Grandes modelos de lenguaje en sistemas de recomendación

Trabajo de Fin de Máster

Máster en Sistemas Inteligentes

VNiVERSiDAD D SALAMANCA

Julio de 2024

Autor

Alberto García Martín

Tutores

Álvaro Lozano Murciego

María N. Moreno García

Departamento de Informática y Automática
Universidad de Salamanca
España

Modelo de Declaración de Autoría para el Trabajo de Fin de Máster

Declaro que he redactado el Trabajo de Fin de Máster (TFM) titulado *"Grandes modelos de lenguaje en sistemas de recomendación"* del Máster Universitario en Sistemas Inteligentes de la Universidad de Salamanca en el segundo semestre del curso académico *2023-24* de forma autónoma, con la ayuda de las fuentes y la literatura citadas en la bibliografía, y que he identificado como tales todas las partes tomadas de las fuentes y de la literatura indicada, textualmente o conforme a su sentido.

Además, soy conocedor de que el citado TFM forma parte de los trabajos de investigación que lleva(n) a cabo mi(s) director(es) *María N. Moreno García y Álvaro Lozano Murciego* dentro del grupo de investigación *MIDA y ESALAB* de la Universidad de Salamanca y, en consecuencia, comparto con el(los) la propiedad intelectual de los resultados alcanzados.

En *Salamanca*, a *12 de julio de 2024*

Firmado por GARCIA MARTIN ALBERTO - ***1611** el día 12/07/2024

Fdo.: *Alberto García Martín*

Resumen

Recientemente, los grandes modelos de lenguaje (LLM) han demostrado tener unas capacidades extraordinarias en la comprensión y generación de lenguaje natural. Esto ha provocado una disrupción en el campo de la inteligencia artificial, donde se han aplicado estos modelos para múltiples tareas que pueden beneficiarse del extenso conocimiento relacionado con las características textuales de los datos que estos modelos han aprendido.

Uno de estos dominios es el de los sistemas de recomendación, donde se observa un crecimiento significativo en el número de trabajos que intentan aprovechar las capacidades de los LLM para mejorar la calidad de las recomendaciones. En este contexto, se realiza una revisión de la literatura para identificar los enfoques más relevantes tanto en los sistemas de recomendación, poniendo especial énfasis en la recomendación secuencial, como en el campo de los LLM, explorando los trabajos más destacados que combinen ambos campos.

En este trabajo, se propone reproducir y evaluar un método de recomendación secuencial basado en LLM llamado LlamaRec. Este método se basa en un enfoque de dos fases, primero un modelo de recuperación, basado en un sistema de recomendación secuencial tradicional, selecciona un conjunto de candidatos de todo el catálogo de ítems disponibles. Estos candidatos luego son reordenados, teniendo en cuenta el historial del usuario, por un LLM entrenado mediante *fine-tuning* para esta tarea.

La investigación de este enfoque tiene como objetivo inicial confirmar la efectividad del método, reproduciendo los resultados originales, y extender su evaluación a otros dominios de recomendación. Además, se propone comparar el rendimiento del método planteado con otros LLM de vanguardia para evaluar si estos modelos pueden mejorar la eficacia del método original. De esta forma, se pretende explorar las posibilidades y limitaciones de la aplicación de LLM en sistemas de recomendación.

La evaluación del sistema se realiza mediante métricas utilizadas habitualmente en sistemas de recomendación, como recall, MRR y NDCG. Los resultados obtenidos en este trabajo muestran que LlamaRec es efectivo en múltiples dominios y que su rendimiento puede mejorarse aún más con la incorporación de modelos de lenguaje más avanzados. Sin embargo, también se identificaron varias limitaciones, como la dependencia de recursos computacionales significativos y la baja eficacia en conjuntos de datos de gran tamaño.

El estudio concluye proponiendo líneas futuras de investigación que incluyen el estudio de métodos para mejorar la eficiencia del entrenamiento de LlamaRec, así como la utilización de infraestructuras más avanzadas para explorar conjuntos de datos más grandes y modelos de lenguaje más complejos. Además, se subraya la importancia de establecer mejores prácticas para la reproducibilidad en la investigación de sistemas de recomendación basados en LLM.

Palabras clave

Sistemas de recomendación, Grandes modelos de lenguaje, Recomendación secuencial, Reproducibilidad.

Abstract

Recently, large language models (LLMs) have demonstrated extraordinary capabilities in understanding and generating natural language. This has caused a disruption in the field of artificial intelligence, where these models have been applied to multiple tasks that benefit from their extensive knowledge of the textual characteristics of the data they have learned.

One such domain is recommender systems, which have seen a significant increase in studies attempting to leverage the capabilities of LLMs to improve the quality of recommendations. In this context, a literature review has been conducted to identify the most relevant approaches in recommender systems, with a special emphasis on sequential recommendation. As well as in the field of LLMs, exploring the most notable works that combine both fields.

In this paper, we propose to reproduce and evaluate an LLM-based sequential recommendation method called LlamaRec. This method employs a two-phase approach: first, a retrieval model, based on a traditional sequential recommender system, retrieves a set of candidates from the entire item catalog available. Then, an LLM fine-tuned to this task reorders these candidates, considering the user's history.

The initial goal of this research is to confirm the effectiveness of the method by reproducing the original results and extending its evaluation to other recommendation domains. Additionally, it aims to compare the performance of the proposed method with other state-of-the-art LLMs to assess if these models can improve the performance of the original method. This exploration seeks to uncover the possibilities and limitations of applying LLMs in recommender systems.

We evaluate the system's performance using standard metrics in recommender systems, such as recall, MRR, and NDCG. The results obtained in this work show that LlamaRec is effective in multiple domains and that its performance can be further improved with the incorporation of more advanced LLMs. However, we also identified several limitations, such as the dependency on significant computational resources and lower performance on large datasets.

The study concludes by proposing future lines of research, including methods to improve the efficiency of LlamaRec training, as well as the use of more advanced infrastructures to explore larger datasets and more complex language models. Furthermore, we emphasize the importance of establishing better practices for reproducibility in the research of LLM-based recommender systems.

Keywords

Recommender systems, Large language models, Sequential recommendation, Reproducibility.

Tabla de contenidos

Lista de tablas

Lista de figuras

Capítulo 1

Introducción

En la actualidad, los sistemas de recomendación son una parte presente en la vida cotidiana de la mayoría de las personas, ya que se utilizan en una amplia variedad de aplicaciones y servicios de internet. Estos sistemas se han convertido en una herramienta fundamental para ayudar a los usuarios a descubrir nuevos contenidos y productos, así como para mejorar la experiencia de usuario en plataformas digitales. Los grandes modelos de lenguaje (LLM) pueden tener un gran potencial para mejorar la calidad de las recomendaciones, ya que son capaces de capturar de manera efectiva las características semánticas de los datos y de esta manera generar recomendaciones más personalizadas y precisas.

Los LLM son modelos de aprendizaje profundo que han sido entrenados en grandes cantidades de texto para aprender a generar la siguiente palabra en una secuencia de texto. En los últimos años, estos modelos han demostrado tener una capacidad excepcional para comprender y generar lenguaje natural, lo que los ha convertido en una herramienta poderosa para una amplia variedad de tareas relacionadas con el procesamiento del lenguaje natural. Los LLM han sido aplicados con éxito en tareas como la traducción automática, el resumen de contenidos y la generación de texto, entre otras.

En este contexto, este trabajo pretende hacer una revisión actualizada de la literatura, identificando los enfoques más relevantes en el campo de los LLM y de los sistemas de recomendación. Se pone especial énfasis en la recomendación secuencial, donde el objetivo es recomendar una secuencia de ítems a los usuarios teniendo en cuenta su historial de interacciones previas. Este enfoque es especialmente relevante en aplicaciones como la recomendación de música, donde la secuencia de reproducción previa puede influir significativamente en las preferencias futuras del usuario.

Los sistemas de recomendación secuenciales y los LLM comparten la característica de que ambos son entrenados sobre secuencias de datos, con el objetivo de predecir un ítem a recomendar en el primer caso, y la siguiente palabra en el segundo. Por lo tanto, la combinación de ambos enfoques puede resultar en sistemas de recomendación más efectivos, que puedan capturar patrones que no serían posibles de identificar con enfoques tradicionales. En este sentido, se propone realizar un estudio de reproducibilidad de un método de recomendación secuencial basado en LLM llamado LlamaRec [135].

1.1. Motivación

La motivación principal de este trabajo radica en explorar las posibilidades y limitaciones de la aplicación de LLM en sistemas de recomendación, con el objetivo de determinar la viabilidad de este tipo de enfoques en la práctica. Este trabajo pretende dar un enfoque pragmático a la investigación en este campo, evaluando la eficacia de un método de recomendación basado en LLM en diferentes dominios de recomendación y comparándolo con otros LLM de vanguardia.

Este trabajo se desarrolla año y medio después del anuncio de ChatGPT, una aplicación con un LLM conversacional que ha demostrado ser capaz de mantener conversaciones coherentes y convincentes con los usuarios, resolviendo todo tipo de preguntas y problemas. La disponibilidad de este tipo de modelos de lenguaje tan avanzados ha acelerado la aplicación de LLM en una amplia variedad de tareas, incluyendo la de recomendación de ítems. Sin embargo, muchos de estos métodos no son posibles de reproducir fácilmente, ya sea por el uso de modelos cerrados que están en constante cambio, o por la ausencia del código relativo a la implementación de los métodos.

La selección de LlamaRec como método de estudio y reproducción en este trabajo se basa en su uso de modelos de lenguaje preentrenados, cuyos pesos son públicos, y la disponibilidad del código fuente de la implementación original. Esto permite una evaluación más transparente y reproducible del método, lo que es fundamental en cualquier trabajo de investigación científica.

Además, el enfoque de dos fases utilizado en LlamaRec, donde primero se recuperan los posibles ítems candidatos utilizando un modelo de recomendación tradicional y luego se reordenan usando un LLM, es una estrategia interesante, ya que puede combinar las bondades de ambos tipos de modelos. Por un lado, el modelo de recuperación puede ser más eficiente en la selección de candidatos, mientras que el LLM puede mejorar la generación de recomendaciones personalizadas, al incorporar información sobre los datos que el otro modelo no tiene en cuenta.

Para evaluar de forma objetiva este método, se ha seleccionado un conjunto de datos con características distintas a los utilizados en la implementación original, lo que permitirá evaluar la generalización del método a diferentes dominios de recomendación. Este conjunto pertenece al dominio de la música, caracterizado por tener secuencias de reproducción donde los usuarios consumen un alto número de ítems en un corto periodo de tiempo, en comparación con otros dominios como el de las películas o la compra de productos.

Evaluando este enfoque sobre un dominio diferente, y empleando distintos LLM para realizar la reordenación de los candidatos, se espera obtener una visión más completa de las capacidades y limitaciones de este tipo de métodos. Además, se espera que los resultados obtenidos en este trabajo puedan servir de base para futuras investigaciones orientadas a mejorar las deficiencias que puedan presentar estos métodos, y que permitan avanzar en el desarrollo de sistemas de recomendación basados en LLM.

1.2. Objetivos

El objetivo principal de este trabajo es estudiar la utilidad y efectividad de los LLM en los sistemas de recomendación, determinando si estos modelos pueden mejorar los resultados de los métodos tradicionales en la generación de recomendaciones personalizadas. Para ello, se propone revisar el estado del arte identificando los enfoques más relevantes en estos campos y realizar un estudio de reproducibilidad del enfoque de LlamaRec. Los objetivos específicos asociados a estas tareas son los siguientes:

1. **Realizar una revisión de la literatura**: Investigar y analizar un compendio de estudios y avances recientes tanto en el campo de los sistemas de recomendación como en el de los LLM. Este objetivo incluye identificar los enfoques más innovadores y efectivos que se han propuesto, así como las métricas de evaluación utilizadas para medir su desempeño. Además, se busca comprender las limitaciones y desafíos actuales de ambos campos.

2. **Reproducir los resultados originales del método LlamaRec**: Llevar a cabo una reproducción exhaustiva de los experimentos originales del método LlamaRec. Esto implica descargar los conjuntos de datos, replicar el entorno experimental y ejecutar los mismos procedimientos para verificar si se obtienen resultados similares a los reportados originalmente. Se prestará especial atención a documentar cualquier deficiencia o dificultad encontrada durante el proceso de implementación para asegurar una mayor reproducibilidad en el futuro.

3. **Evaluar el comportamiento de LlamaRec en un dominio diferente**: Probar el enfoque de LlamaRec en un nuevo dominio de recomendación que no haya sido previamente evaluado, como el de la música. Este objetivo busca determinar si el enfoque de LlamaRec es generalizable a distintos tipos de datos y contextos de recomendación y no únicamente a los dominios en los que fue inicialmente probado. Se analizará su desempeño teniendo en cuenta las diferencias en las características de los conjuntos de datos.

4. **Aplicar distintos LLM del estado del arte**: Implementar y comparar varios modelos de lenguaje grandes recientes para evaluar si estos pueden superar los resultados obtenidos por el método original de LlamaRec. Este objetivo implica seleccionar los LLM más prometedores, integrarlos en el sistema de recomendación y realizar pruebas comparativas para medir mejoras empleando las mismas métricas de evaluación que en los experimentos anteriores.

5. **Desarrollar una herramienta interactiva**: Crear una aplicación o plataforma interactiva que permita a los usuarios experimentar con el método LlamaRec en un dominio de recomendación específico, como el de la música. Esta herramienta deberá ser intuitiva y accesible, proporcionando una interfaz amigable para realizar pruebas y visualizar los resultados. Además, se buscará incorporar funcionalidades que permitan el ajuste de distintos parámetros.

Capítulo 2

Estado del arte

En este capítulo, se presenta una revisión exhaustiva del estado del arte en el campo de los sistemas de recomendación (RS), con un enfoque particular en la integración de grandes modelos de lenguaje (LLM) en estos sistemas.

2.1. Sistemas de recomendación

Los sistemas de recomendación, a veces llamados sistemas recomendadores, son herramientas y técnicas de software que ofrecen sugerencias sobre ítems que probablemente resulten de interés para un usuario en particular [98]. Estas sugerencias suelen estar relacionadas con diversos procesos de toma de decisiones, como qué productos comprar, qué música escuchar o qué publicaciones leer en una red social.

Normalmente, un sistema de recomendación se centra en un dominio específico, adaptando su diseño y los algoritmos subyacentes a las características de dicho dominio. Incluso para problemas que a priori parecen ser comunes, como la recomendación de productos en una tienda *online*, las necesidades y expectativas de los usuarios pueden variar significativamente de una categoría de productos a otra, siendo necesario adaptar los sistemas de recomendación a cada caso concreto.

Los sistemas de recomendación surgen como respuesta a la sobrecarga en el usuario causada por el abrumador número de ítems que podían contener los sistemas digitales. Estos sistemas de recomendación se inspiran en la vida cotidiana, donde se observa que las personas suelen depender de recomendaciones de otra gente [96], ya sean amigos, conocidos o expertos en el tema, para tomar decisiones sobre qué productos comprar, qué películas ver o qué libros leer.

Para replicar este comportamiento, los primeros algoritmos aplicados a sistemas de recomendación empleaban las recomendaciones de otros usuarios para sugerir ítems a un usuario en particular. Este enfoque, llamado filtrado colaborativo [35], se basa en la idea de que si dos usuarios han mostrado preferencias similares en el pasado, es probable que también tengan preferencias similares en el futuro.

2.1.1. Datos empleados por los RS

Los datos utilizados para producir las recomendaciones pueden variar según los algoritmos empleados, y diferir tanto en la naturaleza de estos como en la forma utilizarlos. Sin embargo, por norma general los datos usados en un sistema de recomendación pueden relacionarse con al menos tres tipos de objetos: ítems, usuarios, e interacciones entre ítems y usuarios [98].

- **Ítems**: Los ítems son los objetos que se recomiendan a los usuarios. Los datos sobre los ítems que se pueden utilizar de manera efectiva en los sistemas de recomendación dependen en gran parte de los algoritmos que se emplean para transformarlos y extraer información útil de ellos. También, los metadatos asociados a los ítems suelen ser de gran importancia. Por ejemplo, en un sistema de recomendación de música, los ítems serían las canciones, y los metadatos podrían incluir información sobre el artista o el género. Además, se pueden usar algoritmos especiales para analizar el audio y extraer información adicional, como la tonalidad o el tempo.
- **Usuarios**: Los usuarios son las entidades que interactúan con los ítems. Los sistemas de recomendación pueden aprovechar todo tipo de datos sobre los usuarios para construir un modelo de datos que refleje las preferencias y características de los usuarios. El objetivo de estos modelos de datos es poder ofrecer una recomendación personalizada a cada usuario. Los datos específicos a incluir en estos modelos dependerá de la técnica de recomendación empleada y de la posibilidad de extraer información útil de los mismos. Por ejemplo, en un sistema consciente del contexto, se podrían incluir datos específicos del contexto actual del usuario que está interactuando con el sistema, como la ubicación o la hora del día.
- **Interacciones**: Las interacciones son las instancias en las que un usuario interactúa con un ítem. La información más importante de estas interacciones suele ser la valoración o *feedback* aportado por el usuario. Este *feedback* puede ser explícito o implícito, según contenga o no información directa sobre la preferencia del usuario. Por ejemplo, en un sistema de recomendación de música, la valoración de un usuario indicando que le gusta una canción sería un ejemplo de *feedback* explícito. En cambio, el hecho de haber escuchado una canción muchas veces sería un ejemplo de *feedback* implícito. El *feedback* explícito es más fácil de interpretar y suele ser más útil, pero es más difícil de obtener, ya que requiere una acción consciente por parte del usuario.

