UNIVERSITAT DE BARCELONA

Treball de Fi de Grau

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

Revisión y análisis de Sistemas Recomendadores basados en Sesión

---

Samuel Calabria Cano

Directora: Dra. Maria Salamó
Realitzat a: Departament de
Matemàtiques i Informàtica

Barcelona, 13 de juny de 2022

Resumen

Debido al gran abanico de opciones a la hora de elegir el humano siempre se ha apoyado en expertos para su selección. Aquellas recomendaciones a las que se podía recurrir físicamente en una tienda o centro se han reemplazado por sistemas automáticos de recomendación. Estos reconocen los gustos del usuario y gracias a la información recogida intentan predecir buenas recomendaciones para el usuario.

En los últimos años ha habido una revolución en el campo de la personalización de contenidos debido a las mejoras en aprendizaje automático, Machine Learning en inglés. Es por ello que existe una necesidad de categorizar y evaluar las nuevas aportaciones del estado del arte y comprobar si estos nuevos métodos ofrecen una mejora a los sistemas tradicionales de recomendación.
El punto principal del trabajo se focaliza en reconocer, interpretar y categorizar estos nuevos sistemas que predicen a través de la información recogida durante una sesión o también conocidos como Recomendadores basados en sesión, específicamente nos centraremos en aquellos basados en Deep Learning.

Como resultado del proyecto, se propone una revisión y análisis del estado del arte actual y la profundización en cuatro métodos novedosos, con un análisis a nivel teórico y práctico a nivel de resultados con tres conjuntos de datos.

Es gracias a este estudio y todas las publicaciones que lo sostienen, que se puede afirmar que existe una revolución en este campo y que todos los métodos estudiados ofrecen una mejora a la aproximación tradicional.

Abstract

Due to the wide range of choice, humans have always relied on experts for their selection. The recommendations that could be made physically in a shop or centre have been replaced by automatic systems. They recognise the user's tastes and, based on the information gathered, try to predict good recommendations for the user.

In the last few years there has been a revolution in this matter due to improvements in mainly Machine learning, therefore there is a need to categorise and evaluate the new contributions to the state of the art to check if these new methods offer an improvement to the traditional ones.
The main focus of the work is to recognise, interpret and categorize these new systems that predict through the information collected during a session or also known as Session Based Recommenders, specifically we will focus on those based on Deep Learning.

As a result of the project, a review of the current state of the art and an in-depth study of four novel methods is proposed, with a theoretical and practical analysis of the results.

It is thanks to this study and all the publications that support it that it can be affirmed that there is a revolution in this subject and that all the methods studied in this work offer an improvement on the traditional approach.

Agradecimientos

Transmitir mi más sincero agradecimiento a todos aquellos que me han ayudado a lo largo de esta etapa y han colaborado en esta investigación.

En primer lugar, a mi tutora, la Doctora Maria Salamó, por su ayuda en la planificación, información y organización en este Trabajo de Fin de Grado.

En segundo lugar, a Alejandro Ariza, por su colaboración en las partes más complicadas de la realización del proyecto, otorgando información y recursos para solventarlas.

En tercer lugar, a mi familia y amigos que han estado a lo largo de toda mi carrera apoyándome en todo momento y animándome a seguir adelante.

Y por último a la Universidad de Barcelona por facilitarme los recursos y formación necesaria para el desarrollo tanto de este proyecto como de la carrera.

Índice

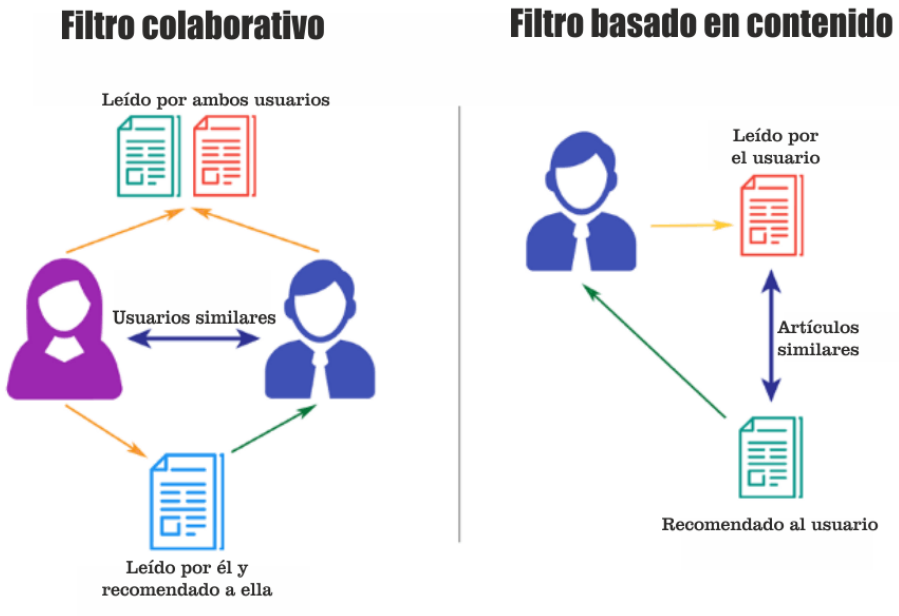
1. Introducción

Durante el siguiente capítulo se introducirán los conceptos necesarios para el entendimiento del proyecto, se contextualizará y se describirán tanto el problema que al que aborda cómo los objetivos de este. Finalmente se explicará la planificación del proyecto y se describirá de una manera escueta la estructura de este.

1.1. Ámbito del proyecto

Los sistemas recomendadores han ganado una relevancia vital en la sociedad actual. Con el aumento exponencial de acceso a la información que se tiene hoy en día, este tipo de sistemas se encargan de ofrecer al usuario alternativas o posibles soluciones a sus problemas. Estas prácticas hacen uso del *Big Data* y el *Machine Learning* como sus principales armas. A través del *Big Data*, estos procesos se nutren de una base de información gigante. Con dicha información, el *Machine Learning* ejecutará una serie de acciones que finalizarán en la obtención de predicciones en las que se tendrán en cuenta aspectos como; aspectos sociales o preferencias del usuario en el momento de la acción. Este tipo de sistemas se describen como sistemas de recomendación. Dentro de este tipos de sistemas existen principalmente dos tipos; aquellos que filtran por contenido, la importancia recae en las características de los ítems, y los colaborativos, que son aquellos que la importancia recae en el usuario, nos centraremos en estos últimos (véase Figura 1).

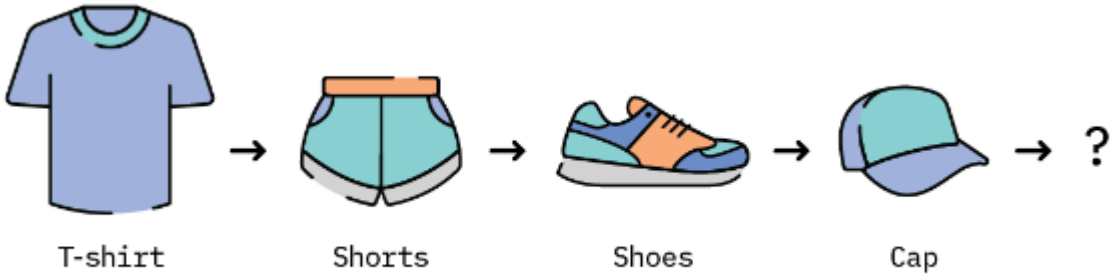
Figura 1. Representación visual de los tipos de sistemas



Durante el proyecto se hará especial hincapié en aquellos sistemas de personalización basados en información de sesiones, comúnmente llamados Sistemas Recomendadores basados en Sesión o SREC. Estos son aquellos en los que la obtención de la predicción viene precedida por unas acciones de un usuario durante un período de tiempo definido, que es la

definición de sesión. Es decir, una secuencia de acciones definidas, normalmente clicks, que se relacionan con otras acciones o productos, (véase Figura 2).

Figura 2. Ejemplificación de una sesión.



Como característica principal en estos sistemas destacamos la captura de relaciones entre diferentes actores, objetos y acciones.

1.2. Descripción del problema

En los últimos años ha habido un incremento sustancial de nuevos algoritmos de recomendación en sesiones. Todos estos nuevos algoritmos tienen un problema de rendimiento debido al largo proceso de obtención de resultados y normalmente se comparan con los métodos tradicionalmente asentados en la literatura.

Además de todo esto, los métodos se comparan con conjuntos de datos específicos y algunos de ellos no proporcionan los métodos de preprocesado, y si lo hacen, únicamente de manera excepcional.

1.3. Objetivos del proyecto

El objetivo principal de este trabajo es el estudio de los Sistemas Recomendadores basados en sesión. Durante el transcurso de este, se revisará el estado del arte junto a las nuevas publicaciones relacionadas con este.

Además de hacer una revisión del estado actual, se analizarán cuatro algoritmos principales. Se lanzarán ejecuciones con tres de ellos, para comprobar que los resultados de las publicaciones están en concordancia con lo revisado.
De estos análisis se obtendrán unos resultados a comparar con los que se sacarán conclusiones sobre el uso de estos sistemas, junto a las ventajas que aportan al estado del arte.
Junto al objetivos principal se recogen los subobjetivos siguientes:

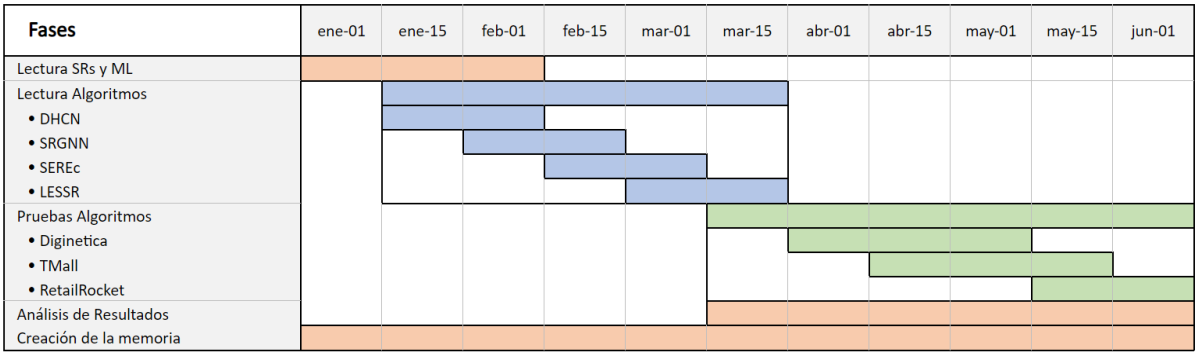
1. Análisis del estado del arte en SREC.
2. Implementación de cuatro algoritmos de SREC.
3. Análisis de resultados en tres conjuntos de datos.
4. Comparativa de los resultados con un método asentado en la literatura.

1.4. Planificación del proyecto

La planificación que se ha seguido durante el proyecto ha sido la siguiente:

Una fase inicial donde se entienden y asientan los conceptos de machine learning necesarios, además de la introducción a los sistemas recomendadores y en especial hincapié a los basados en sesión.
Una segunda fase, en la que se eligen los métodos a estudiar a través de la ayuda y consultoría del director del proyecto. Durante esta fase se sigue leyendo sobre los sistemas recomendadores basados en sesión y sus características frente a los sistemas tradicionales.
En la tercera fase del proyecto, se inician sprints de dos semanas para la lectura y entendimiento de los métodos, además de asíncronamente empezar a revisar el código de estos. El procedimiento es el siguiente, lectura y comprensión de la publicación relacionada, estudio del código para el entendimiento de este y para acabar el lanzamiento de pruebas con los datasets escogidos previamente. Este ciclo se repite secuencialmente con cada uno de los algoritmos propuestos.
Y finalmente, un último ciclo donde recogidos todos los resultados obtenidos se compara con los resultados esperados en las diferentes publicaciones, y si es el caso el relanzamiento de las pruebas pertinentes.

Figura 3. Diagrama de GANTT.



1.5. Descripción de la estructura del TFG

Como ya se ha mencionado anteriormente, la estructura del proyecto inicialmente consistirá en hacer una revisión del estado del arte actual, hacer una presentación teórica a los métodos; resaltando su propuesta de valor frente a la competencia, para finalmente lanzar pruebas con tres conjuntos de datos diferentes y analizar los resultados.
A partir de estos resultados se sacarán unas conclusiones con las que podremos categorizar los métodos, con sus ventajas e inconvenientes.
Esta memoria se divide en los siguientes capítulos:

- **Capítulo 2**: Estado del arte; en este capítulo se definen las características básicas de este tipo de sistemas, además de una categorización de estos parámetros. Finalmente se describirán los cuatro métodos a estudio.

- **Capítulo 3**: Análisis, diseño e implementación; durante el capítulo se describirán los métodos estudiados, se hará una breve presentación a los datos y se presentará el escenario experimental de las pruebas.
- **Capítulo 4**: Análisis y resultados; este apartado se centra en describir los diferentes conjuntos de datos, presentar cuáles son los métodos a comparar y finalmente mostrar los resultados obtenidos de las pruebas.
- **Capítulo 5**: Conclusiones; se mostrarán las conclusiones del trabajo junto a la obtención o no, de los objetivos propuestos.

Finalmente, para acabar el documento, se incluyen las referencias bibliográficas.

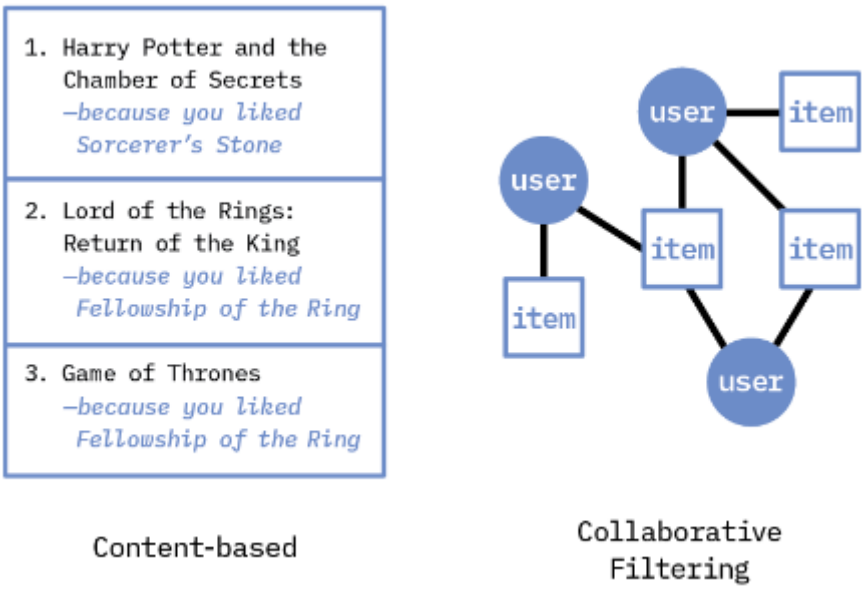
2. Estado del arte

A través del siguiente capítulo se contextualiza el trabajo dentro del ámbito de estudio, se muestra una taxonomía inicial con la que se podrán relacionar los métodos y finalmente una introducción a los métodos de estudio.

2.1. Definición del ámbito de la investigación

Cómo se ha mencionado anteriormente, los sistemas recomendadores se dividen en dos grandes familias; aquellos que son *Content-Based* y aquellos basados en *Collaborative Filtering*.
Explicado de una manera sencilla, los basados en contenido son aquellos crean las recomendaciones a través de las preferencias del usuario por las características de los productos. Estas preferencias se crean a través de las acciones previas del usuario en cuestión o feedback explícito. Por el contrario, los sistemas basados en filtrado colaborativo utilizan las interacciones usuario - ítem a través de toda la red de usuarios para crear recomendaciones a un usuario en específico. Las preferencias del usuario específico se consiguen a través de las preferencias de otros, normalmente usuarios muy parecidos (aquellos que leen lo mismo, ven los mismos shows, compran mismos ítems…). Los Collaborative Filtering tienden a utilizar un histórico de las acciones entre usuario - ítem para crear tendencias de usuario generales, y así diferenciar las acciones inusuales del conjunto general de preferencias. A través de la siguiente figura se ejemplifican estos dos tipos de sistemas.

Figura 4. Ejemplificación de los dos tipos de sistemas.



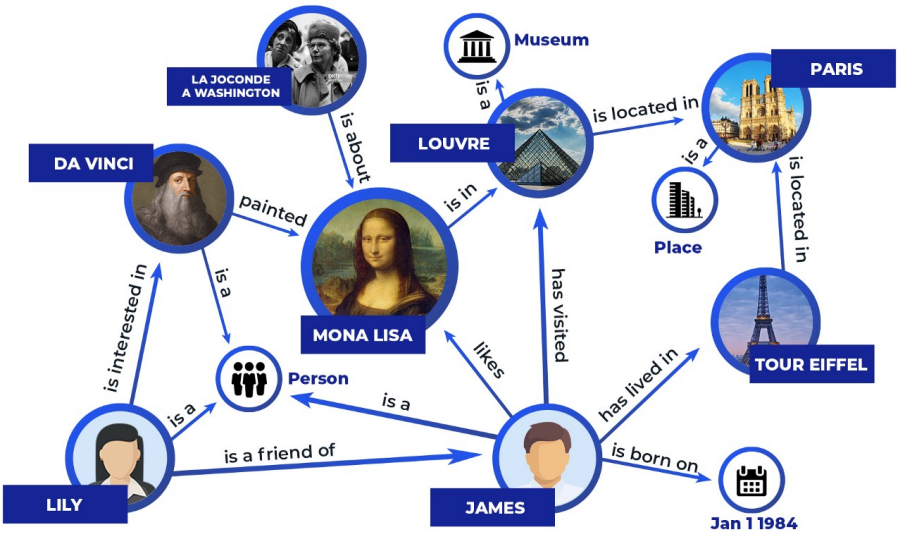
Durante el proyecto nos centraremos en aquellos sistemas que utilizan Collaborative Filtering basados en sesiones. Este tipo de sistemas recogen la información de los usuarios durante un transcurso de tiempo definido, la codifican a través de los diferentes productos o acciones y

se procesan como un conjunto, este conjunto se conoce como sesión. Dentro de los SREC basados en Sesión nos centraremos en aquellos que se basan en *Deep Learning,* y específicamente en aquellos que hacen uso de grafos para representar la información recogida a través de las sesiones, este tipo de sistemas son las GNN o *Graph Neural Networks*.

