtros. La función no lineal $\sigma(\cdot)$ puede ser la función ReLU o alguna de sus variantes (e.g. Leaky ReLU, ELU, Sigmoide, entre otras). En la Figura 4.3 se muestra una GNN de dos capas.

Una GNN será entonces la concatenación de un número L de capas, con cada una siendo una combinación de una convolución (operación lineal) con una operación no-lineal, donde en la capa l se calcula como:

$$x_l = \sigma(\mathbf{H}_l(\mathbf{S})\mathbf{x}_{l-1});\ l = 1, ..., L. \tag{4.13}$$

La combinación de capas en cascada tiene como entrada los datos $x_0 = x$ en la primera capa y como salida $x_L = \Phi(x; S, H)$.

También es posible en las GNN utilizar representaciones *multi-feature* en cada capa a modo de aumentar la capacidad de representación de la red. Se considera una señal de entrada con F_{l-1} *features* $x_1^1, ..., x_{l-1}^{F_{l-1}}$ en la capa l. Cada *feature* x_{l-1}^g, con $g = 1, ..., F_{l-1}$ es procesado en paralelo por F_l filtros diferentes para dar como salida F_l *features* convolucionales:

$$u_l^{fg} = H_l^{fg}(S)x_{l-1}^g = \textstyle\sum_{k=0}^K h_{lk}^{fg} S^k x_{l-1}^g; f = 1, ..., F_l. \tag{4.14}$$

Los *features* convolucionales son luego compactados a lo largo del índice g de la entrada para obtener *features* agregados:

$$u_l^f = \textstyle\sum_{g=1}^{F_{l-1}} H_l^{fg}(S)x_{l-1}^g; f = 1, ..., F_l. \tag{4.15}$$

Estos features agregados son finalmente pasados por una función no-lineal para completar la salida de la capa l:

$$u_l^f = \sigma(u_l^f); f = 1, ..., F_l. \tag{4.16}$$

Los $F_l \times F_{l-1}$ filtros aprendidos aumentan el poder de representación de la red neuronal y explotan la estabilidad de la convolución como operación así como también la topología del grafo. A la entrada de la GNN se tienen los datos $x_0 = x$ y a la

salida una representación con F_l *features* $[x_L^1, ..., x_L^{F_L}] = \Phi(x; S, \mathcal{H})$ donde el conjunto $\mathcal{H} = \{\mathbf{h}_l^{fg}\}_{lfg}$ contiene los *filter-taps* de todas las capas.

Para un $\mathbf{S}$ dado y los hiper-parámetros L, F_l y K el espacio de representaciones $\Phi(x; S, \mathcal{H})$ es caracterizado por los pesos de los filtros en cada capa. Estos pesos son luego aprendidos mediante un método de optimización (e.g. SGD) donde los gradientes se calculan con un algoritmo (e.g. *Backpropagation*), al igual que se hace con las redes neuronales para datos euclideanos.

4.2.1. Invarianza y Equivarianza a Permutaciones

Los grafos son estructuras sin un orden establecido. A diferencia de otras estructuras de datos donde importa el orden (e.g. las imágenes, donde cada píxel tiene su lugar específico), en los grafos un nodo puede estar en cualquier lugar, siempre y cuando se conserve la topología de la red. Es por esto que es importante que las GNN cumplan las propiedades de Invarianza y Equivariancia a Permutaciones [32, pp. 47].

Una forma intuitiva y aparentemente viable de definir una red neuronal sobre un grafo podría ser tomar la matriz de adyacencia como entrada a un MLP. El problema con lo anterior reside en que los resultados dependerían del orden en que se ubican los nodos en la matriz de adyacencia, por lo que dicho modelo no sería invariante a permutaciones. Una función f que tome una matriz de adyacencia $\mathbf{A}$ como entrada debe cumplir con una de las siguientes propiedades:

Invarianza a Permutaciones:

$$f(\mathbf{PAP^T}) = f(\mathbf{A}). \tag{4.17}$$

Equivarianza a Permutaciones:

$$f(\mathbf{PAP^T}) = \mathbf{P}f(\mathbf{A}). \tag{4.18}$$

donde $\mathbf{P}$ es una matriz de permutaciones. La invarianza a permutaciones significa que la función no depende del orden de las filas y columnas de $\mathbf{A}$, mientras que la equivarianza a permutaciones indica que la salida de $\mathbf{f}$ es permutada de forma consistente con las permutaciones de $\mathbf{A}$. A modo de ejemplo, la primera propiedad podría adaptarse bien al problema de clasificación de grafos, mientras que la segunda al de clasificación de nodos en un grafo.

4.2.2. Resumen del Capítulo

En este capítulo se presentaron dos formas distintas de modelar y entender las GNN, ambas con sus ventajas y desventajas. El modelo NMP es más rico en intuición y funciona como una buena analogía de las CNN clásicas usadas en imágenes. Por otro lado, el modelo surgido del área del GSP logra una mayor formalidad teórica. Presentar ambos modelos resulta clave de cara al próximo capítulo, donde se detallan algunos de los RS del estado del arte para GNN modeladas como NMP sobre grafos de conocimiento para luego presentar los RS basados en GNN y grafos de correlación y entrenados con la biblioteca Alegnn.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 5

Estado del Arte en RS Basados en Grafos

El presente capítulo comienza con una definición de un Knowledge Graph (KG), para luego continuar con la descripción del modelo de RS basado KG Knowledge Graph Attention Network for Recommendation (KGAT) [39]. Este modelo se encontraba dentro del estado del arte al momento de ser explorado y tiene como referencia el modelo NMP de GNN. Además de este modelo de terceros, se describen los distintos RS implementados basados en GNN modeladas como convoluciones sobre grafos. Estos últimos tienen la particularidad de haber sido modelados como un problema de clasificación y para todos los ítems y usuarios, siendo esta una forma de abordar el problema no seguida en la bibliografía consultada y por lo tanto considerada de interés.

Knowledge Graphs

Antes que nada un KG es un grafo heterogéneo, es decir un grafo donde tanto nodos como relaciones pueden ser de distinta clase. A modo de ejemplo, en un grafo homogéneo todos los nodos pueden ser películas, mientras que en uno heterogéneo puede haber nodos películas, actores, directores, etcétera. Pero un KG no es solo un grafo heterogéneo, sino también una base de conocimiento representada mediante una estructura de grafo [9]. Los KG se utilizan para representar datos de naturaleza más abstracta. Un grafo de conocimiento está conformado por *entidades* (nodos) y *relaciones* entre ellas (aristas). Al conjunto de dos entidades unidas por una arista se le llama *tripla*. Se define un KG como:

$$\mathcal{G} = \{(h, r, t)\}, \tag{5.1}$$

donde $h \in \mathcal{E}$, $r \in \mathcal{R}$ y $t \in \mathcal{E}$ representan la cabeza, relación y cola de las triplas que conforman el KG, mientras que $\mathcal{E}$ y $\mathcal{R}$ son los conjuntos de entidades y relaciones, respectivamente. En la Figura 5.1 se muestra un KG creado a partir de algunas entradas de la Tabla 5.1. Observar cómo los nodos representan distintas clases de entidades (e.g. películas, actores, género) y las aristas (e.g. origen, género, es_director) indican qué tipo de relaciones se tienen entre ellas. En este caso se tienen nueve entidades de cinco tipos distintos y ocho aristas pertenecientes a cuatro clases de relaciones, conformando un total de ocho triplas.

Los KG resultan de interés para el problema de los sistemas de recomendación dada la capacidad (y facilidad) que tienen para integrar distintas fuentes de datos,

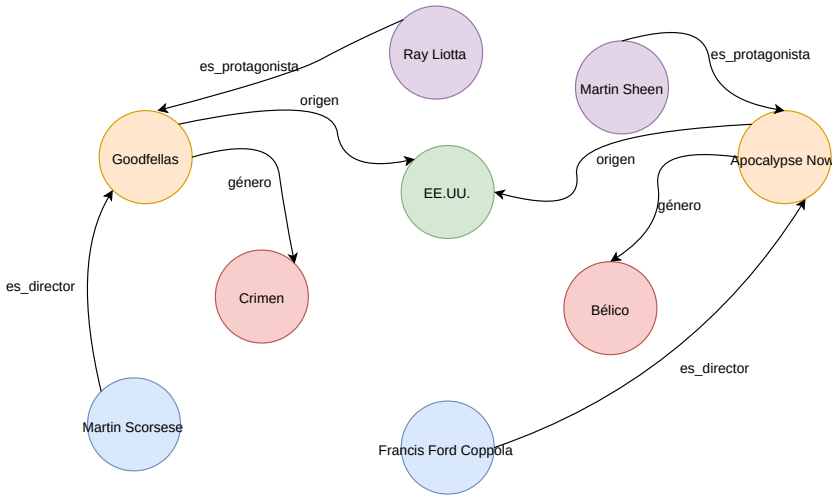

Figura 5.1: Ejemplo de KG creado a partir de Tabla 5.1

Película	Director	Género	Año	País	Protagonista
Goodfellas	Martin Scorsese	Crimen	1990	EE.UU.	Ray Liotta
Apocalipsis Now	Francis Ford Coppola	Bélica	1979	EE.UU.	Martin Sheen
Barry Lyndon	Stanley Kubrick	Drama	1975	R.U.	Ryan O'Neal
Ratatouille	Brad Bird	Animada	2007	EE.UU.	Patton Oswalt

Tabla 5.1: Ejemplo de ítems (películas) con sus atributos.

tanto para usuarios como para ítems. Es principalmente por esta razón que se decidió explorar métodos que usen DL sobre KG.

5.1. Sistemas de Recomendación Basados en Knowledge Graphs

En la *survey* [37] se recopilan distintos métodos para elaborar RS basados en KG. Se los clasifica en tres grandes categorías: basados en *embeddings*, basados en *paths* y los métodos unificados. Los primeros explotan la representación semántica de los ítems/usuarios en el KG, mientras que los segundos hacen lo propio con la conectividad semántica del grafo. Los métodos unificados aprovechan ambas características. Se basan en la idea de *embedding propagation* mediante el uso de Graph Neural Networks (GNN). Debido a que se apoyan en arquitecturas de GNN, se decidió explorar estos métodos, en particular el modelo KGAT [39].

A la hora de correr experimentos se decidió hacer uso de KGAT, ya que se consideró que la exposición del problema era más inteligible, se hacía uso del mecanismo de atención (ampliamente utilizado en el estado del arte de campos como, por ejemplo, el Procesamiento de Lenguaje Natural con las arquitecturas *transformer*) y el repositorio [40] con el código implementado está disponible y documentado.

5.1.1. KGAT: Knowledge Graph Attention Network for Recommendation

El modelo propuesto en [39] explota la conectividad de alto orden del KG. Hace uso de *embedding propagation* para refinar la representación de los nodos y del mecanismo de atención para diferenciar la importancia entre los nodos vecinos (ítems, usuarios o atributos). Las ventajas anteriores presentadas por el modelo introducen un compromiso entre la explotación de la conectividad de alto orden y el costo computacional, así como la necesidad de pesar la contribución de las relaciones y entidades de alto orden. Estos desafíos son enfrentados por el modelo KGAT haciendo uso de la arquitectura GNN y de dos decisiones de diseño:

1. Propagación recursiva de los *embeddings* de las entidades y sus vecinas.
2. Agregación basada en la atención para aprender el peso de cada vecino durante la propagación y así medir la importancia de la conectividad de alto orden.

Grafo Bipartito Usuario-Ítem

En KGAT las interacciones se almacenan en un grafo. Este grafo es de tipo bipartito, ya que tiene nodos de dos clases: usuarios e ítems. Se define el grafo como un conjunto de triplas (cabeza, relación y cola) $\mathcal{G}_1 = \{(u, y_{ui}, i) \mid u \in \mathcal{U}, i \in \mathcal{I}\}$, donde $\mathcal{U}$ es el conjunto de usuarios e $\mathcal{I}$ el conjunto de ítems. Si hubo interacción entre un usuario y un ítem, $y_{ui} = 1$, de lo contrario, $y_{ui} = 0$.

Armado del Knowledge Graph

El KG almacena la información complementaria, o *side-information* (i.e. atributos de los ítems y cualquier fuente externa de información considerada relevante). Es una forma de integrar información adicional, que puede ser recolectada de otras fuentes, como datos demográficos, de redes sociales u otras bases de conocimiento. Se define el KG $\mathcal{G}_2 = \{(h, r, t) \mid h, t \in \mathcal{E}, r \in \mathbb{R}\}$, donde $\mathcal{E}$ es el conjunto de entidades y $\mathbb{R}$ el conjunto de relaciones. El grafo $\mathcal{G}_2$ es un conjunto de triplas (h, r, t), donde h es la cabeza (head), r la relación y t la cola (tail). A modo de ejemplo, una tripla (*Guerra y Paz*, libro.autor, *Tolstoi, León*) indica que León Tolstoi es el autor de la novela *Guerra y Paz*.

Collaborative Knowledge Graphs

Este grafo se construye ensamblando $\mathcal{G}_1$ y $\mathcal{G}_2$ en el grafo unificado (CKG) $\mathcal{G} = \{(h, r, t) \mid h, t \in \mathcal{E}', r \in \mathcal{R}'\}$, donde $\mathcal{E}' = \mathcal{E} \cup \mathcal{U}$ y $\mathcal{R}' = \mathbb{R} \cup \mathcal{R}_{\mathcal{I}}$, con $\mathcal{R}_{\mathcal{I}}$ la relación extra definida como las interacciones usuario-ítem. Este CKG será la entrada al modelo, mientras que la salida será la probabilidad $\hat{y}_{ui}$ de que el usuario u interactúe con el ítem i.

El Modelo KGAT

En la Figura 5.2 se muestra un diagrama del modelo KGAT. El modelo KGAT tiene tres componentes principales:

- Capa de *embedding*. Transforma cada nodo del CKG en un vector de determinada dimensión.
- Capa de propagación de *embeddings* con mecanismo de atención. Propaga los *embeddings*, actualizando su representación, a la vez que aplica el mecanismo de atención para aprender el peso de un nodo vecino.

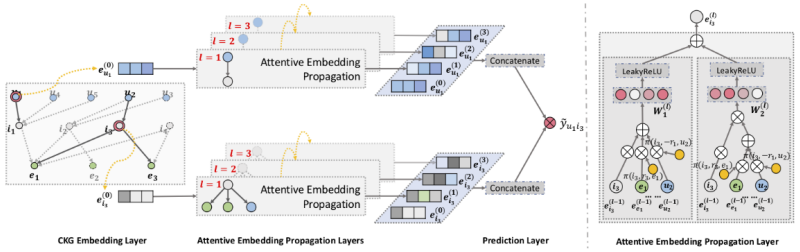

Figura 5.2: El modelo KGAT. Imagen tomada de [39].

- Capa de predicción. Agrega las representaciones de usuarios e ítems a través de todas las capas y devuelve un *score* que indica la probabilidad de que el usuario u interactúe con el ítem i.