2.1.2. Tipos de RS

Los sistemas de recomendación pueden clasificarse en función de diversos criterios, como la naturaleza de los datos disponibles, el tipo de recomendaciones que se ofrecen, o el enfoque algorítmico empleado. Según ha ido evolucionando el campo, han ido surgiendo nuevas técnicas que combinan distintos enfoques; sin embargo, la taxonomía propuesta por Burke [14] sigue siendo una de las más utilizadas para clasificar los sistemas de recomendación. En la figura 2.1 se muestra un diagrama que resume los distintos tipos de sistemas de recomendación. A continuación se detallan los tres tipos más comunes:

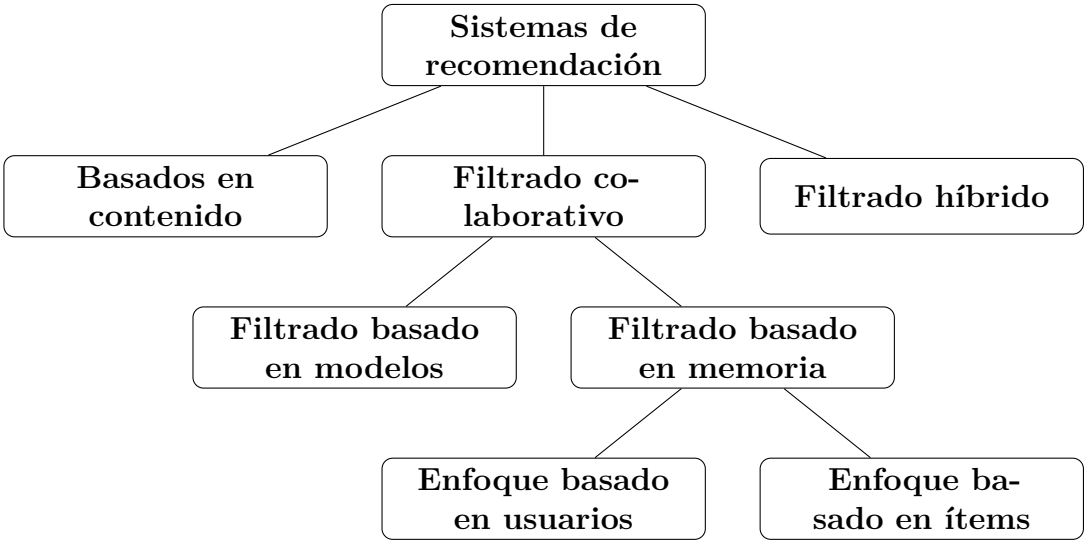

Figura 2.1: Tipos de sistemas de recomendación

- **Basados en contenido**: Los sistemas de recomendación basados en contenido utilizan información sobre los ítems para recomendar ítems similares a los que un usuario ha mostrado interés en el pasado. Los enfoques clásicos de este tipo tiene como objetivo vincular características en los ítems con los datos obtenidos de los usuarios. Una desventaja de este enfoque es que requiere información abundante del ítem para poder realizar recomendaciones precisas.
- **Filtrado colaborativo**: El enfoque de estos sistemas es hacer recomendaciones a partir de las similitudes entre los gustos de los usuarios. La similitud en las preferencias de los usuarios se puede calcular a partir del historial de interacciones de los usuarios con los ítems. La eficacia de este enfoque depende de lo preciso que sea el algoritmo a la hora de encontrar usuarios con gustos supuestamente similares. A diferencia de los sistemas basados en contenido, los sistemas de filtrado colaborativo no necesitan información sobre los ítems. Sin embargo, sufren más del problema de arranque en frío, ya que necesitan un número suficiente de interacciones para poder realizar recomendaciones útiles. El filtrado colaborativo se puede dividir en dos subcategorías: filtrado basado en memoria y filtrado basado en modelos.

  El filtrado basado en memoria usa las interacciones de los usuarios para calcular la similitud entre usuarios o ítems. Si el enfoque es basado en usuarios, la valoración que un usuario le asignaría a un nuevo ítem es calculada a partir de las valoraciones sobre este ítem de los usuarios más similares al usuario objetivo. Mientras que, si el enfoque es basado en ítems, la valoración predicha para un nuevo ítem es calculada a partir de las valoraciones dadas por el usuario objetivo a los ítems más similares al nuevo ítem. A diferencia de los métodos basados en contenido, la similitud entre ítems se calcula a partir de las valoraciones que han recibido de los usuarios.

  Por otro lado, el filtrado basado en modelos utiliza técnicas de minería de datos y de aprendizaje automático para construir un modelo predictivo de las preferencias de los usuarios para ítems que no hayan valorado. Este modelo se entrena con las interacciones de los usuarios con los ítems, y puede ser utilizado

incluso para hacer recomendaciones a usuarios que no estén presentes en los datos iniciales. Este enfoque es más eficiente para realizar recomendaciones a grandes grupos de usuarios. Sin embargo, su eficacia depende en gran medida de la calidad del modelo predictivo.

- **Filtrado híbrido**: Un sistema híbrido agrega dos o más técnicas de recomendación para solventar las limitaciones de cada una de ellas. Las técnicas se pueden combinar de diversas formas, ya sea usando dos modelos por separado y combinando sus resultados, o integrando las técnicas en un único modelo.

2.1.3. Objetivos de los RS

En términos técnicos, el objetivo de un sistema de recomendación, depende de cómo se formule el problema a resolver, habiendo dos modelos principales para plantear el problema: como un problema de predicción de valoraciones, o como un problema de ordenación, es decir, de creación de un ranking de ítems o usuarios.

- **Problema de predicción**: En este caso, el objetivo es predecir la valoración que un usuario daría a un ítem en particular. Se supone que se tiene una matriz incompleta de valoraciones de los usuarios sobre los ítems, donde los valores existentes se usan para entrenar un modelo predictivo que pueda predecir los valores faltantes. Este problema también se conoce como el problema de completar la matriz (*matrix completion*), haciendo referencia a la matriz incompleta de valoraciones cuyos valores deben ser predichos por el algoritmo de recomendación.
- **Problema de ranking**: En la práctica, no es necesario predecir la valoración exacta que un usuario daría a un ítem. En muchos casos, es suficiente con predecir el top-k de ítems a recomendar a un usuario, o el top-k de usuarios a los que recomendar un ítem. Es más común realizar una predicción de los top-k ítems que de los top-k usuarios, pero ambos planteamientos son análogos. Este problema se conoce como el problema de (*top-k recommendation*), y es el enfoque más común en la literatura de sistemas de recomendación.

Desde un punto de vista más pragmático, el objetivo principal de los sistemas de recomendación es generar beneficios [2]. Si se recomiendan ítems que el usuario valorará positivamente, el usuario estará más satisfecho y es más probable que vuelva a utilizar el sistema. Esto genera un beneficio, ya sea directo mediante la compra de un producto, o indirecto, mediante la visualización de anuncios. Con el fin de alcanzar esta meta principal, los sistemas de recomendación pueden tener los siguientes objetivos secundarios:

- **Relevancia**: Es obvio que los ítems que se recomiendan a un usuario deben ser relevantes, ya que es más probable que consuman productos que les interesen. Aunque la relevancia sea de gran importancia, es importante tener en cuenta también los siguientes aspectos para que el sistema de recomendación sea efectivo.
- **Novedad**: Los sistemas de recomendación resultan más útiles cuando los ítems recomendados no son conocidos con anterioridad por el usuario. La repetición en la recomendación de ítems populares no conocidos por el usuario es un

aspecto a evitar, ya que puede llevar a una reducción de la diversidad de las recomendaciones.

- **Serendipia**: La serendipia es la capacidad de un sistema de recomendación de sorprender al usuario con recomendaciones inesperadas, pero que resultan ser de su interés. Este concepto difiere del de la novedad en que no se trata de recomendar ítems simplemente desconocidos para el usuario, sino que se trata de recomendar ítems que el usuario no esperaba que le interesaran.
- **Diversidad**: Si un sistema recomienda una lista de top-k ítems que son muy similares entre sí, el usuario puede no sentirse satisfecho, además de existir un riesgo de que no le guste ninguna de las recomendaciones. Por ello, es importante que las recomendaciones sean diversas, y que cubran una amplia gama de gustos y preferencias. De esta forma se garantiza que al menos una de las recomendaciones sea de interés para el usuario y que no se aburra de las recomendaciones.

Los distintos sistemas de recomendación pueden tener diversos objetivos específicos según el dominio en el que se apliquen. En la tabla 2.1 se muestran ejemplos de algunas aplicaciones que emplean sistemas de recomendación y el ítem objetivo que se recomienda.

Aplicación	**Ítems a recomendar**
Amazon	Productos
Google Search	Anuncios
Google News	Noticias
Netflix	Películas
Spotify	Canciones
TripAdvisor	Hoteles
Twitter	Publicaciones
YouTube	Vídeos

Tabla 2.1: Ejemplos de aplicaciones de sistemas de recomendación

2.1.4. Desafíos en los RS

A pesar de los avances en el campo de los sistemas de recomendación, todavía existen desafíos importantes que limitan la eficacia de estos sistemas. Algunos de los desafíos más importantes con sus posibles soluciones son los siguientes [99].

- **Arranque en frío**: El problema del arranque en frío se refiere a la dificultad de hacer recomendaciones a usuarios nuevos, que se acaban de incorporar al sistema y sobre los que no se dispone de información suficiente. Para solventar este problema, se puede pedir al usuario que proporcione información sobre sus preferencias, o que valore algunos ítems al inicio. También se pueden utilizar técnicas de recomendación basadas en contenido, empleando tanto la información del ítem como la del usuario.
- ***Shilling attacks***: Un ataque de *shilling* es un intento de manipular un sistema de recomendación para promocionar o degradar ciertos ítems [38]. Estos ataques

pueden ser realizados por usuarios malintencionados, que pueden crear cuentas falsas para inflar las valoraciones de ciertos ítems o para degradar los ítems de la competencia. Para mitigar este problema, se pueden emplear técnicas de detección de anomalías para identificar y eliminar las valoraciones falsas antes de que afecten a las recomendaciones.

- **Problema de sinonimia**: El problema de la sinonimia se refiere a la dificultad de identificar que dos términos guardados en el sistema con nombres diferentes se refieren al mismo concepto. Por ejemplo, los términos "coche" y "carro" pueden referirse al mismo concepto, y si esto no se identifica, la precisión de las recomendaciones puede verse afectada. Para solventar este problema, se pueden emplear métodos como la descomposición en valores singulares o técnicas de procesamiento de lenguaje natural.
- **Problema de latencia**: El problema de la latencia es específico de los sistemas de filtrado colaborativo, y ocurre cuando se añaden ítems frecuentemente al sistema. En estos casos, el sistema puede seguir recomendando ítems antiguos, ya que los nuevos ítems no han tenido tiempo de ser valorados por los usuarios. Para mitigar este problema, se pueden emplear técnicas de recomendación basadas en contenido, que no dependen de las interacciones de los usuarios con los ítems.
- **Problema de dispersión**: El problema de la dispersión es común en conjuntos de datos grandes, donde los usuarios han valorado un número muy pequeño de ítems respecto al total. En estos casos, los sistemas de recomendación pueden tener dificultades para encontrar usuarios con gustos similares, ya que la matriz de interacciones es muy dispersa. Para solventar este problema, se pueden emplear técnicas de reducción de la dimensionalidad como la factorización de matrices o sistemas basados en modelos.
- **Problema de oveja negra**: El problema de la oveja negra se refiere a la dificultad de hacer recomendaciones a usuarios que tienen gustos muy diferentes a la mayoría y no se parecen a la de ningún otro grupo de usuarios. Este problema es especialmente importante en filtrado colaborativo, puesto que las recomendaciones dadas a una persona se basan en las valoraciones de los usuarios similares a ella. En este caso, el sistema podría ofrecer recomendaciones que no fueran relevantes para el usuario. Para mitigar este problema, se pueden emplear técnicas de recomendación basadas en contenido.
- **Problema de escalabilidad**: El problema de la escalabilidad se refiere a la dificultad de hacer recomendaciones en conjuntos de datos muy grandes, donde el tiempo de cálculo de las recomendaciones puede ser muy elevado, sobre todo en el caso de los métodos basados en memoria. Para solventar este problema, se pueden emplear técnicas de reducción de la dimensionalidad o sistemas basados en modelos.

2.1.5. Algoritmos para los RS

A continuación se describen algunos de los algoritmos de recomendación top-k más relevantes en el campo [5]. Estos algoritmos se han seleccionado por su relevancia en la literatura y por su eficacia en conjuntos de datos comúnmente utilizados como

MovieLens-1M [40], que contiene valoraciones en una escala de 1 a 5 de películas por parte de usuarios. Se han incluido algoritmos de distintas familias, incluyendo también algunos de los sistemas que sirven de punto de referencia para evaluar los distintos algoritmos.

Métodos no personalizados de referencia

Estos métodos no tienen en cuenta las preferencias de los usuarios, y se utilizan como punto de referencia para comparar la eficacia de los algoritmos de recomendación personalizados.

- **Aleatorio**: Se generan recomendaciones aleatorias para los usuarios.
- **Más popular**: A cada usuario se le recomiendan los ítems más populares, donde la popularidad se mide por el número de interacciones sobre el ítem en el conjunto de datos.

Modelos basados en vecinos y en grafos

Aquí se incluyen los métodos basados en vecinos más cercanos, es decir, aquellos que hacen recomendaciones basadas en la similitud entre los usuarios o los ítems.

- **UserKNN**: Un enfoque de vecinos más cercanos basado en usuarios propuesto por Resnick et al. [97] en 1994 para el sistema GroupLens, un sistema de filtrado colaborativo de noticias. A pesar de su simplicidad, estas técnicas basadas en vecinos más cercanos siguen demostrando ser eficaces en conjuntos de datos pequeños, y siguen siendo utilizadas como un punto de referencia en la literatura.
- **ItemKNN**: Este enfoque de vecinos más cercanos basado en ítems fue propuesto por Sarwar et al. [100] en 2001, y posteriormente, en 2003, fue empleado por Amazon en su sistema de recomendación de productos [73], popularizando su uso en la industria.
- **RP$^3\beta$**: Este algoritmo propuesto por Paudel et al. [83] en 2016 es un método basado en grafos que, conceptualmente, es similar al de ItemKNN, pero que consigue una mayor precisión y diversidad en las recomendaciones.

Modelos lineales

Estos métodos asumen una linealidad en el sistema para determinar los ítems a recomendar.

- **SLIM**: Este es un método basado en regresión lineal y propuesto en 2011 para la recomendación en sistemas top-k por Ning y Karypis [78]. SLIM es capaz de proporcionar recomendaciones precisas con gran rapidez.
- **EASE^R**: Otro modelo lineal propuesto en 2019 por Steck [107], que se basa en autocodificadores superficiales. Es un modelo que, a pesar de su simplicidad, obtiene resultados efectivos.

Modelos de factorización de matrices

Los métodos de factorización de matrices descomponen la matriz de interacciones en el producto de dos matrices de menor dimensionalidad.

- **MF2020**: Los primeros métodos de factorización de matrices fueron inicialmente explorados mediante la descomposición de valores singulares por Billsus y Pazzani [10] en 1998. Posteriormente, especialmente después del anuncio de la competición de Netflix en 2006, se popularizaron distintos métodos que empleaban aprendizaje automático para aprender los factores latentes de los usuarios e ítems. Estos métodos han seguido evolucionando hasta la actualidad, el modelo propuesto por Rendle et al. [95] en 2020, es un ejemplo de ello.
- **iALS**: Este método emplea un enfoque de mínimos cuadrados alternos y es particularmente efectivo a la hora de aprender los factores latentes a partir de *feedback* implícito. Fue propuesto por Hu et al. [53] en 2008, y es una de las técnicas más referenciadas que no emplea redes neuronales.
- **BPRMF**: Otro método que se diseñó para el *feedback* implícito, y que se basa en la optimización de un nuevo criterio derivado del análisis bayesiano. Fue propuesto por Rendle et al. [94] en 2009. BPRMF en concreto es la versión que emplea factorización de matrices, y también es usado como punto de referencia en modelos no basados en redes neuronales.

Modelos basados en redes neuronales

Estos modelos incluyen una red neuronal artificial en su arquitectura, esta estructura tiene como objetivo replicar el comportamiento del cerebro humano.

- **NeuMF**: Este método fue propuesto en 2017 por He et al. [43] y es uno de los modelos de recomendación basados en redes neuronales más influyentes. NeuMF generaliza la factorización de matrices y reemplaza el producto escalar por una arquitectura de redes neuronales. Es uno de los métodos más usados como referencia en sistemas basados en redes neuronales.
- **MultiVAE**: MultiVAE fue diseñado para datos de *feedback* implícito y se basa en autocodificadores variacionales. Fue propuesto por Liang et al. [69] en 2018, y es uno de los métodos más efectivos.
- **LightGCN**: LightGCN, propuesto por He et al. [42] en 2020, es un tipo de red neuronal convolucional basada en grafos (*GCN*) que emplea la agregación de vecinos para el filtrado colaborativo. LightGCN aprende los *embeddings* de usuarios e ítems propagándolos por el grafo de interacciones, y usa la suma ponderada de los *embeddings* aprendidos en todas las capas como el *embedding* final.
- **SGL**: Otro método basado en *GCN* propuesto por Wu et al. [126] en 2021, que suplementa la tarea supervisada clásica de recomendación con una tarea auxiliar auto supervisada, que refuerza el aprendizaje de representaciones de nodos a través de la autodiscriminación.
- **NCL**: NCL es un método basado en *GNN* propuesto por Lin et al. [72] en 2022, es definido como un paradigma de aprendizaje contrastivo enriquecido

por técnicas de vecinos para el filtrado colaborativo basado en grafos. NCL captura explícitamente tanto los vecinos estructurales como los semánticos como objetos del aprendizaje contrastivo.

- **DiffRec**: Este método, propuesto por Wang et al. [118] en 2023, es un modelo generativo de recomendación que infiere las probabilidades de interacción de los usuarios, reduciendo el ruido y evitando interacciones corruptas como en la síntesis de imágenes.

2.2. Sistemas de recomendación secuenciales

Los sistemas de recomendación secuenciales (*SRS*) son una variante de los sistemas de recomendación que ofrecen recomendaciones personalizadas analizando las interacciones históricas de los usuarios de manera secuencial. A diferencia de los sistemas de recomendación tradicionales, que consideran los ítems de manera aislada, la recomendación secuencial tiene en cuenta el orden temporal de las acciones de los usuarios. Este método es particularmente relevante en dominios donde la secuencia de eventos es importante, como en servicios de streaming, plataformas de comercio electrónico y redes sociales.

Algunos autores también emplean el término de recomendación secuencial para referirse a sistemas de recomendación basados en sesiones [56]. La recomendación basada en sesiones es un enfoque secuencial tanto en el sentido de que el objetivo es predecir la siguiente interacción en una sesión o secuencia de interacciones, como en el sentido de que las interacciones están ordenadas secuencialmente en el tiempo. Sin embargo, no todos los sistemas de recomendación secuenciales se basan en sesiones. Un sistema basado en sesiones no tiene en cuenta la información de los usuarios, los sistemas de recomendación conscientes de la sesión, por otro lado, tienen en cuenta tanto la información de los usuarios como la de la sesión. El término de sistemas conscientes de la secuencia sirve para recoger a todos los tipos de sistemas de recomendación mencionados anteriormente.