2.2. Graph Neural Networks

Las GNN son un nuevo tipo de redes neuronales que se basan en el uso de grafos, es por ello que en este tipo de red las relaciones y de qué tipo son, toman una importancia similar o igual que los propios nodos del sistema.

Figura 5. Grafo y de los diferentes tipos de relaciones entre ítems y usuarios.



Este tipo de red neuronal centra la acción en los nodos. Estos se encargan de recopilar información de los nodos vecinos para aprender las relaciones entre los diferentes agentes de la red. De esta manera la GNN es capaz de reconocer las relaciones y asimilarlas, además de transmitir los datos e incorporar la información a los nodos correspondientes. Este tipo de redes son de gran utilidad en nuestro campo de estudio ya que las acciones del usuario están restringidas a un conjunto y tanto las acciones como los ítems tienen una importancia vital.

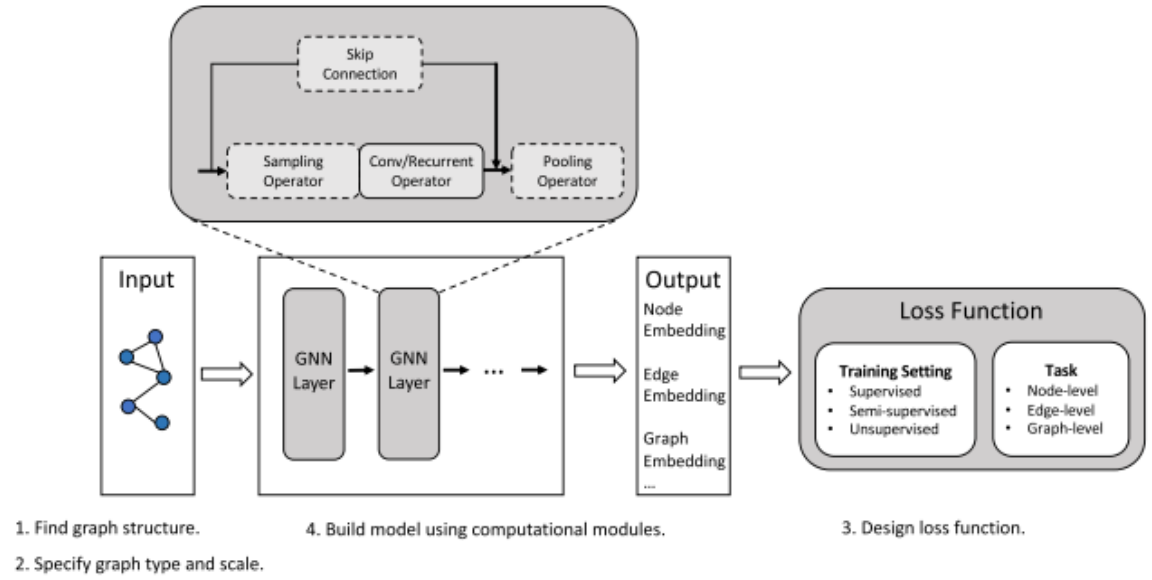
El progreso en los últimos años en el uso de las GNN viene precedido de los avances en las Redes Neuronales Convolucionales (CNN) y en el campo de aprendizaje de grafos (GRL).

Para información como imágenes o textos las CNN son increíblemente efectivas extrayendo características concretas. En cambio, para información recibida como grafos este tipo de redes necesita una generalización a la hora de procesar los datos. Por su parte las GRL se centran en generar vectores de pocas dimensiones para nodos, vértices o subgrafos con el propósito de representar las complejas estructuras de éstos.

Combinando ambos sistemas, CNN y GRL, las GNN son capaces de tratar con la información de manera estructural y aprender representaciones de alto nivel.

Este tipo de sistemas se definen en unos pasos específicos. Normalmente, se constituyen en la obtención de los datos y definición de la estructura de la información con la construcción de el/los grafos pertinentes, el diseño de la red, la optimización del modelo y el resultado de la predicción. A través de la siguiente imagen se muestra un esquema general de una *Graph Neural Network.*

Figura 6. Esquema general del recorrido de una GNN.



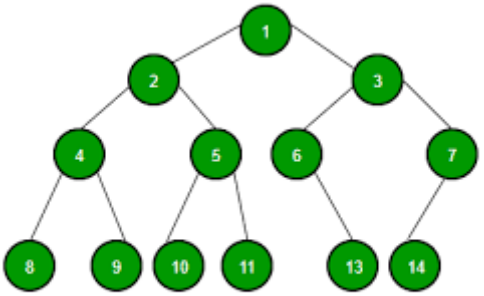
A continuación nombramos los pasos más importantes haciendo aclaraciones.

2.2.1. Construcción del Grafo

En la actualidad, el diseño de los modelos GNN vienen categorizados por tres clases.

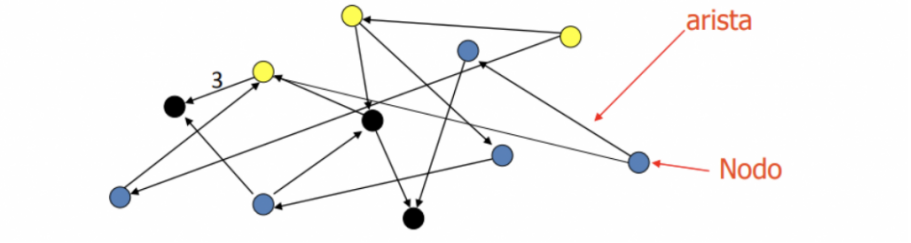
1. **Grafos Homogéneos**: En los que cada vértice solamente conecta con dos nodos y solamente hay un tipo de nodo.

Figura 7. Representación grafo homogéneo.



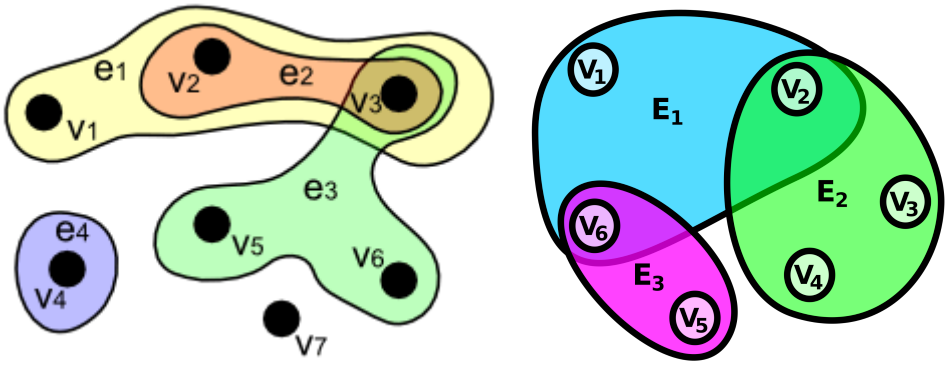
2. **Grafos Heterogéneos**: En los que cada vértice solamente conecta con dos nodos pero hay múltiples tipos de nodo.

Figura 8. Representación grafo heterogéneo.



3. **Hipergrafos**: Aquellos en los que cada vértice puede conectar a más de dos nodos.

Figura 9 y 10. Representaciones de un hipergrafo.



La representación es diferente a cada método, pero normalmente los nodos son los ítems y las relaciones entre estos son las acciones realizadas por un usuario concreto.
Uno de los campos de investigación en los que se están desarrollando avances es el Knowledge Graph (KG), que es una representación de un grafo heterogéneo. Es importante pues el KG integra múltiples atributos de la información con sus relaciones y llegando a poder cubrir una gran cantidad de entidades. Su importancia es clave ya que permite relacionar diferentes entidades o relaciones abstractas en una representación con grafo, creando así una jerarquía dentro de la propia estructura de la información, (véase figura 5).

2.2.2. Diseño de la Red

En términos de red las GNN están categorizadas por espectro o espacio.
Los modelos espectrales consideran los grafos como señales y se procesan con convoluciones a nivel de espectro. Concretamente estas señales primero se transforman al dominio espectral con la transformada de Fourier definida para grafos, después se les aplica un filtro y finalmente son transformadas de vuelta al dominio espacial.
Por otro lado, los modelos espaciales ejecutan la convolución directamente sobre los grafos con el fin de extraer características concretas usando pesos personalizados en la fase de agregación de la red neuronal.

Aunque ambos sistemas nacen de diferentes lugares buscan el mismo principio, recolectar la información de vecindad de manera iterativa para recoger las correlaciones entre nodos y vértices.

2.2.3. Optimización del Modelo

Después del procesado de la red, los nodos y vértices contendrán información así como la información de cómo se ha generado la estructura.
Para ejecutar las tareas próximas al aprendizaje estas incrustaciones deberán ser transformadas a objetivos por una red neuronal general siguiendo un procedimiento estándar. Aquellas incrustaciones (nodes/edges) más relevantes serán mapeadas con etiquetas para la formulación de la función de pérdida, para que luego los optimizadores existentes sean utilizados por el modelo de aprendizaje.
Existen múltiples funciones de mapeado; como el *MultiLayer Perceptron* (MLP) o *inner Product*, y funciones de pérdida como *Pairwise*, *Point-wise* para cada tarea específica.

2.3. Taxonomía de los métodos

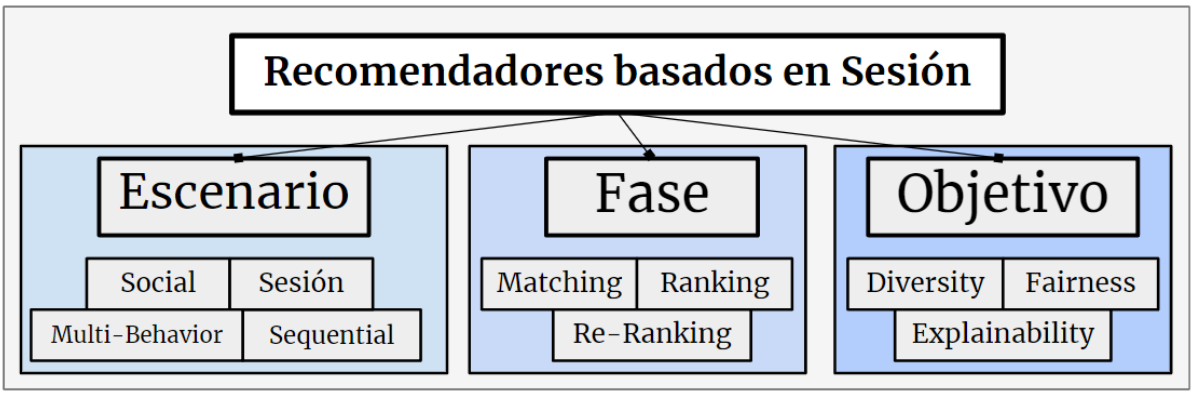
Los sistemas recomendadores basados en sesión se pueden clasificar de diferentes formas. Esta clasificación la dividiremos en dos apartados, a nivel de sistema recomendador y a nivel de GNN.

A nivel general dividiremos los sistemas por tres características:

- En primer lugar, el escenario. En este apartado se diferencian los diferentes contextos en los que se puede aplicar un sistema recomendador.
- En segundo lugar, la fase. A través de este apartado se clasifican los diferentes ciclos por los que puede pasar el método.
- Y finalmente, el objetivo. Es un aspecto adicional en el que se contemplan atributos complementarios al funcionamiento propio del modelo.

A continuación se muestra una representación gráfica de la taxonomía escogida para la elaboración de este proyecto.

Figura 11. Taxonomía sistemas recomendadores basados en Sesión.



Seguidamente se explicarán con más detalle cada una de las características principales de los sistemas recomendadores basados en sesión, además de una pequeña explicación en sus tipos.

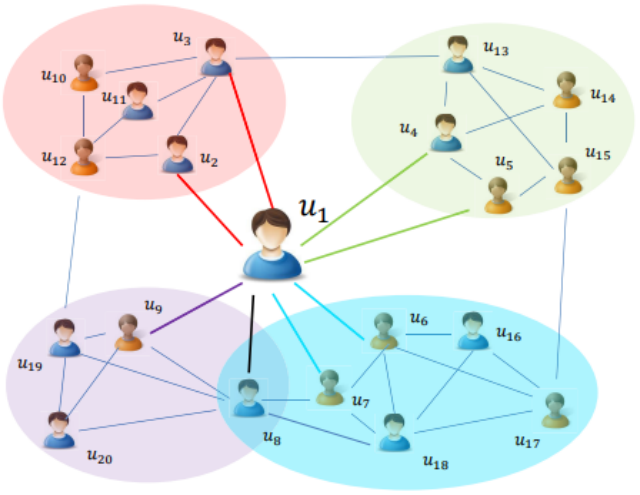
Según el escenario:

Dentro del escenario podemos concretar cuatro grandes posibles situaciones que se pueden llegar a considerar.

Social Recommendation [1]

La habilidad de interaccionar con otros usuarios ha creado la posibilidad de influenciar en el conjunto, conocido como influencia social. Normalmente los usuarios tienden a crear relaciones sociales con aquellos que tienen preferencias similares a ellos. En proyectos de e-commerce por ejemplo, es bastante común aplicar este tipo de escenarios pues usuarios de un grupo en común es más posible que tengan gustos parecidos.
Las relaciones buscan los grupos o usuarios que tienen tendencias más parecidas y los agrupa dentro de la red para clasificarlos.

Figura 12. Representación de social recommendation, ilustrando agrupaciones de usuarios.



Sequential Recommendation [2]

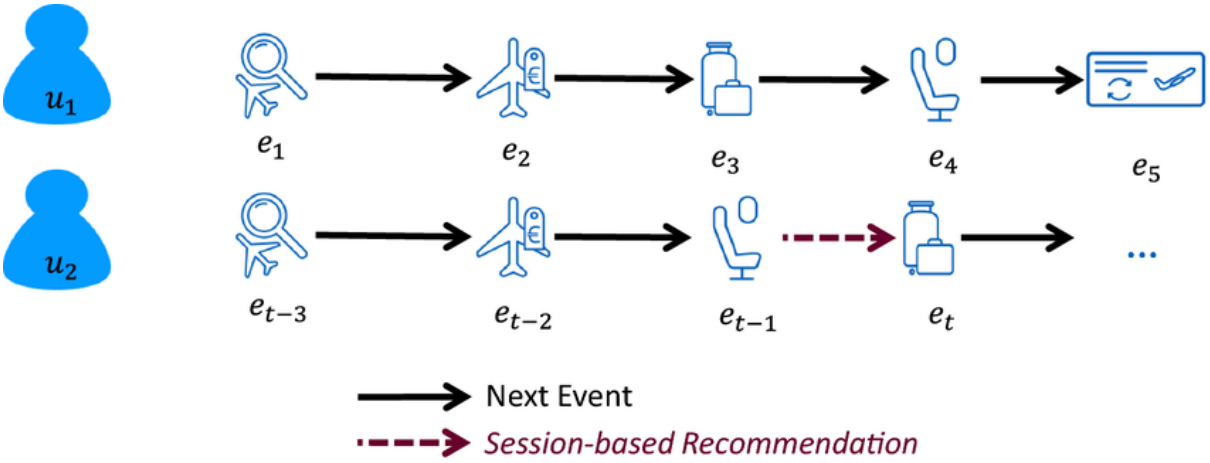
Este tipo de escenario es aquel en el que dado una lista de interacciones ordenadas el sistema es capaz de predecir cuál será la siguiente interacción del usuario debido al orden de las anteriores interacciones. Normalmente utilizados en Collaborative Filtering, poseen dos problemas a la hora de su aplicación; modelar la importancia de los ítems en la cadena de interacciones según su antigüedad y la necesidad de capturar más de una secuencia para mejorar el aprendizaje. Para su representación véase Figura 2.

[1] *Véase más en Bibliografía, Taxonomía - Social Recommendation.*
[2] *Véase más en Bibliografía, Taxonomía - Sequential Recommendation.*

Session-based Recommendation [3]

A veces es imposible monitorizar todos los comportamientos del usuario para un gran período de tiempo debido a los grandes problemas de recursos que ello conlleva, es por ello que gracias a la información anónima del usuario durante el periodo de tiempo a recomendar el sistema predice cuál será el siguiente elemento. Se diferencia a la recomendación secuencial debido a que cada sesión del usuario se procesa de manera independiente.

Figura 13. Ejemplificación de recomendación basada en sesión.



Multi-Behavior Recommendation [4]

Es aquel tipo de sistemas en el que el modelo recoge información de un mismo ítem de diferentes maneras; en un vídeo por ejemplo, el tiempo de visualización, si el usuario ha compartido el vídeo o si ha reaccionado a él de manera positiva o negativa. Existen dos retos principales con este tipo de sistemas. Diferentes influencias comportarán a mismos comportamientos, es por ello que se deberá reconocer cuáles son las señales fuertes y débiles de cada uno para una predicción precisa. Además resulta complicado reflejar las diferentes preferencias de un usuario para los ítems, solamente con su comportamiento pues éste puede tener diferentes significados, por lo cuál para obtener mejor representación los significados de los diferentes comportamientos necesitan estar integrados en el aprendizaje.

Según la fase:

Debido a los requisitos en la actualidad estos sistemas de recomendación se dividen en tres fases: *matching, ranking y re-ranking*, formando una pipeline, en la que cada una de ellas posee diferentes características sobre inputs, outputs o diseño del propio modelo. Además de éstas fases principales cada modelo de recomendación específico puede tener diferentes fases con definiciones más concretas.