Capa de Embedding

La capa de *embedding* tiene como fin generar representaciones vectoriales de las entidades y relaciones en el CKG. Para ello, en el modelo KGAT se emplea el método TransR [41]. Este método aprende los *embeddings* de cada entidad y relación optimizando el principio de traducción $\mathrm{e}_h^r + \mathrm{e}_r \approx \mathrm{e}_t^r$, donde e_h, $\mathrm{e}_t \in \mathbb{R}^d$ y $\mathrm{e}_r \in \mathbb{R}^k$ son los *embeddings* para h, t y r, respectivamente, y e_h^r, e_t^r son las versiones proyectadas en el espacio de las relaciones r. Entonces, para cada tripla (h, r, t) se calcula el *score* de energía como:

$$g(h, r, t) = \| \mathrm{W}_r \mathrm{e}_h + \mathrm{e}_r - \mathrm{W}_r \mathrm{e}_t \|_2^2, \tag{5.2}$$

donde $\mathrm{W}_r \in \mathbb{R}^{k \times d}$ es la matriz de transformación de la relación r, que proyecta las entidades del espacio d-dimensional al k-dimensional. El entrenamiento de TransR considera el orden relativo entre triplas válidas ($\in$ KG) y no-válidas ($\notin$ KG), incentivando su discriminación mediante la siguiente función de pérdida:

$$\mathcal{L}_{KG} = \sum_{(h,r,t,t') \in \mathcal{T}} - \ln \sigma(g(h, r, t') - g(h, r, t)), \tag{5.3}$$

donde $\mathcal{T} = \{(h, r, t, t') | (h, r, t) \in \mathcal{G}, (h, r, t') \notin \mathcal{G}\}$ y (h, r, t') es una tripla no existente construida reemplazando de forma aleatoria una entidad válida t por una no-válida t'. $\sigma(.)$ es la función sigmoide. Se trata de que el modelo aprenda a discriminar entre triplas válidas y no válidas. Esta estretagia es conocida como muestreo negativo y es ampliamente utilizada en métodos de cálculo de *embeddings* de palabras, donde una muestra positiva es una palabra que tiene co-ocurrencia mientas que una muestra negativa es una que no (i.e. no suele estar en el contexto de la palabra a computar el vector de *embedding*) y se elige de forma aleatoria.

Capa de Propagación de Embeddings con Mecanismo de Atención

Esta capa se asienta sobre la arquitectura de GNN para propagar de forma recursiva los *embeddings*, a la vez que hace uso del mecanismo de atención para discriminar la importancia que se le da a determinadas entidades. A continuación se describe una capa única, que consiste en tres etapas: propagación de la información, atención y agregación de información.

- **Propagación de la información**. Se considera $\mathcal{N}_h = \{(h, r, t) | (h, r, t) \in \mathcal{G}\}$ como el conjunto de triplas donde h es la entidad *head*. $\mathcal{N}_h$ es llamada *ego-*

network. Para caracterizar la conectividad de primer orden de la entidad h, se calcula la combinación lineal de la *ego-network*:

$$\mathrm{e}_{\mathcal{N}_h} = \sum_{(h,r,t)\in\mathcal{N}_h} \pi(h,r,t)\mathrm{e}_t, \tag{5.4}$$

donde $\pi(h,r,t)$ indica cuánta información se propaga de h a t a través de r.

- **Atención**. $\pi(h,r,t)$ se implementa mediante el mecanismo de atención, formulado como sigue:

$$\pi(h,r,t) = (\mathrm{W}_r\mathrm{e}_t)^T \tanh\left(\mathrm{W}_r\mathrm{e}_h + \mathrm{e}_r\right), \tag{5.5}$$

donde tanh es la función de activación. De esta manera el *score* de atención depende de la distancia entre e_h y e_t en el espacio de relaciones (e.g. propaga más información para entidades cercanas). Luego se normaliza $\pi(h,r,t)$ a través de todas las triplas conectadas con h con la función softmax. La consecuencia de lo anterior es que el *score* de atención final es capaz de indicar qué nodos vecinos deberían ser tenidos más en cuenta.

- **Agregación de información**. Consiste en agregar la representación de la entidad e_h con su *ego-network* $\mathrm{e}_{\mathcal{N}_h}$ en una nueva representación de la entidad h: $\mathrm{e}_h^{(1)} = f(\mathrm{e}_h, \mathrm{e}_{\mathcal{N}_h})$. Se usan tres clases de funciones de agregación f:

  1. **Agregación tipo GCN**. Suma las dos representaciones y aplica una transformación no-lineal:

$$f_{GCN} = LeakyReLU(\mathrm{W}(\mathrm{e}_h + \mathrm{e}_{\mathcal{N}_h})), \tag{5.6}$$

  donde $\mathrm{W}\in \mathbb{R}^{d'\times d}$ es la matriz de pesos (entrenables) y d' es el tamaño de la transformación.

  2. **Agregación tipo GraphSage**. Concatena dos representaciones y las hace pasar por una transformación no-lineal:

$$f_{GraphSage} = LeakyReLU(\mathrm{W}(\mathrm{e}_h||\mathrm{e}_{\mathcal{N}_h})), \tag{5.7}$$

  donde $||$ es la operación de concatenación.

  3. **Agregación tipo Bi-Interacción**. Diseñada para el modelo KGAT para considerar dos tipos de interacciones entre e_h y $\mathrm{e}_{\mathcal{N}_h}$:

$$\begin{aligned} f_{Bi-Interaction} = &LeakyReLU(\mathrm{W}_1(\mathrm{e}_h + \mathrm{e}_{\mathcal{N}_h})) \\ &+LeakyReLU(\mathrm{W}_2(\mathrm{e}_h \odot \mathrm{e}_{\mathcal{N}_h})), \end{aligned} \tag{5.8}$$

  donde $\mathrm{W}_1, \mathrm{W}_2 \in \mathbb{R}^{d'\times d}$ son matrices de pesos (entrenables) y $\odot$ es el producto elemento a elemento.

- **Propagación de alto orden**. Se agregan capas de propagación para explotar la conectividad de alto orden. En l pasos se representa la entidad como:

$$\mathrm{e}_h^{(l)} = f(\mathrm{e}_h^{(l-1)}, \mathrm{e}_{\mathcal{N}_h}^{(l-1)}), \tag{5.9}$$

donde la información propagada a través de la *l-ego-network* para la entidad h es definida como:

$$\mathrm{e}_{\mathcal{N}_h}^{(l-1)} = \sum_{(h,r,t)\in\mathcal{N}_h} \pi(h,r,t)\mathrm{e}_t^{(l-1)}, \tag{5.10}$$

donde $\mathrm{e}_t^{(l-1)}$ es la representación de la entidad t generada en los pasos previos de agregación de información.

Capa de Predicción

Luego de L capas se obtienen representaciones para nodos u de usuarios ($\{e_u^{(1)}, ..., e_u^{(L)}\}$) y nodos i de ítems ($\{e_i^{(1)}, ..., e_i^{(L)}\}$). En estos conjuntos se codifica la información de conectividad de alto orden. En el modelo se adopta luego el mecanismo de agregación de capas para concatenar las representaciones en cada paso en un único vector:

$$e_u^* = e_u^{(0)}||...||e_u^{(L)}; e_i^* = e_i^{(0)}||...||e_i^{(L)}, \tag{5.11}$$

donde $\|$ es la operación de concatenación. Con esta operación no solo se enriquece la representación inicial de los *embeddings*, sino que se puede de forma adicional controlar la fuerza de la propagación ajustando L. Finalmente se computa el producto interno de las representaciones de usuario e ítem para predecir un *score*:

$$\hat{y}(u, i) = e_u^{*T} \cdot e_i^*. \tag{5.12}$$

Optimización

Para optimizar el modelo de recomendación se usa la función de pérdida BPR (Bayesian Personalized Ranking), descrita más abajo. Esta asume que a las interacciones observadas (que indican la preferencia del usuario, o sea triplas existentes) deberían de serle asignadas mayores valores de predicción que las no-observadas:

$$\mathcal{L}_{CF} = \sum_{(u,i,j)\in\mathcal{O}} - \ln \sigma(\hat{y}(u, i) - \hat{y}(u, j)), \tag{5.13}$$

donde $\mathcal{O} = \{(u, i, j)|(u, i) \in \mathbb{R}^+, (u, j) \in \mathbb{R}^-\}$ es el conjunto de entrenamiento, $\mathbb{R}^+$ indica las interacciones observadas (positivas) entre el usuario u y el ítem i; $\mathbb{R}^-$ son las interacciones (muestreadas) no-observadas (negativas); $\sigma(.)$ es la función sigmoide.

El modelo aprenderá optimizando la función de pérdida conjunta:

$$\mathcal{L}_{KGAT} = \mathcal{L}_{KG} + \mathcal{L}_{CF} + \lambda||\Theta||_2^2, \tag{5.14}$$

donde $\Theta = \{E, W_r, \forall l \in \mathbb{R}, W_1^{(l)}, W_2^{(l)}, \forall l \in \{1, ..., L\}\}$ es el conjunto de parámetros y $\mathbf{E}$ es la tabla de *embeddings* para todas las relaciones y entidades; λ es el parámetro de regularización L_2. En KGAT se optimiza alternando entre $\mathcal{L}_{CF}$ y $\mathcal{L}_{KG}$ con el optimizador Adam.

Experimentos

Para evaluar el modelo se utilizan tres *datasets*, donde en todos los casos se utilizaron tanto usuarios como ítems con un mínimo de diez interacciones:

1. Amazon-book [2]. Contiene reseñas de productos a la venta en Amazon, con 142,8 millones de reseñas pertenecientes al periodo de mayo 1996 hasta julio de 2014.
2. Last-FM [10]. En este conjunto de datos los ítems son canciones. En particular se tomaron datos dentro del período enero-junio de 2015.
3. Yelp-2018 [27]. Aquí los ítems son negocios como restaurantes y bares.

Para ensamblar los KG para los dos primeros *datasets*, se debió hacer uso de la base de conocimiento Freebase para obtener la *side information* necesaria. En el caso de Yelp-2018 se obtuvo esta información extra directo de los restaurantes. En todos los casos se procesan los conjuntos de datos para descartar entidades y relaciones infrecuentes. Para construir los *datasets* de *train* y *test*, se toma el 80 % de la historia

KGAT-Original	Relaciones	Triplas	Usuarios
Amazon-Book	39	2557746	70679
Last-FM	9	464567	23566
Yelp 2018	42	1853704	45919

Tabla 5.2: Número de relaciones, triplas y usuarios de KGs para el modelo KGAT original.

Amazon-Book		Last-FM		Yelp 2018	
recall	ndcg	recall	ndcg	recall	ndcg
0,1489	0,1006	0,087	0,1325	0,0712	0,0867

Tabla 5.3: Resultados de KGAT en paper original.

del usuario (ítems relevados) para entrenamiento y el 20 % restante para evaluación. Un 10 % del conjunto de *train* se usa para validar hiper-parámetros.

Las métricas utilizadas para la evaluación del modelo son Recall@K y NDCG@K con $K = 20$. Como *baselines* se usan métodos tradicionales y métodos basados en grafos. El modelo se implementa con el *framework* TensorFlow [24], se optimiza con Adam y los parámetros se inicializan con el método Xavier. La validación de hiper-parámetros se hace con *grid-search* y para evitar sobre-ajuste se apela a las técnicas de *dropout* y regularización L_2. Se utiliza la estrategia de *early-stopping*, deteniendo el entrenamiento si la Recall@20 no aumenta para 50 *epochs* sucesivas. Para modelar la conectividad de alto orden se usa $L = 3$ y agregación tipo Bi-Interacción en cada capa.

En la Tabla 5.2 se despliega el número de relaciones, triplas y usuarios para los *datasets* utilizados en el modelo KGAT original. Se puede apreciar cómo los *datasets* del modelo KGAT original son considerablemente grandes, generándose KG mayores.

En la Tabla 5.3 es posible ver los resultados del trabajo original de KGAT, obtenidos para tres conjuntos de datos, para las métricas NDCG y Recall, en ambos casos con $K = 20$. Los resultados, según los autores del trabajo, superan en entre un 5 % y 10 % a los métodos *baselines* empleados para comparar.

5.2. Sistemas de Recomendación Basados en Grafos de Correlación: CorrG-RS

Presentado el modelo KGAT, un sistema de recomendación sobre grafos de conocimiento modelado como una GNN de tipo Neural Message Passing, se continúa con RS basados en GNN modeladas con herramientas del mundo del GSP. Estas dos formas de modelar las GNN fueron descritas en el Capítulo 4. En esta sección se presenta el modelo denominado aquí CorrG-RS: Sistemas de Recomendación Basados en Grafos de Correlación.

A grandes rasgos, el enfoque seguido en trabajos como [35] y [34] consistió en entrenar RS basados en GNN (óptica GSP) variando tanto la arquitectura de las redes utilizadas como el ensamblado del grafo, entre otros parámetros (e.g. hiper-parámetros en entrenamientos). Pero en ambas fuentes citadas se tienen los siguientes puntos clave en común:

1. El problema se enfrenta como uno de *regresión* (i.e. predecir el *rating* que un usuario dará a un ítem con un valor entre, por ejemplo, 1 y 5). Por lo tanto, se está en el caso de que la GNN entrenada devolverá una calificación predicha,

esperando que esté lo más cercana a la conocida para un ítem en el conjunto de prueba. Se trata de un caso donde el *feedback* es **explícito** y el problema a resolver es de compleción de la matriz de *ratings*.

2. Al tomarse como un problema de regresión, la forma de evaluar el desempeño de la red (y comparar entre distintos entrenamientos) es mediante el Error Cuadrático Medio (RMSE por sus siglas en inglés). La mejor red entrenada será la que alcance el menor RMSE. Esta es una forma correcta de evaluar modelos de regresión, pero no se realizan predicciones (i.e. recomendaciones) ni se evalúa a la red entrenada como si fuera un RS.
3. Los entrenamientos se realizan para un solo ítem o usuario. A modo de ejemplo, tomando los ítems como nodos, se intenta predecir qué calificaciones dan los usuarios a determinado ítem en particular (e.g. qué calificaciones se le dará a *Star Wars*).
4. Las redes se entrenan sobre grafos de correlación. Ya sean grafos donde los nodos son usuarios, o grafos donde los nodos son ítems, las aristas serán valores escalares de correlación. En el primer caso serán correlaciones entre usuarios, mientras que en el segundo entre ítems. Más adelante se detalla de forma más minuciosa.

Por lo tanto, luego de revisar esta bibliografía (y otros trabajos cercanos) surgieron las siguientes interrogantes: ¿Cómo hacer para entrenar un RS completo, para todos los ítems/usuarios de mi conjunto de datos? ¿Es posible usar estas redes neuronales para el caso de interés de este trabajo, donde se tiene *feedback* implícito? ¿De qué manera se comportaría este RS entrenado a la hora de realizar recomendaciones? ¿Cuál es la mejor manera de evaluarlo? ¿Cómo afectan las decisiones de diseño y entrenamiento de las GNN involucradas? ¿La incorporación de los atributos de los recursos es positiva?

Se decidió entonces explorar el camino de entrenar un RS con todos los ítems y usuarios, para *feedback* implícito, usando también grafos de correlación y el problema visto como uno de clasificación binaria en vez de regresión (i.e. se intenta predecir si al usuario le gustará el ítem en vez de qué calificación le dará). Al tratarse de un problema de clasificación, distinto al de regresión elegido en los trabajos en [35] y [34], se consideró de interés explorarlo. También se entendió relevante dado que este método sería capaz de abrir la puerta a una mayor riqueza a la hora de evaluar los RS entrenados, al poder contar ahora con mayor número de métricas que el RMSE (o similares), algunas específicas del mundo de los sistemas de recomendación, como las denominadas métricas alternativas en la sección 2.2.2.

A continuación se detalla cómo se implementaron **Sistemas de Recomendación Basados en Grafos de Correlación (CorrG-RS)** con la biblioteca Alegnn. Se presenta el problema a resolver, el pre-procesamiento y la preparación de los datos necesarios previo a entrenar, los distintos entrenamientos realizados y otros experimentos tenidos en cuenta.