2.2.1. Características de los SRS

Los SRS son diferentes de los sistemas de recomendación tradicionales en varios aspectos. La figura 2.2 muestra un esquema de alto nivel de un sistema de recomendación secuencial, mostrando sus entradas y salidas. A continuación se detallan las características más importantes de los SRS [87].

- **Entrada**: La entrada principal de un SRS es una lista ordenada en el tiempo de las acciones pasadas de un usuario. Los usuarios pueden ser anónimos o estar identificados por el sistema. Cada acción puede estar asociada con un ítem, y pueden proporcionar *feedback* implícito o explícito sobre el ítem, aunque en la práctica, el *feedback* implícito es más común para este tipo de sistemas. Las acciones, los usuarios y los ítems pueden tener información adicional asociada a ellos.

  Estos registros de interacciones que se usan como entrada de un SRS frecuentemente tienen una serie de características que generan desafíos para estos sistemas [117]. Estas secuencias suelen tener una gran longitud o un orden que

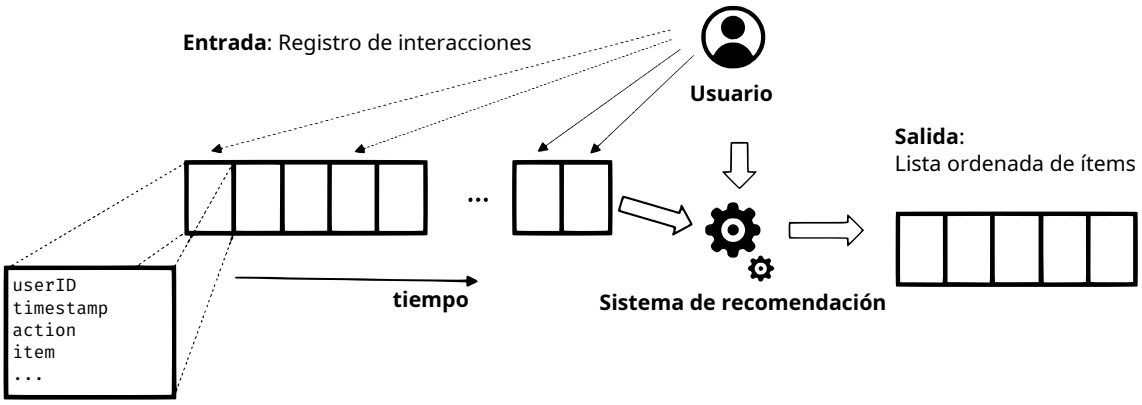

Figura 2.2: Esquema de un sistema de recomendación secuencial, basado en el esquema de Quadrana et al. [87]

en algunos momentos puede ser arbitrario. También puede que la secuencia contenga ruido, al contener interacciones irrelevantes, o que tenga una estructura jerárquica, con interacciones de distinto nivel de detalle.

- **Salida**: La salida de un SRS es una lista ordenada de ítems recomendados para el siguiente paso en la secuencia del usuario. La lista de recomendaciones puede ser generada en un solo paso, o en varios pasos, dependiendo de la arquitectura del sistema. Igual que en la entrada, el orden de los ítems recomendados es importante.
- **Tareas**: Los SRS pueden ser empleados para distintas tareas, como la adaptación de la recomendación al contexto, la detección de tendencias o de patrones, la consideración de límites en el orden de las recomendaciones, o la recomendación repetida de ítems.

2.2.2. Clasificación de los SRS

Los SRS se pueden clasificar en dos grandes grupos según usen métodos tradicionales de recomendación o métodos específicos para recomendación secuencial [130]. Dentro de cada uno de estos grupos, se pueden distinguir distintos tipos de SRS según el enfoque que empleen.

Métodos generales

Estos métodos adaptan técnicas generales de recomendación para hacer recomendaciones secuenciales. Generalmente, presentan precariedades a la hora de tener en cuenta el orden de interacciones de los usuarios, ya que no están diseñados para considerar la dependencia temporal entre las distintas acciones en el historial de un usuario.

- **Minería de patrones frecuentes**: Inicialmente, se utilizaban reglas de asociación para extraer patrones frecuentes de las secuencias de interacciones de los usuarios con el suficiente soporte y confianza. En la recomendación secuencial, los patrones se utilizan para predecir el siguiente ítem en la secuencia del usuario.

- **Vecinos más cercanos**: Los métodos de vecinos más cercanos pueden estar basados en ítems, en usuarios o en sesiones. La idea principal de estos enfoques es encontrar secuencias pasadas en los datos de entrenamiento que sean similares a la secuencia actual, y recomendar los ítems que aparecieron en las secuencias pasadas.
- **Modelos de factorización de matrices**: La factorización de matrices intenta descomponer la matriz de interacciones en dos matrices de menor dimensionalidad. Posteriormente, se predicen las preferencias del usuario realizando el producto escalar de las matrices de factores latentes de los usuarios e ítems. Un problema de estos métodos es que la mayoría solo considera las interacciones de bajo orden entre los factores latentes, pero ignoran las posibles interacciones de mayor orden.

Métodos de recomendación secuencial

Estos métodos están diseñados específicamente para hacer recomendaciones secuenciales, y, por lo general, suelen ser más efectivos a la hora de entender las variaciones a lo largo del tiempo en las preferencias de los usuarios.

- **Cadenas de Markov**: En la recomendación secuencial, los modelos de cadenas de Markov se utilizan para predecir el siguiente ítem en la secuencia del usuario. Estos modelos asumen que los comportamientos futuros de los usuarios solo dependen de las últimas acciones.

  También se pueden emplear procesos de decisión de Markov para implementar técnicas de aprendizaje reforzado en la recomendación secuencial. En este caso, el objetivo es maximizar la recompensa acumulada a lo largo de la secuencia de interacciones del usuario. Si el usuario muestra interés en una recomendación, se asigna una recompensa positiva.

- **Métodos basados en redes neuronales**: Existen varios tipos de redes neuronales que se pueden emplear en la recomendación secuencial.

  Los perceptrones multicapa (MLP) son redes neuronales prealimentadas con múltiples capas ocultas, que pueden aprender representaciones no lineales de las secuencias de interacciones de los usuarios.

  Las redes neuronales recurrentes (RNN) son muy frecuentes al ser capaces de capturar de manera adecuada la dependencia temporal entre las acciones de los usuarios. Sin embargo, es difícil entrenarlas con secuencias largas, al aumentar considerablemente el coste computacional.

  Las redes neuronales convolucionales (CNN) son adecuadas para capturar patrones locales en las secuencias de interacciones de los usuarios, siendo capaces de tener en consideración la información temporal de las secuencias.

  Las redes neuronales de grafos (GNN) son capaces de capturar las relaciones más complejas entre los ítems en una secuencia, y pueden modelar las secuencias como una composición de gustos a largo plazo y de intereses a corto plazo.

- **Mecanismos de atención**: Los mecanismos de atención son capaces de capturar la importancia de cada ítem en una secuencia de interacciones de

un usuario, permitiendo a los modelos centrarse en los ítems más relevantes para hacer recomendaciones. Estos mecanismos son especialmente útiles en secuencias largas, siendo capaces de rendir mucho mejor que otros métodos basados en redes neuronales recurrentes.

2.2.3. Algoritmos para los SRS

En los últimos años, los métodos de aprendizaje profundo han sobrepasado a los métodos tradicionales y a los basados en cadenas de Markov en los problemas de recomendación secuencial [30]. A continuación se describen algunos de los algoritmos más relevantes para la tarea de recomendación secuencial dada una secuencia de interacciones de un usuario con ítems.

Modelos tradicionales

En este apartado se enumeran algunos de los modelos más populares de recomendación secuencial que no están basados en aprendizaje profundo. En vez de redes neuronales, estos modelos emplean técnicas de factorización de matrices y cadenas de Markov.

- **FPMC**: Este modelo propuesto por Rendle et al. [93] en 2010, combina la factorización de matrices con cadenas de Markov de primer orden y sirve frecuentemente como punto de referencia en la literatura.
- **FOSSIL**: Fue propuesto por He y McAuley [41] en 2016, y es un modelo que combina métodos basados en similitud con cadenas de Markov de orden superior. Ofrece un mejor rendimiento en conjuntos de datos dispersos.

Modelos basados en RNN

Los modelos basados en redes neuronales recurrentes son muy populares en la recomendación secuencial, ya que son capaces de capturar las relaciones entre los ítems en una secuencia o entre distintas secuencias.

- **GRU4Rec**: Este es el primer modelo que emplea redes neuronales recurrentes para la recomendación secuencial. Fue propuesto por Hidasi et al. [47] en 2016, y es uno de los modelos más influyentes en la literatura. Toma como entrada una secuencia de interacciones y no tiene en cuenta la identidad del usuario.
- **HRNN**: Este modelo propuesto por Quadrana et al. [88] en 2017, es una extensión de GRU4Rec que modela tanto usuarios como sesiones. HRNN es capaz de capturar la dependencia temporal entre las interacciones de los usuarios y de las sesiones. La capa en el nivel de la sesión considera las actividades de un usuario en una sesión, mientras que la capa en el nivel del usuario modela la evolución de las preferencias de un usuario a lo largo de las sesiones.

Modelos basados en CNN

En la recomendación secuencial, los modelos basados en redes neuronales convolucionales son capaces de capturar patrones locales en las secuencias de interacciones de los usuarios, teniendo en cuenta la información temporal de las secuencias.

- **Caser**: Caser, propuesto por Tang y Wang [109] en 2018, es un modelo que visualiza la matriz de *embeddings* de los ítems del historial como una "imagen", utilizando capas convolucionales para encontrar patrones secuenciales. Utilizando la convolución, es posible percibir saltos en las acciones del usuario. También captura las preferencias a largo plazo del usuario a través del *embedding* del usuario.
- **NextItNet**: Este es un modelo generativo con estructura de bloques residuales, propuesto por Yuan et al. [133] en 2019, y es capaz de capturar relaciones a largo y corto plazo entre los ítems.

Modelos basados en mecanismos de atención

Tanto los mecanismos de atención basados en *RNN* como los basados en *transformers* han sido ampliamente utilizados en la recomendación secuencial. Habiendo ganado una mayor popularidad esta última arquitectura en los últimos años.

- **NARM**: NARM es un modelo codificador-decodificador propuesto por Li et al. [64] en 2017. En el codificador, se emplea una *RNN* con un mecanismo de atención para capturar la importancia de cada ítem en la secuencia de interacciones de un usuario. Mediante este mecanismo de atención, NARM es capaz de eliminar el ruido de acciones no intencionadas, como pueden ser clics accidentales.
- **SASRec**: Este modelo fue propuesto por Kang y McAuley [58] en 2018. Utiliza un mecanismo de autoatención para inferir las relaciones entre ítems a partir de las interacciones históricas de los usuarios. Con este mecanismo, es capaz de estimar los pesos de cada ítem en el historial de interacciones del usuario, aprendiendo las intenciones a corto plazo y las preferencias a largo plazo del usuario.
- **BERT4Rec**: Bert4Rec, propuesto por Sun et al. [108] en 2019, es una versión extendida de SASRec que introduce una arquitectura de *transformer* [115] para realizar la recomendación secuencial. BERT4Rec entrena un modelo bidireccional para modelar los datos secuenciales utilizando la prueba de Cloze [110].
- **S3Rec**: Este modelo, propuesto por Zhou et al. [139] en 2020, incorpora un método de aprendizaje autosupervisado. Utiliza las correlaciones intrínsecas de los datos para derivar señales de autosupervisión y mejorar la recomendación secuencial.
- **CORE**: CORE es un modelo propuesto por Hou et al. [50] en 2022, que unifica la representación espacial para los procesos de codificación y decodificación en la recomendación secuencial y previene el sobreajuste de los *embeddings* en el espacio de representación.
- **FEARec**: FEARec, propuesto por Du et al. [28] en 2023, es un modelo que mejora el mecanismo atención en la recomendación secuencial mediante una estructura de rampa para aprender tanto información de baja frecuencia como de alta frecuencia.

Otros modelos

Otros modelos basados en redes neuronales incluyen los basados en perceptrones multicapa o en redes neuronales de grafos, estos últimos pueden agregar información de una estructura de grafo que representa las secuencias de interacciones.

- **HRM**: HRM fue propuesto por Wang et al. [116] en 2015, y es un modelo basado en perceptrones multicapa. Fue uno de los primeros modelos de recomendación secuencial que empleaba redes neuronales profundas.
- **SR-GNN**: Este modelo propuesto por Wu et al. [128] en 2019, es un modelo basado en redes neuronales de grafos que es capaz de capturar las relaciones más complejas entre los ítems en una secuencia de interacciones de un usuario. En este modelo, cada sesión es modelada como un grafo dirigido, y se utiliza una red neuronal de grafos para obtener representaciones de las sesiones.

2.3. Métodos de evaluación en RS

El diseño adecuado de un sistema de evaluación es fundamental para poder entender la efectividad de distintos algoritmos de recomendación. Un diseño incorrecto puede llevar a una sobreestimación o subestimación de la eficacia de un algoritmo. Los *RS* se pueden evaluar utilizando métodos *online* u *offline* [3]. En un sistema *online*, las reacciones de los usuarios se miden con respecto a las recomendaciones presentadas. Por lo tanto, la participación del usuario es esencial en los sistemas *online*. Sin embargo, los sistemas *online* requieren de una participación activa de los usuarios, por lo que normalmente no es factible utilizarlos en la investigación y en la evaluación de algoritmos.

Hacer pruebas sobre múltiples conjuntos de datos es importante para asegurar una mayor capacidad de generalización del sistema de recomendación, de forma que se pueda estar seguro de que el algoritmo funciona en una variedad de entornos. En estos casos, se utilizan evaluaciones *offline*, es decir, evaluaciones con conjuntos de datos históricos. Los métodos *offline* son, con diferencia, los métodos más comunes para evaluar los sistemas de recomendación desde una perspectiva de investigación.

2.3.1. Paradigmas de evaluación

Existen tres niveles de experimentos que se pueden realizar para comparar distintos algoritmos de recomendación. El primero corresponde a los experimentos *offline*, donde no hace falta que haya interacción con usuarios reales. El segundo nivel corresponde a los estudios de usuario, donde se pide a un grupo controlado de sujetos que usen el sistema y proporcionen su *feedback*. El último nivel corresponde a los experimentos *online*, donde se emplean usuarios reales de un sistema en producción para comparar distintos algoritmos. A continuación, se describen en más detalle estos tres paradigmas de evaluación [37].

Experimentos *offline*

Un experimento *offline* se realiza utilizando un conjunto de datos ya recopilado de usuarios que eligen o valoran ítems. Utilizando este conjunto de datos, se puede

intentar simular el comportamiento de los usuarios que interactúan con un sistema de recomendación. Al hacerlo, se asume que el comportamiento de los usuarios cuando se recopilaron los datos será lo suficientemente similar al comportamiento de los usuarios cuando se implemente el sistema de recomendación, de forma que se puedan tomar decisiones fiables basadas en esta simulación.

La principal ventaja de los experimentos *offline* es que no requieren interacción con usuarios reales, lo que permite comparar un amplio rango de algoritmos a bajo coste. La desventaja es que solo pueden responder a un conjunto muy limitado de preguntas, normalmente preguntas sobre la capacidad predictiva de un algoritmo. No se puede medir directamente la influencia del sistema de recomendación en el comportamiento del usuario en este entorno. Por lo tanto, el objetivo de los experimentos *offline* es filtrar los enfoques inapropiados, permitiendo reducir el número de algoritmos candidatos a ser probados en estudios de usuarios o experimentos *online*, que son más costosos.

Estudios de usuario

Muchos enfoques de recomendación dependen de la interacción de los usuarios con el sistema. Es muy difícil crear una simulación precisa de las interacciones de los usuarios con el sistema, por lo tanto, para evaluar adecuadamente estos sistemas, se pueden recopilar interacciones reales de los usuarios con el sistema que proporcionen información adicional. En estos casos, normalmente se realizan estudios de usuario.

Un estudio de usuario se lleva a cabo reclutando a un conjunto de sujetos de prueba y pidiéndoles que realicen varias tareas que requieran una interacción con el sistema de recomendación. Mientras los sujetos realizan las tareas, se observa y registra su comportamiento, recopilando cualquier número de mediciones cuantitativas. En muchos casos, se pueden hacer preguntas cualitativas, antes, durante y después de que se complete la tarea. Estas preguntas pueden recopilar datos que no son directamente observables. Se pueden recoger datos como cuantas veces se hizo clic en una recomendación o si el usuario miró la recomendación, mediante el seguimiento de ojos.

Hay que tener en cuenta que la consciencia de los usuarios de que están siendo observados puede afectar a su comportamiento, sesgando los resultados. Además, los usuarios reclutados pueden no ser representativos de la población general porque no se dispone de una variedad suficiente de usuarios, lo que puede introducir un sesgo adicional en los resultados.

Experimentos *online*

En los experimentos *online* también se emplean usuarios reales, pero en este caso, los usuarios no son reclutados para el estudio, sino que son usuarios reales de un sistema desplegado y en producción. Este enfoque es normalmente menos susceptible a los sesgos en el proceso de reclutamiento, ya que los usuarios no son conscientes de que están participando en un estudio. Se pueden muestrear aleatoriamente los usuarios del sistema y dividirlos en grupos de control y de tratamiento, para comparar el rendimiento de distintos algoritmos.

La efectividad de los algoritmos en estos entornos se puede medir mediante métricas

de negocio, como el número de clics en las recomendaciones o el tiempo que pasa un usuario en el sistema. También se puede incorporar el beneficio esperado de las recomendaciones en la métrica, para tener en cuenta la importancia de los ítems recomendados. Algunas de las métricas más usadas son los ingresos por mil impresiones (RPM) o la tasa de clics (CTR).

Estos experimentos son también conocidos como *A/B testing*, y miden el impacto directo del sistema de recomendación en el usuario final. La idea básica consiste en comparar dos algoritmos, uno con un grupo de usuarios A y otro con un grupo B, manteniendo todas las demás condiciones lo más similares posible. Al final del proceso, se comparan las métricas de negocio de los dos grupos para determinar cuál de los dos algoritmos es más efectivo.