[3] *Véase más en Bibliografía, Taxonomía - Session Based Recommendation.*
[4] *Véase más en Bibliografía, Taxonomía - Multi-Behavior Recommendation.*

Normalmente la cantidad de elementos a recomendar en un sistema de este tipo puede llegar a ser extremadamente alto, incluyendo millones de ítems a representar. Es por ello que el objetivo en este tipo de arquitecturas de redes neuronales multinivel sea enviar a la siguiente fase un número inferior al del nivel previo. En la actualidad las tres fases principales son el *matching, ranking y re-ranking*.

Matching

Es la primera fase del proceso, donde se generan cientos de elementos candidatos. Durante esta fase no se pueden implementar algoritmos de procesado complejos debido a las limitaciones de latencia online o la gran cantidad de información a procesar. El objetivo de esta fase es obtener elementos potencialmente relevantes y conseguir un modelo compacto de los intereses del usuario. A día de hoy se emplean diferentes sistemas de matching en un mismo modelo como sistemas de popularidad, geográficos o político-sociales.

Ranking

La fase que precede al Matching, es la fase en la que los diferentes ítems candidatos se mezclan en una lista puntuados por un único modelo de puntuación. Se vuelve a reducir la cantidad de elementos candidatos, y debido a ello se pueden utilizar algoritmos más complejos para aumentar la precisión de la recomendación.

Re-ranking

Es la fase final del proceso. Aún habiendo obtenido una lista ordenada según el sistema de ranking puede que no cumpla con otras características importantes del sistema como diversidad o que el sistema sea justo. Esta fase es importante debido a que permite reordenar los elementos con las necesidad del negocio. La mayor preocupación en esta fase es encontrar las múltiples relaciones entre los ítems más valorados para entender y validar su funcionamiento.

Según el objetivo: [5]

El objetivo principal de los sistemas recomendadores es la precisión en sus recomendaciones, además de ésta existen otros objetivos como la diversidad, la explicabilidad o la justícia de selección. Existen más objetivos, pero se hace recopilación de los más importantes.

Diversity

Se consideran dos tipos de diversidad en recomendadores, diversidad a nivel individual y a nivel de sistema. La primera de ellas mide la no similitud entre los ítems recomendados y como de balanceada está la lista de recomendación según éste supuesto. En cambio la diversidad a nivel de sistema se encarga que usuarios diferentes tengan recomendaciones distintas. En ambas encontramos dos problemas

[5] *Este apartado hace referencia a la sección 2.1.4 de la publicación Graph Neural Networks for Recommender Systems: Challenges, Methods, and Directions.*

como la fuerza de señal producida por varios ítems o la recomendación de contenido relevante ante la utilización de estas métricas.

Explainability

La gran mayoría de sistemas de hoy en día utilizan deep learning, eso desencadena en la necesidad de que el sistema sea capaz de justificar sus elecciones y no solo generar buenas recomendaciones basadas en la precisión. Esta capacidad de explicación produce transparencia y confianza que permite a otros refinar los sistemas debido su mayor entendimiento. Actualmente surgen dos obstáculos. El primero es representar esta información requiere una estructura de grafo que es difícil de representar sin las GNN. Y segundo, los razonamientos de las recomendaciones dependen del conocimiento externo del grafo, que dificulta la tarea.

Fairness

Como cualquier sistema dirigido por información, los resultados podrían estar afectados por un mal preprocesado de la información o por la aplicación de ciertos algoritmos que sesgan los resultados. Para aplicar esta característica al sistema o bien a través del entrenamiento de la red se hace que se consiga esta característica o bien se aplican en el post procesado de los datos.

- **Clasificación GNN**

A continuación se explicará la clasificación que se utilizará a nivel de GNN.
Esta consta de tres características a diferenciar; el tipo o tipos de grafo que se utilizan, el tipo de GNN que se emplea y si se utiliza alguna estructura externa para enriquecer el o los grafos, tal y como se puede ver en la siguiente tabla.

Figura 14. Tabla representativa de la clasificación a nivel de GNN.

Aspectos específicos de Session-based		
Grafos	***GNN***	***+ Structure***
Dirigido	Gated GNN	Cross Sessions
Conocimiento	GAT	Edges
Hipergrafo	HyperGCN	
Lineal		

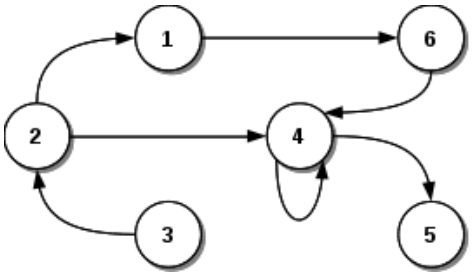
Según el tipo de grafo:

Como se ha comentado previamente existen diferentes tipos de grafo sobre los que crear la red. A continuación se comentarán los más comunes y se anunciarán algunas características.

Grafo dirigido

Un grafo dirigido es aquel en el que las aristas tienen un único sentido. En este caso, una arista se dirige desde el nodo origen hasta el nodo destino.

Figura 15. Ejemplo de grafo dirigido.



Grafo de conocimiento

El grafo de conocimiento es una base que utiliza un modelo con estructura gráfica para integrar datos. Los gráficos de conocimiento a menudo se utilizan para almacenar descripciones interrelacionadas de entidades (objetos, eventos, situaciones o conceptos abstractos), para su representación véase Figura 10.

Hipergrafo

Un hipergrafo es una generalización de un grafo, cuyas aristas aquí se llaman hiperaristas, y pueden relacionar a cualquier cantidad de vértices, en lugar de solo un máximo de dos como en el caso de los grafos. Así, un grafo es una clase particular de hipergrafos, en que cada hiperarista tiene a lo más dos vértices, (véase Figura 9).

Lineal

En teoría de grafos, el grafo lineal, es un grafo que representa las adyacencias entre las aristas de un grafo.

Además de los tipos de grafos comentados existen otros tipos que se pueden utilizar para la creación de estos sistemas, la selección se ha basado en los más comunes.

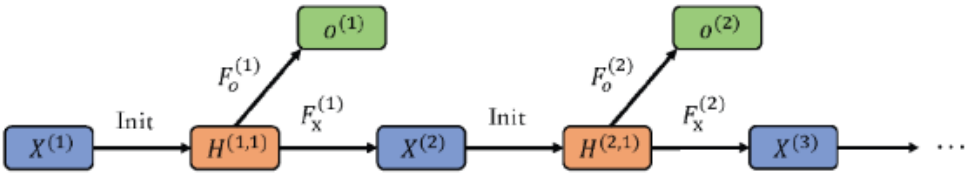
Según el tipo de GNN:

En la actualidad existen múltiples estructuras de GNN, entre las más comunes encontramos las siguientes:

Gated GNN

Son una clase de métodos de Deep Learning diseñados para realizar inferencias sobre datos descritos por grafos. Los GNN son redes neuronales que se pueden aplicar directamente a los grafos y proporcionan una manera fácil de realizar tareas de predicción a nivel de nodo, arista y grafo.

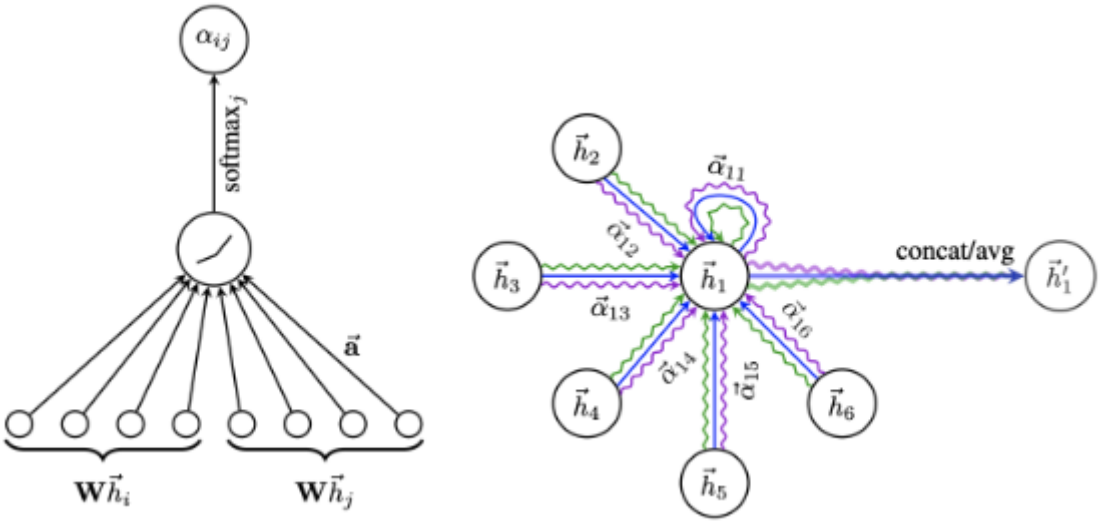
Figura 16. Representación esquemática de una Gated GNN.



GAT

Es una arquitectura de red neuronal que opera con datos estructurados en grafos, aprovechando las capas de autoatención enmascaradas para abordar las deficiencias de los métodos tradicionales basados en convoluciones de grafos.
En la siguiente figura se muestra cómo se transmite y recoge la información en este tipo de sistemas, véase Figura 16.

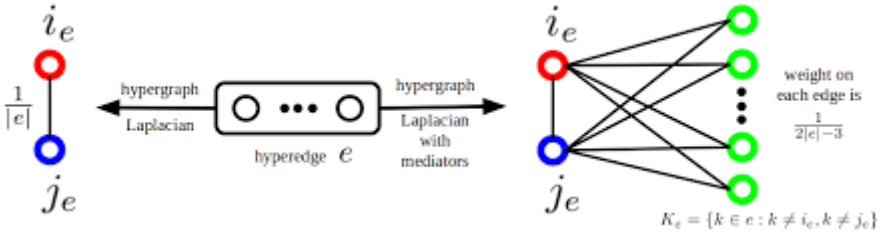
Figura 17. Representación del mecanismo de transmisión de datos en una red GAT.



HyperGCN

Es un enfoque para el aprendizaje semi-supervisado sobre datos estructurados en hipergrafos. Se basa en una variante eficiente de las redes neuronales convolucionales que operan directamente sobre hipergrafos.

Figura 18. Esquema de funcionamiento general de una HyperGCN.



Según la estructura externa adicional:

En el estado del arte actual la gran mayoría de métodos utiliza información adicional que mantiene relación con la sesión en cuestión. Principalmente estos métodos se dividen en dos.

Cross Sessions

> Durante la aplicación de este método se introduce al modelo información relativa a otras sesiones que se realizan de manera simultánea, todo ello para enriquecer las relaciones entre objetos o para descubrir de nuevas.

Additional Edges

> El concepto de este método es sencillo, aplicar relaciones guardadas en el sistema con el fin de descubrir relaciones o fortalecerlas.

En el siguiente apartado se analizarán los artículos más relevantes y se conocerá el estado del arte de los sistemas recomendadores basados en sesión, además de introducir los métodos en los que se basará el trabajo.

2.4. Descripción de los métodos o artículos más relevantes

A continuación se analizarán los métodos propuestos a estudio, todos ellos sistemas recomendadores basados en sesión, además de hacer referencia a los artículos en los que se basa este proyecto.

En primer lugar, Dual Channel Hypergraph Convolutional Networks, es un algoritmo que permite mantener correlaciones entre ítems con un alto grado de vecindad.
Se trata de un algoritmo propuesto en 2021, en el que se presenta la peculiaridad de codificar las sesiones como un hipergrafo y ejecutar convoluciones en este, además de proponer dos canales para un mismo entrenamiento uno con el hipergrafo y otro con un grafo lineal. [6]

[6] *Véase capítulo 3.2.1. DHCN.*

A continuación se presenta Session-based Recommendation with Graph Neural Network, con el que se mantendrán las transiciones complejas entre ítems y se generarán representaciones vectoriales precisas para cada sesión.
Esta arquitectura se contempla con la finalidad de resolver el comportamiento del usuario en una sesión, estimando correctamente las representaciones de usuario. Además de también resolver problemas de correlación entre ítems con contexto local o global. [7]

El siguiente método Lossless Edge-order preserving aggregation and Shortcut graph attention for Session-based Recommendation.
Utilizando este método se atacan a los dos grandes problemas de pérdida de información que se producen en los métodos basados en GNN para sistemas recomendadores de sesión; los problemas de pérdida a la hora de codificar la sesión y de capturar dependencias de alto orden de manera ineficiente.[8]

Finalmente se encuentra el SEREc / SEFrame. Dicho modelo se encarga de crear un marco de trabajo altamente personalizable además de otorgar una alternativa eficiente a los modelos tradicionales. Este sistema también captura comportamientos secuenciales, de gran importancia en sistemas basados en sesión. [9]

[7] Véase capítulo 3.2.2. SR-GNN.
[8] Véase capítulo 3.2.3. LESSR.
[9] Véase capítulo 3.2.4. SERec.

3. Implementación

A través de este apartado se mostrarán todas las características de los métodos estudiados, además de la implementación de los diferentes algoritmos en Google Collaboratory con los cambios pertinentes a los códigos originales para su ejecución.

3.1. Librerías importantes

A continuación se detallan las librerías o módulos más reseñables y comunes en la implementación de estos algoritmos.

- **Numpy**
  Es una librería de Python especializada en el cálculo numérico y el análisis de datos, especialmente para un gran volumen de datos.
  Incorpora una nueva clase de objetos llamados arrays que permite representar colecciones de datos de un mismo tipo en varias dimensiones, y funciones muy eficientes para su manipulación.
  La ventaja de Numpy frente a las listas predefinidas en Python es que el procesamiento de los arrays se realiza mucho más rápido (hasta 50 veces más) que las listas, lo cual la hace ideal para el procesamiento de vectores y matrices de grandes dimensiones.

- **Pandas**
  Pandas es una muy popular librería de código abierto dentro de los desarrolladores de Python, y sobre todo dentro del ámbito de Data Science y Machine Learning, ya que ofrece unas estructuras muy poderosas y flexibles que facilitan la manipulación y tratamiento de datos.
  Pandas surgió como necesidad de aunar en una única librería todo lo necesario para que un analista de datos pudiese tener en una misma herramienta todas las funcionalidades que necesitaba en su día a día, como son: cargar datos, modelar, analizar, manipular y prepararlos.

- **Pickle**
  El módulo pickle implementa protocolos binarios para serializar y deserializar una estructura de objetos Python. *Pickling* es el proceso mediante el cual una jerarquía de objetos de Python se convierte en una secuencia de bytes, y el *unpickling* es la operación inversa, mediante la cual una secuencia de bytes de un archivo binario o un objeto tipo binario es convertido nuevamente en una jerarquía de objetos.

- **PyTorch**
  PyTorch es una biblioteca de aprendizaje automático de código abierto basada en la biblioteca de Torch, utilizado para aplicaciones que implementan visión artificial y

procesamiento de lenguajes naturales Es un software libre y de código abierto liberado.

- **Tensorflow**
  Creado por el equipo de Google Brain, TensorFlow es una biblioteca de código abierto para la computación numérica y Machine Learning a gran escala. TensorFlow reúne una serie de modelos y algoritmos de Machine Learning y Deep Learning y los hace útiles mediante una metáfora común.
  Utiliza Python para proporcionar una práctica API para crear aplicaciones con el marco de trabajo, a la vez que ejecuta esas aplicaciones en C++ de alto rendimiento.

- **Cuda**
  CUDA son las siglas de Compute Unified Device Architecture que hace referencia a una plataforma de computación en paralelo incluyendo un compilador y un conjunto de herramientas de desarrollo creadas por Nvidia que permiten a los programadores usar una variación del lenguaje de programación C para codificar algoritmos en GPU de Nvidia.
  Por medio de wrappers se puede usar Python, Fortran, Julia y Java en vez de C/C++.

3.2. Algoritmos de recomendación

Como se ha enunciado previamente se estudiarán cuatro métodos diferentes.
Se presentará una aproximación al algoritmo, se presentarán los problemas a los que da solución el método. Para acabar, se explicará el método en cuestión y cómo resuelve los problemas anteriormente citados.
El orden de presentación de los algoritmos será el siguiente; DHCN, SR-GNN, LESSR y SERec.

Debido a la falta de recursos no se han podido lanzar todas las pruebas en una máquina en local, es por ello que se encontró la plataforma de Google Collab en la que se pueden lanzar entornos de ejecución con recursos limitados.
Se han conseguido saltar las limitaciones de la plataforma gracias a trozos de código de terceros y a poder guardar el estado del modelo en cada iteración por si se desconectaba el entorno.

En los diferentes notebooks se encontrará un algoritmo a testear, con un conjunto de celdas ordenadas ya para la ejecución, que normalmente, se corresponden con los diferentes ficheros del proyecto original.

3.2.1. DHCN

3.2.1.1. Introducción y planteamiento del problema

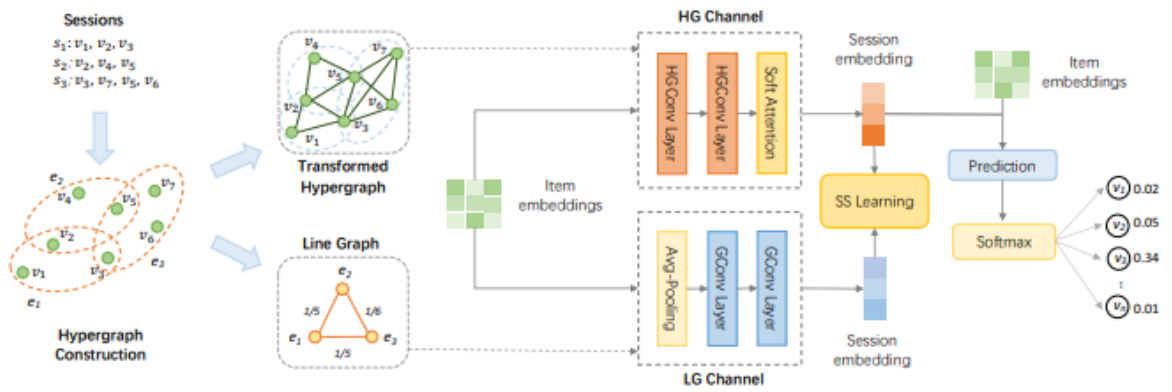
En los métodos de RNN (Recurrent Neural Net), los tradicionales, las secuencias de información unidireccionales son la clave, pues vienen generadas en un período corto de tiempo que nos lleva a pensar que pueden ser temporalmente dependientes.
Sin embargo esta afirmación puede engañar a las RNN ya que ignoran la coherencia entre ítems. Recientemente la efectividad de las GNN ha sido aplicada a muchos campos, por ejemplo en los SBR.