5.2.1. Presentación del Problema

El problema a resolver será completar la matriz de *ratings*, o interacciones, ya que, como se detalla más adelante, el *feedback* será implícito (i.e. se tendrá en la celda (i, j) un 1 cuando al usuario i le gustó un recurso y un 0 en caso contrario). Para esto primero se debe ensamblar un grafo de correlación como sigue:

1. Los n **nodos** del grafo son los ítems (en este caso, libros).
2. Una **arista** (i, j) tiene un **peso** proporcional a qué tanta información aporta el porcentaje de avance de lectura (o llegado al caso otro tipo de interacción) del

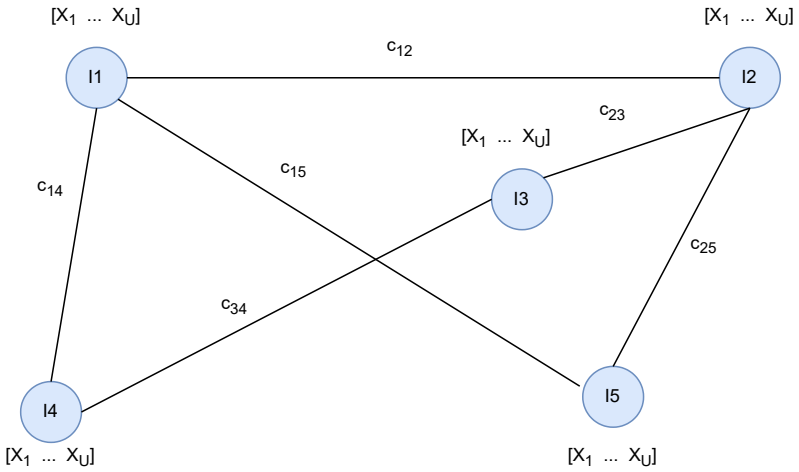

Figura 5.3: Ejemplo de grafo de correlación ensamblado para entrenar RS con GNN.

libro i sobre el de j. En particular, se calcula una matriz de correlación $n \times n$, que luego de normalizada es el GSO utilizado para entrenar la GNN.

3. Sobre cada uno de los n nodos se define una señal $[x_1, ..., x_u]$ con los porcentajes de avance dados por los u usuarios al ítem representado por el nodo. Cada dimensión de esta señal vectorial indica si el usuario interactuó con el nodo (ítem) en particular, siendo 0 cuando no hubo interacción.
4. Cada calificación en el conjunto de entrenamiento tendrá una etiqueta y. El problema se enfrenta como uno de clasificación binaria: se entrena una GNN capaz de predecir si determinado usuario interactuará con determinado ítem. En ese caso la etiqueta predicha será 1, de lo contrario será 0. Más adelante se describe cómo se dividió el conjunto de entrenamiento y la asignación de las etiquetas y.

En la Figura 5.3 se muestra un ejemplo con cuatro nodos, con las correlaciones C_{ij} como pesos de las aristas y las señales $\mathbf{x} \in \mathbb{R}^u$ soportadas sobre los nodos. En este ejemplo los nodos corresponden a ítems y las señales a calificaciones (interacciones).

Los RS implementados son capaces de recomendar ítems a los usuarios en base a los datos disponibles de porcentaje de avance de lectura. Se trata de predecir si a un usuario le gustará determinado ítem, considerando que a un usuario le gustó un ítem si tiene un porcentaje de avance de lectura mayor o igual al 70 %. Con este umbral se definen las $\mathbf{y}$ para cada $\mathbf{x}$, que vale 1 o 0 según se esté por encima del umbral o debajo, respectivamente. Estas son las etiquetas que se intentarán predecir. Para armar el grafo se usan solo los ítems **con** *feedback* (i.e. de los que se tiene el porcentaje de avance de lectura), filtrando los que tengan menos de dos interacciones y también se hace lo mismo con los usuarios que no hayan interactuado con más de dos recursos. Al variar estos filtros se modifica el tamaño del grafo, como se muestra más adelante.

5.2.2. Procesamiento y Preparación de los Datos

A continuación se enumeran los distintos pasos realizados previo al entrenamiento:

1. Se filtran tanto usuarios como ítems por perfil 'Adulto'. En el caso de los ítems, se considera dentro del perfil 'Adulto' a todo ítem que entre las palabras clave

del atributo 'AUDIENCIA' tenga la palabra 'Adultos' (i.e. algunos ítems tienen por ejemplo este campo con el valor 'Adultos;Jóvenes;Niños').

2. A continuación se filtran ítems que tengan menos de 2 interacciones, así como usuarios que hayan interactuado con menos de 2 ítems. Este será el filtrado por defecto, con el fin de contar con un buen número de interacciones (i.e. el tamaño del grafo). También se llevaron a cabo experimentos con un filtrado más restrictivo, para así tener grafos más pequeños, de manera de poder llevar a cabo experimentos con los recursos computacionales disponibles. A modo de ejemplo, filtrando con 2 se tiene un grafo de 3130 nodos, mientras que con un filtro de 20 la red pasa a ser de 406 nodos.
3. Luego de los dos filtrados anteriores, de las 356352 interacciones disponibles en la tabla de porcentaje de avance, quedan 40291. Estos valores de porcentaje de avance filtrados, que valen entre 0 y 100, serán divididos entre 100 para evitar inestabilidades numéricas, quedando finalmente en el rango $[0, 1]$.
4. De las 40291 interacciones resultantes, se tienen 3311 títulos únicos. Estos ítems serán los nodos del grafo a construir, mientras que las aristas serán las correlaciones entre los ítems. Esta matriz de correlaciones será el GSO a utilizar para entrenar el RS con la biblioteca Alegnn.
5. Las correlaciones se calculan entre las columnas de una matriz (i.e. la matriz de interacciones) que tendrá en las filas a los usuarios y en las columnas a los ítems, con los valores de cada celda el porcentaje de avance dividido entre 100. Por lo tanto el GSO, luego de realizados ajustes, será una matriz de dimensiones 3130×3130 (i.e. son 3130 ítems o nodos). Los valores en NaN (i.e. cuando no hay *feedback*) se hacen 0.
6. Además del grafo, representado por el GSO, se tendrá una matriz de interacciones de dimensiones 4391×3130, siendo las filas los usuarios y las columnas los ítems del grafo. De esta matriz saldrán las señales soportadas sobre los nodos del grafo que se usarán para entrenar el RS.
7. El próximo paso consiste en dividir el conjunto de datos, la matriz de interacciones del punto anterior, en los conjuntos de entrenamiento, validación y prueba. La división se realiza a nivel de usuario, seleccionando para cada sub-conjunto cierta cantidad de usuarios con todas sus interacciones. El factor de división utilizado fue de 80/10/10 para aprovechar al máximo los datos para el entrenamiento. A modo de reproducir cada experimento, a la hora de dividir el conjunto de datos se usa la misma semilla aleatoria.
8. Por último, previo a comenzar el entrenamiento, se deben preparar los datos para que puedan ser entrada de la arquitectura LocalGNN disponible en Alegnn. Para ello es necesario definir cómo serán las x y las y, o sea los valores de porcentaje de avance en las x y las etiquetas a clasificar en las y.
    - Para las x se fijarán los valores NaN (cuando el usuario de la fila no dio *feedback* al ítem de la columna) a 0.
    - En el caso de las y, se binarizarán los valores, fijando un umbral de 0,7 (70 % de porcentaje de avance de lectura). Las interacciones mayores al umbral valdrán 1, mientras que las menores o iguales valdrán 0. Lo anterior significa que se considera que a un usuario le gustó un recurso si alcanzó un 70 % de avance en la lectura (hay que considerar que muchos libros tienen al final referencias, notas, bibliografías, agradecimientos, imágenes, epílogos, etc).
9. Por último se re-dimensionan ambos conjuntos para que puedan ser entrada del modelo LocalGNN a entrenar.

5.2.3. Entrenamientos

Para entrenar los RS se utilizó la biblioteca Alegnn [1] [30]. Esta biblioteca está programada sobre PyTorch [18] para implementar distintas arquitecturas de GNN. Los diferentes tipos de redes disponibles en la biblioteca tienen como fin atacar determinados problemas y tipos de grafos. Se eligió en particular el modelo LocalGNN por su afinidad para con el problema de clasificación binaria a resolver y a su buen desempeño a nivel computacional. Algunas arquitecturas de interés, como por ejemplo Graph Attention Networks (GAT) o Graph Convolution Attention Networks (GCAT), no pudieron ser probadas debido a la elevada demanda de recursos que insumían. Asimismo, por las mismas razones fue posible realizar experimentos muy acotados con la arquitectura LocalGNN y con GSO *multi-edge* (GSO-ME) (i.e. solo fue posible experimentar con grafos más chicos).

A continuación se presentan las principales decisiones en cuanto a arquitectura de las GNN y los hiper-parámetros de los entrenamientos:

1. Función de pérdida: CrossEntropyLoss de PyTorch [18]. Se fija el parámetro **ignore_index** en 0. Esto hará que se ignore la clase 0 al computar la pérdida, aprendiéndose así solo la etiqueta 1, que es lo que se desea aprender en el caso de un RS. Esto se ve reflejado en los bajos valores de acierto obtenidos durante el entrenamiento, tanto en el conjunto de entrenamiento como en el de validación. Antes de fijar este parámetro en 0, se observaba en los gráficos de *accuracy*, tanto en entrenamiento como validación, valores muy altos. Esto se debía a que la clase 0 es mayoritaria dada la alta dispersión de los datos (i.e. la mayoría de los recursos tienen pocas interacciones), por lo que se estaba frente a un problema clásico de desbalanceo. Se lograban buenos resultados de *accuracy* en la clase 0, dominante, ocultando los de todas formas esperables malos resultados en la clase 1 (como suele suceder en un RS).
2. Optimizador: Adam.
3. *Learning rate*: de 0.005.
4. *Batch size*: 64.
5. *Epochs*: 100.
6. *Bias*: se fija en *True* para todos los experimentos.
7. *Pooling*: No se aplica.
8. *Filter taps*: experimentos con 3 y 5.
9. Número de Capas: experimentos con 1, 2 y 3. Se intentó entrenar con 5 capas, pero el tiempo de entrenamiento se disparaba de forma considerable dada la mayor cantidad de pesos a calcular.
10. *Features* a nivel de capas: de 32 y 64, intercaladas.
11. *Features* a nivel de aristas: con 1 y 7. El primer caso corresponde al uso de un GSO simple (matriz), mientras que el segundo a un GSO-ME (conjunto de matrices, una para cada atributo). En la próxima sección se describe en detalle este GSO múltiple.
12. Capas de *Readout* individual y múltiples. Se experimentó de las dos maneras, con una sola capa totalmente conectada o múltiples. Para la arquitectura LocalGNN estas capas se agregan localmente en cada nodo.

5.2.4. Otros Experimentos: GSO Multi-Edge

Uno de los objetivos del presente trabajo consistió en explorar la posibilidad de enriquecer un RS implementado con GNN mediante el uso de los atributos a nivel de ítem disponibles (podrían haber sido atributos a nivel de usuarios de contar con ellos). Con ese fin se llevaron a cabo experimentos con la funcionalidad de la biblioteca Alegnn para entrenar GNN sobre grafos con aristas *multi-edge* (i.e. aristas con pesos vectoriales en vez de escalares). Estos pesos vectoriales se almacenan en un GSO *multi-edge* (GSO-ME).

En vez de una matriz, este GSO-ME es un conjunto de matrices, una por cada dimensión del vector de pesos de arista. Computacionalmente el GSO-ME se representa como un *tensor* de tres dimensiones (E, N, N) donde E es el número de *features* en cada arista y N es el número de nodos. Cada una de las E matrices será también una de correlación, y siempre se incluirá en primer lugar el GSO simple descrito más arriba (i.e. la matriz de correlación entre ítems). Las restantes matrices de correlación que conforman el GSO-ME se calculan luego de seleccionar un conjunto de atributos considerados de interés. En los experimentos realizados fueron los siguientes seis atributos, debido a que se evaluaron como los que más podrían aportar información sobre las preferencias de los usuarios: 'TIPO_DE_RECURSO', 'IDIOMA', 'AUTOR', 'ANO_DE_PUBLICACION', 'AUDIENCIA', 'GENERO'. Las correlaciones se calculan entre las columnas de una matriz con los ítems como columnas y los distintos valores de cada atributo como filas.

Debido a la alta exigencia computacional a la hora de entrenar con un GSO-ME, solo se pudo llevar a cabo una cantidad acotada de experimentos para arquitecturas de dos capas nada más y grafos de pocos ítems. Como ya fue mencionado anteriormente, el tamaño del grafo se controla según el filtrado a nivel de interacciones por ítem y recursos consumidos por usuarios. Para poder hacer entrenamientos con los recursos disponibles, hubo que subir estos filtros de 2 a 20, pasando el grafo de tener 3130 nodos a 406. Es necesario mencionar que por esta razón resulta arriesgado comparar ambos tipos de experimentos, ya que este filtrado repercute a la hora de entrenar, debido a que se tienen recursos con mayor cantidad de interacciones, de esta manera facilitando a las GNN aprender de los datos.

Para los experimentos con GSO-ME se tomaron las mismas decisiones de diseño y valores de hiper-parámetros que para el GSO simple, salvo estas diferencias:

1. *Filter taps*: experimentos con 5.
2. Número de Capas: 2. Por alto consumo de recursos computacionales.
3. *Features* a nivel de aristas: 7.

En [34] se informa que no se incorporaron *meta-features* como artistas, álbumes, géneros, etc. (se usó el conjunto de datos MovieLens-100K [13]) debido a que previamente en [36] se demostró que tener en cuenta estos atributos no mejora el desempeño de las GNN entrenadas.

5.2.5. Resumen del Capítulo

Al comienzo de este capítulo se presentó un modelo del estado del arte para RS basados en grafos de conocimiento y el modelo NMP de GNN, con el mecanismo de atención: KGAT. Para este modelo se explicó en detalle su arquitectura, entrenamientos y resultados para tres conjuntos de datos. Luego se pasó a detallar el modelo CorrG-RS, que tiene a la biblioteca Alegnn como base para las arquitecturas de GNN. Se presentó el problema a resolver, se describió la preparación y filtrado de los datos, el armado de los grafos de correlación, las arquitecturas de GNN utilizadas y los principales parámetros de entrenamiento. El capítulo se cierra con el uso de los atributos

de los ítems, incorporados al grafo de correlación mediante *features* vectoriales a nivel de aristas y a la GNN como múltiples GSO encapsulados en uno solo, denominado aquí GSO-ME. La principal motivación para experimentar con un GSO-ME fue verificar si la incorporación de atributos a nivel de ítems mejoraría el desempeño de los RS, así como también para contrastar con conclusiones de otros trabajos, que no la recomiendan. En el siguiente capítulo se despliegan los resultados obtenidos para los modelos y experimentos mencionados.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 6

Experimentos y Resultados

En este capítulo se presentan los resultados obtenidos para los distintos modelos entrenados y experimentos llevados a cabo. Se describe lo trabajado con los modelos de base (i.e. KGAT, ALS, BPR, RS Aleatorio y RS de Popularidad) para luego comparar con los desarrollados en la sección 5.2 (i.e. CorrG-RS). Para el caso de KGAT, al tratarse de código de terceros, y para ALS, BPR, RS Aleatorio (RS-A) y el RS de Popularidad (RS-P), por ser implementaciones relativamente sencillas, se describe con mayor detalle la preparación de los datos y los resultados obtenidos. En el caso de CorrG-RS sí se desarrolla de forma más minuciosa particularidades de los entrenamientos, así como los resultados obtenidos.

6.1. Modelos de Base

6.1.1. KGAT

Datos interacciones usuario-ítem

Se utiliza la tabla matriz_porcentaje_av debido a que es de menor *sparsity* y a que se considera al porcentaje de avance como un buen indicador de la preferencia del usuario. Luego se filtra por perfil Adulto y para las interacciones con avance de lectura mayor o igual a 80 %. Además, se excluyen los recursos leídos por menos de dos personas y los usuarios que hayan leído menos de dos.