La principal desventaja de este enfoque es que requiere de una extensa base de usuarios para obtener resultados significativos. Por lo tanto, es muy difícil de implementar en sistemas que están en fase de desarrollo. Además, estos experimentos no suelen ser públicos y están restringidos a los propietarios del sistema. Esto significa que los resultados no pueden ser comparados con otros *benchmarks* realizados por otros investigadores. También, se corre el riesgo de que un algoritmo que se despliegue en producción como parte de una prueba *online* pueda tener un impacto negativo en la experiencia del usuario, lo que puede llevar a una pérdida de usuarios. Por estos motivos, lo más recomendable es comenzar con una evaluación *offline*, seguida de un estudio de usuario, y finalmente, si es posible, realizar un experimento *online*.

2.3.2. Métricas de evaluación en experimentos *offline*

Como se explicó en la sección 2.1.3, los sistemas de recomendación pueden tener distintos objetivos, que influyen en como el usuario valora las recomendaciones. En los experimentos realizados con usuarios, obtener un *feedback* explícito de los usuarios es relativamente sencillo, ya que simplemente se puede pedir a los usuarios que valoren las recomendaciones. Sin embargo, en los experimentos *offline*, no se dispone de esta información.

En los sistemas de recomendación se asume que si un sistema proporciona predicciones precisas respecto a un conjunto de prueba, estas predicciones serán preferidas por el usuario sobre otras menos precisas [37]. Además, la precisión es adecuada para ser medida en experimentos *offline*. Por lo tanto, se suelen buscar algoritmos que maximicen la precisión de estas recomendaciones.

Según como se plantee el problema de la recomendación (ver 2.1.3), la evaluación se realizará calculando la exactitud de las predicciones de valoración o del ranking de los ítems recomendados. Las métricas para el problema de recomendación top-k normalmente solo se centran en la exactitud del ranking de los top-k ítems, en vez de en todos los ítems que devuelva el sistema de recomendación. Además, algunas veces el orden de los ítems recomendados no es tan importante como la presencia de los ítems correctos en la lista de recomendaciones, por lo que según el caso se emplearán métricas de clasificación o de ranking. Tanto las métricas de clasificación como las de ranking se pueden emplear para evaluar a los sistemas de recomendación secuencial, aunque para estos sistemas, las métricas de ranking son preferibles, al tener en cuenta el orden de las recomendaciones.

A continuación, se describen algunas de las métricas más comunes para evaluar sistemas de recomendación.

Métricas de predicción de valoraciones

- **Raíz del error cuadrático medio (*RMSE*)**: El RMSE es una de las métricas más populares en la evaluación de la exactitud de las valoraciones predichas. El sistema genera valoraciones $\hat{r}_{ui}$ para un conjunto de prueba $\mathcal{T}$ de pares de usuarios e ítems (u, i) para los cuales se conocen las valoraciones reales r_{ui}. El RMSE entre las calificaciones predichas y reales se da por la fórmula:

$$\text{RMSE} = \sqrt{\frac{1}{|\mathcal{T}|} \sum_{(u,i)\in\mathcal{T}} (\hat{r}_{ui} - r_{ui})^2} \tag{2.1}$$

- **Error absoluto medio (*MAE*)**: El MAE es una alternativa popular al RMSE, que penaliza en menor medida los errores grandes en las predicciones. Se define por:

$$\text{MAE} = \frac{1}{|\mathcal{T}|} \sum_{(u,i)\in\mathcal{T}} |\hat{r}_{ui} - r_{ui}| \tag{2.2}$$

- **RMSE y MAE normalizados**: Para comparar los resultados de distintos conjuntos de datos, se pueden normalizar los valores de RMSE y MAE dividiendo por el rango de las valoraciones. De esta forma, se obtiene una medida de la precisión relativa de las predicciones.

Métricas de clasificación

- **Precisión**: La precisión es la proporción de ítems relevantes entre los ítems recomendados. Se define como:

$$\text{Precision@k} = \frac{\#(\text{k-ítems relevantes} \cap \text{k-ítems recomendados})}{\#\text{k-ítems recomendados}} \tag{2.3}$$

- **Exhaustividad (Recall)**: El *recall* es la proporción de ítems relevantes que han sido recomendados. Se define como:

$$\text{Recall@k} = \frac{\#(\text{k-ítems relevantes} \cap \text{k-ítems recomendados})}{\#\text{k-ítems relevantes}} \tag{2.4}$$

- **F-score (F_1)**: El F-score es la media armónica de la precisión y el *recall*. Se define como:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{2.5}$$

- **Área bajo la curva ROC (*AUC*)**: El AUC [7] se calcula como el área bajo la curva ROC, que es la curva que representa la tasa de verdaderos positivos frente a la tasa de falsos positivos.
- **Tasa de aciertos (Hit Rate@k)**: El *hit rate* es el número de listas de recomendaciones de longitud k que contienen al menos un ítem relevante.
- ***MRR***: El *Mean Reciprocal Rank* es la posición media del primer ítem relevante en la lista de recomendaciones. Dado r_u como el rango (posición) del primer ítem en la lista de recomendaciones que u ha elegido. Entonces, el MRR se define como:

$$\text{MRR} = \frac{1}{|U|} \sum_{u \in U} \frac{1}{r_u} \tag{2.6}$$

Métricas de ranking

- ***NPDM***: La *Normalized Distance based Performance Measure* es comúnmente usada en la recuperación de información. Distingue entre órdenes correctos de pares, órdenes incorrectos y empates. Formalmente, sea $\delta_{\succ_1,\succ_2}(i_1, i_2)$ una función de distancia entre un ranking de referencia $\succ_1$ y un ranking propuesto $\succ_2$. La distancia total sobre todos los pares de ítems en I es:

$$\beta_{\succ_1,\succ_2}(I) = \sum_{(i_1,i_2) \in I} \delta_{\succ_1,\succ_2}(i_1, i_2) \tag{2.7}$$

  Sea $m(\succ_1)$ un factor de normalización. La puntuación NDPM comparando un ranking propuesto de ítems $\succ_2$ con un ranking de referencia $\succ_1$ es

$$NDPM(I, \succ_1, \succ_2) = \frac{\beta_{\succ_1,\succ_2}(I)}{m(\succ_1)} \tag{2.8}$$

- ***AP***: La precisión media (*Average Precision*) da más peso a los errores sobre los ítems que aparecen en las primeras posiciones en el ranking de referencia. Sea $\succ_1$ el ranking de referencia y $\succ_2$ un ranking propuesto sobre un conjunto de ítems. La medida AP compara el orden de cada ítem en el ranking propuesto $\succ_2$ con respecto a sus ítems precedentes en el ranking de referencia $\succ_1$.

  Para cada $i_1 \in I$, el conjunto $Z^{i_1}(I, \succ)$ denota todos los pares de ítems (i_1, i_2) en I tal que $i_2 \succ i_1$. Estos son todos los ítems que son preferidos sobre i_1.

  Se define la función $\delta(i_1, i_2, \succ_1, \succ_2)$ para que sea igual a 1 cuando $\succ_1$ y $\succ_2$ están de acuerdo en los ítems i_1 e i_2, y cero en caso contrario.

  Sea $A^{i_1}(I, \succ_1, \succ_2)$ la puntuación normalizada entre $\succ_2$ y el ranking de referencia $\succ_1$ para todos los ítems i_2 tal que $i_2 \succ_1 i_1$. La puntuación AP de un ranking $\succ_2$ sobre I dado un ranking $\succ_1$ se define como

$$AP(I, \succ_1, \succ_2) = \frac{1}{|I| - 1} \sum_{i \in I} A^i(I, \succ_1, \succ_2) \tag{2.9}$$

- **R-score (R)**: En muchas aplicaciones, el usuario puede elegir solo un ítem, o un conjunto muy pequeño de ítems. En esos casos, se puede esperar que los usuarios solo observen unos pocos ítems de la parte superior de la lista de recomendaciones. El R-score asume que el valor de las recomendaciones disminuye exponencialmente a lo largo del ranking para obtener la siguiente puntuación para cada usuario u:

$$R_u = \sum_j \frac{\text{máx}(r_{u,i_j} - d, 0)}{2^{\frac{j-1}{\alpha-1}}} \tag{2.10}$$

  donde i_j es el ítem en la posición j, $r_{u,i}$ es la valoración del usuario u del ítem i, d es una valoración neutral, y α es un parámetro que controla la disminución exponencial del valor de las posiciones en el ranking.

- **NDCG**: En otras aplicaciones, se espera que el usuario vea una parte relativamente grande de la lista de recomendaciones. En esos casos, se necesita una disminución más lenta del descuento posicional. La NDCG es una medida de recuperación de información, donde las posiciones se descuentan logarítmicamente. Suponiendo que cada usuario u tiene una "ganancia" $g_{u,i}$ por ser recomendado el ítem i, la DCG promedio para una lista J de ítems se define como

$$\text{DCG} = \frac{1}{N} \sum_{u=1}^{N} \sum_{j=1}^{J} \frac{g_{u,i_j}}{\log_b(j+1)} \tag{2.11}$$

  donde i_j es el ítem en la posición j de la lista. La base del logaritmo es un parámetro libre, pero normalmente se usa la base 2. La ganancia $g_{u,i}$ se puede entender como la relevancia del ítem i para el usuario u. Normalmente, el valor de la ganancia es una función exponencial definida por

$$g_{u,i} = 2^{rel_{u,i}} - 1 \tag{2.12}$$

  donde la relevancia $rel_{u,i}$ es una función heurística de las valoraciones o *hits*. La NDCG es la versión normalizada de la DCG dada por

$$\text{NDCG} = \frac{\text{DCG}}{\text{DCG}^*} \tag{2.13}$$

  donde DCG^* es la DCG ideal.

2.4. Grandes modelos de lenguaje

Los grandes modelos de lenguaje (*LLM*) son algoritmos de aprendizaje profundo que pueden reconocer, extraer, resumir, predecir y generar texto basado en el conocimiento adquirido durante el entrenamiento en conjuntos de datos masivos. Todos los modelos de lenguaje tienen en común que realizan tareas relacionadas con el procesamiento del lenguaje natural (*NLP*).

Los *LLM* son considerados grandes porque están compuestos por una gran cantidad de parámetros (los pesos y sesgos de las redes neuronales subyacentes), del orden de miles de millones, y son entrenados con cantidades masivas de datos. Gracias a esta combinación, los LLM pueden generar respuestas más precisas y sofisticadas en comparación con otros modelos de arquitectura similar, pero con un número de parámetros mucho menor. La calidad de un modelo depende de su tamaño y de la cantidad de datos de entrenamiento, pero también de la calidad de los datos.

Además, los modelos necesitan datos de naturaleza muy diversa para poder realizar varias tareas distintas de *NLP*. Sin embargo, si ya se tiene un modelo destinado a ser bueno en resolver un conjunto amplio de tareas, se puede emplear un método de "ajuste fino" (*fine-tuning*) para adaptarlo a las tareas descritas por un conjunto de datos más pequeño. De esta forma, un modelo de lenguaje fundacional, con altas capacidades de generalización, se transforma en un modelo especializado (*fine-tuned*) en realizar las tareas de un dominio más concreto.

La forma más habitual de interactuar con los LLM es mediante una interfaz de chat, donde el usuario puede introducir un texto de entrada, conocido como *prompt*, con una pregunta o tarea a resolver y el modelo generará una respuesta o solución basada en el conocimiento adquirido durante el entrenamiento. Este conocimiento se puede extender adjuntando información textual adicional al *prompt*, proporcionando un contexto sobre el que basar las respuestas. La longitud del contexto que puede procesar un LLM es limitado, siendo normalmente del orden de miles de tókenes, siendo un token una unidad de texto, como una palabra o un carácter.

Aunque los LLM son capaces de generar respuestas de gran calidad, también pueden generar respuestas incorrectas o sin ninguna relación con la entrada. Estas generaciones se conocen como "alucinaciones" y es uno de los mayores problemas de los LLM, presente incluso en los modelos más avanzados como GPT-4 [80].

2.4.1. Evolución de los LLM

Los primeros modelos de procesamiento del lenguaje natural desarrollados en la década de 1950 se basaban en reglas humanas en lugar de usar algoritmos de aprendizaje automático para las tareas de NLP [76]. Esto los hacía adecuados para tareas más simples que no requerían demasiadas reglas, como la clasificación de texto, pero inadecuados para tareas más complejas, como la traducción automática. Los modelos basados en reglas también tenían deficiencias en los casos límite, ya que no podían hacer predicciones o clasificaciones precisas para datos nunca vistos para los que no se habían establecido reglas claras.

A finales de la década de los 80, los métodos basados en aprendizaje estadístico comenzaron a ganar popularidad. La idea principal de estos métodos es construir el modelo de predicción basándose en el contexto más reciente. Los modelos de lenguaje con un contexto fijo de longitud n también se conocen como modelos de n-gramas. Estos modelos fueron ampliamente usados en tareas de NLP, y fueron superados por un modelo basado en *MLP* por primera vez en 2003 con el modelo propuesto por Bengio et al. [9] (premio Turing de 2018).

A partir de entonces, se comenzaron a desarrollar modelos de lenguaje basados en métodos de aprendizaje neuronal más complejos, como las redes neuronales

recurrentes (*RNN*), especialmente las de memoria a corto y largo plazo (*LSTM*) [48]. Estos modelos pueden memorizar datos pasados hasta cierto punto y, por lo tanto, proporcionar predicciones y clasificaciones precisas. Sin embargo, estos modelos también tienen limitaciones, como la incapacidad de capturar dependencias en los datos y la dificultad para utilizar secuencias largas en el entrenamiento, ya que aumentaban considerablemente el costo computacional.

El desarrollo de la arquitectura *transformer* [115] en 2017 marcó un hito en el desarrollo de los modelos de lenguaje. Los modelos basados en esta arquitectura emplean un mecanismo de atención que les permite capturar mejor las dependencias en los datos, además su estructura paralela les permite procesar secuencias más largas de datos. Esto hace que los modelos basados en *transformer* sean más eficientes y precisos y tengan mejores capacidades de generalización que los modelos anteriores basados en RNN y LSTM. Esta arquitectura permitió mejorar los modelos de NLP para entender secuencias mucho más largas de texto y realizar una gama mucho más amplia de tareas.

Sin embargo, en retrospectiva, los modelos desarrollados en este periodo, como BERT [25], tenían capacidades limitadas, principalmente debido a la falta de grandes conjuntos de datos y recursos computacionales adecuados. Además, estos modelos, a pesar de haber despertado la atención de investigadores y expertos en el campo, no eran lo suficientemente precisos para ser comercializados.

El desarrollo de LLM fue principalmente impulsado por el lanzamiento de GPT-3 [13] por OpenAI en 2020. Este modelo fue entrenado sobre un corpus de 300 mil millones de tókenes, alcanzado un tamaño de 175 mil millones de parámetros. Esto le permitió producir respuestas a tareas de NLP más precisas y completas comparadas con las de modelos anteriores, además, presentaba habilidades no presentes en modelos más pequeños, lo que se conoce como habilidades emergentes [122]. Esta capacidad de mejora se intenta predecir mediante leyes de escalado (*scaling laws*), que relacionan el rendimiento de un modelo con su tamaño, el tamaño del conjunto de datos de entrenamiento y las horas de cómputo usadas durante el entrenamiento [59].

En 2022, OpenAI lanzó ChatGPT [1], una aplicación web basada en GPT-3.5, una versión de GPT-3 optimizada para mantener conversaciones, que permite a los usuarios interactuar con un LLM de forma gratuita y sin necesidad de conocimientos técnicos. Este lanzamiento generó un gran interés tanto en la comunidad científica como en el público en general, ya que desbloqueó nuevas posibilidades en el campo de la inteligencia artificial y democratizó el acceso a tecnologías punteras de NLP.

Esta evolución fue principalmente impulsada por avances científicos relacionados con las redes neuronales, mecanismos de atención, métodos de aprendizaje autosupervisado (*Self Supervised Learning* o *SSL*), etc. Pero también ha sido acompañada por avances en la infraestructura de computación, que permiten a los investigadores entrenar modelos más grandes y complejos cada vez en menos tiempo y con menos recursos. Actualmente, estamos en un periodo de rápida evolución en el campo de los LLM, con modelos cada vez más capaces que se lanzan regularmente, tanto de forma cerrada, como ChatGPT, como de forma abierta.

[1] `https://openai.com/index/chatgpt/`

2.4.2. Métodos de entrenamiento y adaptación de LLM

Los grandes modelos de lenguaje se entrenan desde cero utilizando un método de aprendizaje autosupervisado y semisupervisado sobre conjuntos masivos de datos. Este proceso se conoce como preentrenamiento, y es extremadamente costoso en términos de recursos computacionales y tiempo. A este proceso también hay que sumarle el tiempo y los recursos necesarios para obtener, almacenar y preprocesar los datos de entrenamiento, una tarea igual de compleja y costosa que el entrenamiento en sí. En la figura 2.3 se muestra un esquema de un *pipeline* de entrenamiento de un LLM, incluyendo las fases de preentrenamiento, *fine-tuning* y *RLHF*, por Wolfe [125].

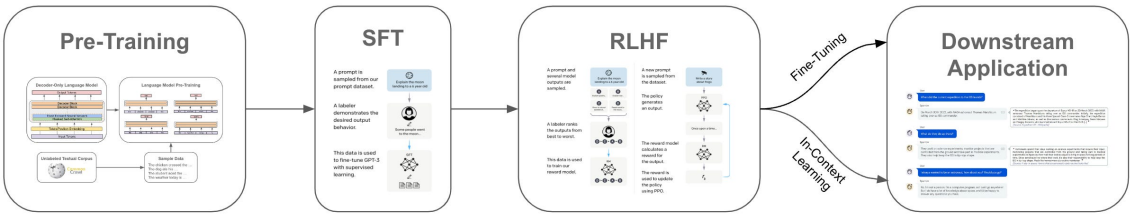

Figura 2.3: *Pipeline* de entrenamiento de un LLM, incluyendo las fases de *pre-training*, *supervised fine-tuning* y *RLHF*. Por Wolfe [125]

Una de las ventajas de entrenar un LLM desde cero, es la posibilidad de elegir la arquitectura más adecuada para la tarea que se quiere realizar. Sin embargo, a pesar de que se han desarrollado arquitecturas para LLM distintas a la de *transformer*, como RWKV [84], basada en *RNN*, o Mamba [36], por ahora no han demostrado ser tan efectivas como las basadas en *transformer*. Por lo tanto, la decisión en la arquitectura se suele limitar a elegir entre una arquitectura de *transformer* de codificador-decodificador, como T5 [92], o de solo-decodificador, como los modelos de la familia GPT.