A diferencia de las RNN, los métodos de las GNN modelan la información de la sesión como subgrafos dirigidos y las transiciones de ítems como relaciones por parejas, lo que hace que no se asuma la dependencia temporal entre ítems consecutivos.

Los modelos tradicionales existentes solo presentan mejoras triviales comparados con las RNN. La razón principal es por el hecho de que no tienen en cuenta las correlaciones complejas entre ítems en las SBR; como el efecto conjunto de los anteriores clicks, las relaciones many-to-many o las relaciones de alto orden entre ítems. Para resolver estos problemas, se presenta un nuevo modelo basado en un hipergrafo en el que capturar estas relaciones de alto orden. Asumimos que los ítems de la sesión tienen correlación temporal pero no estrictamente de manera secuencial.

Para situar nuestro primer algoritmo lo definiremos dentro de la taxonomía que se ha visto en la sección 2.3. Se trata de un método de recomendación basado por sesiones, que cumple todas las fases que hemos definido; *Matching, Ranking y Re-ranking*. Este método usa la estructura de un hipergrafo y un grafo lineal como complementación. El tipo de estructura de GNN que utiliza pues, se trata de una HyperGCN. Finalmente, utiliza información relacionada con otras sesiones (cross-sessions), a través del uso del grafo lineal.Las siglas de **DHCN** provienen de **D**ual Channel **H**ypergraph **C**onvolutional **N**etworks.

Figura 19. Funcionamiento general de DHCN.



Técnicamente, modelamos cada sesión como una hiperarista, que nos da la flexibilidad y capacidad de mantener las complejas relaciones entre ítems. Y diferentes hiperaristas, que están conectadas entre los ítems compartidos.
Todo ello constituye el hipergrafo que contiene las diferentes correlaciones de alto nivel.

Acumulando las diferentes capas en el canal del hipergrafo, podemos tomar prestadas las fortalezas de la convolución hipergráfica para generar resultados de recomendación de mayor calidad.

Aunque, debido a que cada hiperarista contiene un número limitado de ítems, la poca densidad de información limitará estos beneficios.
Para resolver este problema introducimos el grafo lineal e integramos supervisión autónoma a nuestro modelo, todo ello para mejorar el modelado del hipergrafo. El grafo lineal se crea a través del hipergrafo modelando cada hiperarista como un nodo y centrándose en la conectividad de las hiperaristas.

3.2.1.2. Construcción del modelo de datos

Para capturar las relaciones en el recomendador de sesión, adoptamos la estructura de hipergrafo para representar cada sesión como una hiperarista. Y cada hipervértice como los diferentes ítems. En el grafo lineal, cada sesión es modelada como un nodo y las diferentes sesiones están conectadas a través de los ítems en común. El peso de cada arista es el número de ítems conectados entre el total.

En el código se puede encontrar esta conversión de las sesiones a los diferentes grafos desde la línea 39 del main.py. Esta llamada instancia el objeto DHCN en el que se configuran los diferentes hiperparámetros de la red y donde se crean los dos objetos que representan los grafos, el hipergrafo y el grafo lineal, véase Figura 20.

Figura 20. Clase DHCN. [10]

```
class DHCN(Module):
    def __init__(self, adjacency, n_node,lr, layers,l2, beta,dataset,emb_size=100, batch_size=100):
        super(DHCN, self).__init__()
        self.emb_size = emb_size
        self.batch_size = batch_size
        self.n_node = n_node
        self.L2 = l2
        self.lr = lr
        self.layers = layers
        self.beta = beta
        self.dataset = dataset

        values = adjacency.data
        indices = np.vstack((adjacency.row, adjacency.col))
        if dataset == 'Nowplaying':
            index_fliter = (values < 0.05).nonzero()
            values = np.delete(values, index_fliter)
            indices1 = np.delete(indices[0], index_fliter)
            indices2 = np.delete(indices[1], index_fliter)
            indices = [indices1, indices2]
        i = torch.LongTensor(indices)
        v = torch.FloatTensor(values)
        shape = adjacency.shape
        adjacency = torch.sparse.FloatTensor(i, v, torch.Size(shape))
        self.adjacency = adjacency
        self.embedding = nn.Embedding(self.n_node, self.emb_size)
        self.pos_embedding = nn.Embedding(200, self.emb_size)
        self.HyperGraph = HyperConv(self.layers,dataset)
        self.LineGraph = LineConv(self.layers, self.batch_size)
        self.w_1 = nn.Linear(2 * self.emb_size, self.emb_size)
        self.w_2 = nn.Parameter(torch.Tensor(self.emb_size, 1))
        self.glu1 = nn.Linear(self.emb_size, self.emb_size)
        self.glu2 = nn.Linear(self.emb_size, self.emb_size, bias=False)
        self.loss_function = nn.CrossEntropyLoss()
        self.optimizer = torch.optim.Adam(self.parameters(), lr=self.lr)
        self.init_parameters()
```

- **Hypergraph**

El desafío principal de diseñar una convolución para un hipergrafo es cómo propagar las representaciones vectoriales de los ítems. La convolución del hipergrafo puede ser vista como dos procesos de refinamiento 'nodo-hipervértice-nodo'. El interés del usuario está representado a través de la agregación de las representaciones vectoriales usando mecanismos de atención en los que los ítems tienen diferentes niveles de prioridad. Durante esta fase abandonamos las técnicas de modelado de secuencias y otros modelos de atención en SBR, para que el único factor temporal sea la posición de la representación vectorial, esto hace que el modelo sea poco pesado y eficiente.

- **Lineal Graph**

Que la información de la sesión sea escasa puede llevar a recomendaciones subóptimas, para resolver este problema aplicamos el aprendizaje autosupervisado.

El primer paso del proceso es diseñar otra *Graph Convolutional Network* creada a través del grafo lineal con la que generaremos las señales de supervisión. Maximizando la información

[10] Figura 20, se hace referencia a la creación de los dos grafos necesarios, el hipergrafo y el grafo lineal.

mútua obtenida a través de los dos canales usando aprendizaje de contraste, con información positiva y negativa.
A continuación se muestra cómo se implementa este tipo de red dentro del fichero de model.py del proyecto de DHCN.

Figura 21. Clase LineConv, representación del grafo lineal.

```
class LineConv(Module):
    def __init__(self, layers,batch_size,emb_size=100):
        super(LineConv, self).__init__()
        self.emb_size = emb_size
        self.batch_size = batch_size
        self.layers = layers
    def forward(self, item_embedding, D, A, session_item, session_len):
        zeros = torch.cuda.FloatTensor(1,self.emb_size).fill_(0)
        # zeros = torch.zeros([1,self.emb_size])
        item_embedding = torch.cat([zeros, item_embedding], 0)
        seq_h = []
        for i in torch.arange(len(session_item)):
            seq_h.append(torch.index_select(item_embedding, 0, session_item[i]))
        seq_h1 = trans_to_cuda(torch.tensor([item.cpu().detach().numpy() for item in seq_h]))
        session_emb_lgcn = torch.div(torch.sum(seq_h1, 1), session_len)
        session = [session_emb_lgcn]
        DA = torch.mm(D, A).float()
        for i in range(self.layers):
            session_emb_lgcn = torch.mm(DA, session_emb_lgcn)
            session.append(session_emb_lgcn)
        #session1 = trans_to_cuda(torch.tensor([item.cpu().detach().numpy() for item in session]))
        #session_emb_lgcn = torch.sum(session1, 0)
        session_emb_lgcn = np.sum(session, 0)/ (self.layers+1)
        return session_emb_lgcn
```

Como se puede comprobar esta clase setea los hiperparámetros necesarios y define cómo se transmitirá la información a las siguientes capas a través de la función forward.
El grafo lineal puede verse como un grafo que contiene las diferentes sesiones y la semejanza entre estas. Es por ello que en cada convolución las sesiones recogen información de sus vecinos, así pues podemos capturar las relaciones entre estas.

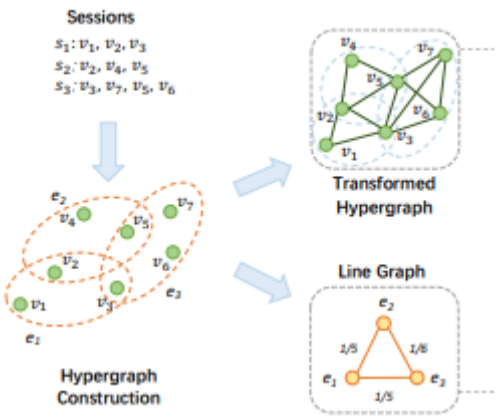
3.2.1.3. Señales de supervisión

Hasta ahora aprendemos a través de los dos canales, el hipergrafo y el grafo lineal. Cada canal codifica la información a nivel de ítem o de sesión, esto hace que cada uno de los grupos sepan poco del otro pero que se puedan complementar.
Directamente los dos grupos pueden reflejar información importante para el otro a través del aprendizaje autosupervisado, es decir, al estar nutriendo a la red con dos canales distintos podemos utilizar la información resultante para nutrirlos entre sí.

3.2.1.4. Explicación del método y funcionamiento

El DHCN es un algoritmo que se aplica para la recomendación de ítems en una sesión. Este método recoge la información de diferentes sesiones con el que crea un hipergrafo, donde los hipernodos serán los objetos contenidos en las sesiones y las hiperaristas las diferentes sesiones que relacionan los objetos entre sí (véase Figura 22). A partir de este hipergrafo se crean dos estructuras con las que se trabajará en la red.

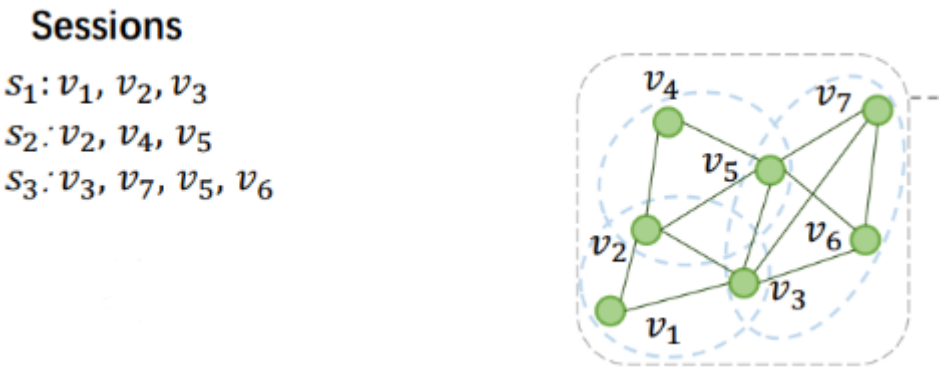
Figura 22. Transformación de las sesiones a los dos tipos de grafos.



Por una parte, un grafo lineal encargado de mantener las informaciones de las sesiones y cómo se vinculan entre ellas. Siendo los nodos las representación de las sesiones y las aristas denotaran si los nodos en cuestión mantienen objetos en común, dando otorgando peso a cada arista; siendo el peso el número de objetos en común entre la suma de objetos totales de ambas sesiones.

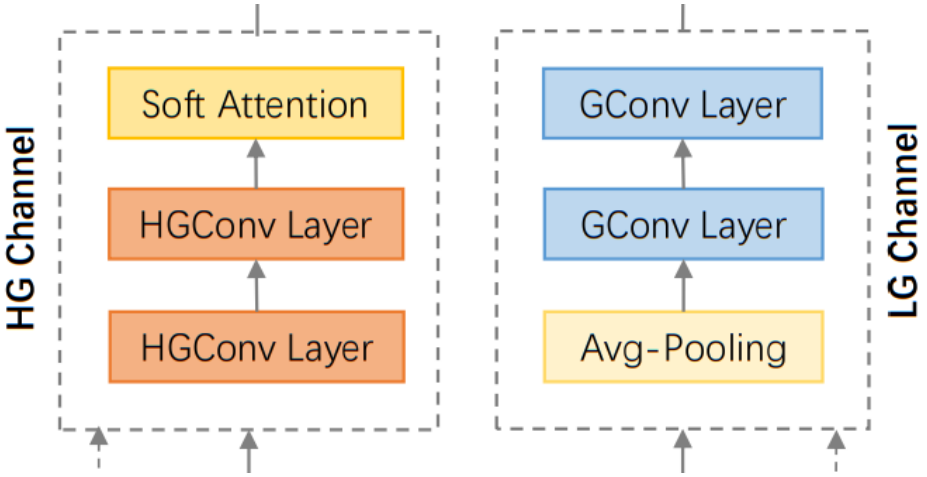
Por la otra parte, la transformación del hipergrafo, encargada de mantener la información de los ítems y las relaciones que mantienen. La representación de los vértices son los diferentes objetos y sus conexiones indicarán si mantienen relación en alguna sesión. Como se puede observar en la Figura 23, el nodo v2 forma parte de las sesiones s1 y s2. Es por ello que v2 mantiene relación tanto con v1 y v3, que forman parte de s1; como con v4 y v5 que conforman s2.

Figura 23. Transformación de las sesiones a hipergrafo.



Obtenidas ya ambas partes se aplicarán una serie de convoluciones a nivel de grafo que nos permitirán conservar las relaciones de vecindad de alto orden, uno de los objetivos en este tipo de recomendadores. Estas convoluciones se harán en ambos grafos, pero serán diferentes en forma, aunque ambas utilizarán las representaciones vectoriales de los ítems.

Figura 24. Representación de las convoluciones en DHCN. [11]



Obtenidas ya las representaciones vectoriales de los grafos convolucionados se usará el hipergrafo transformado para entrenar a la red auto-supervisada y el grafo lineal como soporte de ground-truth. Una vez entrenada la red, dada una sesión, calcularemos las puntuaciones para todos los ítems candidatos aplicando el inner product entre la representación aprendida del hipergrafo transformado y la representación de la sesión.

Finalmente, ordenaremos los ítems candidatos de mayor a menor puntuación y devolveremos dicha lista con los parámetros a desear.

3.2.2.5. Revisión del código

A través de esta sección veremos un par de apuntes reseñables de la implementación a través del proyecto original de Python.

El proyecto se divide en tres ficheros:

- main.py: Controlador del proyecto, desde este fichero se hacen las diferentes llamadas al entreno de la red y obtención de predicciones.
- model.py: Se definen las dos clases de los grafos además de la clase principal DHCN.
- util.py: Contiene la clase Data y los métodos relacionados con la partición y tratamiento de los datos.

A continuación se adjunta una captura de cómo se obtienen las puntuaciones, que en resumidas cuentas, se consiguen gracias a la representación vectorial de la Sesión devuelta por el hipergrafo y la representación vectorial del ítem calculada gracias al grafo lineal.

[11] *Figura 24, aplicación de las diferentes convoluciones tanto en el hipergrafo (izquierda) como en el grafo lineal (derecha).*

Figura 25. Obtención de las predicciones. [12]

```
print('start predicting: ', datetime.datetime.now())

model.eval()
slices = test_data.generate_batch(model.batch_size)
for i in slices:
    ## Llamada de Forward de DHCN, dentro del Forward de DHCN se llama a al forward de HyperG y LineG.
    ## El forward de DHCN devuelve el sess_emb_hgnn devuelto a través del forward de HyperG.
    ## El forward de DHCN devuelve el item_emb_hg devuelto a través del forward de LineG.
    ## Ambos se utilizan para calcular el score general, a través de la siguiente línea en el forward general.
    ## scores = torch.mm(sess_emb_hgnn, torch.transpose(item_emb_hg, 1,0))
    tar, scores, con_loss = forward(model, i, test_data)
    scores = trans_to_cpu(scores).detach().numpy()
```

Además de algún pequeño cambio para solucionar warnings de rendimiento se han pasado los hiper parámetros directamente al modelo con las opciones default y en el caso del parámetro n_node con el adecuado para cada dataset.

3.2.2. SR-GNN

3.2.2.1. Introducción y planteamiento del problema

El segundo algoritmo vuelve a tratarse de un algoritmo de recomendación basado en sesiones, que vuelve a implementar las tres fases de estos modelos.
La estructura del grafo del SR-GNN se compone de grafos dirigidos, y su estructura de GNN se centra en una gated GNN. Las Gated GNN, como se han introducido anteriormente en la taxonomía, son un tipo de red neuronal que utiliza unidades recurrentes cerradas y técnicas modernas de optimización, que luego se extienden a las secuencias de salida.

El objetivo del presente método es predecir las acciones del usuario en sesiones anónimas. Algoritmos previos a éste modelaban la sesión como una secuencia de datos, en la que se estimaban las representaciones de los usuarios además de la de los objetos.
En el método propuesto, las sesiones son estructuradas como grafos. Utilizando las GNN podemos capturar las transiciones complejas entre ítems, que són complicadas de obtener con los métodos tradicionales.

La gran mayoría de sistemas de recomendación asumen que el sistema posee información del usuario y las actividades pasadas que este realizó. Sin embargo, en muchos de estos servicios el usuario no dispone la capacidad de identificarse y la única información que tenemos de este es la de la propia sesión.
Aunque el estado del arte está desarrollado y ofrece buenas alternativas a estos problemas existen algunas limitaciones.