Ensamblado de los KG

Los KG se arman con la tabla de recursos, haciendo uso de la biblioteca de Python Networkx [14] (referida de aquí en más como nx). Se construye un grafo para cada relación y luego se ensamblan todos estos sub-grafos con la función nx.ensamble_all(). Se construyen tres KGs distintos, considerando diferentes tipos de atributos:

- El KG1 es el más grande. Se consideran los atributos a priori más relevantes: TIPO_DE_RECURSO, IDIOMA, AUTOR, EDITOR, ANO_DE_PUBLICACION, GENERO, PRESTAMOS_REALIZADOS. Las relaciones son de TITULO a los atributos anteriores: res_type, lang, author, editor, year, genre, borrows.
- En el KG2 se quitan atributos con pocos valores, como TIPO_DE_RECURSO o IDIOMA. Estos atributos forman nodos de alto grado que impactan fuertemente en la topología del grafo. Se desea ver cómo impacta dejarlos afuera. Los atributos finalmente elegidos son: AUTOR, EDITOR, ANO_DE_PUBLICACION,

KGAT-B. Ceibal	Relaciones	Triplas	Usuarios
KG1	7	52222	1807
KG2	5	37278	1807
KG3	3	22430	1807
KG4	7	52222	1843
KG5	5	37278	1843
KG6	3	22430	1843

Tabla 6.1: Número de relaciones, triplas y usuarios de KGs para la Biblioteca Ceibal.

GENERO, PRESTAMOS_REALIZADOS. Las relaciones son de TITULO a los atributos anteriores: author, editor, year, genre, borrows.

- KG3 es el de menor tamaño. Al KG2 se le quita los atributos EDITOR y PRESTAMOS_REALIZADOS partiendo de la base de que al usuario no le suele interesar quién es el editor ni sabe cuántos préstamos fueron hechos para ese libro. Al igual que con KG2, se quiere indagar en el impacto de atributos como los mencionados. Atributos: AUTOR, ANO_DE_PUBLICACION, GENERO. Las relaciones son de TITULO a los atributos anteriores: author, year, genre.
- Los KG 4, 5, y 6 son iguales a 1, 2, 3, respectivamente. Se diferencian en la forma en que se dividen los datos usuario-ítem, lo que se detallará más abajo.

Para ser input del modelo KGAT, los KGs deben ser transformados a un archivo de texto (kg_final.txt). En cada línea del archivo se tendrá una tripla $(head, relation, tail)$. Los valores son enteros generados con el método nx.convert_node_labels_to_integers(). Para poder recuperar los identificadores originales, se hace uso de diccionarios Python para mapear los valores y se almacenan en los archivos de texto (ids_items.txt y ids_users.txt). Las relaciones se guardan en relation_list.txt, con una línea para cada relación, con su identificadores en caracteres y en enteros. En la Tabla 6.1 se despliega el número de relaciones, triplas y usuarios para cada KG construido.

División Entrenamiento/prueba

Se experimentó con dos tipos distintos de división de los datos de interacción usuario-ítem en entrenamiento y prueba.

- División 80/20 por usuario (D82-U). Es el tipo de arreglo usado por KGAT, donde para cada usuario, se toma el 80 % de los ítems con interacciones para entrenamiento y el restante 20 % para pruebas. El modelo toma como input dos archivos de texto, uno para cada conjunto, donde cada usuario tiene una línea con sus respectivos ítems.
- División *leave-one-out* (LOO). Es análoga al tipo de división anterior, con la salvedad de que para el conjunto de pruebas solo se toma un ítem (seleccionado de forma aleatoria) y el resto para el conjunto de entrenamiento.

Para considerar los datos de interacción, el modelo KGAT toma como entrada dos archivos de texto (train.txt y test.txt). En cada línea se tiene el id del usuario y los ids de los respectivos ítems. Los ids de usuario deben comenzar en 0, por lo que se construyeron diccionarios para mapear estos a los ids originales. Los ids de los ítems son los generados en los KGs. Aquí también se elaboran diccionarios para mapear valores. Tanto para los usuarios como para los ítems, se generan archivos ids_items.txt y ids_users.txt.

KGAT – Split LOO				
KG1				
k	**hit**	**ndcg**	**precision**	**recall**
20	**0,1489**	0,0688	**0,0074**	**0,1489**
40	**0,2103**	0,0813	**0,0053**	**0,2103**
60	0,2524	0,0887	0,0042	0,2524
80	**0,2966**	0,0959	0,0037	**0,2966**
100	**0,3282**	0,1007	**0,0033**	**0,3282**
KG2				
k	**hit**	**ndcg**	**precision**	**recall**
20	0,1278	0,0563	0,0064	0,1278
40	0,1843	0,0678	0,0046	0,1843
60	0,2241	0,0749	0,0037	0,2241
80	0,2645	0,0814	0,0033	0,2645
100	0,2955	0,0862	0,0030	0,2955
KG3				
k	**hit**	**ndcg**	**precision**	**recall**
20	0,1450	**0,0709**	0,0073	0,1450
40	0,2048	**0,0830**	0,0051	0,2048
60	**0,2562**	**0,0921**	**0,0043**	**0,2562**
80	0,2922	**0,0979**	**0,0037**	0,2922
100	0,3243	**0,1028**	0,0032	0,3243

Tabla 6.2: Resultados para KGAT con Split LOO.

KGAT: Split LOO y D82-U

Para cada KG construido con Split tipo LOO (KG 1, 2, 3) o D82-U (KG 4, 5, 6), se aplica el modelo KGAT. En las Tablas 6.2 y 6.3 es posible ver los resultados para todas las métricas, con $K = [20, 40, 60, 80, 100]$ y para las divisiones del conjunto de datos tipo LOO y D82-U.

En la Figura 6.1 se grafica de forma conjunta, para todos los KG ensamblados, los resultados para las métricas MeanAverageHitRate@K (MAH@K), MeanAveragePrecision@K (MAP@K), MeanAverageRecall@K (MAR@K) y MeanAverageNDCG@K (MANDCG@K), respectivamente. Se observa cómo el MAHR@K aumenta con K, ya que es más probable encontrar un ítem en el historial del usuario. Lo mismo pasa con MAR@K, debido a que también aumenta la probabilidad de encontrar ítems relevantes entre los recomendados. MANDCG@K aumenta con K también por motivos similares al anterior (aumento de numerador en cálculo, o sea la suma de *scores*). La única métrica que baja con K es MAP@K, algo esperable dado que para muchos usuarios se contará con pocos ítems en su historial como para que el sistema tenga buena precisión al subir K (que impacta en denominador de cálculo de la métrica).

No se observan comportamientos muy disímiles KG a KG, pero sí se ve el agrupamiento de las curvas según tipo de división del conjunto de datos en entrenamiento y pruebas. Las curvas de los KG 1, 2 y 3 suelen estar juntas, al igual que las de los KG 4, 5 y 6. Además de lo anterior, los resultados para los KG 4, 5 y 6 suelen ser superiores, salvo para la métrica MAR@K, donde lo es, de forma marginal, los KG 1, 2 y 3. Por lo tanto, es posible concluir que, para los experimentos realizados, tuvo más peso el tipo de división del conjunto de datos en entrenamiento y prueba que el ensamblado de los KG.

KGAT – Split D82-U				
KG4				
k	**hit**	**ndcg**	**precision**	**recall**
20	**0,2062**	0,0866	**0,0120**	**0,1429**
40	**0,2773**	0,1029	**0,0087**	**0,1930**
60	0,3321	0,1148	**0,0074**	0,2328
80	0,3814	**0,1250**	**0,0065**	**0,2737**
100	**0,4178**	0,1325	**0,0059**	**0,3043**
KG5				
k	**hit**	**ndcg**	**precision**	**recall**
20	0,1861	0,0797	0,0111	0,1299
40	0,2545	0,0945	0,0079	0,1803
60	0,3142	0,1066	0,0067	0,2243
80	0,3592	0,1154	0,0058	0,2595
100	0,3923	0,1225	0,0053	0,2888
KG6				
k	**hit**	**ndcg**	**precision**	**recall**
20	0,2029	**0,0877**	0,0116	0,1363
40	0,2746	**0,1034**	0,0083	0,1874
60	**0,3391**	**0,1165**	0,0071	**0,2362**
80	**0,3755**	0,1244	0,0062	0,2646
100	0,4151	**0,1329**	0,0057	0,2987

Tabla 6.3: Resultados para KGAT con Split D82-U.

Dataset	**K**	**ndcg**	**recall**
Amazon-Book	20	10.06 %	14.89 %
Biblioteca Ceibal (AVG)	20	7.50 %	13.85 %

Tabla 6.4: Resultados de KGAT para Amazon-Book y valores promedio obtenidos para la Biblioteca Ceibal (con distintos KG y divisiones del conjunto de datos).

En la Tabla 6.4 se despliegan los resultados obtenidos en el trabajo original para el conjunto de datos Amazon-Book y el promedio de los distintos entrenamientos para la Biblioteca Ceibal, para $K = 20$ y las métricas Recall y NDCG. Se observa que los resultados originales para un conjunto de datos de una plataforma de venta de libros son comparables a los obtenidos para los datos de la Biblioteca Ceibal.

6.1.2. ALS y BPR

División en entrenamiento/prueba

Para los modelos de base ALS y BPR se utilizó una división del conjunto de datos en entrenamiento/prueba tipo 80/20 (Split D82-U). Se trata de la división habitual para este tipo de RS, consistente en usar el 80 % de los interacciones de los usuarios para entrenar y el restante 20 % para prueba. Recordar que para KGAT las divisiones del conjunto de datos eran tipo D82-U y además LOO. Los modelos ALS y BPR necesitan contar con todos los usuarios en su conjunto de entrenamiento, por lo tanto el tipo de división del conjunto de datos. Esto también debe tenerse en cuenta a la hora de agregar un nuevo usuario: se deberá agregarlo al entrenamiento para poder

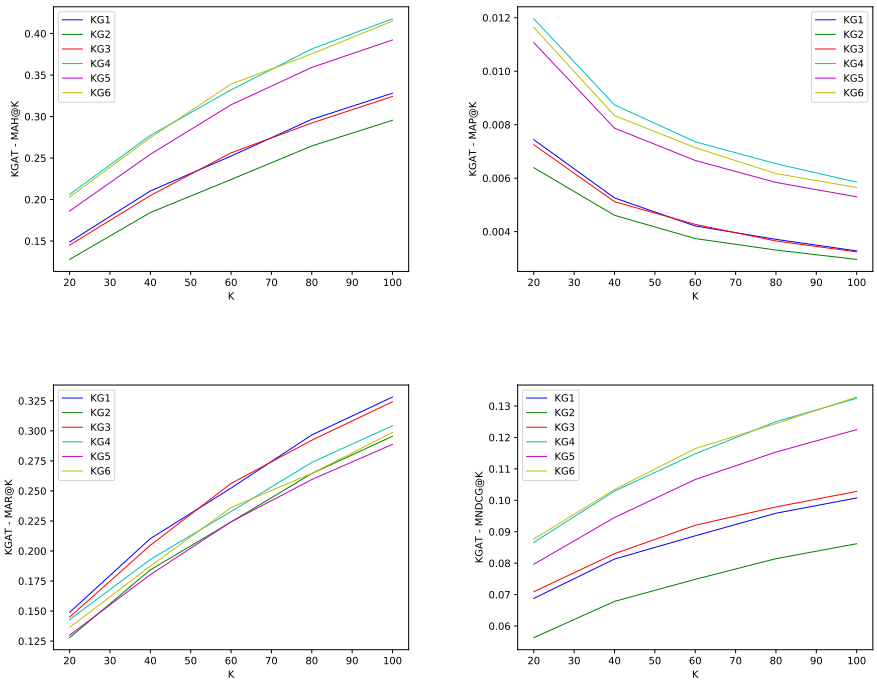

Figura 6.1: MAH@K, MAP@K, MAR@K y MNDCG@K para el modelo KGAT, para todos los KGs y $K = [20, 40, 60, 80, 100]$.

realizar recomendaciones luego. Se trata del problema conocido como arranque en frío y que fue mencionado en el Capítulo 2.

Resultados

Es las Tablas 6.5 y 6.6 se muestran los resultados obtenidos para ALS y BPR, respectivamente, con $K = [10, 20, 50]$ para las métricas HitRate@K, NDCG@K, Precision@K y Recall@K, con división del conjunto de datos tipo D82-U. En la Figura 6.2 se grafican los resultados anteriores. Se verifica la superioridad de ALS frente a BPR, en todas las métricas, y respecto a KGAT en las métricas **ndcg** y **recall** para $K = 20$, como se puede apreciar en la segunda fila de la Tabla 6.4.

ALS y BPR se comportan de forma similar a KGAT respecto a cómo mejoran las métricas al aumentar K salvo para la Precision@K, que como ya fue mencionado, se trata de algo esperable dado que para muchos usuarios se contará con pocos ítems en su historial como para que el sistema tenga buena precisión al crecer K. Además se observa que los resultados obtenidos para el modelo BPR son notoriamente inferiores respecto a ALS. Una de las posibles razones para que se presente este fenómeno es que BPR está diseñado para fortalecer el *rankeo* de los ítems recomendados. De esta manera, el modelo, durante el proceso de entrenamiento, optimizará la capacidad de recomendar ítems en cierto orden más que la capacidad de estimar el grado de preferencia (i.e. nivel de *feedback*). Esta última es la capacidad para la que ALS está diseñado, netamente dedicado a acercarse lo más posible al valor de *feedback* en la

K	hit	ndcg	precision	recall
10	16,91 %	10,43 %	2,09 %	11,00 %
20	21,65 %	11,67 %	1,39 %	14,19 %
50	29,15 %	12,93 %	0,82 %	19,86 %

Tabla 6.5: Resultados para ALS.

K	hit	ndcg	precision	recall
10	2,70 %	1,16 %	0,29 %	1,14 %
20	3,69 %	1,37 %	0,23 %	1,48 %
50	6,56 %	1,84 %	0,19 %	2,77 %

Tabla 6.6: Resultados para BPR.

matriz de interacciones.

6.2. CorrG-RS

En esta sección se muestran los entrenamientos llevados a cabo con los modelos CorrG-RS así como los resultados obtenidos. Los resultados se miden con las métricas descritas en el Capítulo 2 a modo de comparar entre los distintos experimentos de CorrG-RS y contra los RS Aleatorio (RS-A) y RS de Popularidad (RS-P). Con los modelos KGAT, ALS y BPR es posible llevar a cabo comparaciones de resultados, pero sin el mismo grado de exactitud dada la distinta forma de dividir el conjunto de datos.

6.2.1. Entrenamientos

En la Tabla 6.7 se despliegan los resultados obtenidos para distintos entrenamientos al variar el número de capas, la cantidad de *filter taps*, el número de atributos en las aristas y el filtrado a nivel de ítems y usuarios. En todos los entrenamientos de la tabla se usó la arquitectura LocalGNN, el sesgo activado, 100 épocas, un umbral de binarización de 0,7 y la división de los datos en entrenamiento/validación/prueba fue de 80/10/10. El *hardware* utilizado en los entrenamientos fue una GPU para laptop marca NVIDIA GeForce RTX 3070 de 8 GB de memoria RAM.

Lo primero a observar es cómo aumentar el número de capas y la cantidad de *filter taps* incrementa de forma considerable los tiempos de entrenamiento, dada la

id	**layers**	**taps**	**edge_feats**	**readout**	**filt_user**	**filt_item**	**time(min)**	**train_loss**	**val_loss**
1	1	5	1	single	2	2	6	1.83E-01	1.96E-03
2	2	5	1	single	2	2	148	3.85E-02	4.85E-07
3	3	5	1	single	2	2	426	5.08E-02	1.44E-09
4	2	3	1	single	2	2	75	4.22E-02	7.78E-06
5	2	5	1	multi	2	2	160	3.29E-02	0.00E+00
6	2	5	1	single	10	10	4	3.53E-02	6.30E-04
7	2	5	7	single	20	20	3	8.55E-03	5.79E-07
8	2	5	1	single	20	20	2	2.59E-02	1.39E-03
9	2	5	7	multi	20	20	3	1.36E-02	5.96E-11
10	2	5	1	multi	20	20	2	2.83E-02	9.35E-04

Tabla 6.7: Resultados para CorrG-RS. En todos los experimentos se usó la arquitectura LocalGNN, el sesgo activado, 100 épocas, un umbral de binarización de 0,7 y la división de los datos en entrenamiento/validación/prueba fue de 80/10/10.