Por estos motivos, pre-entrenar un LLM desde cero no suele ser práctico para la mayoría de equipos de investigación o empresas, que no pueden permitirse el lujo de gastar tanto tiempo y recursos en este proceso. En su lugar, se recurre al *fine-tuning*, un enfoque que permite entrenar los parámetros de un modelo preentrenado para que aprendan sobre un conjunto de datos nuevo. De esta manera, se pueden extender las habilidades de los LLM sin tener que incurrir en los costes del preentrenamiento. A continuación, se muestran los métodos más comunes para adaptar el comportamiento un LLM de manera eficiente [137].

Instruction Tuning

Instruction tuning es un método de *fine-tuning* que consiste en entrenar un LLM en una colección de conjuntos de datos que contienen instrucciones descritas en lenguaje natural para realizar tareas específicas. Para llevar a cabo el *instruction tuning*, primero es necesario recopilar o construir instancias formateadas de instrucciones. Luego, se emplean estas instancias formateadas para hacer *fine-tuning* al LLM de manera supervisada. Después de este proceso, los LLM demuestran habilidades superiores para generalizar tareas no vistas [123]. En la figura 2.4 se muestra un esquema comparativo entre *fine-tuning*, *prompting* e *instruction tuning*, por Wei et al. [123].

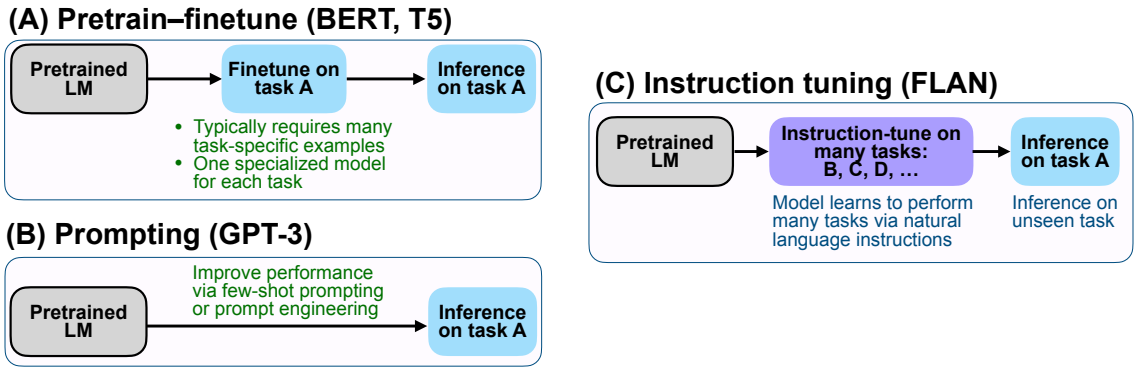

Figura 2.4: Comparación entre *fine-tuning*, *prompting* e *instruction tuning*. Por Wei et al. [123]

Normalmente, una instancia formateada como una instrucción contiene la descripción de la tarea (o instrucción), una entrada, la correspondiente salida y, opcionalmente, un pequeño número de ejemplos. Los datos para construir estas instancias se pueden derivar de conjuntos de datos diseñados para tareas tradicionales de NLP, para ello tan solo habría que formatear los datos añadiendo una descripción de la tarea en lenguaje natural. Sin embargo, estos conjuntos de datos suelen ser limitados en cuanto a la diversidad de instrucciones, por lo que no son adecuados para entrenar modelos de lenguaje que se quieran especializar en tareas más concretas.

No obstante, construir instrucciones a partir de conjuntos de datos tradicionales puede resultar en una variedad limitada en las instrucciones, o que las salidas no se adecúen a las necesidades del usuario. Para resolver esto, InstructGPT [82] propone utilizar las consultas que los usuarios han enviado a la API de OpenAI como descripciones de tareas. De esta forma, se pueden obtener instrucciones más diversas y que se ajusten mejor a las necesidades de los usuarios. También se han propuesto métodos semiautomáticos para construir instancias, como Self-Instruct [120], que utiliza el propio LLM para generar instrucciones a partir de ejemplos existentes.

A diferencia del preentrenamiento que utiliza un enfoque no supervisado y conjuntos de datos muy grandes, el *instruction tuning* es más eficiente, ya que solo se necesita un número moderado de datos para el entrenamiento. Además, hay tener en cuenta que es un proceso supervisado, por lo que la optimización es diferente a la del preentrenamiento, requiriendo una configuración de hiperparámetros diferente.

A pesar de no usar una gran cantidad de datos, el *instruction tuning* ha demostrado ser clave para mejorar las habilidades de los LLM. El tamaño del modelo fundacional no es de vital importancia, ya que modelos de diferentes tamaños han demostrado verse beneficiados por el *instruction tuning*, y modelos más pequeños con *instruction tuning* pueden superar a modelos fundacionales mucho más grandes [20].

Este proceso habilita a los LLM a entender y seguir mejor las instrucciones en lenguaje natural, lo que les permite realizar tareas específicas sin necesidad de ejemplos de entrenamiento. También puede otorgar al LLM conocimiento sobre un dominio específico, lo que le permite realizar tareas específicas de ese dominio con mayor precisión. Por ejemplo, Med-PaLM [105] es un modelo que ha sido entrenado con instrucciones del dominio clínico y que ha demostrado alcanzar niveles de rendimiento comparables a los de los médicos expertos en *benchmarks* de preguntas médicas.

Alignment Tuning

Los LLM pueden exhibir comportamientos no deseados, inventándose información falsa, persiguiendo objetivos indeseados, o produciendo lenguaje dañino, engañoso y sesgado. Esto se debe a que en el proceso de preentrenamiento no se tienen en consideración los valores o preferencias humanas, por ejemplo, si partes del corpus de entrenamiento incluyen mensajes racistas de usuarios de internet, este comportamiento se puede filtrar en las respuestas. Para evitar estos comportamientos, se ha propuesto el *alignment tuning*, un proceso que permite alinear a los LLM con las expectativas humanas [82]. Este proceso, a diferencia del *instruction tuning*, requiere considerar criterios muy diferentes, como la utilidad, la honestidad y la inocuidad, además, esta tarea de alineamiento puede reducir las capacidades generales de los LLM.

El *feedback* humano de alta calidad es indispensable para poder ajustar adecuadamente a los LLM con las preferencias y valores humanos. El método más común para generar datos de *feedback* humano es la anotación humana, esto destaca el papel crítico de seleccionar anotadores humanos adecuados. Para proporcionar *feedback* de alta calidad, los anotadores humanos deben tener un nivel de educación cualificado y un gran nivel en el idioma que se esté utilizando.

Para recoger el *feedback* humano, normalmente los anotadores ordenan las distintas respuestas del modelo según su idoneidad. Además, los anotadores pueden proporcionar respuestas a preguntas diseñadas por los investigadores sobre el alineamiento de la salida del modelo. También se pueden establecer sistemas basados en reglas o en el propio LLM para detectar automáticamente cuando la respuesta generada viola unas reglas predefinidas [80].

El aprendizaje por refuerzo a partir de *feedback* humano (*RLHF*) [19] es el método más usado para ajustar a los LLM con los valores humanos. Este método emplea algoritmos de aprendizaje por refuerzo, como *PPO* [104], para adaptar los LLM a partir de los datos de *feedback* humano, aprendiendo a optimizar un modelo de recompensa. Este enfoque incorpora a los humanos en el proceso de entrenamiento para desarrollar LLM alineados adecuadamente.

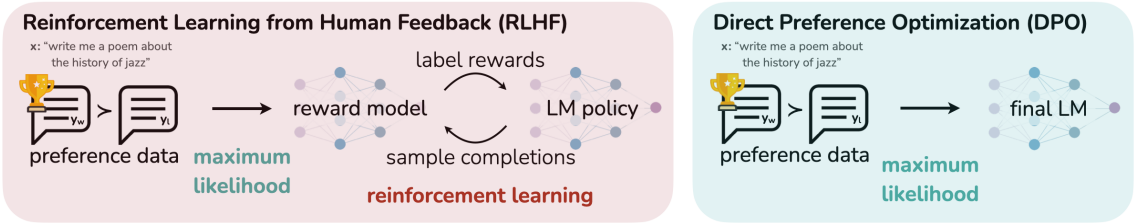

Figura 2.5: Comparación de RLHF y DPO. Por Rafailov et al. [91]

Aunque el RLHF ha logrado con éxito alinear los comportamientos de los LLM, también sufre de limitaciones notables. En primer lugar, se necesita entrenar múltiples modelos, incluyendo el modelo que se está ajustando, el modelo de recompensa y el modelo de referencia al mismo tiempo, un proceso tedioso y que consume muchos recursos. Además, el algoritmo de PPO usado en RLHF es bastante complejo y sensible a los hiperparámetros. Como alternativa, se está explorando la posibilidad de optimizar directamente los LLM para adherirse a las preferencias y valores humanos,

utilizando un *fine-tuning* supervisado sin emplear aprendizaje por refuerzo, como se ha propuesto en DPO [91] u ORPO [49]. En la figura 2.5 se muestra una ilustración comparativa entre RLHF y DPO, por Rafailov et al. [91].

Parameter Efficient Fine-Tuning

Como los LLM están compuestos por una gran cantidad de parámetros, realizar un *fine-tuning* completo sobre todos los parámetros puede ser costoso. Por ello, se han desarrollado métodos para realizar un *fine-tuning* eficiente de parámetros (*PEFT*), reduciendo el número de parámetros a entrenar, pero manteniendo un buen rendimiento. En la figura 2.6 se muestra una ilustración comparativa de los distintos métodos de *PEFT*, por Zhao et al. [137].

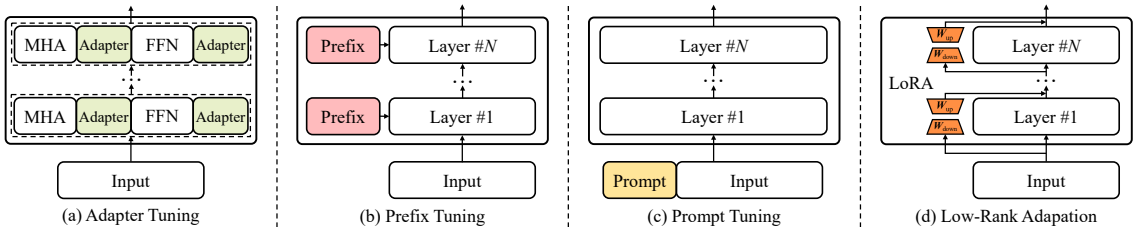

Figura 2.6: Comparación de los distintos métodos de *PEFT*. Por Zhao et al. [137]

Para lograr el objetivo del *PEFT*, se pueden incorporan módulos de redes neuronales más pequeñas, llamados adaptadores, en los modelos de *transformer*. Para implementar el adaptador, se propone utilizar una arquitectura de cuello de botella [51] que primero comprime el vector de características original en una dimensión más pequeña y luego lo recupera a la dimensión original. Los módulos de adaptador se integrarían en cada capa de tipo *transformer*.

También se ha propuesto una alternativa, conocida como *prefix tuning* [68] que consiste en adjuntar una secuencia de prefijos, que consisten en un conjunto de vectores entrenables, a cada capa de tipo *transformer*. Estos vectores de prefijos son específicos de la tarea, y se pueden considerar como tókenes virtuales.

De forma similar al *prefix tuning*, el *prompt tuning* [63] se centra en incorporar vectores entrenables de un *prompt* en la capa de entrada. De esta forma, extiende el texto de entrada, incluyendo un grupo de tókenes de *prompt*, y luego toma la entrada extendida para resolver tareas específicas.

LoRA [52] propone una forma de reducir el número de parámetros entrenables mediante la imposición de una restricción de rango bajo (*low-rank*) para aproximar la matriz de actualización en cada capa densa. La principal ventaja de LoRA es que puede ahorrar en gran medida el uso de memoria y almacenamiento, ya que se puede mantener una única copia del LLM, mientras se mantienen una serie de matrices de descomposición de rango bajo específicas de la tarea para adaptarlo a diferentes tareas.

Con la popularización de los LLM, los métodos de *fine-tuning* eficientes han atraído cada vez más la atención de la comunidad científica para desarrollar modelos entrenados en tares concretas de manera más rápida y ligera. En particular, LoRA ha ganado una gran popularidad, habiéndose aplicado a una multitud de LLM de código abierto para mejorar su rendimiento.

Memory Efficient Fine-Tuning

Debido al gigantesco número de parámetros de los LLM, estos requieren de una gran cantidad de memoria para realizar tanto el entrenamiento como la inferencia, haciéndolos muy costosos de desplegar en aplicaciones reales. Para reducir la huella de memoria de los LLM, se han propuesto métodos para comprimir los modelos, como la cuantización, de tal forma que los LLM puedan ser utilizados en entornos con recursos limitados, reduciendo también la latencia en la inferencia.

La cuantización se refiere al proceso de mapeo de números de coma flotante a enteros, siendo la cuantización a 8 bits (int8) la más común. Para los modelos basados en redes neuronales, los datos que se pueden cuantizar son los pesos o parámetros del modelo, al estar representados originalmente en números de coma flotante. En la figura 2.7 se muestra una comparación de la precisión de los distintos tipos de coma flotante, por Labonne [62].

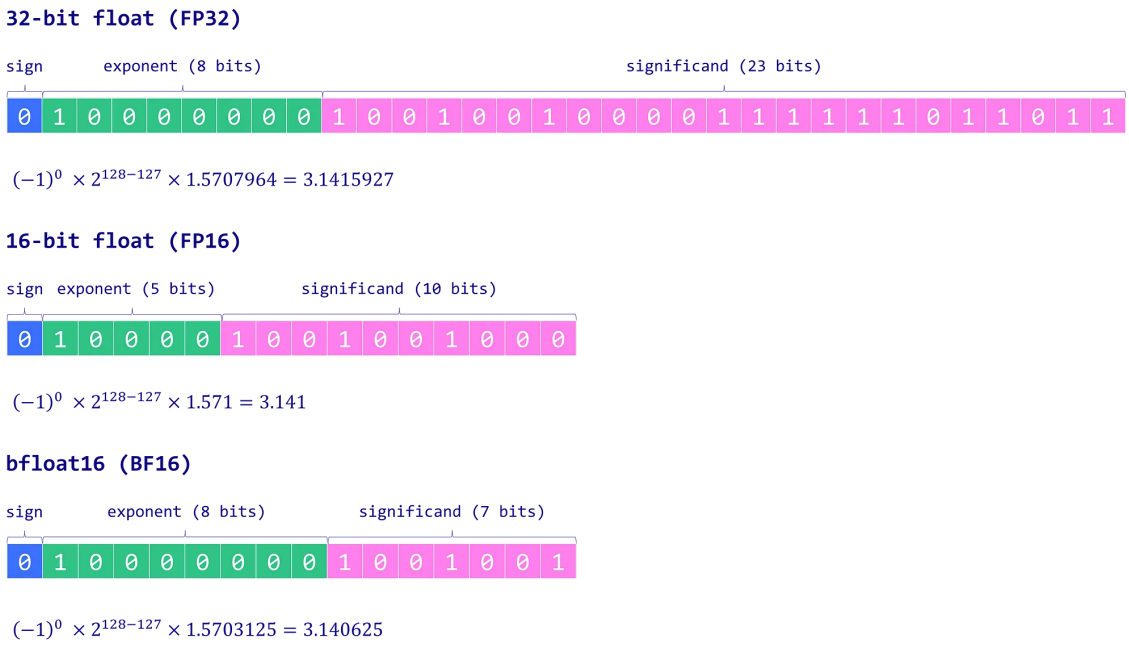

Figura 2.7: Comparación de la precisión de los distintos tipos de coma flotante. Por Labonne [62]

Existen dos enfoques principales para la cuantización de LLM, el entrenamiento consciente de la cuantización (QAT), y la cuantización posterior al entrenamiento (PTQ). El entrenamiento consciente de la cuantización requiere de un proceso de reentrenamiento completo del modelo [75], mientras que la cuantización posterior al entrenamiento no requiere de reentrenamiento [23]. Como los LLM contienen una gran cantidad de parámetros, los métodos de PTQ son preferidos por su coste computacional mucho más reducido comparado con el de los métodos de QAT, GPTQ [31] y AWQ [70] son dos ejemplos de este tipo de métodos.

En la cuantización posterior al entrenamiento, la cuantización directa a un número bajo de bits (int4) suele resultar en una degradación notable en el rendimiento del modelo. Para superar este desafío, se ha propuesto QLoRA [24], que incorpora pequeños adaptadores entrenables (de 16 bits de precisión) en los modelos cuantizados,

ofreciendo un proceso de *fine-tuning* eficiente y de alta precisión. QLoRA combina las ventajas de LoRA, comentado anteriormente, y de los métodos de cuantización, y ha demostrado que es posible cuantizar modelos a 4 bits con QLoRA y obtener un rendimiento muy similar al de los modelos de 16 bits. Estos métodos han permitido democratizar el entrenamiento y despliegue de LLM en entornos con recursos limitados, pudiendo realizar estas tareas en tarjetas gráficas de consumo general.

La cuantización de los LLM ha demostrado ser una técnica eficaz para reducir la huella de memoria y la latencia de los LLM tanto en la inferencia como en el entrenamiento, por lo que se ha convertido en una técnica esencial en el despliegue de LLM. Es importante encontrar el nivel de precisión que el modelo puede soportar sin que su rendimiento se vea afectado. Pero comparados con los modelos originales, los LLM cuantizados tienen una huella de memoria más pequeña y una velocidad de inferencia más rápida a cambio de una degradación mínima en el rendimiento.

2.4.3. Inferencia de LLM

Después del preentrenamiento y del *fine-tuning*, una técnica para mejorar las respuestas de los LLM es diseñar estrategias de *prompting* adecuadas para resolver diversas tareas. Los *prompts* pueden ser optimizados de forma manual o automática. Siguiendo métodos como el aprendizaje en contexto (*in-context learning*), el aprendizaje en cadena de pensamiento (*chain-of-thought*) y la planificación, se pueden diseñar *prompts* que mejoran el rendimiento de los LLM en tareas específicas.

Prompt engineering

El proceso de crear manualmente un *prompt* que se adecúe a una tarea específica se conoce como *prompt engineering* [74]. Un *prompt* bien diseñado es muy útil a la hora de solicitar al LLM que realice tareas específicas. Aunque la creación manual de *prompts* es un proceso intuitivo, también puede llegar a consumir mucho tiempo y, lo que es más importante, los modelos son muy sensibles a los *prompts* creados de forma incorrecta, lo que puede llevar a un bajo rendimiento en la tarea a resolver. Por ello, se han propuesto enfoques automáticos para optimizar *prompts* y lograr un rendimiento óptimo.