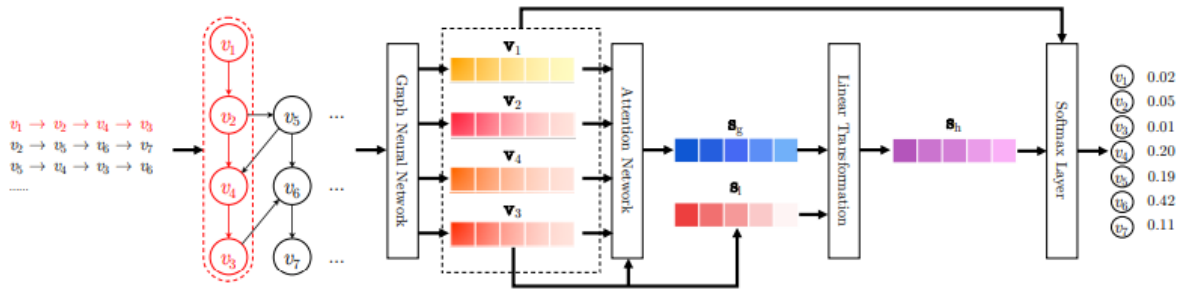
La primera de ellas es que sin adecuar el comportamiento del usuario en una sesión, los métodos tienen dificultad en estimar las representaciones de usuario.

[12] *Figura 25, código de model.py (obtención de las predicciones).*

El segundo de estos puntos trata sobre los patrones de transición entre ítems. Trabajos previos demuestran que estos patrones son importantes y pueden ser usados como factores locales en recomendadores de sesión. El problema surge cuando todos los métodos modelan las transiciones de manera individual y omiten las transiciones entre contextos o ítems en otras sesiones.

Para solventar estas limitaciones se propone el siguiente método; **S**ession-based **R**ecommendation with **G**raph **N**eural **N**etwork, con el que mantendremos las transiciones complejas entre ítems y se generarán vectores latentes precisos.

Figura 26. Esquema general de SR-GNN.



3.2.2.2. Construcción del modelo de datos

Para los recomendadores de sesión, primero se construyen grafos dirigidos a través del histórico de secuencias. A través del grafo de la sesión, la GNN es capaz de capturar las transiciones entre ítems y generar representaciones vectoriales precisas, que es la limitación de los métodos convencionales.

Figura 27. Partición de los datos para el entreno. [13]

```
train_data = Data(train_data, sub_graph=True, method='ggnn', shuffle=True)
test_data = Data(test_data, sub_graph=True, method='ggnn', shuffle=False)
model = GGNN(hidden_size=100, out_size=100, batch_size=100, n_node= 48727,
                lr=0.001, l2=1e-5,  step=1, decay=3 * len(train_data.inputs) / 100, lr_dc=0.1,
                nonhybrid='store_true')
print('CREACION DEL MODELO')
best_result = [0, 0,0,0,0,0,0,0]
best_epoch = [0, 0,0,0,0,0,0,0]
for epoch in range(30):
    print('epoch: ', epoch, '==========================================')
    slices = train_data.generate_batch(model.batch_size)
    fetches = [model.opt, model.loss_train, model.global_step]
    print('start training: ', datetime.datetime.now())
    loss_ = []
    for i, j in zip(slices, np.arange(len(slices))):
        adj_in, adj_out, alias, item, mask, targets = train_data.get_slice(i)
        _, loss, _ = model.run(fetches, targets, item, adj_in, adj_out, alias,  mask)
        loss_.append(loss)
    loss = np.mean(loss_)
    slices = test_data.generate_batch(model.batch_size)
    print('start predicting: ', datetime.datetime.now())
```

Al principio, todas las sesiones se modelan como grafos dirigidos, en el que cada sesión puede ser tratada como un subgrafo. Todo seguido se representa cada sesión como una composición del interés general y del interés del usuario actual, donde estas representaciones globales y locales están compuestas por los vectores latentes de los nodos. Finalmente, para cada sesión se predice la probabilidad de que cada ítem sea clicado en la próxima iteración.

3.2.2.3. Generación de las representaciones vectoriales

Los métodos convencionales de recomendación asumen que existe una representación latente distinta para el usuario en cada sesión. Por el contrario, utilizando el SR-GNN no hacemos suposiciones de este tipo.

Para predecir mejor los siguientes clicks del usuario planeamos desarrollar una estrategia que combine las preferencias a largo término con los intereses actuales de la sesión con el fin de crear la representación final de la sesión.

Después de alimentar a la GNN obtenemos los vectores de todos los nodos. Para representar cada sesión consideramos un vector local que estará definido por el último ítem clicado y un vector global en el que agregaremos todos los vectores de los ítems. Finalmente, calcularemos la representación vectorial final como la transformación lineal de la concatenación de ambos vectores. Esta parte se produce a partir de la línea 69 de la primera celda del fichero SR_GNN.ipynb.

[13] *Figura 26, se muestra en qué parte del código se entrena el modelo para cada partición de datos.*

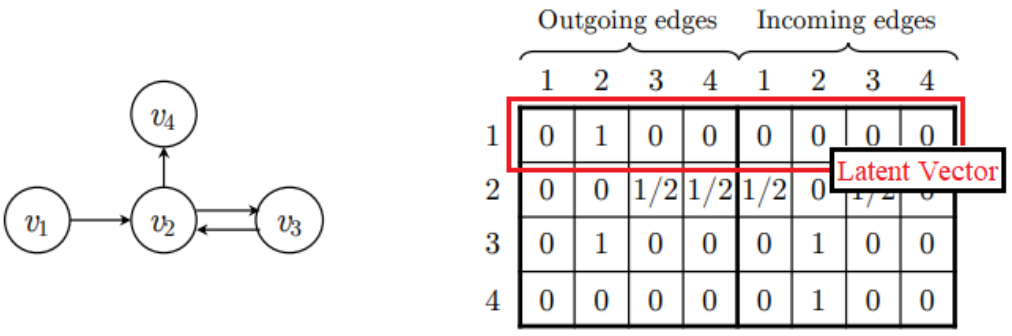
3.2.2.4. Explicación del modelo y funcionamiento

Cada secuencia de sesión se modela como un grafo dirigido, donde cada nodo representa un ítem y cada arista la transición entre objetos. Dado que múltiples ítems pueden aparecer en más de una ocasión se asigna le un peso normalizado a cada arista, que se calcula a través de la ocurrencia de la arista entre el grado del nodo del inicio de la arista.

Utilizando los vectores latentes de cada ítem, cada sesión puede ser representada por un conjunto de vectores, y el conjunto de esta será una parte de la representación vectorial, véase Figura 28.

Este tipo de sistema es capaz de mantener diferentes matrices de conexión para diferentes grafos de sesión. Si existen diferentes estrategias de construcción dicha matriz cambiará respectivamente.

Figura 28. Ejemplo de representación de sesión. [14]



Utilizando este tipo de sistemas conseguimos que se propague la información a través de los nodos bajo las restricciones de la matriz, específicamente extrae los vectores latentes de los vecinos y los alimenta como input a la GNN. Más adelante dos puertas son las encargadas de elegir qué información se mantiene y se descarta.

Después de obtener la representación vectorial de cada sesión, se calcula la puntuación para cada ítem candidato multiplicando su representación vectorial por la representación de la sesión.

3.2.3. LESSR

3.2.3.1. Introducción y planteamiento del problema

Para este método se vuelve a repetir la casuística de recomendadores basados en sesión, con todas las fases del proceso incluídas. La estructura de este método se basa en el uso de grafos

[14] *Figura 28, representación de la conversión de una sesión a las diferentes representaciones vectoriales de los objetos.*

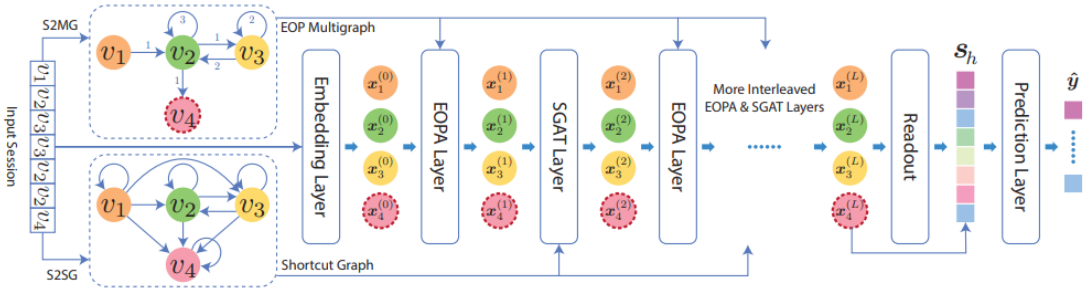
dirigidos sobre una GNN de tipo GAT, además durante el proceso se utilizan aristas adicionales para complementar las predicciones.

El objetivo de este es atacar a los dos grandes problemas de pérdida de información que se producen en los métodos basados en GNN para sistemas recomendadores de sesión; los problemas de pérdida a la hora de codificar la sesión y de capturar dependencias de alto orden de manera ineficiente.

El primer problema hace referencia a que alguna información sobre las transiciones entre objetos se ignora debido a cómo se codifican las sesiones a grafos y las permutaciones invariantes durante la transmisión de la información.
El segundo problema trata de la ineficiencia de cómo se capturan las transiciones entre vecinos de alto orden.

Para la resolución del primer problema se propone un esquema de codificación sin pérdida además de una capa de preservación de relaciones basada en GRU.
En cambio para el segundo problema se propone un atajo a la capa de atención, que captura las relaciones de alto orden propagando la información a través de las conexiones de bajo orden.
Combinando ambas capas se solucionan los problemas de pérdida de información.

Figura 29. Esquema de funcionamiento de LESSR.



El primer problema de pérdida de información en los métodos basados en GNN se debe a cómo se convierten las sesiones a grafos. Para procesar sesiones utilizando GNNs, las sesiones deben estar convertidas a grafos primero.
En el método propuesto cada sesión se convierte en un grafo dirigido en el que los nodos son objetos únicos en la sesión y las relaciones entre estos son las transiciones ocurridas durante la sesión, estas relaciones pueden tener un peso o no.

Pongamos como ejemplo una sesión, S1 = {v1, v2, v3, v3, v2, v2, v4} que se convierte a grafo y resulta en la siguiente figura, véase Figura 30.

Tal y cómo se convierten las sesiones en los modelos GNN podría surgir una segunda sesión tal que S2 = {v1, v2, v2, v3, v3, v2, v4} que resultaría en el mismo grafo.

Esto conlleva a que en determinadas ocasiones ambas sesiones podrían reproducir mismos resultados aún siendo diferentes. Todo esto acaba repercutiendo en la predicción final en la que estos modelos no son capaces de dar buenas recomendaciones cuándo surgen este tipo de cuestiones.

Figura 30. representación de la Sesión (S1).

El segundo problema trata de la ineficiencia de estos modelos de capturar todas las relaciones de alto orden.

En cada capa de un modelo GNN, la información transmitida por los nodos es propagada a través de las aristas para cada paso, así pues, cada capa puede capturar únicamente las relaciones de un paso. Apilando múltiples capas, el modelo GNN puede capturar tantas relaciones como número de capas tenga este.
Desde que apilar capas no necesariamente incrementa el rendimiento del modelo, debido a problemas de *overfitting* y *over-smoothing*, el número óptimo de capas normalmente no supera a tres. Esto conlleva a que las relaciones entre ítems que superen este número no se conservan y existen importantes patrones secuenciales que son mayores a esto.

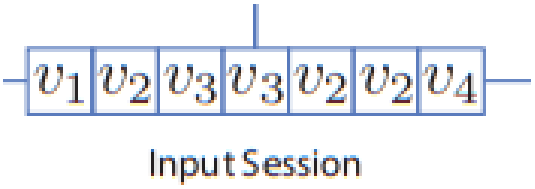
Para resolver los problemas mencionados se presenta el **L**ossless **E**dge-order preserving aggregation and **S**hortcut graph attention for **S**ession-based **R**ecommendation.

3.2.3.2. Métodos de conversión a grafos

Para procesar sesiones utilizando GNNs, éstas han de convertirse previamente a grafos. Durante esta sección introduciremos dos métodos: S2MG y S2SG.
El primero de ellos se encargará de procesar las sesiones a multigrafo, **S**ession **T**o **M**ulti-**G**raph. En cambio, el segundo las convertirá en el grafo de acceso directo, **S**ession **T**o **S**hort**C**ut Graph.

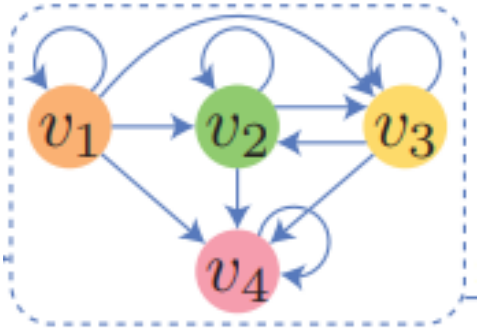
En la literatura de los sistemas recomendadores diferenciamos dos tipos de algoritmos para convertir las sesiones a grafos.

Figura 31. Representación de la Sesión 'input'.



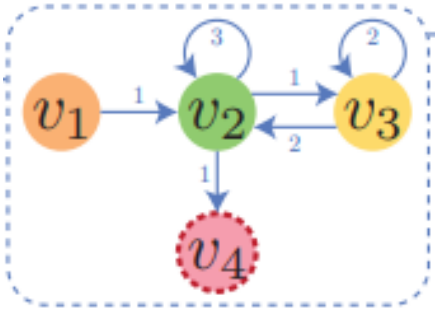
El primer método, S2G (**S**ession **T**o **G**raph), consiste en convertir una sesión a un grafo sin pesos donde el conjunto de nodos consiste únicamente en los ítems de la sesión, y las relaciones entre estos se establecen en forma de aristas.

Figura 32. Grafo de la Sesión 'input' usando S2G.



De otra forma, el segundo método propone dar pesos a estas relaciones con las veces que ha surgido esa transición en la sesión, este método se llama S2WG (**S**ession **T**o **W**eighted **G**raph).

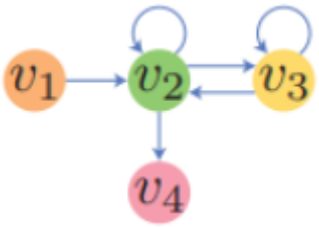
Figura 33. Grafo de la sesión 'input' usando S2WG.



El problema con estos métodos surgen cuándo no se es capaz de reconstruir la sesión dado el grafo, para confirmar estas afirmación únicamente necesitamos probar que S2WG captura más información que su análogo el S2G, por ejemplo, con el número de transiciones entre objetos.
Para demostrar que S2WG también pierde información durante la conversión lo ejemplificaremos con dos sesiones.

Las sesiones son SX = {v1,v2,v3,v3,v2,v2,v4} y SY = {v1,v2,v2,v3,v3,v2,v4}.
Si las convertimos utilizando S2WG nos codifica ambas sesiones como el grafo de la Figura 34, una vez dado este grafo no podemos saber cuál de las dos sesiones pertenece a la representación ya que es ambiguo.

Figura 34. Representación de SX y SY usando S2WG.



- **EOPA**

Para resolver estos problemas se propone el S2MG, que convierte cada sesión en un multigrafo dirigido (EOPA) que conserva el orden de relaciones. Para cada transición en la sesión original se crea una arista. Se considera multigrafo pues para cada relación entre dos nodos se creará una arista. Después, cada nodo se ordenará por número de veces que aparece en la sesión y se ordenará todo el conjunto en función del número de ocurrencias. El último nodo de la sesión se marcará distintamente para comprobar que es el final de las transiciones.

- **SGAT**

El otro punto a tener en cuenta es cómo capturar las relaciones entre vecinos de alto orden. La solución que se propone es utilizar un grafo directo de atención, SGAT (**S**hortcut **G**raph **A**ttention).
Esta capa requiere que se introduzca un grafo diferente al EOP multigrafo, es por ello que se utiliza el S2SG para obtener el grafo de entrada.

Dada una sesión, se crea el grafo en el que el conjunto de nodos son los objetos únicos de la sesión y donde las aristas de este son las transiciones recíprocas entre objetos.

Este grafo se denomina de acceso directo porque conecta los objetos sin ítems intermediarios, se añaden bucles en los nodos para que después la capa SGAT pueda pasar el mensaje a los nodos vecinos.

A continuación se muestra la parte de código donde se instancia ambas capas; dentro del __init__ del objeto LESSR.

Figura 35. Inicialización de ambas capas SGAT y EOPA. [15]

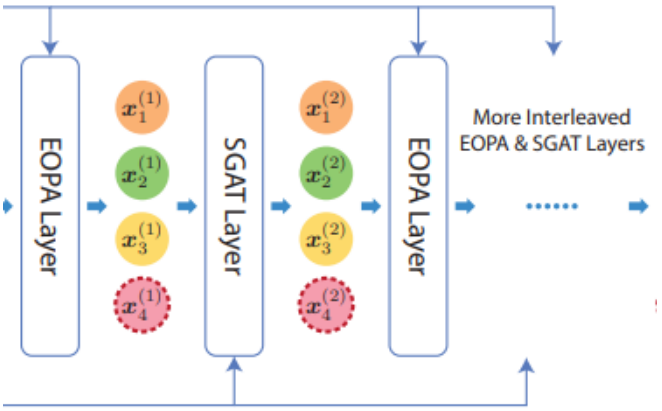
```
for i in range(num_layers):
    if i % 2 == 0:
        layer = EOPA(
            input_dim,
            embedding_dim,
            batch_norm=batch_norm,
            feat_drop=feat_drop,
            activation=nn.PReLU(embedding_dim),
        )
    else:
        layer = SGAT(
            input_dim,
            embedding_dim,
            embedding_dim,
            batch_norm=batch_norm,
            feat_drop=feat_drop,
            activation=nn.PReLU(embedding_dim),
        )
    input_dim += embedding_dim
    self.layers.append(layer)
```

3.2.3.3. Explicación del modelo y funcionamiento

Ambas capas se han propuesto con la finalidad de solventar los dos problemas principales de las GNNs en sistemas recomendadores. En vez de conectar las capas EOPA después de las SGAT, estas se intercalan por las siguientes razones.

El grafo de acceso directo es un método de conversión con pérdida, así pues agrupar múltiples capas SGAT produciría el mismo problema de pérdida. Intercalando ambas capas, la pérdida de información puede ser capturada y retenida en la siguiente capa EOPA y la capa SGAT puede centrarse en capturar las dependencias de alto orden, véase Figura 35.

Figura 36. Intercalado de capas SGAT y EOPA en el modelo.