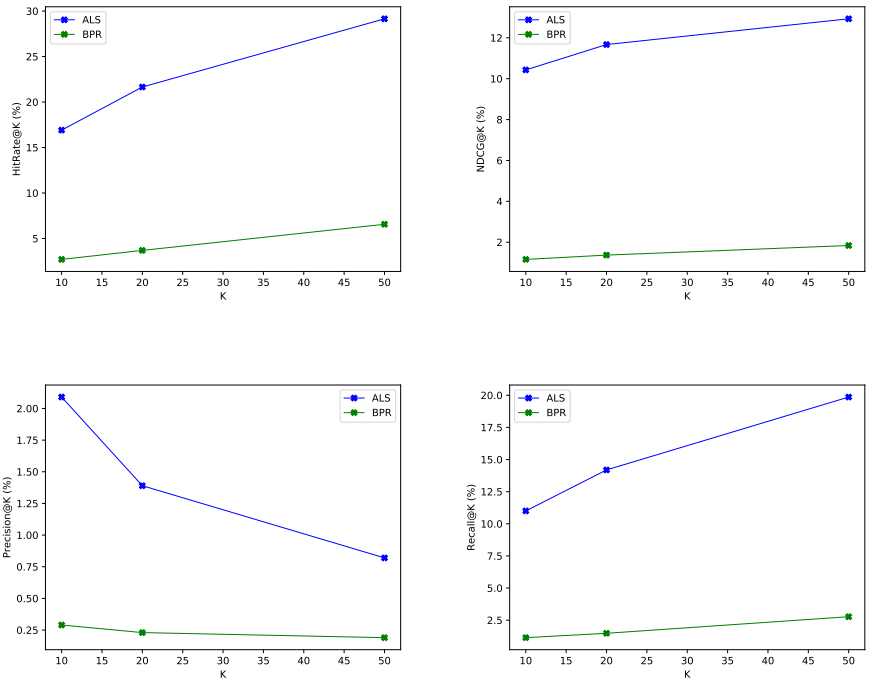

Figura 6.2: Resultados para ALS y BPR. Observar cómo ALS supera a BPR en todas las métricas, para todo K.

mayor cantidad de parámetros a calcular. Se trata de un aspecto a considerar dado el compromiso resultados versus tiempo de entrenamiento y consumo de recursos computacionales. Otro aspecto que impacta de forma severa en los tiempos de entrenamiento es el tamaño de los grafos. Al aumentar los umbrales de filtrado a 10 y 20 se observa una drástica disminución en los tiempos.

GSO Simple

Considerando los valores de pérdida alcanzados tanto en el conjunto de entrenamiento como en validación, se puede concluir que el mejor modelo (i.e. el que a iguales condiciones alcanzó niveles más bajos de pérdida en el conjunto de entrenamiento) fue el del entrenamiento 5, seguido por el entrenamiento 2, que solo se diferencia en el número de capas de *readout*. Lo anterior demuestra la ventaja de utilizar múltiples capas totalmente conectadas a la salida.

En la Figura 6.3 se grafica la *accuracy* y las pérdidas para el entrenamiento 2. Se puede apreciar cómo las pérdidas disminuyen a medida que transcurren las épocas así como también la oscilación de la *accuracy* entorno a valores bajos, como es de esperar en un RS. En cuanto a este último gráfico, recordar que al ignorar la clase 0 mayoritaria mediante el uso del parámetro ignore_index en la función de pérdida, hace que esta clase no se tenga en cuenta al calcular la pérdida. En la Figura 6.4 es posible ver un caso donde no se fijó este parámetro en la función de pérdida. Ver cómo los valores alcanzados son realmente alentadores, pero engañosos: se está aprendiendo la

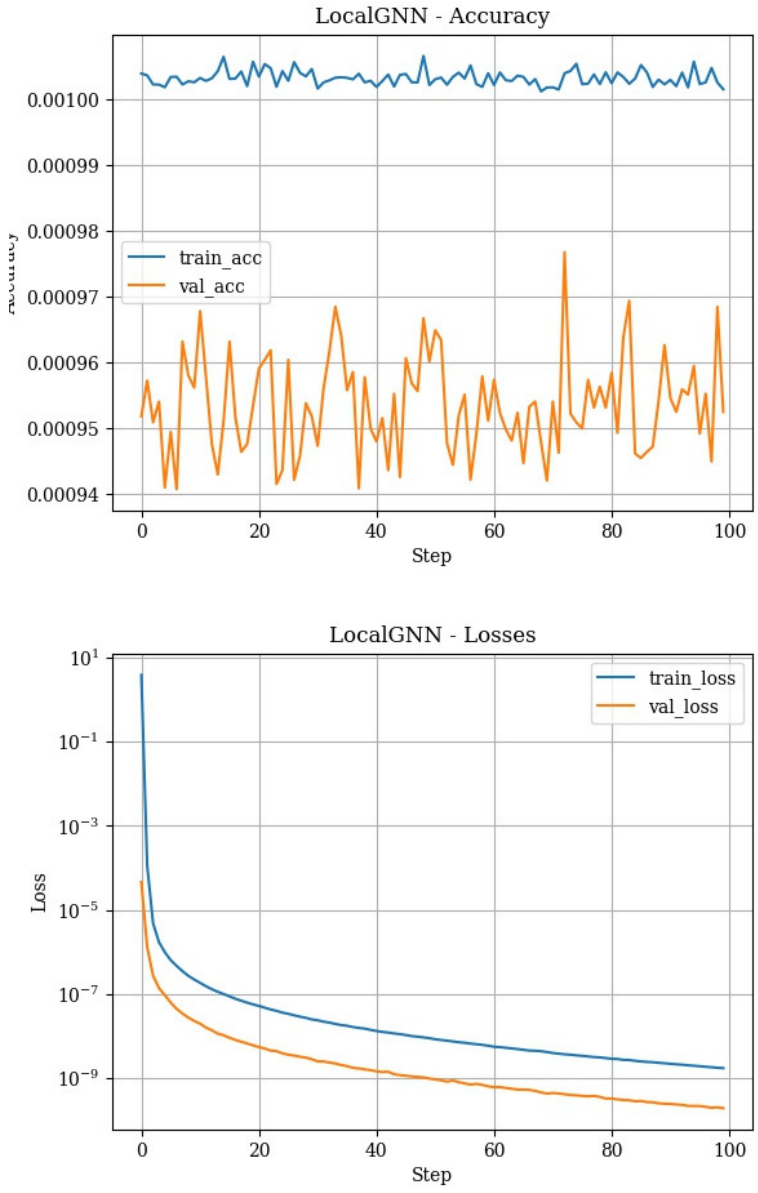

Figura 6.3: *Accuracy* y pérdidas para entrenamiento 2.

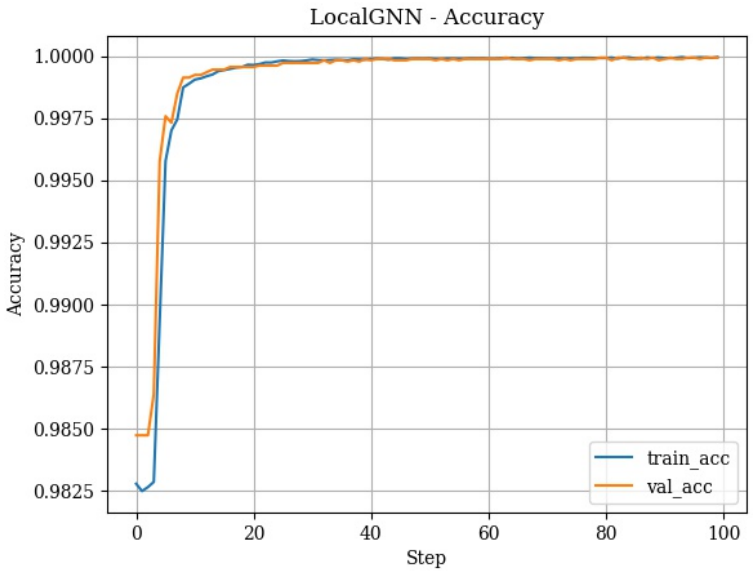

Figura 6.4: *Accuracy* para entrenamiento **sin** parámetro ignore_index en función de pérdida.

clase 0 ampliamente mayoritaria, ocultando lo mal que se desempeñaría en la clase de interés 1.

GSO-ME

Como se mencionó más arriba, para poder llevar a cabo estos experimentos se debió utilizar un grafo más pequeño debido a limitaciones computacionales. Con este fin, se utilizaron umbrales de filtrado a nivel de usuarios e ítems de 20 (i.e. usuarios con al menos 20 ítems consumidos e ítems con 20 interacciones o más), generándose grafos de 406 nodos en vez de 3130 como con un umbral de 2. A iguales condiciones se entrenó un modelo con un GSO simple (i.e. un solo *edge-feature*, experimento 8) y otro con un GSO-ME de siete *edge-features* (experimento 7). En las Figuras 6.5 y 6.6 es posible apreciar las curvas de *accuracy* y pérdida para cada experimento. Los valores de pérdida alcanzados por el experimento 7, donde se usa el GSO-ME, indica que *a priori* sería positiva su utilización.

6.2.2. Resultados

Resultados: GSO Simple

En la Tabla 6.8 se despliegan los resultados obtenidos para las métricas **tradicionales** en los modelos entrenados con un GSO simple. Se puede apreciar cómo, salvo para $K = 20$, el modelo que obtiene los mejores resultados es el 4. En todos los casos se logra superar al RS-A, algo esperable, mientras que no sucede lo mismo con el RS-P, el cual supera al mejor modelo basado en GNN en todos los casos. Esto se debe al fuerte peso que tienen los ítems populares en un RS. En la Figura 6.7 se muestra la evolución con K en cada métrica tradicional para el mejor modelo CorrG-RS-4. Para

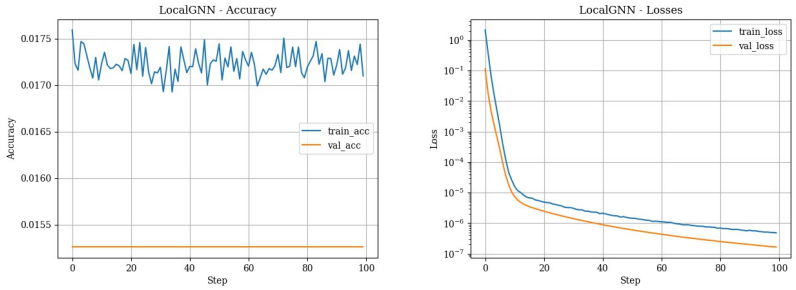

Figura 6.5: *Accuracy* y pérdida para entrenamiento 8.

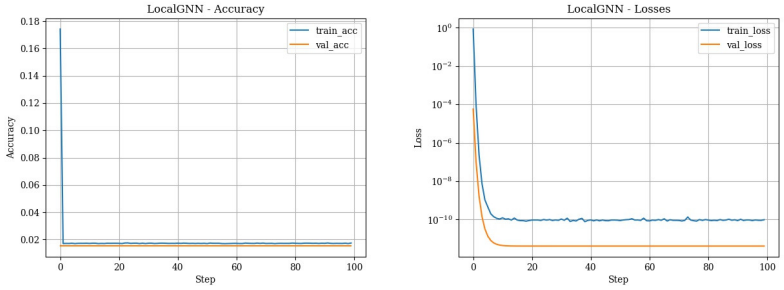

Figura 6.6: *Accuracy* y pérdida para entrenamiento 7.

este modelo también se observa, al igual que en los modelos de base, cómo la única métrica que no mejora con el incremento de K es la Precision@K.

En la Tabla 6.9 se despliegan los resultados obtenidos para las métricas **alternativas** en los modelos entrenados con un GSO simple. Para esta clase de métricas el modelo que mejores resultados arroja es el 1, el cual corresponde al modelo de una capa y cinco *filter taps*. En todos los casos RS-A obtiene resultados superiores, algo esperable en métricas como la novedad, diversidad, la serendipia o la cobertura a nivel de ítem. También, como era de esperar, se supera al RS-P en todas las métricas, ya que un RS que solo recomiende ítems populares no podría desempeñarse bien en cuanto a la novedad, diversidad o serendipia de sus recomendaciones. En la Figura 6.8 se muestra la evolución con K en cada métrica alternativa para el mejor modelo CorrG-RS-1.

Salvo para la diversidad, las métricas varían de forma muy ligera a medida que aumenta K. En el caso de la diversidad, se observa que empeora de forma considerable. Esto se debe a cómo se calcula la métrica. Al calcularse la *Intra-List Similarity* (ILS), a medida que aumenta K también lo hará la proporción de ítems similares (y populares), elevando la métrica. En el caso de la novedad se tiene en cuenta solo la popularidad de los ítems recomendados y además se calcula su logaritmo, aplanando el peso que tiene un ítem a medida que aumenta su popularidad. La novedad es la métrica que más mejora con K, aunque también de forma modesta. Para la personalización se calculan las distancias *inter* listas de recomendación, por lo tanto, a medida que aumenta K es de esperar que aparezcan más ítems repetidos entre las recomendaciones de los usuarios, haciendo que la métrica empeore (i.e. las listas cada vez se tornan menos personalizadas). Respecto a la *serendipia*, esta se mantiene prácticamente constante a medida que aumenta K, observándose muy leves mejorías nada más. Para esta

RS	K	MAH	MAP	MAR
RS-A	10	1.80 %	0.22 %	0.27 %
RS-P	10	16.91 %	1.98 %	8.96 %
1	10	6.47 %	0.68 %	2.03 %
2	10	10.43 %	1.44 %	2.72 %
3	10	10.43 %	1.40 %	2.32 %
4	**10**	**12.23 %**	**1.55 %**	**3.81 %**
5	10	10.79 %	1.47 %	2.72 %
RS-A	20	3.24 %	0.16 %	0.63 %
RS-P	20	26.98 %	1.76 %	13.20 %
1	20	7.91 %	0.41 %	2.73 %
2	20	15.47 %	1.24 %	5.28 %
3	20	15.11 %	1.15 %	4.60 %
4	20	15.11 %	1.17 %	4.70 %
5	**20**	**15.47 %**	**1.21 %**	**4.94 %**
RS-A	50	5.76 %	0.12 %	1.36 %
RS-P	50	44.96 %	1.32 %	23.35 %
1	50	23.74 %	0.63 %	10.05 %
2	50	25.54 %	0.93 %	9.48 %
3	50	23.74 %	0.87 %	8.62 %
4	**50**	**27.34 %**	**1.11 %**	**11.19 %**
5	50	26.62 %	0.96 %	9.78 %
RS-A	100	11.51 %	0.15 %	3.85 %
RS-P	100	54.32 %	1.00 %	30.68 %
1	100	28.06 %	0.48 %	12.35 %
2	100	37.41 %	0.75 %	17.88 %
3	100	35.25 %	0.73 %	16.42 %
4	**100**	**38.49 %**	**0.78 %**	**18.56 %**
5	100	36.69 %	0.74 %	17.49 %

Tabla 6.8: Resultados de métricas **tradicionales** para CorrG-RS, RS-A y RS-P, con $K = [10, 20, 50, 100]$. **GSO Simple**.

métrica, se calculan las similitudes entre el historial del usuario y los ítems en las listas de recomendación, dividiéndolas entre K y luego entre la cantidad de ítems en el historial, así suavizando el impacto que pueda tener un mayor K.