In-Context Learning (ICL)

Esta forma de *prompting* es propuesta inicialmente junto con GPT-3 [13], y se ha convertido en una forma habitual de utilizar los LLM. En *ICL* se usa un *prompt* formateado en lenguaje natural, que consiste en la descripción de una tarea a realizar y/o algunos ejemplos de la tarea como demostraciones. Para la construcción de la plantilla del *prompt*, se comienza con la descripción de la tarea, se seleccionan algunos ejemplos del conjunto de datos, y se combinan en un orden específico. Finalmente, se añade una instancia del conjunto de prueba a las demostraciones para que el LLM genere la salida. Basándose en las demostraciones de la tarea, los LLM pueden reconocer y realizar una nueva tarea sin necesidad de un entrenamiento explícito. La eficacia de ICL está altamente influenciada por el diseño de las demostraciones.

Chain-of-Thought (CoT)

CoT es una estrategia de *prompting* que mejora el rendimiento de los LLM en tareas complejas de razonamiento [121]. En lugar de construir los *prompts* con pares de entrada-salida, como en ICL, CoT va un paso más allá e incorpora pasos de razonamiento intermedios, que sirven de puente entre las entradas y las salidas. CoT se propuso por primera vez como una extensión de ICL, que aumenta cada demostración $\langle \text{entrada}, \text{salida} \rangle$ como $\langle \text{entrada}, \text{CoT}, \text{salida} \rangle$. Un CoT es una serie de pasos de razonamiento intermedios para conectar la entrada y la salida. Con estas demostraciones mejoradas, los LLM pueden seguirlas para generar otros CoT y la respuesta para una nueva entrada. Sin embargo, a diferencia de los pares $\langle \text{entrada}, \text{salida} \rangle$ en ICL, los CoT son difíciles de obtener y suelen requerir de intervención humana. Afortunadamente, se ha descubierto que los LLM pueden generar CoT a través de instrucciones simples como *"Let's think step by step."* [61], lo que hace que este método sea fácil de usar. A pesar de la mejora en el rendimiento en las tareas de razonamiento complejas, CoT sigue sufriendo problemas debido a la generación de razonamiento incorrecto. En la figura 2.8 se muestra una comparación ilustrativa de ICL y CoT, por Zhao et al. [137].

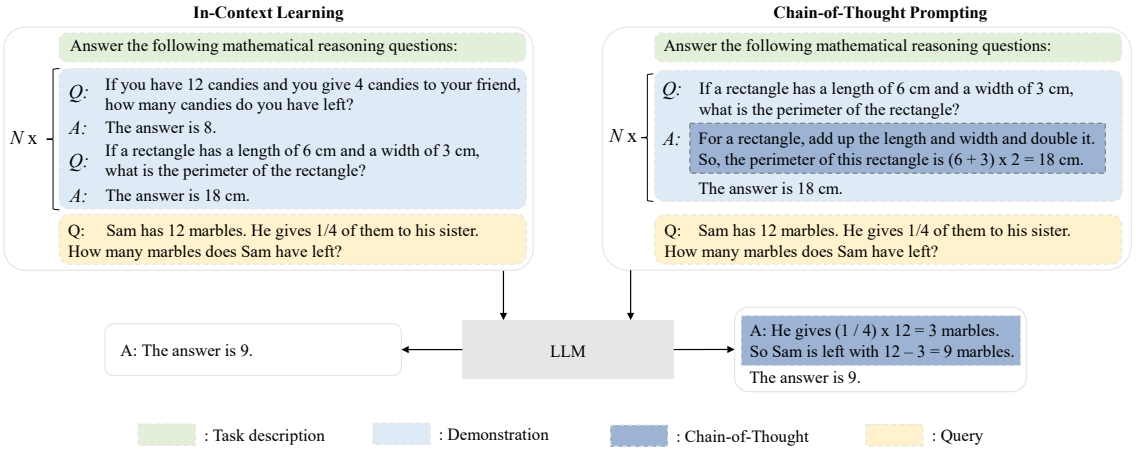

Figura 2.8: Comparación ilustrativa de ICL y CoT. Por Zhao et al. [137]

Planificación

Los métodos de ICL y CoT son enfoques simples y efectivos para mejorar de forma general el rendimiento de los LLM en la resolución de diversas tareas. Sin embargo, estos métodos sufren de carencias a la hora de resolver tareas más complejas. Por eso, se han propuesto distintos métodos de planificación que dividen las tareas complejas en subtareas más pequeñas y generan un plan de acciones para llevar a cabo la tarea [39].

2.4.4. Evaluación de los LLM

Se han propuesto un gran número de métodos y *benchmarks* para evaluar la las habilidades de los distintos LLM de forma empírica y comprobar su efectividad. Dentro de los métodos de evaluación, se pueden distinguir dos tipos principales: la evaluación automática y la evaluación humana [16].

Evaluación automática

La evaluación automática es el método más común de evaluación, emplea métricas estandarizadas y herramientas de evaluación para evaluar el rendimiento de los modelos. En comparación con la evaluación humana, la evaluación automática no requiere de una participación humana, lo que ahorra tiempo y reduce el impacto de los factores subjetivos humanos, haciendo que el proceso de evaluación sea más estandarizado y fácil de comparar.

Este proceso se centra principalmente en examinar las habilidades básicas de los modelos, como la capacidad de razonamiento y el conocimiento general. Estas habilidades se pueden evaluar con tareas bien definidas, por lo que se han diseñado *benchmarks* para evaluar los LLM. Los *benchmarks* más comunes son diseñados como problemas cerrados como las preguntas de opción múltiple. Estos *benchmarks* se pueden dividir en dos categorías principales, los orientados al conocimiento, como MMLU [44] y los orientados al razonamiento, como BigBench [106]. También existen *benchmarks* para dominios más específicos, como MATH [45], para las matemáticas, o HumanEval [17], para la programación (concretamente en Python).

Para realizar la evaluación, cada problema se formatea como un *prompt* para que los LLM generen el texto de la respuesta. Luego, se compara la respuesta generada con la respuesta real para calcular el rendimiento del modelo, utilizando métricas estándares como la precisión, la exactitud y la puntuación F1. En la evaluación se pueden emplear distintas técnicas de *prompting* para mejorar el rendimiento del modelo, incluyendo N demostraciones o empleando CoT en los casos más complejos.

Una desventaja de los *benchmarks* tradicionales es que la respuesta esperada es rígida, esto puede no suponer un gran problema para los LLM fundacionales, pero los LLM que han recibido *fine-tuning* para adaptarse a las preferencias humanas pueden verse penalizados injustamente. Para resolver este problema, se han propuesto métodos que explotan las capacidades de los LLM más potentes, como GPT-4, para simular el trabajo de un evaluador humano. Por ejemplo, MT-bench [138] recoge un conjunto de preguntas de múltiples turnos para evaluar a los LLM y mejora la fiabilidad de los evaluadores basados en LLM mediante métodos como ICL y CoT. AlpacaEval [29], por otro lado, utiliza un modelo como GPT-4 como juez para juzgar pares de respuestas generadas por los LLM.

Evaluación humana

La creciente mejora en las capacidades de los LLM ha ido más allá de lo que pueden capturar las métricas de evaluación estándar en tareas de NLP. Por ello, la evaluación humana se ha convertido en una opción clara en casos donde la evaluación automática no sea adecuada. Aunque la evaluación automática puede ser capaz de evaluar algunas tareas de generación de texto, especialmente si se emplean métodos basados en LLM, la evaluación humana sigue siendo más fiable y precisa en la evaluación de tareas con respuestas más abiertas.

En este proceso, los evaluadores humanos evalúan la calidad y exactitud de los resultados generados por los LLM, en un contexto mucho más cercano a un escenario de aplicación real, por lo que se puede obtener un *feedback* más completo y preciso. Estos evaluadores pueden ser tanto expertos en un dominio, como usuarios ordinarios.

El número de evaluadores involucrados en el proceso es un factor clave para lograr una representación adecuada y que los resultados sean estadísticamente significativos.

Dentro de este enfoque, los evaluadores humanos juzgan la calidad de las respuestas generadas por los LLM en preguntas abiertas. La evaluación puede ser en forma de comparación de pares, donde los evaluadores comparan dos respuestas generadas por distintos LLM y eligen la mejor, o en forma de puntuación única, donde los evaluadores asignan una puntuación a una respuesta en función de su calidad. El ejemplo más destacado de este tipo de evaluación es Chatbot Arena [18], una plataforma de *crowdsourcing* que permite a los usuarios interactuar con dos LLM anónimos y comparar sus respuestas, los LLM ganan una puntuación Elo en función de la evaluación de los usuarios y son clasificados en un ranking.

Además de tener en cuenta el *feedback* de los evaluadores humanos, estos sistemas de evaluación también evitan los problemas de contaminación de datos, que afectan a los *benchmarks* tradicionales. Esto se debe a que el conjunto de test de estos *benchmarks* puede filtrarse en los datos de entrenamiento del LLM, lo que puede llevar a una evaluación incorrecta.

2.4.5. LLM destacados en la actualidad

La rápida evolución de los LLM ha llevado a la aparición de una gran cantidad de modelos de lenguaje, cada uno con sus propias características y capacidades. En la figura 2.9 se muestra un árbol evolutivo de los modelos de lenguaje modernos, mostrando la evolución de los LLM en los últimos años y destacando algunos de los modelos más conocidos [132]. Las ramas del árbol están codificadas con colores para indicar la arquitectura de *transformer* utilizada por cada modelo.

En la tabla 2.2 se muestran algunos de los modelos de lenguaje más destacados a fecha de redacción de este trabajo, junto con sus características más relevantes y su rendimiento en distintos *benchmarks*. Estos modelos han sido seleccionados por su alto rendimiento respecto a su tamaño, tanto en *benchmarks* automáticos como en evaluaciones humanas. Hay que tener en cuenta que la B en los números de parámetros indica miles de millones (*billions*), y que la T en los tókenes indica billones (*trillions*).

El orden de los modelos en la tabla 2.2 se ha establecido en función del número de parámetros, de mayor a menor, ya que el rendimiento en los *benchmarks* suele estar relacionado con el tamaño del modelo. Los modelos que no han publicado sus pesos se han colocado al principio de la tabla, ya que son los que tienen un rendimiento más alto, pero apenas proporcionan información sobre su arquitectura.

Los resultados de los *benchmarks* de evaluación tradicionales se han obtenido directamente de los trabajos originales. En el caso de los modelos que ofrecen datos tanto para la versión fundacional como para una versión *instruct-tuned*, se ha optado por mostrar los resultados de la versión *instruct-tuned*, ya que suelen ser los más altos.

Algunas publicaciones no comparten la información del método de *prompting* empleado en los *benchmarks*, esto se indica con una cruz al lado de los valores que no proporcionan esta información. En el resto de casos, las distintas publicaciones suelen emplear las mismas técnicas para cada uno de los *benchmarks*:

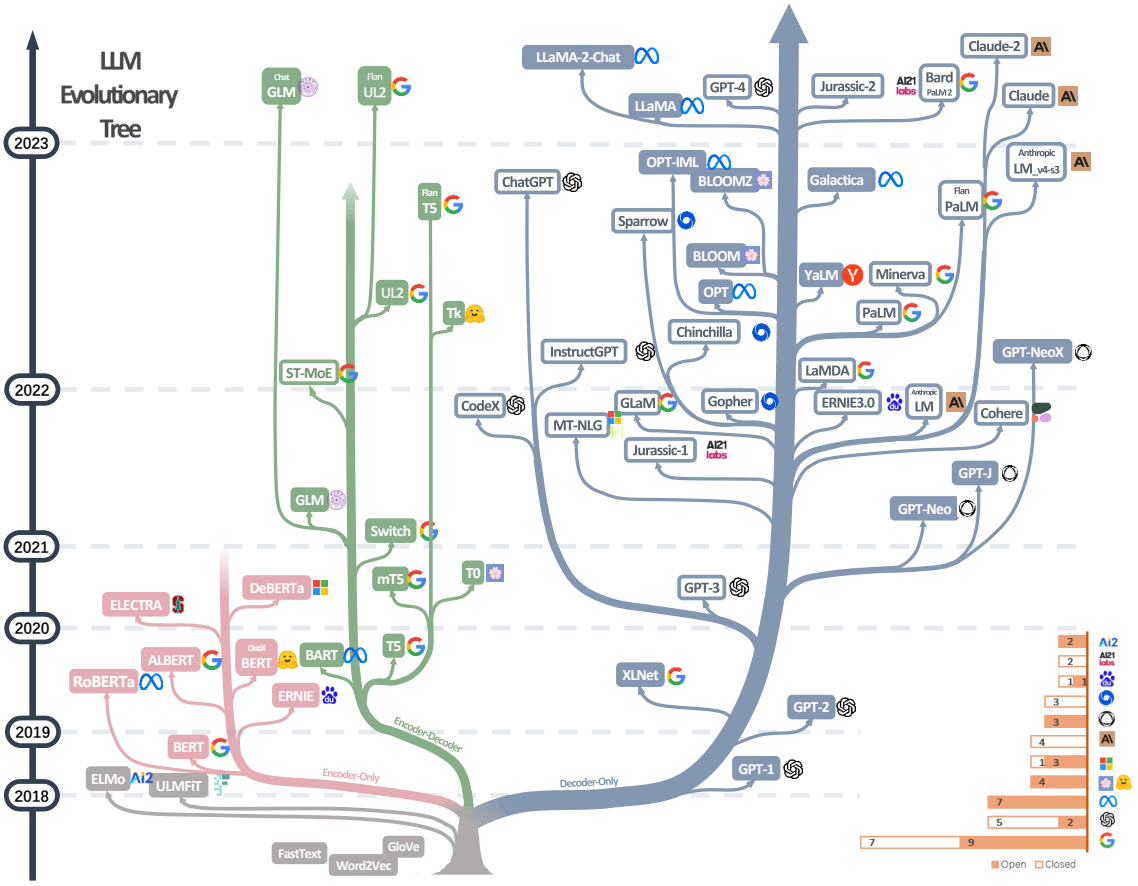

Figura 2.9: Árbol evolutivo de los LLM en los últimos años, destacando algunos de los modelos más conocidos, el color de las ramas indica la arquitectura del modelo y el relleno de la etiqueta de los nombres si es abierto. Por Yang et al. [132]

Modelo	OW	Parám	Tokens	Fecha	MMLU[44]	BBH[106]	MATH[21]	HuEval[17]	AlEval[29]	Arena[18]
Claude 3.5 Sonnet [6]	✗	N/A	N/A	06/2024	88.7	93.1	71.1*	92	N/A	1272
GPT-4o [79]	✗	N/A	N/A	05/2024	88.7†	N/A	76.6†	90.2†	57.5†	1287
Gemini 1.5 Pro [111]	✗	N/A	N/A	02/2024	85.9	89.2	67.7	84.1	N/A	1267
Qwen 2 72B [89]	✓	72B	N/A	06/2024	84.2	82.4	51.1	64.6	N/A	1186
Llama 3 70B [4]	✓	70B	15T	04/2024	82	81.3	50.4	81.7	34.4	1207
Llama 2 70B [113]	✓	70B	2T	07/2023	69.7	65.7	11.6	25.6	14.7	1093
Gemma 2 27B [33]	✓	27B	13T	06/2024	76.2	74.9	42.3	51.8	N/A	1214
Phi 3 14B [1]	✓	14B	4.8T	04/2024	78	81.4	N/A	62.2	N/A	1122
Llama 2 13B [113]	✓	13B	2T	07/2023	53.8	47	6.7	14	8.4	1063
Gemma 2 9B [33]	✓	9B	8T	06/2024	72.3	68.2	36.6	40.2	N/A	1182
Llama 3 8B [4]	✓	8B	15T	04/2024	68.4	61.1	30	62.2	22.9	1152
Qwen 2 7B [89]	✓	7B	N/A	06/2024	70.3	62.6	44.2	51.2	N/A	N/A
Gemma 7B [112]	✓	7B	6T	02/2024	64.3	55.1†	24.3	32.3	10.4	1084
Phi 3 7B [1]	✓	7B	4.8T	04/2024	75.7	79.1	N/A	61	N/A	1101
Mistral 7B [57]	✓	7B	N/A	09/2023	60.1	N/A	13.1	30.5	17.1	1072
Llama 2 7B [113]	✓	7B	2T	07/2023	45.7	38.1	3.8	7.9	5.4	1037
Phi 3 3.8B [1]	✓	3.8B	3.3T	04/2024	68.8	71.7	N/A	58.5	N/A	1066
Gemma 2 2.6B [33]	✓	2.6B	2T	06/2024	51.3	41.9	15	17.7	N/A	N/A
Gemma 2B [112]	✓	2B	6T	02/2024	42.3	35.2†	11.8	22	5.4	1021
Qwen 2 1.5B [89]	✓	1.5B	N/A	06/2024	56.5	37.2	21.7	31.1	N/A	N/A
Qwen 2 0.5B [89]	✓	0.5B	N/A	06/2024	45.4	28.4	10.7	22	N/A	N/A

Tabla 2.2: Comparación de los distintos LLM más destacados en la actualidad, mostrando si tienen los pesos abiertos (*open weights*), el número de parámetros, el número de tókenes de preentrenamiento, la fecha de lanzamiento, y el rendimiento en distintos *benchmarks* de evaluación. Los modelos están divididos según la disponibilidad de los pesos y ordenados por el número de parámetros.

- **MMLU**: 5-shot.
- **BBH**: 3-shot con CoT.
- **MATH**: 4-shot con CoT, excepto para Claude 3.5 que es 0-shot con CoT.
- **HumanEval**: 0-shot.

Es importante tener en consideración que la fecha de corte del conjunto de entrenamiento para todos los modelos listados, es posterior a la fecha de publicación de los conjuntos de datos de los *benchmarks*. Por lo que es posible que el conjunto de test de los *benchmarks* forme parte del conjunto de entrenamiento de los modelos, o que se haya contaminado indirectamente con información del conjunto de test.

Esto no supone un problema para las puntuaciones de AlpacaEval y Chatbot Arena, ya que como se menciona en la sección 2.4.4, estos métodos emplean jueces basados en LLM o humanos, respectivamente, para evaluar las respuestas generadas por los modelos. Sin embargo, esta puntuación puede fluctuar en el futuro y solo es precisa en el momento de la publicación de este trabajo.