Otra ventaja de intercalar ambas capas es que este tipos de capas pueden utilizar características de otro tipo de capas. En este caso, las capas EOPA son capaces de capturar el contexto local de la información; y las SGAT capturan las dependencias globales. Intercalando ambas obtenemos ambas ventajas y mejoramos las capacidades del modelo aprendiendo relaciones más complejas.

[15] *Figura 35, código de la segunda celda de LESSR.ipynb dentro de la clase LESSR.*

Después de pasar por todas las capas se obtienen las representaciones finales de todos los nodos. Para representar la sesión actual como una representación vectorial, se aplica una función de lectura en la que se computa la representación a nivel de grafo agregando nodos.

Ya para acabar, después de la función de lectura se aplica la capa de predicción en la que se ordenarán los ítems en función de su relevancia.

3.2.2.5. Revisión del código

A través de esta sección se mostrarán apuntes importantes del algoritmo.
Para empezar el formato del conjunto de datos de entrada para LESSR es diferente a DHCN y SR-GNN, donde únicamente se necesitaban dos ficheros de texto. En este algoritmo además de los ficheros de train y test se deberá incluir un fichero num_items que incluirá el número de ítems únicos. Es por ello que se han hecho tres métodos de preprocesado diferentes para los diferentes conjuntos de datos empleados.
El preprocesado de Diginetica es una copia a un método del repositorio original.[16]
Los otros dos toman partes de este pero se han modificado para el correcto funcionamiento en cada *dataset*.

Otro apunte importante es que LESSR hace uso específico de la librería dgl. DGL es un paquete de Python creado para facilitar la implementación de la familia de modelos de redes neuronales de grafos. Ofrece un control versátil del envío de mensajes, la optimización de la velocidad a través del *auto-batching* y los núcleos de matriz dispersa altamente ajustados, además del entrenamiento multi-GPU/CPU para escalar gráficos de cientos de millones de nodos y aristas.

En relación al código es un código limpio en el que se instancia la clase LESSR. Donde al crear el objeto se crean las diferentes capas que van asociadas a las clases de EOPA y SGAT, (véase Figura 37).

[16] Acceso al fichero referenciado, enlace:
https://github.com/twchen/lessr/blob/master/preprocess.py

Figura 37. Clase LESSR.

```
class LESSR(nn.Module):
    def __init__(
        self, num_items, embedding_dim, num_layers, batch_norm=True, feat_drop=0.0
    ):
        super().__init__()
        self.embedding = nn.Embedding(num_items, embedding_dim, max_norm=1)
        self.indices = nn.Parameter(
            th.arange(num_items, dtype=th.long), requires_grad=False
        )
        self.num_layers = num_layers
        self.layers = nn.ModuleList()
        input_dim = embedding_dim
        for i in range(num_layers):
            if i % 2 == 0:
                layer = EOPA(
                    input_dim,
                    embedding_dim,
                    batch_norm=batch_norm,
                    feat_drop=feat_drop,
                    activation=nn.PReLU(embedding_dim),
                )
            else:
                layer = SGAT(
                    input_dim,
                    embedding_dim,
                    embedding_dim,
                    batch_norm=batch_norm,
                    feat_drop=feat_drop,
                    activation=nn.PReLU(embedding_dim),
                )
            input_dim += embedding_dim
            self.layers.append(layer)
        self.readout = AttnReadout(
            input_dim,
            embedding_dim,
            embedding_dim,
            batch_norm=batch_norm,
            feat_drop=feat_drop,
            activation=nn.PReLU(embedding_dim),
        )
        input_dim += embedding_dim
        self.batch_norm = nn.BatchNorm1d(input_dim) if batch_norm else None
        self.feat_drop = nn.Dropout(feat_drop)
        self.fc_sr = nn.Linear(input_dim, embedding_dim, bias=False)
```

Además de estas clases en la inicialización de la clase LESSR se encuentra la clase AttnReadout, encargada de computar la representación de la sesión juntamente con los intereses a corto plazo del usuario, (véase Figura 37).

3.2.4. SERec

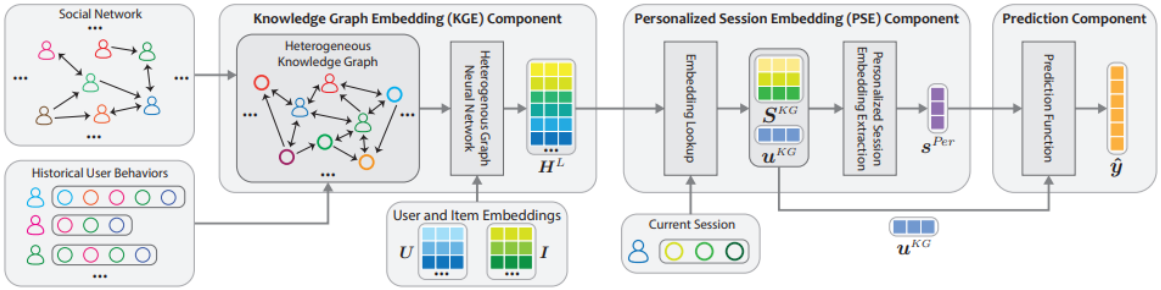
3.2.4.1. Introducción y planteamiento del problema

El primer problema que surge con el estado actual del arte en los sistemas recomendadores es la poca eficiencia de estos. Para predecir el próximo ítem que recomendar a un usuario los métodos necesitan procesar más sesiones para otorgar influencias sociales o de grupo, y esto de manera individual.

Para solventar dicho problema se propone un framework para recomendaciones basadas en sesión.

En primer lugar se utilizará una red creada a partir de un grafo heterogéneo para aprender las representaciones de usuario e ítems, en las que se integrará información relativa a la red social.
A continuación, para generar la predicción, únicamente el usuario y las representaciones de ítems relevantes se transmitirán al modelo que no tiene las relaciones sociales. Durante este proceso, como las representaciones vectoriales se pueden precomputar, el modelo global se ejecutará tan rápido como lo haga el NSA.

Figura 38. Esquema general de SERec.



Además de ser eficiente el modelo ofrece dos ventajas adicionales.
La primera de ellas, la flexibilidad. La estructura del framework ofrece la posibilidad de ser compatible con cualquier NSA modelo.
La segunda, es la capacidad que provee de capturar transiciones entre diferentes sesiones cuándo los métodos tradicionales solamente capturan las relaciones de la propia sesión.

3.2.4.2. Presentación de los componentes

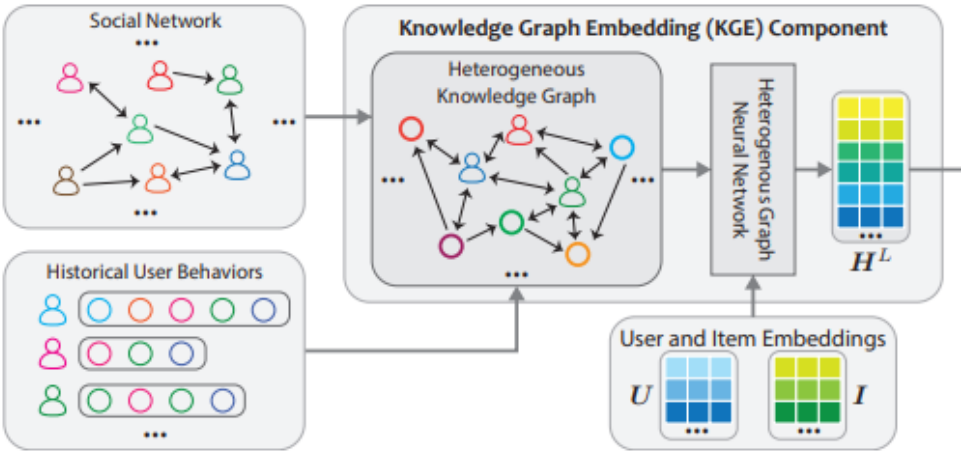
En este método se diferencian tres grandes componentes, de los cuáles veremos dos en detalle; el KGE y el PSE.

- **KGE Knowledge Graph Embedding**

El componente KGE se encarga principalmente de dos tareas. La primera, crear un grafo de conocimiento heterogéneo a partir de todos los comportamientos de usuario y la red social de estos. La segunda, aprender las representaciones del usuario y los ítems que combinan el conocimiento en el grafo heterogéneo a través de la HGNN o heterogeneous graph neural network.

En dicho grafo, las representaciones de los nodos serán todos los usuarios y ítems recogidos a partir de el histórico de comportamientos y la red social de usuarios.

Figura 39. Esquema del componente KGE.



El grupo de aristas se define como el subgrupo de aristas de estos cuatro tipos:

Usuario-Usuario:
Definen la relación A es seguido por B. Se utiliza el 'seguido por' en vez del 'sigue' porque las GNNs actualizan los nodos utilizando las aristas que reciben estos y eso hace que los usuarios sean más influenciables por los usuarios que siguen que por los que les siguen a ellos. El peso de estas aristas es de uno.

Usuario-Ítem / Ítem-Usuario:
Representan las interacciones de los usuarios con los ítems. El peso de las aristas viene definido por el número de veces que ocurre la interacción.

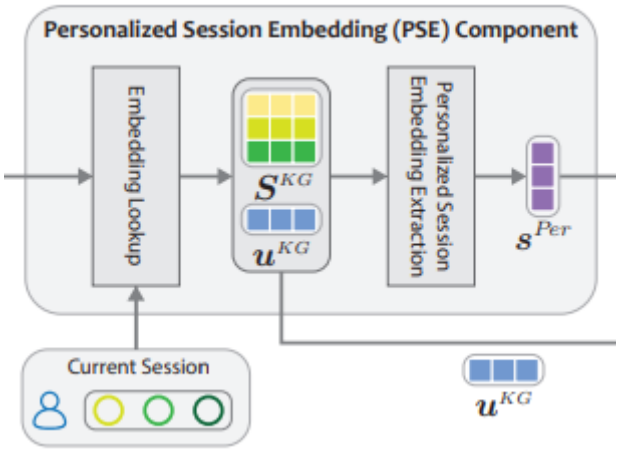
Ítem-Ítem:
Representan las transiciones entre ítems en el conjunto de las sesiones. El peso de estas aristas viene definido por el número de veces que ocurre.

- **PSE Personalized Session Embedding**

La segunda tarea de este framework es aplicar la HGNN para aprender las representaciones, donde las preferencias del usuario capturarán las influencias sociales y, donde los ítems la información colaborativa que hay en el grafo.
De todas formas, el comportamiento del usuario en la sesión actual es de vital importancia para determinar los intereses dinámicos.

Es por ello que el PSE genera una representación específica personalizada de la sesión. En ella se capturan las preferencias actuales del usuario y las propiedades de los ítems en dicho contexto.

Figura 40. Esquema del componente PSE.



En la primera etapa, para capturar las preferencias, se ejecuta una operación llamada *Embedding Lookup*. Esta operación permite extraer las representaciones vectoriales relevantes del usuario y los diferentes ítems.

Durante la segunda etapa, se ejecuta la *personalized session embedding extraction.* La cual se encarga de extraer calcular los vectores personalizados de los ítems de la sesión actual a partir del *Embedding Lookup*.

3.2.4.3. Explicación del modelo y funcionamiento

En primer lugar SEFrame crea un grafo heterogéneo de conocimiento a partir de la red social y el historial global de los comportamientos de los usuarios.
A partir de este grafo se crea una red neuronal de la que se aprenderán las representaciones del usuario y los ítems más relevantes, en las que se toman en cuenta relaciones sociales, interacciones usuario-ítem y transiciones entre ítems.
Una vez obtenidas estas representaciones, se transmiten al modelo PSR o Personalized Session-based Recommendation. Y finalmente, cómo estas representaciones contienen información del conjunto, dicho modelo puede aprovechar el conocimiento de la red social para proveer las recomendaciones finales.

Podemos decir que este framework es eficiente pues la información del usuario y las representaciones de los ítems pueden estar precomputadas a través de inferencia. Por lo tanto, durante la inferencia, el modelo SSR únicamente necesita procesar la sesión actual, que es igualmente de eficiente que el modelo original PSR.

4. Análisis y resultados

Durante el siguiente capítulo se explicarán al detalle los diferentes conjuntos de datos, el escenario experimental, las implementaciones realizadas de los métodos y sus respectivos resultados.

4.1. Descripción de los datasets

Para realizar los diferentes experimentos se han utilizado tres conjuntos de datos diferentes; dos principales Diginetica y TMall, y Retail Rocket como complementario.

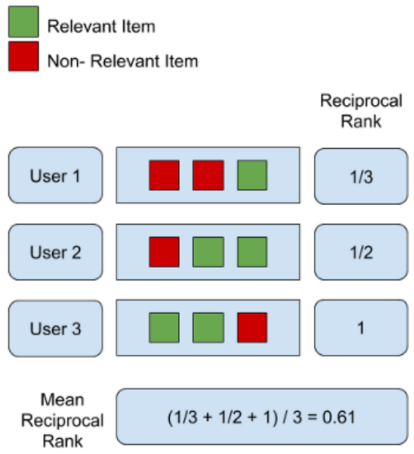
El primero de ellos, Diginetica, es un dataset de e-commerce utilizado para el entrenamiento de modelos predictivos basados en sesión. Este dataset está formado por diferentes archivos csv. De estos podemos obtener atributos como los diferentes identificadores necesarios y los timestamps de las acciones, además de otros atributos complementarios.

Por otro lado tenemos a TMall. Este dataset está compuesto por los datos recogidos de Taobao y TMall, dos páginas chinas de compra por internet, además de información recopilada por la aplicación china de Alipal. A destacar de este dataset tenemos que no tiene identificadores de sesión, es por ello que, para las pruebas necesarias de los algoritmos se han agrupado las diferentes acciones entre días, una práctica bastante común en conjuntos de datos que no tienen estos atributos.

En tercer lugar, Retail Rocket originado a través de información de comercios web reales presenta información sobre transacciones, carritos de compra, clicks además de otras interacciones durante un período de tiempo de 4 meses y medio.
Finalmente, se presentan las métricas utilizadas para el estudio y evaluación de los métodos; MRR y el Hit.

El MRR es el rango recíproco medio. También se conoce como ratio de aciertos recíprocos medios (ARHR). Trata de medir "¿Dónde está el primer elemento relevante?".

Figura 41. Ejemplo de cálculo de MRR.



$$\mathrm{MRR} = \frac{1}{Q}\sum_{i=1}^{Q}\frac{1}{\mathrm{rank}_i}$$

Para acabar, el Hit es la proporción de aciertos. Simplemente la fracción de usuarios para los que la respuesta correcta está incluida en la lista de recomendaciones de longitud L.[17]

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

4.2. Detalle de los datos

En este proyecto se utilizarán tres de los datasets más reconocidos y utilizados para comparar algoritmos en la literatura: Diginetica, TMall y RetailRocket.

Diginetica es un dataset para recomendadores de sesión utilizado para e-commerce. Este conjunto de datos se basa en seis archivos .csv y una carpeta comprimida con las imágenes de los diferentes productos. En este dataset el fichero más importante a utilizar es train-item-views.csv de 42.7MB. En este fichero se recoge la siguiente información:

- sessionId
- userId
- itemId
- timeframe, (tiempo desde la primera consulta de una sesión, en milisegundos).
- eventdate, (*datetime*).

Un ejemplo seguiría este patrón: *1;;81766;526309;2016-05-09*

Diginetica	*Views*	*Purchases*	*Users*	*Items*
	1.235.380	18.025	232.816	184.047

TMall, por otro lado, es también un dataset para sistemas de recomendación basados en sesión que se creó a través de las páginas chinas Taobao y Tmall, además de recoger información de la aplicación Alipay. Este conjunto de datos está formado por cuatro archivos .csv del que se puede sacar información. TMall no tiene un sessionId por default, así que para lanzar en LESSR se ha definido un intervalo y se han agrupado todos los movimientos de cada usuario. El intervalo definido ha sido de un día, sin ningún motivo justificable sino igual que se utiliza para Gowalla, otro conjunto de datos importante en este ámbito.

Finalmente, Retail Rocket publicado a través de la misma empresa con el mismo nombre, consta de cinco ficheros de información de un e-commerce. Retail Rocket tiene la misma

[17] *Para más información sobre las métricas: véase Métricas I y Métricas II en Bibliografía.*

peculiaridad que TMall, no tiene un identificador de sesión. Dado este hecho las sesiones se harán en agrupaciones en días sobre las interacciones de un mismo usuario.

RetailRocket	*Transactions*	*Users*	*Items*
	2.756.101	1.407.580	235.061

4.3. Escenario experimental

El escenario donde se han podido realizar los experimentos principalmente ha sido en la plataforma de Google Collab, debido al largo tiempo de procesamiento de las operaciones se han ido guardando estados intermedios para guardar los progresos, tal y como se ha comentado en la sección 3.2. Las librerías que vienen instaladas por defecto en el entorno son suficientes para lanzar todos los experimentos, a excepción de LESSR, que se ha hecho la puntuación necesaria en el apartado 3.2.3.5.

También se ha utilizado un ordenador de sobremesa con una tarjeta gráfica Nvidia GeForce 1080 con 8 GBytes de RAM para el caso del DHCN.

Se ha habilitado un repositorio de Github con todos los Collabs utilizados, el proyecto de DHCN y los diferentes métodos de preprocesado.

https://github.com/SamuelCalabriaC/Recommender-Systems

En dicho repositorio se han hecho los cambios necesarios para poder lanzar las pruebas en Collab, menos en DHCN como se ha comentado previamente.

4.4. Descripción de los métodos comparados

Esto trata de una recapitulación de los diferentes métodos a comparar, que son tres; DHCN, SR-GNN y LESSR.
Cada uno de ellos propone algo diferente a los métodos tradicionales que hace evolucionar el estado del arte.

En primer lugar, DHCN. Este método consiste en modelar los datos como un hipergrafo, para después crear una red convolucional de dos canales basada a partir de este hipergrafo. La utilización del hipergrafo permite a la red mantener relaciones de alto orden, que es la propuesta de valor que ofrece este método.