Resultados: Filtrado de Usuarios e Ítems en 20 y GSO-ME

En la Tabla 6.10 se despliegan los resultados obtenidos para las métricas **tradicionales** en los modelos entrenados con un filtrado a nivel de usuarios e ítems de 20, para comparar cuando se usa un GSO-ME y no. En este caso el mejor modelo, salvo para $K = 20$, es el 8, que corresponde a un modelo donde se utilizó un GSO **simple**. Otra vez, al igual que para los entrenamientos de la sección anterior, se supera al RS-A y no al RS-P. En la Figura 6.9 se muestra la evolución con K en cada métrica tradicional para el mejor modelo CorrG-RS-8. Para este modelo también se observa, al igual que en los modelos de base, cómo la única métrica que no mejora con el incremento de K es la Precision@K.

En la Tabla 6.11 se despliegan los resultados obtenidos para las métricas **alternativas** en los modelos con un filtrado a nivel de usuarios e ítems de 20, para comparar cuando se usa un GSO-ME y no. Aquí el modelo que mejor se desempeña es el 7, que corresponde a un modelo entrenado con un GSO-ME. Como era de esperar, suele estar por encima del RS-P y por debajo del RS-A. En la Figura 6.10 se muestra la evolución con K en cada métrica alternativa para el mejor modelo CorrG-RS-7. En este caso se observa el mismo comportamiento que en los modelos de GSO simple y filtrado de 2 a medida que aumenta K.

RS	K	item_coverage↑	user_coverage↑	personalization↑	novelty↓	diversity↓	serendipity↓
RS-A	10	76.00 %	100.00 %	1.00	1.17	1.70	0.76
RS-P	10	0.00 %	100.00 %	0.45	5.32	1.81	0.78
1	**10**	**37.00 %**	**100.00 %**	**1.00**	**3.23**	**1.84**	**0.78**
2	10	1.00 %	100.00 %	0.91	4.94	02.08	0.81
3	10	0.00 %	100.00 %	0.64	4.91	02.08	0.81
4	10	13.00 %	100.00 %	0.99	4.83	02.07	0.82
5	10	0.00 %	100.00 %	0.71	4.94	02.08	0.81
RS-A	20	94.00 %	100.00 %	1.00	1.15	3.58	0.75
RS-P	20	1.00 %	100.00 %	0.48	5.32	3.90	0.79
1	**20**	**49.00 %**	**100.00 %**	**0.99**	**3.17**	**3.88**	**0.78**
2	20	4.00 %	100.00 %	0.97	4.62	4.39	0.82
3	20	1.00 %	100.00 %	0.70	4.52	4.45	0.82
4	20	20.00 %	100.00 %	0.99	4.49	4.35	0.82
5	20	1.00 %	100.00 %	0.68	4.62	4.41	0.82
RS-A	50	100.00 %	100.00 %	0.99	1.17	9.27	0.76
RS-P	50	2.00 %	100.00 %	0.49	4.73	9.99	0.79
1	**50**	**62.00 %**	**100.00 %**	**0.99**	**3.00**	**9.96**	**0.78**
2	50	22.00 %	100.00 %	0.99	04.07	11.04	0.81
3	50	2.00 %	100.00 %	0.66	04.02	11.10	0.81
4	50	27.00 %	100.00 %	0.99	04.07	10.96	0.81
5	50	2.00 %	100.00 %	0.64	4.14	11.12	0.81
RS-A	100	100.00 %	100.00 %	0.98	1.15	18.69	0.75
RS-P	100	3.00 %	100.00 %	0.50	4.37	19.35	0.77
1	**100**	**71.00 %**	**100.00 %**	**0.98**	**2.83**	**20.08**	**0.78**
2	100	29.00 %	100.00 %	0.98	3.74	20.93	0.79
3	100	4.00 %	100.00 %	0.62	3.74	20.94	0.79
4	100	32.00 %	100.00 %	0.98	3.76	20.79	0.79
5	100	4.00 %	100.00 %	0.60	3.74	20.75	0.79

Tabla 6.9: Resultados de métricas **alternativas** para CorrG-RS, RS-A y RS-P, con $K = [10, 20, 50, 100]$. Con los caracteres ↑ y ↓ se indica en si la métrica es mejor al aumentar o al disminuir, respectivamente. **GSO Simple**.

RS	K	MAH	MAP	MAR
RS-A	10	9.52 %	0.95 %	0.92 %
RS-P	10	52.38 %	8.33 %	8.84 %
7	10	19.05 %	4.29 %	4.65 %
8	**10**	**30.95 %**	**5.48 %**	**4.42 %**
9	10	21.43 %	3.10 %	1.88 %
10	10	33.33 %	4.76 %	4.41 %
RS-A	20	35.71 %	2.50 %	4.19 %
RS-P	20	61.90 %	6.19 %	15.32 %
7	20	47.62 %	6.31 %	11.60 %
8	**20**	**50.00 %**	**4.76 %**	**8.53 %**
9	20	38.10 %	3.45 %	5.61 %
10	20	50.00 %	4.52 %	7.77 %
RS-A	50	50.00 %	1.52 %	8.15 %
RS-P	50	83.33 %	4.76 %	25.80 %
7	50	69.05 %	4.43 %	21.60 %
8	**50**	**76.19 %**	**4.90 %**	**23.00 %**
9	50	71.43 %	3.67 %	16.68 %
10	50	76.19 %	4.48 %	22.14 %
RS-A	100	83.33 %	2.40 %	34.09 %
RS-P	100	90.48 %	4.43 %	47.75 %
7	100	85.71 %	3.43 %	36.02 %
8	**100**	**88.10 %**	**3.83 %**	**40.45 %**
9	100	88.10 %	3.60 %	39.19 %
10	100	88.10 %	3.79 %	39.69 %

Tabla 6.10: Resultados de métricas **tradicionales** para CorrG-RS, RS-A y RS-P, con $K = [10, 20, 50, 100]$ y **filtrado en** 20. **GSO Simple y GSO-ME**.

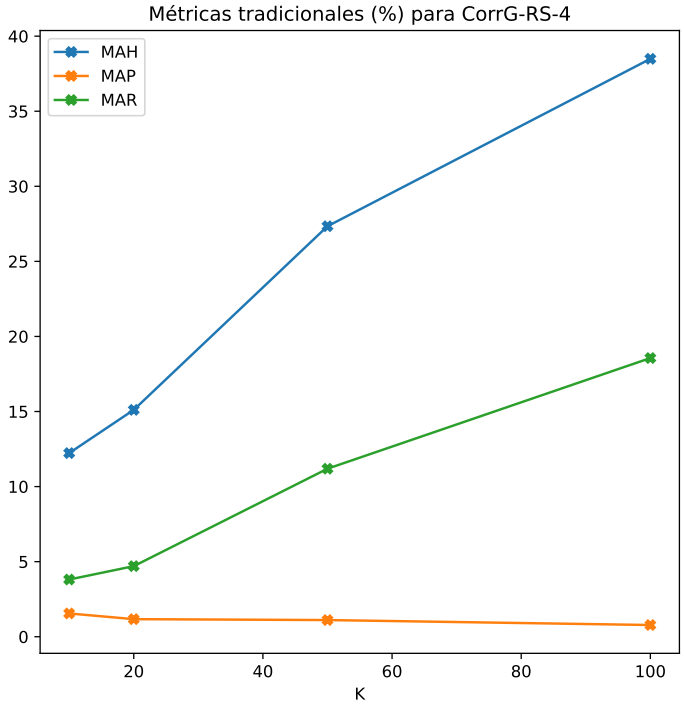

Figura 6.7: Evolución con K en cada métrica tradicional para el modelo CorrG-RS-4.

Pérdidas y Diferencias entre Entrenamientos y Evaluaciones para Modelos CorrG-RS

Como se puede observar en la Figuras 6.3, 6.6 y 6.5, la pérdida en validación es menor que en entrenamiento. Esto se debe a que (1) la pérdida en validación se calcula al final de cada época y a que (2) se trata de un conjunto de validación pequeño (10 %) y desbalanceado (recordar que predomina ampliamente la clase 0), provocando que pequeñas diferencias en la distribución de los datos puedan impactar de mayor manera. Por estas razones se toma la pérdida en entrenamiento como referencia a la hora de evaluar qué modelo alcanzó menores valores.

Al observar los resultados de los entrenamientos a nivel de pérdidas y a nivel de métricas se observan dos fenómenos:

1. Los modelos que obtuvieron menores pérdidas no necesariamente arrojan los mejores resultados a nivel de métricas.
2. En las métricas se verifica también que los mejores modelos no lo son en ambos tipos: en las tradicionales arroja mejores resultados uno y en las alternativas otro.

Los dos puntos anteriores pueden deberse a dos motivos principales:

1. Las pérdidas se reportan para los conjuntos de entrenamiento y validación, mientras que las métricas se calculan para el conjunto de prueba. Dada la naturaleza

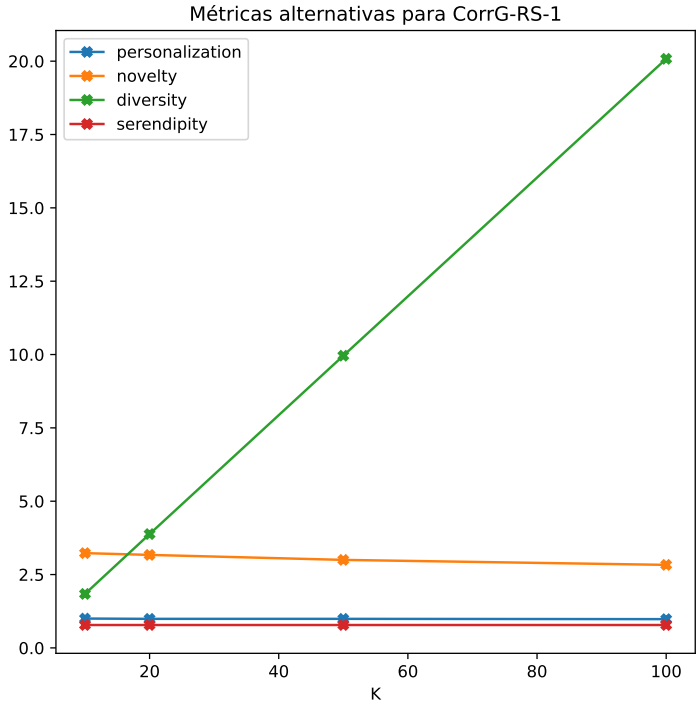

Figura 6.8: Evolución con K en cada métrica alternativa para el modelo CorrG-RS-1.

dispersa de los datos y lo relativamente pequeño del conjunto de prueba (10 % del conjunto de datos), no es de extrañar que se dé este fenómeno. Las diferencias reportadas en las métricas entre los modelos son relativamente menores, dentro de valores en sí bajos, como es de esperar en un RS.

2. Que entre dos de los mejores modelos uno alcance métricas tradicionales superiores y otro haga lo propio entre las alternativas no es de extrañar dada la forma en que se calculan estas métricas. Para las alternativas es necesario primero generar representaciones vectoriales de los ítems. Posteriormente, y según de qué métrica se trate, se computan distancias entre los vectores. La representación de los vectores es independiente de las interacciones entre los usuarios y los ítems, por lo tanto puede darse que algunos vectores tengan mayor peso a la hora de calcularse una métrica alternativa, generando así variaciones. Esto no quiere decir que las métricas alternativas estén del todo desacopladas de los modelos. Estas se calculan en base a los historiales de los usuarios y las listas de recomendación generadas. Pero sí es menester señalar posibles diferencias generadas por la forma en que se vectorizan los ítems.

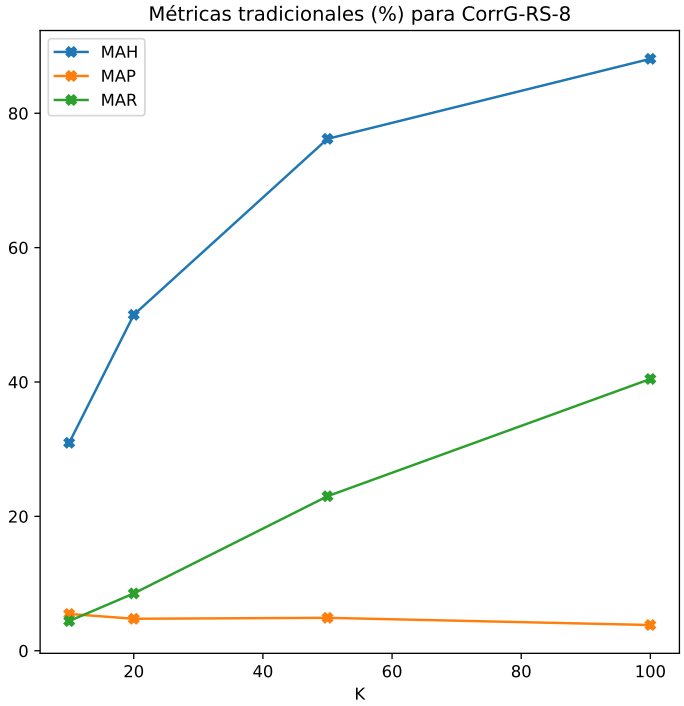

Figura 6.9: Evolución con K en cada métrica tradicional para el modelo CorrG-RS-8.

6.3. Análisis Comparativo entre Modelos de Base y CorrG-RS

En la Tabla 6.12 se comparan, para las métricas tradicionales, los distintos modelos de base con los modelos tipo CorrG-RS, con $K = [20]$. Para cada modelo KGAT se indica el KG utilizado y el tipo de división del conjunto de datos, *Leave-one-out* (LOO) y División a nivel de Usuario 80/20 (D82-U). Las comparaciones deben ser tomadas con cautela dada la distinta forma de dividir el conjunto de datos según el tipo de modelo. De todas maneras, se puede verificar cómo los mejores resultados en MAH y MAP se obtienen para CorrG-RS-8 y en MAR para KGAT-KG1-LOO. Respecto a CorrG-RS-1, supera en MAH a KGAT cuando se usa LOO, salvo en MAR, mientras que en D82-U es competitivo salvo, otra vez, en MAR.

Los mejores resultados se obtienen para el modelo $CorrG - RS - 8$, que tiene un filtrado de 20 y un GSO simple. Al ser el filtrado más alto, se entrena y prueba el modelo con tanto ítems como usuarios con buenos niveles de interacciones. De esta manera, lo que se está haciendo es bajar la dispersión de los datos, algo que era de esperar repercutiera positivamente en los resultados. La alta dispersión de las matrices de interacciones es algo a tener en cuenta para cualquier tipo de RS. Es una de las razones por las que se suele filtrar a nivel de usuario e ítems. El filtrado de base que se usó, de 2, es muy poco restrictivo y tuvo como fin experimentar con la mayor cantidad

RS	K	item_coverage↑	user_coverage↑	personalization↑	novelty↓	diversity↓	serendipity↓
RS-A	10	66.00 %	100.00 %	0.98	2.10	1.66	0.76
RS-P	10	2.00 %	100.00 %	0.45	4.44	1.78	0.79
7	**10**	**46.00 %**	**100.00 %**	**0.97**	**3.21**	**1.89**	**0.79**
8	10	9.00 %	100.00 %	0.90	3.78	1.95	0.81
9	10	28.00 %	100.00 %	0.96	3.46	1.93	0.81
10	10	5.00 %	100.00 %	0.76	3.82	1.96	0.82
RS-A	20	89.00 %	100.00 %	0.97	2.20	3.49	0.76
RS-P	20	5.00 %	100.00 %	0.48	4.34	3.98	0.80
7	**20**	**56.00 %**	**100.00 %**	**0.96**	**3.32**	**3.97**	**0.80**
8	20	17.00 %	100.00 %	0.90	3.73	04.09	0.81
9	20	37.00 %	100.00 %	0.95	3.54	04.08	0.82
10	20	8.00 %	100.00 %	0.73	3.77	4.11	0.81
RS-A	50	100.00 %	100.00 %	0.94	2.16	09.04	0.76
RS-P	50	12.00 %	100.00 %	0.49	3.77	10.23	0.80
7	**50**	**60.00 %**	**100.00 %**	**0.94**	**3.29**	10.31	0.80
8	50	45.00 %	100.00 %	0.92	3.37	9.91	0.80
9	50	48.00 %	100.00 %	0.92	3.33	10.42	0.81
10	50	19.00 %	100.00 %	0.73	3.39	**9.91**	**0.79**
RS-A	100	100.00 %	100.00 %	0.89	2.13	18.14	0.75
RS-P	100	25.00 %	100.00 %	0.50	3.43	19.67	0.79
7	**100**	**66.00 %**	**100.00 %**	**0.88**	**2.94**	20.80	0.80
8	100	63.00 %	100.00 %	0.87	03.05	19.43	0.78
9	100	56.00 %	100.00 %	0.85	2.95	20.90	0.80
10	100	35.00 %	100.00 %	0.70	03.06	**19.41**	**0.78**

Tabla 6.11: Resultados de métricas **alternativas** para CorrG-RS, RS-A y RS-P, con $K = [10, 20, 50, 100]$ y **filtrado en** 20. **GSO Simple y GSO-ME**. Con los caracteres ↑ y ↓ se indica en si la métrica es mejor al aumentar o al disminuir, respectivamente.