Entre todos los modelos, destacan los resultados de la familia de modelos Phi-3, al mostrar un rendimiento muy alto respecto a su tamaño en los *benchmarks* tradicionales. La versión de Phi-3 de 3.8B obtiene resultados superiores a Llama 3 8B en MMLU y BigBench-Hard, un modelo que tiene el doble de parámetros y fuer publicado en el mismo mes. Mientras tanto, en Chatbot Arena, la puntuación Elo de Phi-3 3.8B es 1066, considerablemente peor a la de Llama 3 8B, igual a 1152. Llama 3 8B también supera en Chatbot Arena a la versión de Phi-3 de 14B, que tiene una puntuación Elo de 1122 y supera de forma significativa a Llama 3 8B en los *benchmarks* tradicionales.

Esta discrepancia en los resultados tradicionales y en los de evaluación humana pueden sugerir una contaminación en el conjunto de entrenamiento de la familia Phi-3. Pero sin más información sobre los datos de entrenamiento, es difícil determinar la causa de esta discrepancia. También conviene mencionar que la familia de Phi-3 son los únicos modelos abiertos que no han publicado los pesos de los modelos fundacionales, ofreciendo solo versiones *instruct-tuned* para servir como asistentes de chat, lo que puede limitar su aplicación en otros dominios.

En definitiva, debido a la vertiginosa velocidad con la que avanzan el desarrollo de LLM en la actualidad, es importante tener en cuenta que la información sobre modelos específicos proporcionada en este y otros trabajos puede quedar obsoleta en poco tiempo. Por ello, es importante hacer una revisión actualizada de los LLM disponibles y evaluar de manera contrastada sus capacidades antes de emplear un modelo específico en un proyecto.

2.5. LLM en sistemas de recomendación

Con el creciente interés en los *LLM*, la comunidad científica ha comenzado a explorar su aplicación en diversos campos, incluyendo el de los sistemas de recomendación. La principal ventaja de incorporar LLM en los sistemas de recomendación radica en su capacidad para extraer representaciones de alta calidad de las características textuales y aprovechar el conocimiento del mundo codificado en ellos [66].

A diferencia de los sistemas tradicionales de recomendación, los enfoques basados en LLM pueden capturar información contextual y entender las consultas de los usuarios, las descripciones de los ítems y otros datos textuales de manera más efectiva. Al comprender este contexto, los sistemas de recomendación basados en LLM pueden mejorar la precisión y relevancia de las recomendaciones, lo que lleva a una mayor satisfacción del usuario. Mientras tanto, los LLM también pueden mejorar las habilidades de recomendación respecto al problema de arranque en frío, causado por la falta de interacciones históricas. Estos modelos tienen la capacidad de adaptarse a candidatos nuevos, donde un candidato representa un ítem que podría ser sugerido a un usuario. Esto es posible gracias al amplio conocimiento adquirido durante la fase de preentrenamiento, permitiéndoles ofrecer recomendaciones pertinentes aun sin haber interactuado previamente con ciertos ítems o usuarios [127].

Además, la combinación de modelos generativos con sistemas de recomendación abre la posibilidad de desarrollar aplicaciones innovadoras y más prácticas. Por ejemplo, se puede mejorar la interpretabilidad de las recomendaciones, ya que los sistemas basados en LLM pueden proporcionar explicaciones en lenguaje natural, ayudando a los usuarios a entender los factores que influyen en sus recomendaciones. También, los LLM permiten ofrecer recomendaciones más personalizadas y contextualizadas, mediante *prompts* configurables por el usuario que se pasan a un sistema de recomendación, mejorando la interacción del usuario con el RS y ofreciendo una mayor diversidad de resultados [32].

En la tabla 2.3 se muestra una comparación de diferentes modelos de recomendación basados en LLM que se mencionan en esta sección, destacando el LLM base utilizado, si se ha realizado *fine-tuning* y los dominios en los que se ha probado cada modelo.

Modelo	**LLM base**	**Fine-tuning**	**Dominio**
RecMind [119]	ChatGPT	Ninguno	Comercio, Empresas
Chat-REC [32]	ChatGPT	Ninguno	Películas
LlamaRec [135]	Llama 2 (7B)	QLoRA	Comercio, Películas, Juegos
TALLRec [8]	Llama 2 (7B)	LoRA	Libros, Películas
PALR [131]	Llama 2 (7B)	Completo	Comercio, Películas
P5 [34]	T5 (223M)	Completo	Comercio, Empresas
GPT4Rec [65]	GPT2 (110M)	Completo	Comercio
POD [67]	T5 (60M)	Completo	Comercio

Tabla 2.3: Comparación de diferentes modelos de recomendación basados en LLM

2.5.1. Categorización de los RS basados en LLM

Los trabajos existentes sobre sistemas de recomendación basados en LLM se pueden dividir en tres categorías principales, según el rol que tenga el LLM con respecto al sistema de recomendación [54].

LLM como sistema de recomendación

En este enfoque, el LLM se utiliza directamente como un sistema de recomendación. La secuencia de entrada generalmente consiste en la descripción del perfil del usuario,

la descripción del ítem y la instrucción de la tarea. En la respuesta devuelta se espera obtener una recomendación razonable.[34]

Estos sistemas, como Chat-REC [32], hacen uso de un LLM potente para combinar una interfaz conversacional con un sistema de recomendación. Los resultados devueltos por el LLM pueden ser filtrados y/o modificados antes de ser presentados al usuario, lo que permite asegurar la calidad de las respuestas.

LLM en el sistema de recomendación

En este tipo de propuestas, el LLM se utiliza como un componente dentro de un sistema de recomendación, encargándose tan solo de mejorar una parte del proceso. A continuación se describen algunos de los roles que los LLM pueden desempeñar en el *pipeline* de un sistema de recomendación [71].

- **Mejora de las características de ítems y usuarios**: Los LLM, al tener potentes capacidades de razonamiento y un extenso conocimiento del mundo, pueden ser tratados como una base de conocimiento. De esta forma, pueden proporcionar características adicionales para representar mejor las preferencias de los usuarios y el contenido de los ítems. Por ejemplo, KAR [129] utiliza un LLM para generar el conocimiento relacionado con el razonamiento que justifica las preferencias de los usuarios, además de añadir conocimiento factual sobre los ítems, que se utilizan como características adicionales en un modelo de recomendación tradicional.

- **Generación de instancias**: También se pueden aprovechar los LLM para generar instancias sintéticas que se utilizan para enriquecer el conjunto de datos de entrenamiento y mejorar la calidad del modelo de recomendación. Por ejemplo, GReaT [11] emplea un LLM *fine-tuned* para esta tarea, generando datos tabulares realistas como instancias de entrenamiento adicionales.

- **Codificación de características**: Para mejorar la representación de los ítems, se emplea un LLM como codificador de características para escenarios con abundantes características textuales disponibles. Mientras el contenido de los ítems es generalmente estático, el interés de los usuarios es muy dinámico y evoluciona con el tiempo, por lo que se requiere un modelado secuencial sobre los comportamientos de los usuarios y las preferencias subyacentes. Por ejemplo, U-BERT [86] mejora la representación del usuario codificando las reseñas en una secuencia de vectores densos a través de BERT [25], seguido de modelos neuronales con mecanismos de atención diseñados para modelar los intereses del usuario.

- **Puntuación de ítems candidatos**: En las tareas de puntuación de candidatos, el LLM sirve como una función que estima la puntuación de cada ítem candidato para un usuario objetivo. La lista final de ítems se obtiene ordenando la puntuación calculada entre el usuario objetivo y cada ítem en el conjunto de candidatos. Se puede realizar un filtrado para reducir el número de ítems candidatos, ahorrando así coste computacional.

  Generalmente, un LLM toma como entrada los tókenes de texto discretos del *prompt*, y genera el siguiente token en la secuencia de salida. Sin embargo, la tarea de puntuación de ítems requiere que el modelo realice una puntuación

para una entrada correspondiente a un par usuario-ítem, y la salida debe ser un número real, en lugar de un token discreto, además debe estar dentro de un rango numérico que indique la valoración del usuario. Por lo tanto, es necesario abordar el problema de que la salida requiere valores numéricos continuos, mientras que el LLM produce tókenes discretos. Para ello, se puede estimar la valoración basándose en la distribución de probabilidad generada por el LLM para el siguiente token.

Propuestas como TALLRec [8] o PALR solucionan este problema añadiendo una pregunta binaria (de sí o no) dirigida a evaluar la preferencia del usuario, al final del *prompt*, compuesto por la descripción del perfil del usuario, los comportamientos del usuario y el ítem candidato. De esta forma, convierte la tarea de puntuación de ítems en un problema de respuesta a una pregunta binaria. Posteriormente, se puede obtener la probabilidad asignada a los tókenes correspondientes con "sí" y "no", para asignar una puntuación al ítem.

- **Ordenamiento de ítems**: En esta tarea, el LLM produce como salida la lista final y ordenada de ítems a recomendar. Para evitar el problema de la alucinación, es necesario proporcionar como entrada un conjunto de ítems candidatos de los que seleccionar u ordenar las recomendaciones. Este enfoque depende de la capacidad intrínseca de razonamiento de los LLM para deducir las preferencias de los usuarios y generar una lista ordenada que sea adecuada.

  El LLM tiene que ordenar los candidatos de un conjunto de ítems, por lo que se usa un modelo de recuperación para filtrar el conjunto universal de ítems y así obtener un conjunto con un número limitado de ítems. El número de este conjunto suele ser muy reducido (e.g. 20 ítems), debido a la limitación del tamaño del contexto que presentan los LLM. El contenido de los ítems candidatos se presenta en el *prompt* de entrada junto con la descripción de la tarea para que el LLM genere la lista de ítems ordenada. Por ejemplo, LlamaRec [135], emplea LRURec [134] como el modelo de recuperación de los ítems candidatos, y emplea una versión *fine-tuned* de Llama-2 para obtener una lista ordenada a partir del conjunto filtrado de ítems.

  El uso de un modelo de recuperación permite mitigar el problema de alucinación de los LLM. Sin embargo, la introducción de los ítems candidatos en el *prompt* del LLM puede causar otros problemas. Por un lado, los LLM no pueden manejar un gran número de candidatos debido a la limitación del tamaño del contexto (la cantidad máxima de tókenes que un LLM puede tomar como entrada). Además, el orden de los ítems candidatos en el *prompt* puede afectar a la salida del ranking, lo que lleva a resultados de recomendación inestables.

Hay que tener en cuenta que un LLM puede desempeñar más de uno de los roles mencionados, ya que los LLM pueden aprender a realizar múltiples tareas. Por ejemplo, P5 [34] hace *fine-tuning* de un modelo para que realice una serie de tareas de recomendación empleando diferentes plantillas de *prompting*, entre estas tareas se encuentran tanto la puntuación, como la recomendación secuencial de ítems candidatos. En la figura 2.10 se muestra un esquema de las distintas tareas que puede realizar P5.

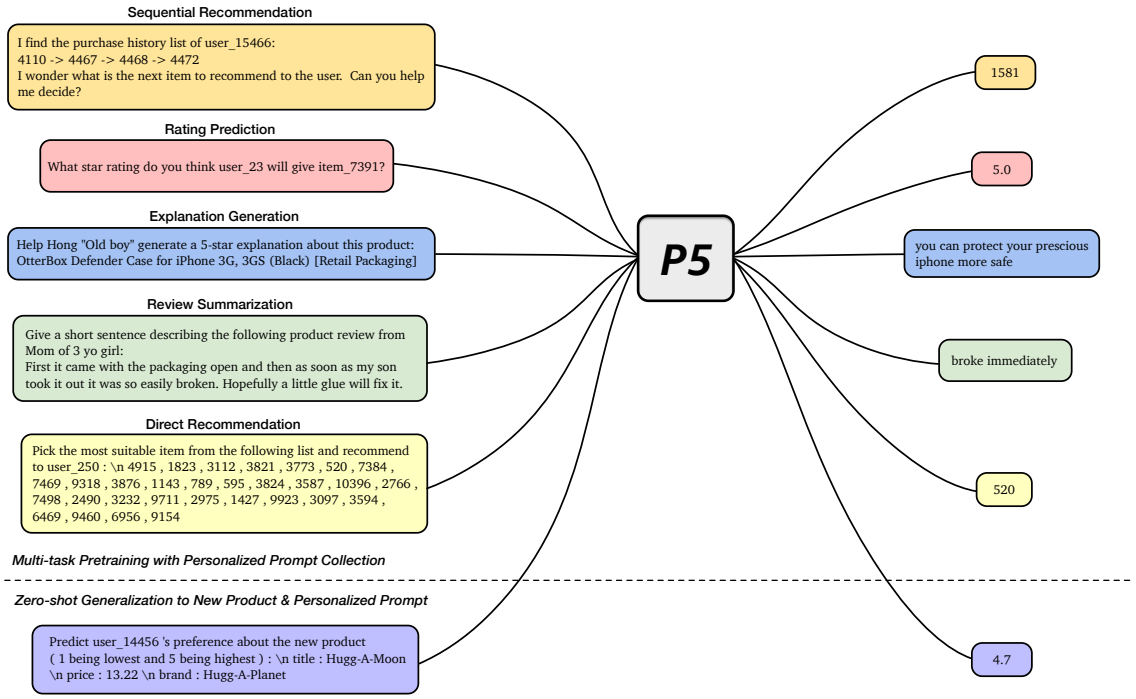

Figura 2.10: Ilustración de las distintas tareas que puede realizar P5. Por Geng et al. [34]

Sistema de recomendación en el LLM

Estos métodos hacen uso de un LLM que llama a distintos agentes, entre los que se encuentra al menos un sistema de recomendación, para obtener recomendaciones. El LLM se encarga de la interacción con el usuario y de presentar las recomendaciones, mientras que el sistema de recomendación se encarga de la generación de las recomendaciones en sí. Este enfoque permite una mayor flexibilidad en la recomendación, ya que se pueden emplear distintos sistemas según las necesidades del usuario.

Un ejemplo de este enfoque es RecMind [119], que propone un sistema que controla el proceso de recomendación dividiendo el proceso en subtareas. Este sistema atiende las consultas de los usuarios mediante una estrategia de planificación en la que hace uso de múltiples herramientas, como modelos expertos, motores de búsqueda y herramientas de SQL, para proporcionar recomendaciones más precisas y personalizadas.

2.5.2. Adaptación de LLM para RS

Para adaptar un LLM y usarlo para tareas relacionadas con sistemas de recomendación existen dos paradigmas principales, según si se realiza *fine-tuning* sobre un LLM o no [127].

Sin *fine-tuning*

Los LLM muestran grandes capacidades en tareas no vistas con anterioridad. Por lo tanto, se puede asumir que los LLM ya tienen la habilidad de realizar recomendaciones y el objetivo es construir *prompts* que activen estas habilidades. Emplean las técnicas descritas en la sección 2.4.3 para adaptar los LLM a tareas de recomendación sin

necesidad de entrenar parámetros adicionales. Destacan sobre todo las técnicas de *prompt engineering*, sobre las de *in-context learning*, que apenas han sido exploradas en este contexto. Algunos ejemplos de este enfoque son RecMind [119] y ChatREC [32], ambos basados en ChatGPT.

Con *fine-tuning*

Como se ha mencionado anteriormente, los LLM tienen buenas capacidades de realizar tareas no vistas anteriormente y su capacidad para proporcionar recomendaciones supera a la de métodos de referencia como la elección aleatoria o más popular. Sin embargo, no es extraño que los sistemas de recomendación construidos de esta manera no superen el rendimiento de los modelos de recomendación entrenados específicamente para una tarea concreta y en datos específicos. Por ello, muchos investigadores buscan mejorar la capacidad de recomendación de los LLM mediante un *fine-tuning* adicional. Estos métodos ya se han comentado en la sección 2.4.2, en este campo destacan sobre todo los métodos de *instruction tuning* y *prompt tuning*. Algunos ejemplos de modelos que han aplicado un *fine-tuning* completo sobre el modelo base son P5 [34], PALR [131], GPT4Rec [65] o POD [67].

2.5.3. Problemas de los RS basados en LLM

Los desafíos en la implementación de LLM en los sistemas de recomendación, surgen principalmente de las características únicas de los sistemas de recomendación en las aplicaciones del mundo real [71].

Coste del entrenamiento

Los métodos principales para mejorar el rendimiento de los sistemas de recomendación modernos basados en aprendizaje profundo son aumentar el tamaño de los datos de entrenamiento, e incrementar la frecuencia con la que se actualiza el modelo. Ambos factores requieren una gran eficiencia en el entrenamiento del modelo para no incurrir en costes computacionales y de tiempo excesivos. Aunque la técnica de *fine-tuning* es efectiva para alinear los LLM con las tareas de los sistemas de recomendación y mejorar su rendimiento, conlleva costes prohibitivos que solo se ven aliviados ligeramente cuando se emplean técnicas de *PEFT*. Por lo tanto, asegurar la eficiencia del entrenamiento al adaptar los LLM a los sistemas de recomendación es uno de los principales desafíos para las aplicaciones del mundo real.

Latencia en la inferencia

Los sistemas de recomendación en línea normalmente son servicios que trabajan en tiempo real y son muy sensibles al tiempo, donde todos los procesos deben ser completados en intervalos de tiempo del orden de decenas de milisegundos. La inferencia de los LLM, por lo tanto, da lugar a un problema de latencia, ya que el tiempo de inferencia de los LLM es considerablemente más alto, sin mencionar el coste adicional de tiempo que supone la generación del *prompt*. Este problema puede aliviarse empleando técnicas de cuantización sobre el LLM, aunque incluso con este método hace falta un *hardware* muy potente para obtener resultados en un tiempo aceptable.

Modelado de secuencias largas de texto

Al adaptar los LLM a las tareas de recomendación, hay que construir *prompts* mediante plantillas e insertar instrucciones y demostraciones adecuadas al principio si es necesario. Sin embargo, para hacer recomendaciones efectivas se requiere de un historial de usuario muy largo, un conjunto de candidatos muy grande y un gran número de características para lograr el mejor rendimiento, lo que puede llevar a *prompts* excesivamente largos para los LLM.

La longitud de estas entradas pueden plantear los siguientes desafíos. Primero, una entrada de texto excesivamente larga causaría un incremento exponencial en el uso de memoria, y puede que llegue a alcanzar el límite de la ventana de contexto del LLM, lo que resultaría en la pérdida de información parcial y en un resultado degradado. Por otra parte, incluso si la longitud de la entrada no excede la ventana de contexto, puede haber problemas para que el LLM comprenda y razone completamente sobre los datos de recomendación, al tener que buscar información relevante en un contexto muy largo. Afortunadamente, los últimos trabajos en el campo de los LLM han mejorado tanto la longitud de la ventana de contexto, como la capacidad de recuperación de la información en contextos muy largos [27].