Se seguirán los experimentos con el SR-GNN. Dicho algoritmo se centra en el reconocimiento de patrones y la representación vectorial correcta del usuario. Cada secuencia de sesión se modela como un grafo dirigido, donde cada nodo representa un ítem y cada arista la transición entre objetos. El procedimiento consiste en concatenar diferentes matrices de sesión y realizar operaciones sobre estas, para finalmente obtener una puntuación a través de una multiplicación entre el vector del ítem candidato y la representación de la sesión.

Y finalmente, analizaremos el LESSR. Este método se basa en resolver los dos grandes problemas de pérdida de información que se producen en los métodos basados en GNN para sistemas recomendadores de sesión; los problemas de pérdida a la hora de codificar la sesión y la captura ineficiente de dependencias de alto orden.

4.5. Resultados

Para la ejecución de las diferentes pruebas en DHCN y SR-GNN se ha utilizado el propio conjunto de datos codificados que se proporciona en la página de la publicación de DHCN. En cambio para LESSR se han tenido que preprocesar los datos para cada lanzamiento. Además de comparar los datos entre los diferentes conjuntos se compararan con los resultados en uno de los métodos asentados en la literatura GRU4REC durante el apartado 4.6.1.[18]
Después de ver todas las métricas de los algoritmos se hará una comparativa entre ellos para comparar cual, en los conjuntos de datos estudiados ha dado mejor rendimiento.

4.5.1. DHCN

Empezaremos analizando los resultados por orden de presentación.
El DHCN, uno de los pilares actuales en el estado del arte.
Los resultados obtenidos han sido los mejores, como ya comentan sus creadores, las consecuencias de su aplicación son una mejora considerable sobre los métodos tradicionales, ya que consideran la relación secuencial en los datos.

Los resultados obtenidos son los esperados, bastante parecidos a los encontrados en el artículo de referencia. Los resultados obtenidos en TMall son inferiores a los de Diginetica y Retail Rocket.

Resultados DHCN

	Hit@5	MRR@5	Hit@10	MRR@10	Hit@20	MRR@20
Diginetica	27,74	15,60	39,59	17,16	52,89	18,04
Epoch	*19*	*19*	*19*	*17*	*10*	*14*
TMall	19,13	12,92	24,21	13,53	29,25	13,88
Epoch	*3*	*13*	*6*	*13*	*6*	*13*
RR	37,89	25,04	45,98	26,78	52,52	27,12
Epoch	*8*	*8*	*8*	*8*	*8*	*8*

[18] *Se puede consultar toda la información del método en este repositorio de Github: https://github.com/hidasib/GRU4Rec .*

4.5.2. SR-GNN

En segundo lugar revisaremos los resultados del SR-GNN.
Los resultados obtenidos han sido los peores de la comparativa, ya que se trata del método más antiguo. Las puntuaciones siguen siendo inferiores en TMall, siendo bastante parecidas en Retail Rocket y Diginetica.

Resultados SR-GNN

	Hit@5	MRR@5	Hit@10	MRR@10	Hit@20	MRR@20
Diginetica	25,34	14,52	36,50	16,06	51,34	17,81
Epoch	3	5	5	3	3	3
TMall	18,41	12,24	23,40	13,11	27,54	13,36
Epoch	7	8	3	3	7	7
RR	37,46	25,42	45,21	26,47	52,38	25,42
Epoch	5	5	3	3	3	3

Seguidamente compararemos los resultados obtenidos con los obtenidos del algoritmo GRU4REC.

4.5.3. LESSR

Y finalmente LESSR, con unos resultados ligeramente inferiores que el DHCN. Este algoritmo se postula como uno de los nuevos trabajos a tener en cuenta, su velocidad de recomendación es bastante superior a los dos métodos anteriores y devuelve unos resultados ligeramente parecidos a DHCN, un método con grandes resultados.

Resultados LESSR

	Hit@5	MRR@5	Hit@10	MRR@10	Hit@20	MRR@20
Diginetica	27,70	14,77	39,36	16,32	52,56	17,95
Epoch	13	12	9	12	12	12
TMall	21,45	13,44	28,15	14,33	34,72	14,79
Epoch	3	4	4	4	3	3
RR	34,71	22,42	44,46	23,76	54,47	24,46
Epoch	13	10	10	10	10	10

4.5.4. Comparativa métodos

Ya para acabar haremos unas comparativas entre los métodos estudiados a través de los diferentes conjuntos de datos.
En primer lugar, se mostrará el Hit@5 y el MRR@5 para el Dataset de Diginetica.

Figura 42. Comparativa Diginetica Hit@5

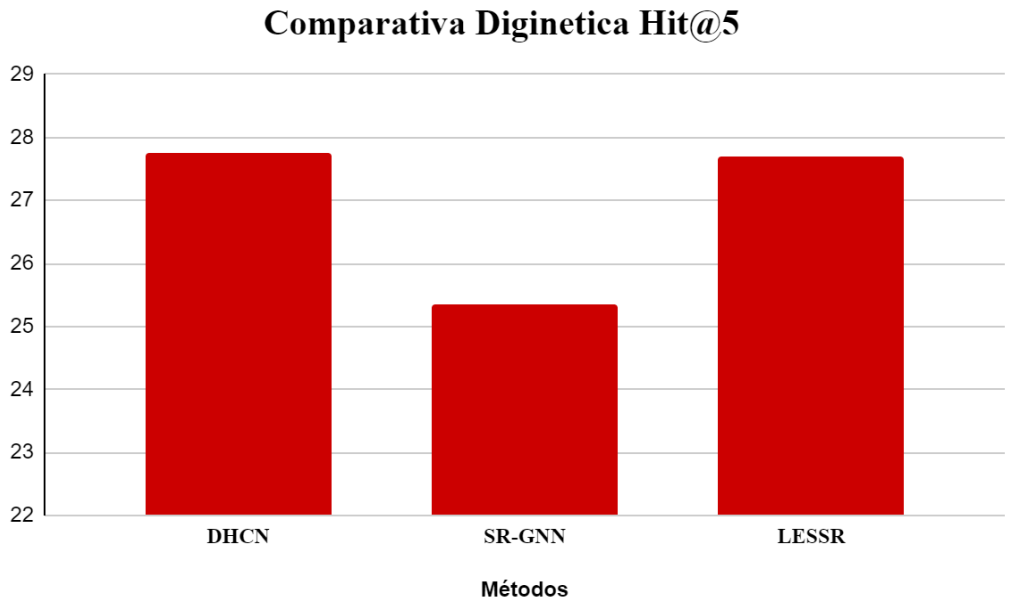
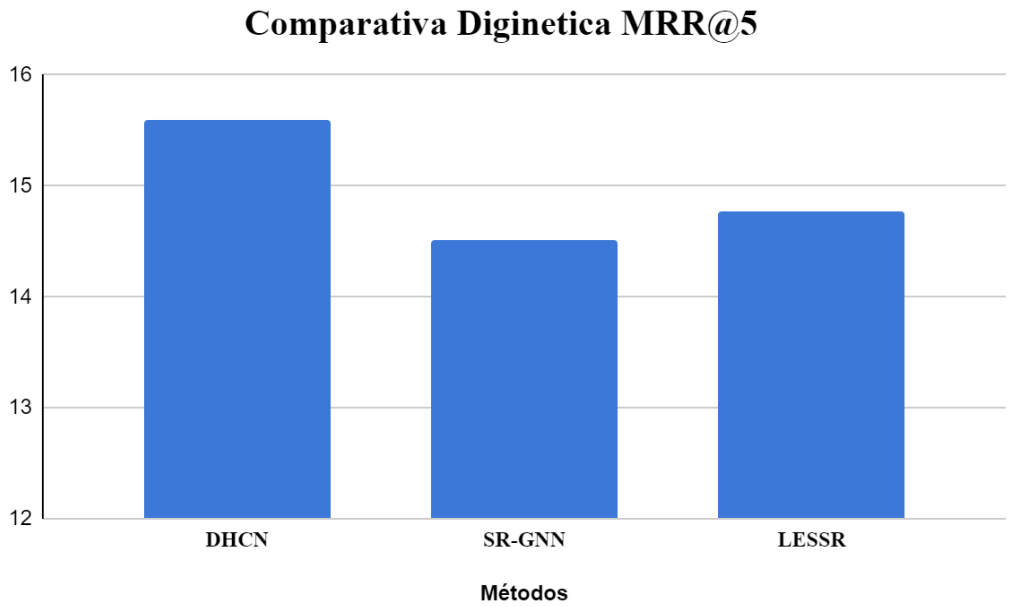


Figura 43. Comparativa Diginetica MMR@5



Como se puede comprobar tanto DHCN como LESSR obtienen resultados similares para este *dataset*, estos resultados se preveían ya que SR-GNN se posiciona como el algoritmo más sencillo dentro de los tres estudiados y se suponía que daría menos rendimiento. Podemos observar que LESSR se distancia a DHCN en MRR@5 y esto nos confirma que aunque ambos

métodos consiguen acertar el mismo número de ítems correctos DHCN es capaz de predecir más ítems correctos a la hora de observar sesiones de longitud cinco.

A continuación revisaremos los mejores resultados obtenidos en la métrica MRR@20 y Hit@20 con todos los datasets.

Figura 44. Comparativa Hit@20.

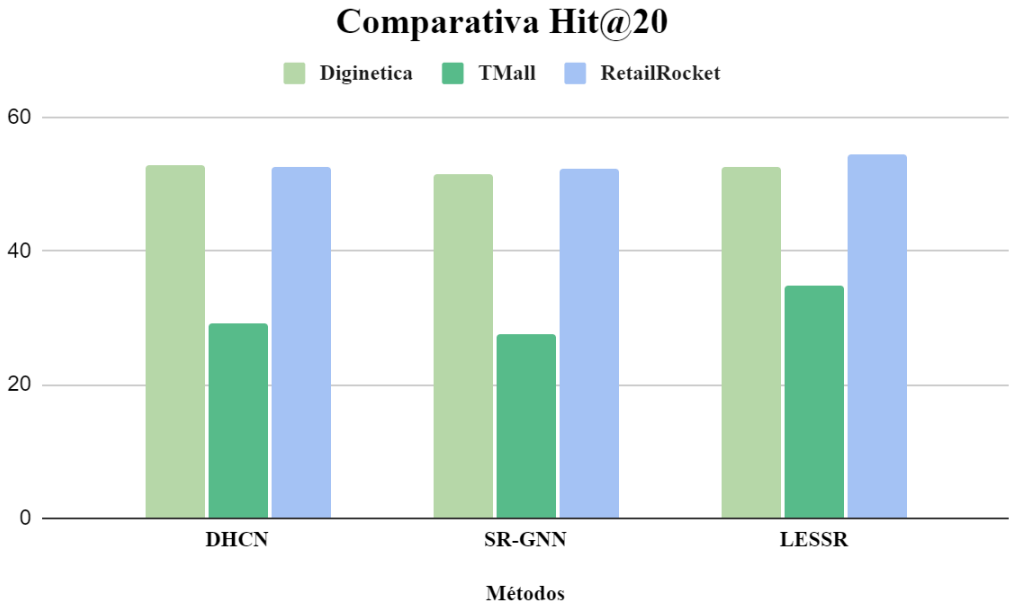
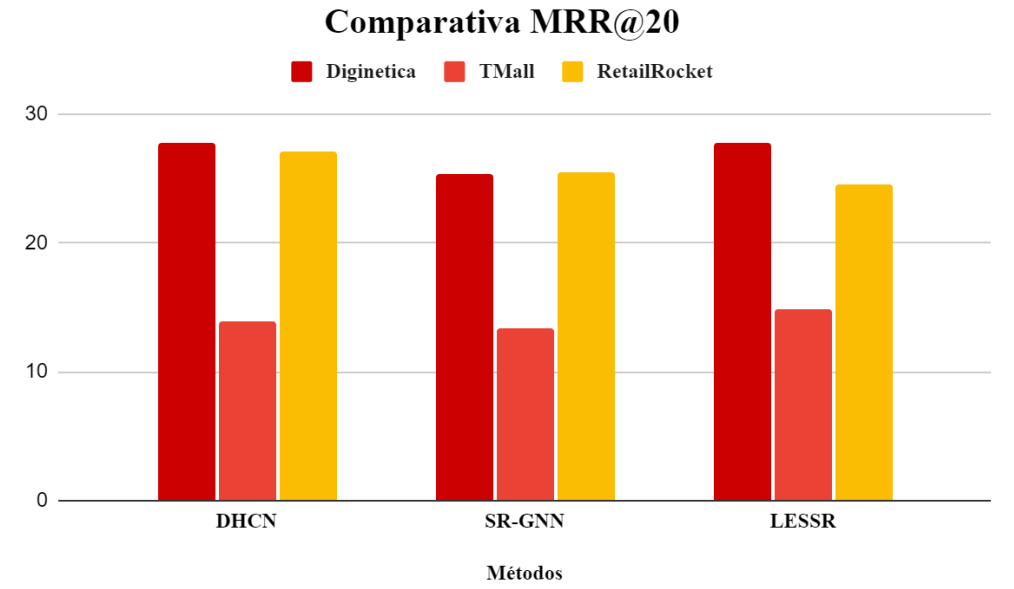


Figura 45. Comparativa MRR@20.



Los resultados obtenidos para Hit@20 y MRR@20 nos confirman que DHCN sigue postulándose como el mejor algoritmo, LESSR le sigue de cerca pero teniendo un comportamiento irregular a través de los diferentes conjuntos de datos.

Aunque los tres algoritmos ofrecen puntuaciones similares, DHCN sigue siendo aquel que de media predice más objetos relevantes en las sesiones de longitud veinte, igual que hemos mencionado para aquellas de cinco.

Gracias a estos gráficos podemos concluir que, el método SR-GNN se trata de un algoritmo muy regular y sus recomendaciones no tienen muy en cuenta la naturaleza del conjunto de datos, ya que para todos los conjuntos de datos ha obtenido puntuaciones similares, observado durante todas las gráficas.

Por otro lado, LESSR, como se ha mencionado anteriormente, nos indica que puede ser un algoritmo igualmente potente que DHCN, en términos de precisión, pero es más susceptible a los datos del conjunto haciendo que su rendimiento no sea tan regular, como podemos observar en la Comparativa de MRR@20.

Finalmente, DHCN ha obtenido los mejores resultados en conjunto siendo el algoritmo más potente y siendo fiable a través de todos los conjuntos de datos.

4.6. Discusión y resumen final

Gracias a las diferentes pruebas hemos podido comprobar diferentes tipos de algoritmos con diferentes conjuntos de datos. Los resultados obtenidos han ido acorde con lo esperado comparado con los resultados de los diferentes *papers*. Con ello hemos podido reafirmar que este tipo de sistemas son una clara mejora a los métodos tradicionales, gracias al nuevo enfoque que aportan al campo.

Dicho enfoque se centra en recoger los gustos del usuario específico, lo que hace que no se requiera datos de otros usuarios. El hecho de que no requiera de muchos datos hace que este tipo de sistemas sean escalables, un factor diferenciador en la actualidad.

4.6.1 Comparativa con resultados de GRU4REC

Durante esta subsección compararemos los resultados obtenidos de nuestras pruebas con resultados de un método tradicional utilizando el mismo dataset. En este caso el método asentado de la literatura es GRU4REC del que sacaremos sus métricas a través de dos publicaciones, la propia de DHCN y Graph-Enhanced Multi-Task Learning of Multi-Level Transition Dynamics for Session-based Recommendation ya que se utiliza unos de los métodos estudiados, SR-GNN en los que las métricas obtenidas son muy parecidas a las conseguidas en el proyecto.

La comparativa con DHCN en las métricas de Hit@10 y MRR@10 es la siguiente.

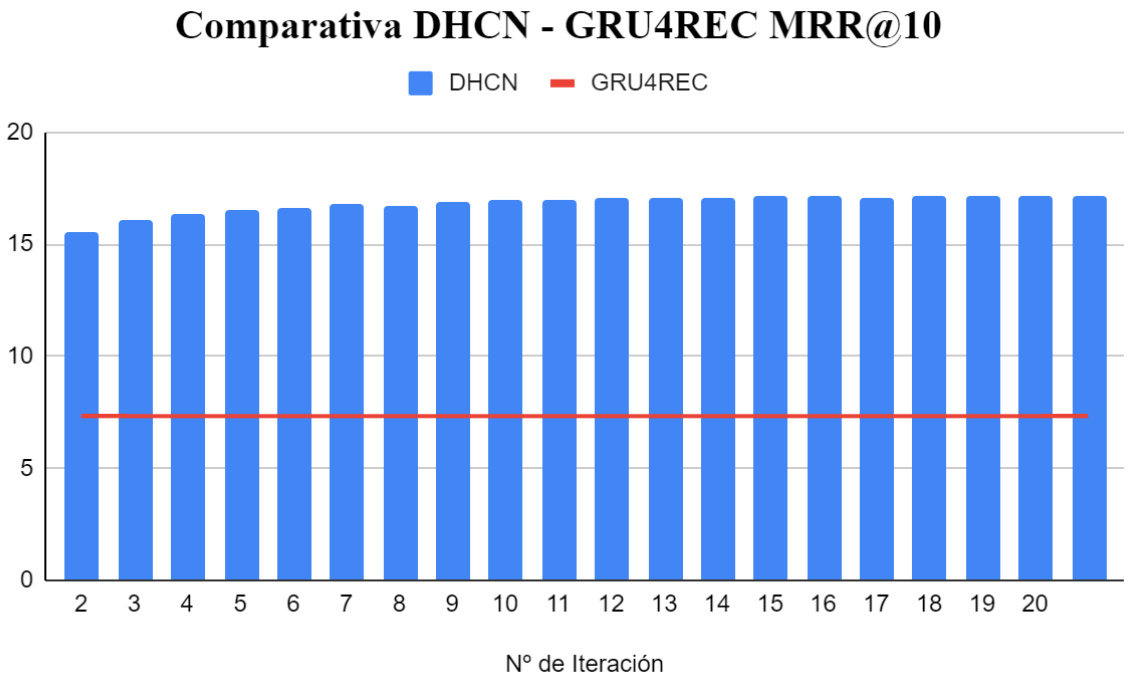
Comparativa DHCN - GRU4REC

		Hit@10	**MRR@10**
Diginetica	*DHCN*	39,59	17,16
	GRU4REC	*17,93*	*7,33*
TMall	*DHCN*	24,21	13,53
	GRU4REC	*9,47*	*5,78*
RetailRocket	*DHCN*	45,98	26,78
	GRU4REC	*31,01*	*14,96*

Como se puede comprobar en los tres conjuntos de datos DHCN ha conseguido mejores puntuaciones en las métricas de Hit@10 y MRR@10.