RS	K	MAH	MAP	MAR
RS-A	20	3.24 %	0.16 %	0.63 %
RS-P	20	26.98 %	1.76 %	13.20 %
ALS	20	21.65 %	1.39 %	14.19 %
BPR	20	3.69 %	0.23 %	1.48 %
CorrG-RS-4	20	15.11 %	1.17 %	4.70 %
CorrG-RS-8	20	**47.62 %**	**6.31 %**	11.60 %
KGAT-KG1-LOO	20	14.89 %	0.74 %	**14.89 %**
KGAT-KG2-LOO	20	12.79 %	0.64 %	12.78 %
KGAT-KG3-LOO	20	14.50 %	0.73 %	14.50 %
KGAT-KG4-D82U	20	20.62 %	1.2 %	14.29 %
KGAT-KG5-D82U	20	18.61 %	1.11 %	12.99 %
KGAT-KG6-D82U	20	20.29 %	1.16 %	13.63 %

Tabla 6.12: Resultados de métricas **tradicionales** para CorrG-RS (4 y 8), ALS, BPR, RS-A y RS-P y KGAT, con $K = [20]$. Para cada modelo KGAT se indica el KG utilizado y el tipo de división del conjunto de datos.

de datos posible. Filtrar en 20 permitió alcanzar mejores resultados, pero a costo de perder usuarios e ítems a recomendar.

6.3.1. Análisis Cualitativo

En esta sección se realiza un análisis comparativo entre los modelos de base ALS y KGAT y distintos entrenamientos del modelo CorrG-RS. Se seleccionan dos usuarios con el fin de explorar más a fondo qué tipo de recursos son recomendados por los distintos modelos. En todos los casos se generan listas de recomendación de largo $K = 20$ para compararlas entre ellas y con el historial de cada usuario elegido. Los dos usuarios se seleccionaron con el criterio de que para uno de ellos ALS (el mejor modelo de base para $K = 20$) devuelve mejores recomendaciones y para el otro lo hace CorrG-RS. Las recomendaciones para el modelo KGAT se realizaron con el entrenamiento

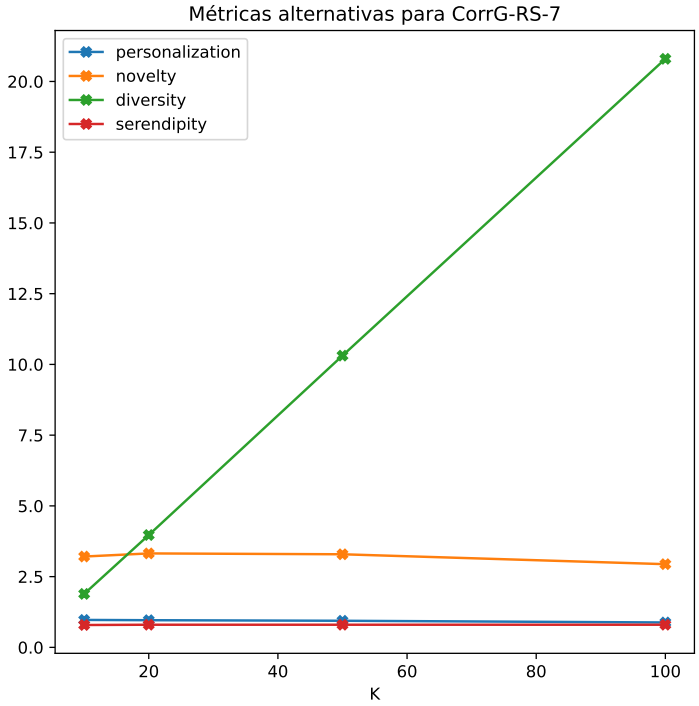

Figura 6.10: Evolución con K en cada métrica alternativa para el modelo CorrG-RS-7.

para el KG4 y división del conjunto de datos tipo D82-U, el mejor entrenamiento para $K = 20$.

Los historiales de los usuarios que se despliegan más adelante consisten en todos los recursos con los que interactuaron. Además, se filtran los de menor popularidad por una cuestión de brevedad. De esta manera se quiere obtener un perfil de cada usuario, observar qué tipo de recursos y autores consume, más allá de cómo se dividió el conjunto de datos según el RS entrenado. Al analizar las recomendaciones para cada modelo, cuando se afirma que determinado recurso recomendado estaba en el historial del usuario, se refiere a que se encuentra entre los ítems seleccionados en el conjunto de prueba.

Como se hizo mención en los Capítulos 1 y 2, la privacidad es un aspecto delicado a tener en cuenta en los sistemas de recomendación. Por esta razón es menester aclarar que, con el fin de mantener el anonimato, no se muestran los identificadores de los usuarios, utilizando las nomenclaturas U_1 y U_2, respectivamente.

Usuario U_1

En el caso de U_1 se tiene que ALS se desempeña muy por encima de CorrG-RS, siendo capaz de realizar mayor cantidad de recomendaciones encontradas en el historial del usuario. Los recursos consumidos por U_1 se despliegan en la Tabla 6.13, junto con el autor, género y el número de interacciones para tener una mejor idea de la

Título	Autor	Género	#Interacciones
Los 3 primeros casos del inspector Mascarell:	Sierra i Fabra, Jordi	Narrativa	55
La cúpula	King, Stephen	Narrativa	39
Mr. Mercedes (Trilogía Bill Hodges 1)	King, Stephen	Narrativa	38
La mujer que no existió	Moretti, Kate	Narrativa	27
Poirot investiga	Christie, Agatha	Narrativa	26
Irène (Un caso del comandante Camille Verhoeven 1)	Lemaitre, Pierre	Narrativa	26
El Ángel (Colomba y Dante 2)	Dazieri, Sandrone	Narrativa	21
Alex (Un caso del comandante Camille Verhoeven 2)	Lemaitre, Pierre	Narrativa	20
Un asesino en escena	Marsh, Ngaio	Narrativa	20
El poeta	Connelly, Michael	Narrativa	20
Mañana te toca a ti	Ahnhem, Stefan	Narrativa	19
Echo Park	Connelly, Michael	Narrativa	19
Tesis sobre un homicidio	Paszkowski, Diego	Narrativa	15
La restauradora	Stevens, Amanda	Narrativa	15
El soborno	Grisham, John	Narrativa	15
La vidente : inspector Joona Linna 3	Kepler, Lars	Narrativa	13
El caso del banquero asesinado	Angelis, Augusto de	Narrativa	13
El asesino vive en el 21	Steeman, Stalisnas-André	Narrativa	12
Muerte blanca: el tercer caso de la agente Marian Dahle	Lindell, Unni	Narrativa	12
Al calor del verano	Katzenbach, John	Narrativa	12
Quien pierde paga (Trilogía Bill Hodges 2)	King, Stephen	Narrativa	12
Fluye el Sena: tres casos del comisario Adamsberg	Vargas, Fred	Narrativa	11
Farsa	Erikson, Thomas	Narrativa	11
Juegos de ingenio	Katzenbach, John	Narrativa	11
El francotirador paciente	Pérez-Reverte, Arturo	Narrativa	11
Cosmética del enemigo	Nothomb, Amélie	Narrativa	11
El observatorio	Connelly, Michael	Narrativa	11
El secreto de Gray Mountain	Grisham, John	Narrativa	10
Una cierta justicia	James, P.D.	Narrativa	10
Personas desconocidas	Katzenbach, John	Narrativa	10
Cuando pase tu ira	Larsson, Åsa	Narrativa	10
Muerte en Estambul	Márkaris, Petros	Narrativa	10

Tabla 6.13: Historial del usuario U_1.

popularidad del título. En la tabla se muestran los libros en el historial con un mínimo de 10 interacciones.

En el historial de U_1 se observan autores muy populares (e.g. King, Connelly, Lemaitre, Katzenbach) y títulos más bien policiales (e.g. *Poirot investiga*, *Iréne*, *Alex*, *Echo Park*). La mayoría de los autores no son de lengua española y tampoco se encuentran autores nacionales entre los listados.

Entre las recomendaciones de ALS se encuentran las siguientes en el historial de U_1: Offshore, El camino blanco, El peor remedio, El huevo de oro, La ira de los ángeles, El ángel negro, Justicia uniforme. Estas seis recomendaciones se encuentran en el historial del usuario, pero no figuran en la Tabla 6.13 debido a que son recursos con menos de diez interacciones y por lo tanto poco populares. Se trata de un aspecto a destacar, dado que se considera positivo que un RS sea capaz de recomendar ítems poco populares. Esto también puede ser una de las razones por las que ALS es capaz de alcanzar mejores resultados en las métricas llamadas tradicionales.

En las demás recomendaciones se encuentran los títulos: *Veneno de cristal*, *Luna funesta*, *El misterio de la mosca dorada: el primer caso de Gervase Fen*, *Universidad para asesinos*, *Oso*, *Macbeth*, *Castigos justificados*, *Tan buenos chicos*, *La palabra se hizo carne*, *El Silmarillion*, *Un jardín entre viñedos*, *Anatomía de un instante*, *El tren de las 4.50*. En este conjunto de recursos también se observa que se compone más bien de autores de *bestsellers*, no uruguayos y de género policial (e.g. *La palabra se hizo carne*, *Veneno de cristal*), con autores que se encuentran también en el historial de U_1 (e.g. Connelly, Márkaris).

El modelo KGAT recomienda los ítems: *Más allá de la verdad*, *1222*, *La luz del*

Título	Autor	Género	#Interacciones
Una jaula de oro	Läckberg, Camilla	Narrativa	101
No está solo	Dazieri, Sandrone	Narrativa	39
El mar	Casacuberta, Pablo	Narrativa	27
Té de benteveo	Lamolle, Guillermo	Narrativa	23
El Ángel (Colomba y Dante 2)	Dazieri, Sandrone	Narrativa	21
La muerte en Venecia	Mann, Thomas	Narrativa	19
Un jardín entre viñedos	Santos, Carmen	Narrativa	17
Las horas distantes	Morton, Kate	Narrativa	17
Los gritos del pasado	Läckberg, Camilla	Narrativa	14
Fóllame	Despentes, Virginie	Narrativa	12
La hermandad de la Sábana Santa	Navarro, Julia	Narrativa	10

Tabla 6.14: Historial del usuario U_2.

diablo, *Una mujer en tu camino*, *El contrato*, *La sed*, *En la mente del hipnotista*, *La broma*, *En las fauces del león*, *Presagios*, *El cazador*, *Veneno de cristal*, *Echo Park*, *No mires atrás*, *Crímenes duplicados*, *¿Quién teme al lobo?*, *El ojo de Eva*, *Crepúsculo en Oslo*, *Cucarachas*, *El guardián*. Entre las recomendaciones solo se encuentra el título *Echo Park* en el historial de U_1 y tampoco hay ítems entre los cien más populares para el perfil Adulto. Sí es posible resaltar que el recurso recomendado pertenece a un autor del gusto del usuario, como es Michael Connelly.

Para este usuario los modelos tipo CorrG-RS que logran recomendar ítems en el historial son los entrenamientos $CorrG - RS - 4$ y $CorrG - RS - 8$, con filtrado de 2 y 20, respectivamente. En ambos casos recomiendan el título *Los 3 primeros casos del inspector Mascarell*, policial del autor Sierra i Fabra. Se trata del ítem más popular en el historial de U_1. Ambos RS coinciden en catorce de veinte recomendaciones (70 %): *Noches de bonanza*, *El caso Fitzgerald*, *La madre*, *El día que se perdió el amor*, *El buzón del tiempo*, *Sabotaje (Serie Falcó)*, *Los 3 primeros casos del inspector Mascarell* , *La mirada de los ángeles*, *La borra del café*, *El quinto hijo*, *Cadáver se necesita (inútil sin experiencia)*, *Bajo la luna de Hawai*, *La reina descalza*, *El fin de la inocencia*. En estas recomendaciones se encuentran obras uruguayas (e.g. *Noches de bonanza*, *Cadáver se necesita (inútil sin experiencia)*, entre otros títulos), algo no deseable para U_1, al menos desde un punto de vista de las métricas llamadas tradicionales.

Es necesario reflexionar si este listado aportaría al usuario de recomendaciones relevantes desde una perspectiva de las métricas alternativas: se trata de ítems sin duda *novedosos* y *diversos*, sí poco *personalizados* y tal vez capaces de generar el efecto de *serendipia*. La naturaleza más bien subjetiva de estas métricas dificulta su evaluación sin ningún tipo de *feedback* por parte del usuario.

Usuario U_2

En el caso de U_2 se tiene que ALS se desempeña de manera deficiente, no siendo capaz de realizar ninguna recomendación incluida en el historial del usuario. Los recursos consumidos por U_2 se despliegan en la Tabla 6.14, junto con el autor, género y el número de interacciones para tener una mejor idea de la popularidad del título. En la tabla se muestran los libros en el historial con un mínimo de 10 interacciones.

Al observar el historial de U_2 es posible concluir que se trata de una persona más bien lectora de novelas, de autores de *bestsellers* (e.g. Camila Läckberg o Julia Navarro), con muchas mujeres en la lista, aunque también se encuentran dos autores uruguayos (e.g. Casacuberta y Lamolle). Todos los recursos en el historial superaron el 70 % de umbral fijado en el porcentaje de avance de lectura para considerar que el libro fue completado.

El modelo KGAT recomienda los ítems: *Todo lo que no te pude decir*, *El valle de los leones*, *Noches de bonanza*, *Los 3 primeros casos del inspector Mascarell*, *Los amores*

equivocados, *El día que se perdió la cordura*, *El monarca de las sombras*, *Los clásicos según Fontanarrosa*, *Un asesino en escena*, *La mujer que volvió del abismo*, *El caso Fitzgerald*, *Las cenizas del cóndor*, *Habitaciones privadas*, *Los dueños del mundo*, *La chica del tren*, *El día que se perdió el amor*, *La madre*, *El quinto hijo*, *Mujer equivocada*, *Bajo los vientos de Neptuno*. Entre las recomendaciones no se encuentra ningún título en el historial de U_2 pero sí trece (65 % de los ítems) entre los cien más populares. En la lista recomendada se encuentran obras de autores nacionales (e.g. *Las cenizas del cóndor*, *Noches de bonanza*), *bestsellers* (e.g. *La chica del tren*, *El caso Fitzgerald*) y autoras mujeres (e.g. Cristina Peri Rossi, Paula Hawkings, Doris Lessing), tres aspectos que se encuentran dentro de las preferencias (historial) de U_2. Esto podría ser considerado positivo desde un punto de vista de las métricas alternativas.

Para este usuario y el K elegido, el modelo de base ALS tampoco logra devolver recomendaciones en el historial, al igual que los modelos $CorrG - RS - 7$ y $CorrG - RS - 8$ de CorrG-RS, con filtrado de 20. En cambio los modelos CorrG-RS con filtrado de 2 número $CorrG - RS - 1$ y $CorrG - RS - 4$, recomiendan los recursos *No está solo* y *Una jaula de oro*, respectivamente. No es de extrañar que los dos ítems exitosamente recomendados (al menos desde un punto de vista de las métricas tradicionales) sean los dos de mayor popularidad en el historial de U_2, ya que se trata de un comportamiento usual en los RS.

El modelo $CorrG - RS - 1$ recomienda los ítems *Noches de bonanza*, *La muy fiel y reconquistadora*, *Los herederos de la tierra*, *Cabrera según Fernando*, *Ideación*, *Sartoris*, *La fiebre del carbón*, *La mano de Fátima*, *Los vagabundos de la cosecha*, *La primera orden*, *Luis Batlle Berres: el Uruguay del optimismo*, *Cuentos completos*, *Mil de fiebre*, *Aguafuertes porteñas*, *El buzón del tiempo*, *Galeano*, *No está solo*, *La madre*, *El idioma de la fragilidad*, *Las cenizas del cóndor*.

Por su parte, $CorrG - RS - 4$ recomienda los ítems *Una jaula de oro*, *Noches de bonanza*, *El caso Fitzgerald*, *La madre*, *Más allá del invierno*, *Cuentos completos*, *El día que se perdió el amor*, *El tigre y la nieve*, *El buzón del tiempo*, *El invierno más largo*, *Sabotaje (Serie Falcó)*, *Los 3 primeros casos del inspector Mascarell*, *La mirada de los ángeles*, *La borra del café*, *El quinto hijo*, *Cadáver se necesita (inútil sin experiencia)*, *Bajo la luna de Hawai*, *La reina descalza*, *Mil de fiebre*, *El fin de la inocencia*.

Ambos modelos coinciden en cinco de las veinte recomendaciones (un 25 %) y ambos son capaces de recomendar varias obras nacionales (e.g. *Mil de fiebre* o *Noches de bonanza*, entre otros títulos), algo que se puede considerar positivo dada la presencia de libros uruguayos en el historial de U_2, al menos desde un punto de vista de las métricas alternativas.

Presencia de Ítems Populares en las Recomendaciones

En los puntos anteriores se evidenció para dos usuarios específicos cómo la popularidad de los ítems en su historial influye en las recomendaciones y cómo algunos modelos tienden a recomendar recursos populares más que otros. Este último es un aspecto frecuente en los RS y por lo tanto resulta de interés evaluarlo de forma más detallada. En la Tabla 6.15 se despliega la media de ítems entre los 100 más populares (MIP100) encontrados en las recomendaciones de los seis modelos analizados en los dos puntos anteriores y para los RS de Popularidad y Aleatorio, a modo de referencia.

Como era de esperar, se puede apreciar cómo la MIP100 es de 20 para el RS de Popularidad (este RS recomienda también en base a los 100 recursos más populares). ALS es capaz de recomendar con una MIP100 debajo del RS Aleatorio, algo que reafirma lo observado para el U_1, cuando ALS recomendaba numerosos ítems de baja popularidad. El modelo KGAT recomienda una media de 7,20 ítems populares, aproximadamente el promedio si se toman los usuarios U_1 (0) y U_2 (13). Respecto a los

Modelo	K	MIP100
ALS	20	2.87
KGAT	20	7.20
CorrG-RS-1	20	6.47
CorrG-RS-4	20	14.1
CorrG-RS-7	20	10.3
CorrG-RS-8	20	13.0
RS-Popularidad	20	20.0
RS-Aleatorio	20	4.47

Tabla 6.15: Media de Ítems Populares en las listas de recomendación para $K = 20$.

modelos CorrG-RS, se comprueba cómo los RS 4, 7 y 8 tienden a recomendar una alta proporción de ítems populares, aspecto también encontrado en los análisis para U_1 y U_2.

Este comportamiento es esperable en los modelos $CorrG-RS-7$ y $CorrG-RS-8$ debido que se filtra por usuarios e ítems con frecuencias mayores o iguales a 20, por lo tanto aprendiendo de datos conformados principalmente por ítems de alta popularidad. $CorrG-RS-1$ es el único capaz de recomendar menor cantidad de ítems altamente consumidos. Este modelo tiene la particularidad de ser el único entrenado sobre una GNN de una sola capa ($CorrG-RS-4$, también con GSO Simple, tiene 2 capas).

La mayor proporción de ítems entre los 100 más populares encontrada en las listas de recomendación generadas por los modelos CorrG-RS puede deberse en parte a que se entrenan como clasificadores, con la particularidad de que se trata de un conjunto de datos desbalanceado (i.e. la mayoría de los ítems son poco consumidos). A estas dos características se suma el hecho de que los ítems populares son dominantes: este pequeño grupo tendrá aún más peso entre los ítems consumidos. Por lo tanto es más probable que sean recomendados.

6.3.2. Resumen del Capítulo

En este capítulo se presentaron los resultados obtenidos para los algoritmos KGAT, ALS y BPR entrenados con los datos de la Biblioteca Ceibal y que fueron usados como modelos de base. También se agregaron a estos modelos los RS-Aleatorio y RS-Popularidad, de sencilla implementación y comportamiento esperado, y en consecuencia valiosos a la hora de comparar. Luego se detallan los resultados retornados por los modelos CorrG-RS, para distintos entrenamientos, variando principalmente arquitecturas, tamaño de grafos (i.e. filtrado a nivel de usuarios e ítems) y GSO (i.e. ME o simple). El capítulo finaliza con una comparación, tanto cuantitativa como cualitativa, entre los distintos modelos, teniendo en cuanto el grado al que se pueden comparar, principalmente fijado por el tipo de división del conjunto de datos en entrenamiento/validación/prueba. Las principales conclusiones adquiridas de estos análisis serán resumidas en el capítulo que sigue.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 7

Conclusiones y Trabajo a Futuro

En el presente trabajo se abordó el problema de implementar Sistemas de Recomendación basados en Redes Neuronales sobre datos en Grafos. Resultó de interés estudiar cómo este tipo de datos (i.e. los grafos), y las arquitecturas de redes neuronales diseñadas para aplicar técnicas de aprendizaje profundo sobre ellos, podrían desempeñarse a la hora de entrenar sistemas de recomendación. En particular se desarrolló en profundidad la implementación de RS basados en GNN sobre grafos de correlación, modelos aquí denominados CorrG-RS, y se los comparó con modelos de referencia, tanto basados en métodos clásicos (ALS y BPR) como en GNN sobre KG (KGAT). Se trabajó con datos de la Biblioteca Ceibal, demostrándose la capacidad de los modelos tipo CorrG-RS de competir tanto con métodos clásicos como con modelos del estado del arte en su clase.

El problema se enfrentó como uno de clasificación binaria y teniendo en cuenta todos los usuarios e ítems. Esto resultó de interés no solo por ser una vía distinta a la tomada en trabajos anteriores citados, sino también por las oportunidades que se presentan a la hora de evaluar un RS. Al ser los RS entrenados clasificadores, se abre una amplia variedad de métricas para evaluarlos, permitiendo medir diversos aspectos de estos modelos. Esto permitió explorar e implementar métricas específicas de los RS, ya fueran las llamadas aquí tradicionales o las denominadas como alternativas. De todas maneras es necesario tomar con cautela los resultados obtenidos en estas últimas, debido a la ya mencionada complejidad a la hora de representar vectorialmente a los ítems y cómo esto impacta a la hora de computar las métricas. La clave estuvo en aumentar la cantidad de atributos más que en el tipo de procesamiento.

Fue de particular interés evaluar si la incorporación de los atributos de los ítems a los entrenamientos sería positiva. Estos atributos se incorporaron a los grafos en forma de *features* a nivel de aristas, en un GSO-ME. Los entrenamientos realizados con el GSO-ME resultaron alentadores para grafos pequeños y con usuarios e ítems con mayores niveles de interacciones. Sería de interés más adelante poder entrenar modelos con grafos de mayor número de nodos.

El algoritmo ALS devuelve buenos resultados para lo rápido de entrenar y relativamente sencillo de implementar. Por su parte, BPR, también rápido y sencillo de implementar, tiene un bajo desempeño. En cuanto a KGAT, los resultados no son tan alentadores considerando los largos tiempos de entrenamiento y la preparación de los datos. Los RS entrenados con la biblioteca Alegnn obtuvieron resultados competitivos con ALS y KGAT y superiores a BPR para grafos grandes (filtrado bajo). Al entrenarse los RS con grafos pequeños (filtrado alto), los resultados son mejores, lo que demuestra la ventaja de usar ítems y usuarios con mayor número de interacciones.

A la hora de entrenar los RS se demostró que las GNN de dos capas dan muy buenos resultados a bajo costo computacional y tiempos de entrenamiento. No se trabajó de manera exhaustiva en la búsqueda de hiper-parámetros de entrenamiento, por lo que se trata de algo a explorar a futuro junto con variaciones en las arquitecturas de las GNN y el *pooling* con el fin de encontrar mejoras en los resultados.

El problema de seleccionar los atributos que mejor representen a los ítems no es trivial, por lo que se debería en caso de considerarse necesario introducir mejoras más adelante. Este problema en sí podría tratarse de un campo de estudio. También a futuro, y relacionado con el punto anterior, dar con una forma de aprovechar el atributo 'MATERIA' (i.e. tesauro), de alto contenido semántico, podría ser positivo para mejorar la representación vectorial de los ítems y así obtener mediciones de similitud más precisas. El uso de un tesauro introduciría desafíos a la hora de explotar esta información así como también a nivel computacional dado el gran tamaño del árbol.

Dado lo relativamente sencillo y rápido de entrenar RS basados en GNN, en caso de obtenerse mejoras en determinadas métricas, resulta válido preguntarse si estos modelos podrían ser parte de un motor de recomendación más amplio, que sea un ensamblado de distintos modelos. De esta manera se podría atacar determinadas debilidades que algunos RS presentan, como recomendar ítems más bien populares, listas de recomendación poco diversas o baja cobertura a nivel de catálogo, entre otras.

Referencias

[1] Alegnn: Graph neural networks. https://github.com/alelab-upenn/graph-neural-networks.

[2] Amazon-book dataset. http://jmcauley.ucsd.edu/data/amazon/.

[3] Christopher olah. understanding lstm networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[4] Collaborative filtering advantages disadvantages. https://developers.google.com/machine-learning/recommendation/collaborative/summary.

[5] Cyberspace administration of china: on public solicitation of comments on the provisions on the administration of internet information service algorithms recommendation. http://www.cac.gov.cn/2021-08/27/c_1631652502874117.htm.

[6] Implicit: Fast python collaborative filtering for implicit datasets. https://implicit.readthedocs.io/en/latest/index.html.

[7] Item and user coverage. https://gab41.lab41.org/recommender-systems-its-not-all-about-the-accuracy-562c7dceeaff.

[8] Knowledge base. https://en.wikipedia.org/wiki/Knowledge_base.

[9] Knowledge graphs. https://en.wikipedia.org/wiki/Knowledge_graph.

[10] Last-fm dataset. https://grouplens.org/datasets/hetrec-2011/.

[11] Loading graphs from csv. https://pytorch-geometric.readthedocs.io/en/latest/notes/load_csv.html.

[12] Matrix factorization. https://developers.google.com/machine-learning/recommendation/collaborative/matrix.

[13] Movielens-100k. https://grouplens.org/datasets/movielens/100k/.

[14] Networkx: Network analysis in python. https://networkx.org/.

[15] Normalized discounted cumulative gain. https://en.wikipedia.org/wiki/Discounted_cumulative_gain.

[16] Plan ceibal. https://www.ceibal.edu.uy.

[17] Plan ceibal: Biblioteca país plan ceibal. https://bibliotecadigital.ceibal.edu.uy/.

[18] Pytorch. https://pytorch.org/.

[19] Pytorch geometric. https://pytorch-geometric.readthedocs.io/en/latest/.

[20] Recmetrics. https://github.com/statisticianinstilettos/recmetrics.

[21] Sentence transformers. https://www.sbert.net/.

[22] Sparse matrices. https://docs.scipy.org/doc/scipy/reference/sparse.html.

[23] Stanford university. cs231n: Convolutional neural networks for visual recognition. `http://cs231n.stanford.edu/`.

[24] Tensorflow. `https://www.tensorflow.org/`.

[25] Tesauro de la biblioteca país plan ceibal. `http://contenidos.ceibal.edu.uy/tematres/vocab/index.php`.

[26] Transformers. `https://huggingface.co/docs/transformers/index`.

[27] Yel2018 dataset. `https://www.yelp.com/dataset/challenge`.

[28] Charu C. Aggarwal. *Recommender Systems: The Textbook*. Springer, Switzerland, 2016.

[29] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, jul 2017.

[30] G. Leus F. Gama, A. G. Marques and A. Ribeiro. Convolutional neural network architectures for signals supported on graphs. *IEEE Trans. Signal Process*, 67(4):1034–1049, Feb 2019.

[31] Fernando Gama, Elvin Isufi, Geert Leus, and Alejandro Ribeiro. Graphs, convolutions, and neural networks: From graph filters to graph neural networks. *IEEE Signal Processing Magazine*, 37(6):128–138, Nov 2020.

[32] William L. Hamilton. Graph representation learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 14(3):1–159, 2020.

[33] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. pages 263–272, 2008.

[34] Weiyu Huang, Antonio Marques, and Alejandro Ribeiro. Rating prediction via graph signal processing. *IEEE Transactions on Signal Processing*, PP:1–1, 08 2018.

[35] Elvin Isufi, Fernando Gama, and Alejandro Ribeiro. Edgenets: Edge varying graph neural networks. *CoRR*, abs/2001.07620, 2020.

[36] Jeremy Ma, Weiyu Huang, Santiago Segarra, and Alejandro Ribeiro. Diffusion filtering for graph signals and its use in recommendation systems. 03 2016.

[37] Chuan Qin Hengshu Zhu-Xing Xie Hui Xiong Qingyu Guo, Fuzhen Zhuang and Qing He. A survey on knowledge graph-based recommender systems. 2020.

[38] Zeno Gantner Lars Schmidt-Thieme Steffen Rendle, Christoph Freudenthaler. Bpr: Bayesian personalized ranking from implicit feedback. *UAI '09: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, page 452–461, Jun 2009.

[39] Yixin Cao Meng Liu-Tat-Seng Chua Xiang Wang, Xiangnan He. Kgat: Knowledge graph attention network for recommendation. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 950–958, 2019.

[40] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu and Tat-Seng Chua. Knowledge graph attention network. tensorflow implementation for the paper: *KGAT: Knowledge Graph Attention Network for Recommendation*, 2019. `https://github.com/xiangwang1223/knowledge_graph_attention_network`.

[41] Maosong Sun Yang Liu Yankai Lin, Zhiyuan Liu and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, page 2181–2187, Jan 2015.

[42] Chris Volinsky Yifan Hu, Yehuda Koren. *Eighth IEEE International Conference on Data Mining*, Dec 2008.

[43] Yuan Cao Zhang, Diarmuid Ó Séaghdha, Daniele Quercia, and Tamas Jambor. Auralist: Introducing serendipity into music recommendation. page 13–22, 2012.

Esta página ha sido intencionalmente dejada en blanco.

Índice de tablas

Índice de figuras

Esta es la última página.
Compilado el miércoles 26 octubre, 2022.
http://iie.fing.edu.uy/