Gracias a poder guardar información de todas las iteraciones poder hacer una comparativa entre la mejora a través de las iteraciones de DHCN y el mejor resultado de GRU4REC, con ello se observa que desde la primera iteración DHCN provee de mejores resultados que el método tradicional, véase a través de la siguiente gráfica.

Figura 46. Comparativa DHCN-GRU4REC.



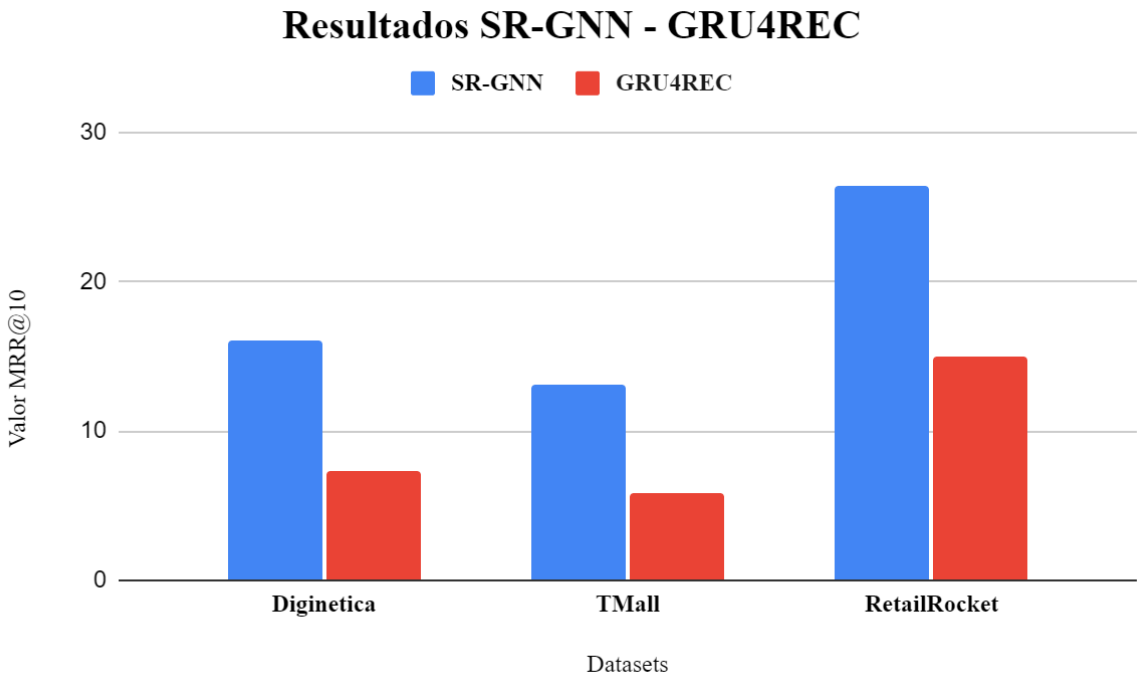
A continuación veremos la misma comparativa para el algoritmo SR-GNN utilizando las mismas métricas.

Comparativa SR-GNN - GRU4REC

		Hit@10	MRR@10
Diginetica	*SR-GNN*	36,50	16,06
	GRU4REC	*17,93*	*7,33*
TMall	*SR-GNN*	23,40	13,11
	GRU4REC	*9,47*	*5,78*
RetailRocket	*SR-GNN*	45,21	26,47
	GRU4REC	*31,01*	*14,96*

Como podemos comprobar SR-GNN aún no siendo tan potente como DHCN sigue mostrando una mejora abismal con GRU4REC, en todos los conjuntos de datos. A continuación se mostrarán los resultados en forma gráfica para la métrica de MRR@10.

Figura 47. Comparativa SR-GNN - GRU4REC.



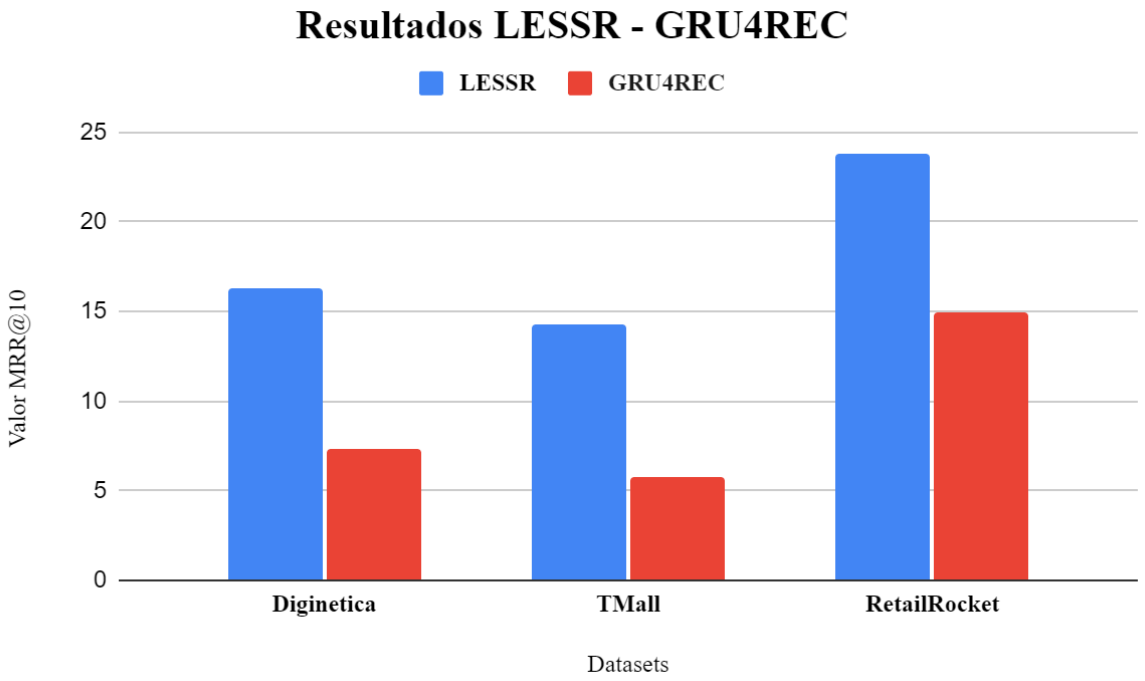
Seguidamente compararemos los resultados obtenidos con los obtenidos del algoritmo LESSR.

Comparativa LESSR - GRU4REC

		Hit@10	MRR@10
Diginetica	*SR-GNN*	39,36	16,32
	GRU4REC	*17,93*	*7,33*
TMall	*SR-GNN*	28,15	14,33
	GRU4REC	*9,47*	*5,78*
RetailRocket	*SR-GNN*	44,46	23,76
	GRU4REC	*31,01*	*14,96*

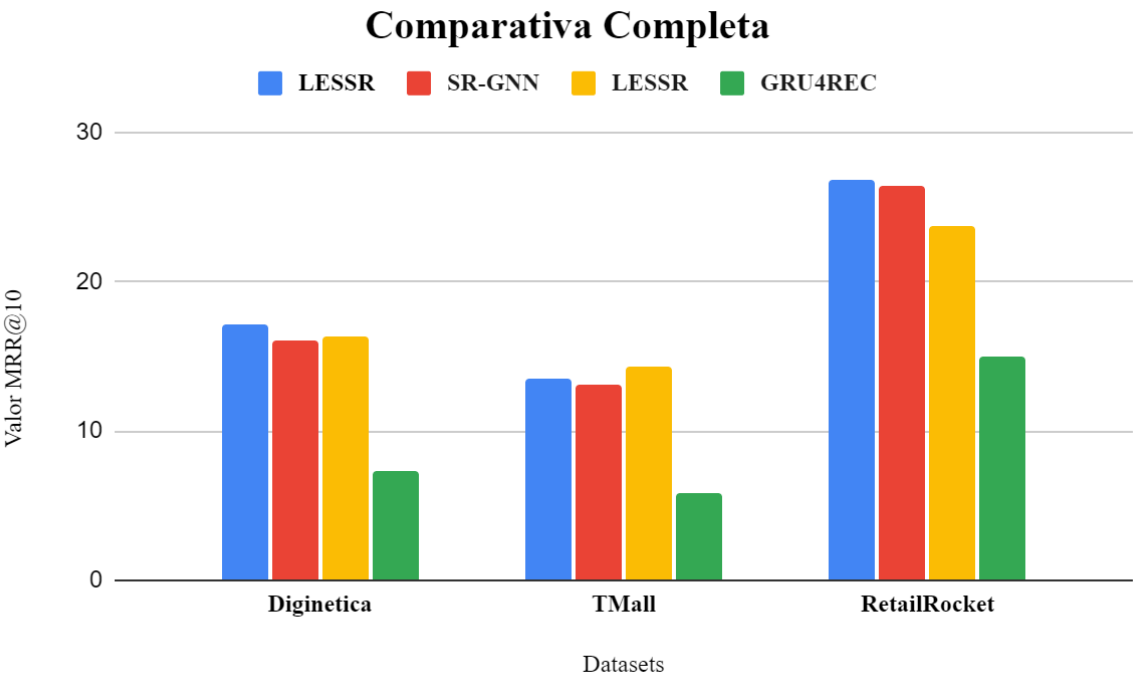
Como era de esperar, se repite lo mismo, en los tres conjuntos de datos LESSR ha conseguido mejores puntuaciones en las métricas de precisión@10 y MRR@10. Como se ha hecho con SR-GNN se mostrarán las mejores iteraciones para LESSR y GRU4REC de forma gráfica en la métrica MRR@10.

Figura 48. Comparativa LESSR - GRU4REC.



Ya para finalizar, se muestra una gráfica con las mejores iteraciones de los tres métodos en MRR@10 comparados con el resultado de GRU4REC.

Figura 49. Comparativa de todos los métodos en MRR@10.



5. Conclusiones

El estudio y comprensión del estado del arte nos ha hecho capaz de entender cuáles son los problemas de este tipo de sistemas; como la ineficiencia de agrupar capas, los problemas de *cold start* o la ineficiencia de capturar relaciones de alto orden, entre otros.

Gracias al reconocimiento de estos problemas hemos encontrado las justificaciones de las nuevas propuestas y a qué problemas se dirigen. Todas ellas resuelven problemas del pasado y plantean nuevos, como la baja optimización computacional o la falta de un referente o manera de hacer.

A diferencia del análisis teórico, la parte práctica además de reafirmar las suposiciones teóricas nos ha ayudado a entender el funcionamiento de estos sistemas y reconocer cuáles son las partes vitales a la hora de poder hacer un sistema de este estilo y sus características. También ha servido como confirmación de la realización correcta de las pruebas, al tener en algunos casos resultados con los conjuntos de datos utilizados.

Para recapitular; se ha hecho una presentación teórica de cada uno de los algoritmos, se ha revisado el código y hecho una implementación que se recoge en el Github del proyecto[19]. Se han realizado pruebas con tres conjuntos de datos diferentes para evaluar su funcionamiento y finalmente se han comparado con uno de los métodos asentados en el estado del arte.

En relación con los objetivos, se han cumplido todos a falta de poder hacer comparativas con SERec. El problema que ha ocurrido con este algoritmo es que era necesaria información relativa a la red social de los usuarios, y para los conjuntos de datos escogidos no se podía obtener esa información.

Es por ello que se aumentó el número de *datasets* a comparar y se añadió la comparativa con un método asentado en la literatura.

Además de todo el conocimiento teórico aprendido sobre los sistemas recomendadores, se han podido ver resultados prácticos de su aplicación y el impacto que tiene hacer una buena elección de algoritmo dependiendo de las necesidades de cada proyecto, cuándo inicialmente no se tenía constancia de ello.

Como se ha podido comprobar en el apartado de Análisis y resultados, todos los algoritmos revisados mejoran el estado del arte de hace unos años y se asientan como el estado del arte actual en los sistemas recomendadores basados en sesión.

5.1. Trabajo futuro

Se podría trabajar en la continuación del proyecto aumentando el repositorio creado con diferentes métodos propuestos en la literatura y los futuros, además de comparar los resultados

[19] *Se puede encontrar en: https://github.com/SamuelCalabriaC/Recommender-Systems* .

obtenidos con los que se tienen ya para poder sacar conclusiones, ya que únicamente con la evaluación de tres algoritmos es difícil catalogar un orden de jerarquía en las características de cada uno de ellos.

Otra forma de poder seguir trabajando en el tema sería proponer un algoritmo propio que recogiera las mejores partes de cada método; como por ejemplo, crear un método utilizando la estructura de grafo del DHCN juntamente con la aplicación de las diferentes capas de retención de información del LESSR para el grafo lineal complementario.

Ya para finalizar, crear un repositorio donde se tuviese todo el código a mano con entornos de ejecución mejoraría el entendimiento y uso de estos métodos, ya que, gran parte de los problemas surgidos durante la realización de este trabajo han sido relacionados con la puesta en marcha de los algoritmos y/o preprocesado de datos.

Bibliografía

Para la realización del trabajo se han consultado las siguientes fuentes:

- Introducción a los Sistemas Recomendadores

GENERAL Chen Gao, Yu Zheng, Nian Li, Yinfeng Li, Yingrong Qin, Jinghua Piao, Yuhan Quan, Jianxin Chang, Depeng Jin, Xiangnan He, Yong Li (2021). Graph Neural Networks for Recommender Systems: Challenges, Methods, and Directions. Retrieved from https://doi.org/10.48550/arXiv.2109.12843

GENERAL FIB LAB, Tsinghua University (2021). GNN-Recommender Systems. Retrieved from https://github.com/tsinghua-fib-lab/GNN-Recommender-Systems.

GENERAL Google Developers (2019). Content-based Filtering. Retrieved from https://developers.google.com/machine-learning/recommendation/content-based/basics.

GENERAL Cloudera (2019). Session-based Recommender Systems. Retrieved from https://session-based-recommenders.fastforwardlabs.com/

Previous State (NARM) Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Jun Ma (2017). Neural Attentive Session-based Recommendation. Retrieved from arXiv:1711.04725v1.

Previous State (GRU4REC) Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, Domonkos Tikk (2016). Session-based Recommendations with Recurrent Neural Networks. Retrieved from arXiv:1511.06939v4.

Métricas I Big Data and Multi-modal Computing Group (2019). Personalized Graph Neural Networks with Attention Mechanism for Session-Aware Recommendation. Retrieved from https://github.com/CRIPAC-DIG/A-PGNN/blob/bff3d189e81fdfbd7ea179c735f269add142d506/model_last.py#L415

Métricas II Benjamin Wang, Ranking Evaluation Metrics for Recommender Systems Retrieved from: https://towardsdatascience.com/ranking-evaluation-metrics-for-recommender-systems-263d0a66ef54

GRU4REC Results Chao Huang, Jiahui Chen, Lianghao Xia, Yong Xu, Peng Dai, Yanqing Chen, Liefeng Bo, Jiashu Zhao, Jimmy Xiangji Huang. Graph-Enhanced Multi-Task Learning of Multi-Level Transition Dynamics for Session-based Recommendation. Retrieved from: https://arxiv.org/pdf/2110.03996.pdf

- Publicaciones de los algoritmos

DHCN Xia, X., Yin, H., Yu, J., Wang, Q., Cui, L., & Zhang, X. (2021). Self-Supervised Hypergraph Convolutional Networks for Session-based Recommendation. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(5), 4503-4511. Retrieved from https://ojs.aaai.org/index.php/AAAI/article/view/16578

SRGNN Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, Tieniu Tan (2019). Session-based Recommendation with Graph Neural Networks. Retrieved from https://doi.org/10.48550/arXiv.1811.00855

LESSR Tianwen Chen and Raymong Chi-Wing Wong, KDD (2020). Handling Information Loss of Graph Neural Networks for Session-based Recommendation. Retrieved from http://home.cse.ust.hk/~raywong/paper/kdd20-informationLoss-GNN.pdf.

SEFrame & SERec Tianwen Chen and Raymond Chi-Wing Wong (2021) An Efficient and Effective Framework for Session-based Social Recommendation. Retrieved from http://home.cse.ust.hk/~raywong/paper/wsdm21-SEFrame.pdf.

- Conjunto de datos

Datasets Diginetica Diginetica, Retail Technology Company (2016). CIKM Cup 2016 Track 2: Personalized E-Commerce Search Challenge. Retrieved from https://competitions.codalab.org/competitions/11161#learn_the_details-data2

Datasets TMall TMall & Taobao (2018). User Behavior Data on Taobao/Tmall IJCAI16 Contest. Retrieved from https://tianchi.aliyun.com/dataset/dataDetail?dataId=53

Datasets Retail Rocket Retail Rocket (2021). Retailrocket recommender system dataset. Retrieved from https://www.kaggle.com/datasets/retailrocket/ecommerce-dataset.

- Repositorios utilizados

Repositorio Github Samuel Calabria (2022). Recommender Systems repository. Retrieved from https://github.com/SamuelCalabriaC/Recommender-Systems.

Repositorio DHCN Xiaxin1998 (2021). DHCN Repository. Retrieved from https://github.com/xiaxin1998/DHCN.

Repositorio SRGNN Big Data and Multi-modal Computing Group, CRIPAC (2019). SRGNN Repository. Retrieved from https://github.com/CRIPAC-DIG/SR-GNN.

Repositorio LESSR TwChen (2021). LESSR Repository. Retrieved from https://github.com/twchen/lessr.

Repositorio SEFrame/SEREC TwChen (2021). SeFrame repository. Retrieved from https://github.com/twchen/SEFrame.