Modelado de identificadores

En los sistemas de recomendación, existen una serie de características de identificación que no contienen información semántica (por ejemplo, el identificador de un usuario o de un ítem). Si se incluyen estos identificadores en el *prompt*, la tokenización de estos identificadores no tendrá sentido para los LLM, y no podrán aprender nada de ellos. Por lo tanto, muchos trabajos optan por eliminar estos identificadores y reemplazarlos por información semántica, como el título o la descripción de un ítem. Sin embargo, algunos trabajos han demostrado que estos identificadores pueden mejorar significativamente el rendimiento de los sistemas de recomendación, aunque a costa de sacrificar la capacidad de generalización entre distintos dominios [34]. Por lo que aún es una cuestión abierta si se deben incluir estos identificadores en la entrada de los LLM o no.

Sesgos

Cualquier sistema que emplee un LLM puede contener un sesgo indeseado, ya que el corpus de preentrenamiento puede incluir información discriminatoria u ofensiva. Aunque existen métodos para mitigar estos sesgos, como los mencionados en la sección 2.4.2, los trabajos existentes ya han detectado problemas de sesgo en los sistemas de recomendación basados en LLM, tanto desde el punto de vista del usuario como del ítem [55, 136].

Capítulo 3

Metodología

La reproducibilidad de los resultados empíricos es un pilar fundamental de la investigación científica y un requisito para asegurar el progreso continuo en cualquier campo científico. La reproducibilidad no solo verifica los hallazgos originales, sino que también garantiza que estos hallazgos sean confiables y aplicables en diversas circunstancias. En muchos casos, la falta de reproducibilidad puede indicar problemas subyacentes en la metodología, los datos o la interpretación de los resultados, lo que subraya la importancia de realizar estudios que verifiquen y extiendan investigaciones previas.

Por ello, este estudio se enfoca en repetir y analizar un trabajo previo en contextos alternativos, esperando contribuir a un mayor nivel de reproducibilidad en el futuro. En concreto, este trabajo se basa en el análisis y replicación del estudio llevado a cabo por Yue et al. [135], que propone LlamaRec, un sistema de recomendación basado en *LLM*. Sin embargo, este estudio no se limita a una simple repetición del experimento original, sino que introduce variaciones significativas, como la aplicación de la metodología a diferentes conjuntos de datos y la inclusión de nuevos modelos de lenguaje que no fueron considerados en el estudio inicial. Estas adaptaciones permiten una evaluación más amplia y robusta de la metodología de LlamaRec en diversos contextos, aportando así una perspectiva más generalizable sobre su aplicabilidad y eficacia.

A continuación, se presenta un análisis detallado de la metodología empleada por LlamaRec en el contexto de los sistemas de recomendación basados en *LLM*. Las características principales del estudio incluyen la revisión de los resultados reportados originalmente, la implementación del algoritmo en diferentes dominios y la comparación de los resultados empleando distintos LLM. De esta forma se extiende el análisis original mediante la incorporación de resultados adicionales derivados de pruebas en distintos escenarios, buscando identificar fortalezas y limitaciones en la metodología propuesta por LlamaRec.

LlamaRec es un marco de trabajo con dos etapas que hace uso de LLM para ofrecer recomendaciones basadas en el ranking de los ítems. En la primera etapa, se emplean sistemas de recomendación secuencial más livianos para recuperar candidatos basados en el historial de interacción del usuario. En la segunda etapa, tanto el historial del usuario como los ítems candidatos recuperados se pasan al LLM en formato de texto

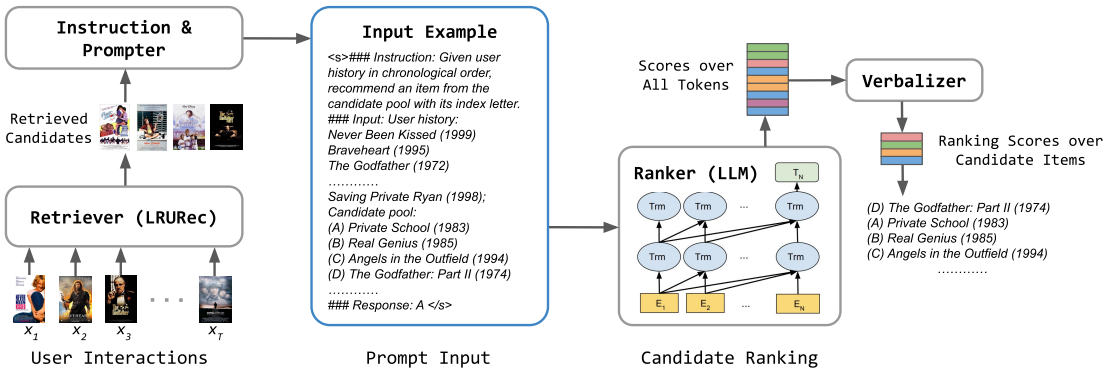

Figura 3.1: Arquitectura de LlamaRec. Por Yue et al. [135]

mediante una plantilla del *prompt*. Posteriormente, en vez de generarse los nombres de los ítems a recomendar, se fuerza al modelo a solo generar distribuciones de probabilidad sobre los candidatos. De esta forma, se pueden clasificar eficientemente los ítems sin necesidad de generar múltiples tókenes y evitando las alucinaciones. En la figura 3.1 se muestra un esquema de la arquitectura de LlamaRec [135].

Este método se basa en la recomendación secuencial, donde el historial de interacción del usuario se utiliza como entrada del sistema. Este historial está compuesto por una secuencia de ítems en orden cronológico con los que ha interactuado previamente. En la etapa de recuperación (parte izquierda de la figura 3.1), estos ítems se representan mediante identificadores únicos debido al gran volumen de datos presentes al comienzo. Mientras que en la etapa de ordenación (parte central y derecha de la figura 3.1), se utilizan los títulos de los ítems para aportar información semántica que se corresponda con el conocimiento derivado del preentrenamiento del LLM.

Debido al gran volumen de ítems candidatos en los conjuntos de datos de recomendación, normalmente del orden de millones, es común utilizar un enfoque similar al que propone LlamaRec, donde un modelo más rápido recupera los ítems candidatos a ordenar por un modelo más potente, esta arquitectura se ha utilizado, por ejemplo, para recomendar vídeos en YouTube [22].

De manera más formal, se puede describir a LlamaRec como un sistema de recomendación que toma como entrada el historial de interacciones de un usuario x de un conjunto de datos $\mathcal{X}$. Este historial es una secuencia de ítems en orden cronológico $[x_1, x_2, \ldots, x_T]$ con los que el usuario ha interaccionado, donde cada ítem se define en el espacio de ítems $\mathcal{I}$ ($x_i \in \mathcal{I}, i = 1, 2, \ldots, T$). La salida del sistema es la lista de puntuaciones del *ranking* $\hat{y} \in \mathbb{R}^{|\mathcal{I}|}$, con el ítem realmente más relevante para el usuario denotado por $y \in \mathcal{I}$. El modelo completo se denota con f parametrizado por θ (es decir, $\hat{y} = f(\theta; x)$), y está formado por un modelo de recuperación (*retriever*) $f_{\text{retriever}}$ y un modelo de ordenación (*ranker*) basado en un LLM f_{ranker} ($f = f_{\text{ranker}} \circ f_{\text{retriever}}$). Idealmente, el ítem con la puntuación más alta en $\hat{y}$ debería ser el ítem realmente más relevante y (es decir, $y = \arg\max \hat{y}$). Por lo tanto, el objetivo del sistema es maximizar la puntuación de dicho ítem. Es decir, se busca minimizar la pérdida $\mathcal{L}$ con respecto a los parámetros θ sobre $\mathcal{X}$:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{X}}[\mathcal{L}(f(\theta; x), y)] \tag{3.1}$$

El modelo de recuperación (*retriever*) particular que emplea LlamaRec es LRURec [134], propuesto por los mismos autores, en la figura 3.2 se muestra un esquema de su arquitectura. Este es un sistema de recomendación secuencial basado en redes neuronales recurrentes lineales. Implementa por primera vez la unidad recurrente lineal (LRU), propuesta por Orvieto et al. [81], en un sistema de recomendación, con el objetivo de mejorar la eficiencia y rendimiento sobre modelos anteriores. De esta forma, LRURec se encarga de recuperar los top-20 ítems por cada secuencia de entrada, independientemente de su longitud. El uso de un *retriever* diseñado por los mismos autores del método principal puede plantear dudas sobre la idoneidad de su elección y su rendimiento comparado con otros sistemas de recomendación secuencial más populares. Este trabajo pretende reproducir los resultados obtenidos por los autores tanto en los métodos propuestos como en los que sirven de referencia.

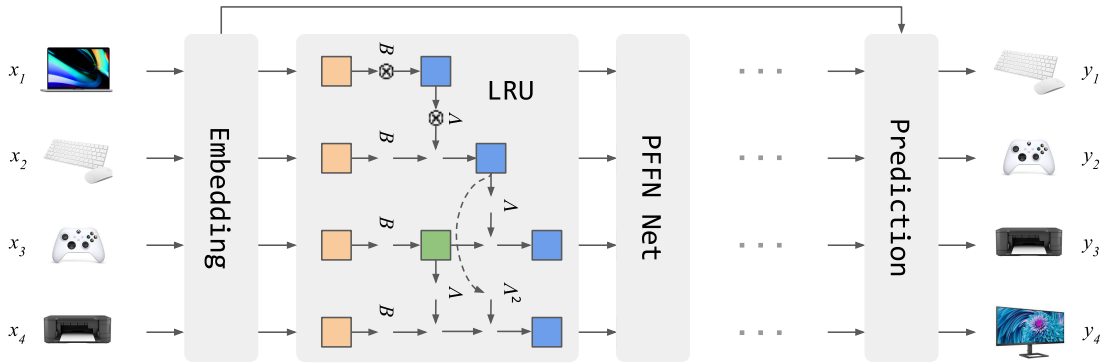

Figura 3.2: Arquitectura de LRURec. Por Yue et al. [134]

LlamaRec emplea Llama 2 7B como modelo base para la etapa de ordenación (*ranker*). En esta etapa se construye un *prompt* que contiene una descripción de la tarea, el historial del usuario y los ítems candidatos, representados por sus nombres. Para limitar la salida del LLM para tan solo incluir la puntuación de los ítems candidatos se utiliza un verbalizador, que proyecta los logits de las etiquetas originales a otro conjunto de etiquetas [26], definido por letras índice (A, B, C...) que identifican a los ítems candidatos.

En la implementación de LlamaRec, se emplea *fine tuning* para ajustar el comportamiento del LLM a la tarea de recomendación. Para ello se utiliza la plantilla de *prompt* definida anteriormente y se calcula la pérdida solo para los tókenes correspondientes al conjunto de etiquetas del verbalizador. Para poder llevar a cabo un entrenamiento eficiente, se utilizan métodos de *PEFT*, concretamente QLoRa, haciendo posible el entrenamiento del modelo base en tarjetas gráficas de consumo general, como en una RTX 3090 o 4090, ambas con 24GB de VRAM.

El avance explosivo de los LLM, comentado en el capítulo del estado del arte, lleva a plantearse si el uso de otros modelos más modernos y con mejores resultados en *benchmarks* tradicionales que Llama 2 7B, puede mejorar los resultados de LlamaRec. En este trabajo se comparará el rendimiento del sistema original usando como *ranker* otros LLM de distintos tamaños, para evaluar si el uso de estos modelos puede mejorar los resultados originales.

Para evaluar el sistema original se emplearon los conjuntos de datos MovieLens-100K [40], Amazon Beauty [77] y Amazon Games [77]. Estos conjuntos de datos se

han procesado construyendo secuencias de entradas en orden cronológico y filtrando usuarios e ítems con menos de 5 interacciones, así como ítems sin metadatos asociados a ellos. En la tabla 3.1 se muestran las estadísticas de los conjuntos de datos después de ser procesados. Estas estadísticas, que fueron reportadas por los autores en su trabajo original y han sido validadas en este estudio, muestran el número de usuarios, ítems, interacciones, longitud media de las secuencias y densidad de los conjuntos de datos.

Datasets	**Usuarios**	**Ítems**	**Interac.**	$\overline{\textbf{Long.}}$	**Densidad**
ML-100k	610	3650	100K	147.99	4e-2
Beauty	22332	12086	198K	8.87	7e-4
Games	15264	7676	148K	9.69	1e-3

Tabla 3.1: Estadísticas de los conjuntos de datos después de ser procesados.

Se puede observar que los conjuntos de datos escogidos son de un tamaño relativamente pequeño, siendo Beauty el mayor con casi 200K interacciones. Mientras tanto, los conjuntos de datos más comunes para problemas de recomendación secuencial suelen tener millones de interacciones, como es el caso de MovieLens-1M [40], una iteración de ML-100K mucho más utilizada en la literatura actual. Este hecho puede plantear dudas sobre la capacidad o eficiencia de este método en conjuntos de datos más grandes. Además, todos los conjuntos contienen secuencias basadas en el historial completo de los usuarios, en vez de estar basados en sesiones de uso. Por lo tanto, se planteará la evaluación de LlamaRec sobre un conjunto de datos diferente a los planteados en el trabajo original.

Capítulo 4

Caso de estudio

En este capítulo se plantea el estudio de reproducibilidad del método de LlamaRec [135], evaluándolo en diversos dominios y comparando el método base usando otros *LLM* punteros de distintos tamaños. Este caso de estudio pretende obtener resultados que respondan a las siguientes preguntas de investigación (*RQ*):

I **RQ1**: ¿Son reproducibles los resultados del artículo original de LlamaRec?

II **RQ2**: ¿Cómo funciona LlamaRec en otros dominios de recomendación?

III **RQ3**: ¿Se pueden utilizar otros LLM del estado del arte para mejorar los resultados de LlamaRec?

4.1. Descripción del caso de estudio

En este caso de estudio, se busca reproducir y extender los resultados obtenidos por el método de LlamaRec, evaluándolo en diferentes dominios y utilizando otros LLM pertenecientes al estado del arte y de diversos tamaños. El objetivo principal es validar la eficacia y versatilidad del enfoque en dos fases de LlamaRec en distintos contextos y explorar la potencial mejora en su rendimiento mediante la incorporación de otros modelos de lenguaje. Para ello, el estudio se estructura en distintas fases.

Primero, se reproducen los resultados obtenidos por LlamaRec en el artículo original, utilizando los mismos conjuntos de datos y configuraciones experimentales. Se comparan los resultados obtenidos con los sistemas de referencia propuestos por los autores y se evalúa la eficacia del método en la tarea de recomendación secuencial.

Posteriormente, se evalúa el rendimiento de LlamaRec en otros dominios de recomendación, utilizando conjuntos de datos diferentes a los empleados en el trabajo original. Se busca comprobar si el método es efectivo en distintos contextos y si es capaz de generalizar su rendimiento a otros dominios.

Finalmente, se evalúa el rendimiento de LlamaRec utilizando otros LLM del estado del arte, con el objetivo de explorar si el uso de modelos más modernos y con mejores resultados en *benchmarks* tradicionales puede mejorar los resultados del método original. Se comparan los resultados obtenidos con los distintos modelos de lenguaje

y se analiza la influencia del tamaño y la capacidad de los modelos en el rendimiento de LlamaRec.

4.2. Reproducción de resultados

La primera fase del caso de estudio se centra en la reproducción de los resultados reportados en el artículo original de LlamaRec, utilizando los mismos conjuntos de datos y configuraciones experimentales descritos por los autores. Este paso es fundamental para validar la efectividad del método propuesto y establecer una línea base confiable para las comparaciones subsecuentes.

Los conjuntos de datos utilizados ya se han mencionado en el capítulo 3, y sus estadísticas tras filtrar los usuarios e ítems con menos de 5 interacciones, e ítems sin metadatos asociados, se encuentran en la tabla 3.1. Concretamente son:

- **MovieLens-100K** [40]: un conjunto de datos de recomendación de películas que contiene aproximadamente 100K valoraciones de películas. Estos datos fueron recogidos a partir de 610 usuarios entre el 29 de marzo de 1996 y el 24 de septiembre de 2018. Habiendo sido seleccionados al azar para su inclusión, todos los usuarios habían valorado al menos 20 películas. Cada usuario está representado por un identificador, y no se proporciona ningún otro tipo de información sobre ellos.
- **Amazon Beauty** [77]: este conjunto de datos forma parte de un conjunto mayor de *datasets* de *reviews* de productos de Amazon [1]. Estos conjuntos de datos contienen valoraciones del 1 al 5 y metadatos de productos de Amazon que abarcan desde mayo de 1996 hasta julio de 2014. En concreto, el conjunto de la categoría Beauty incluye 198K valoraciones de productos de belleza.
- **Amazon Games** [77]: este conjunto también forma parte de los conjuntos de datos de Amazon mencionado anteriormente. En este caso, contiene 231K valoraciones de videojuegos vendidos en Amazon.

Para tener un punto de referencia en la evaluación de los resultados, los autores de LlamaRec adoptan distintos sistemas de recomendación secuencial que pueden ser considerados estado del arte, estos están basados en redes neuronales recurrentes y en *transformers*. Los resultados reportados en estos casos base también tendrán que ser validados en el estudio. En concreto, se emplean los siguientes modelos:

- **NARM**: NARM es un modelo basado en *RNN* propuesto por Li et al. [64] en 2017. Emplea un codificador local y global para realizar la recomendación secuencial.
- **SASRec**: Este modelo fue propuesto por Kang y McAuley [58] en 2018. Adopta un mecanismo de atención unidireccional para procesar la secuencia de entrada y generar los siguientes ítems.
- **BERT4Rec**: Bert4Rec, propuesto por Sun et al. [108] en 2019, es un modelo con un mecanismo de atención bidireccional entrenado mediante la predicción de ítems enmascarados.

[1] `https://cseweb.ucsd.edu/~jmcauley/datasets/amazon/links.html`

- **LRURec**: LRURec fue propuesto por los mismos autores de LlamaRec [134] en 2023. Está basado en redes neuronales recurrentes lineales y pretende mejorar la eficiencia y rendimiento de la recomendación sobre modelos anteriores mediante la utilización de la unidad recurrente lineal (LRU). Este modelo se emplea como *retriever* en LlamaRec.

En la partición de los conjuntos de datos, se sigue la estrategia *leave-one-out*, de tal forma que para cada usuario se toma su historial íntegro de interacciones ordenado de forma temporal, se reserva la última interacción para el conjunto de test y la penúltima para el conjunto de validación. El resto de interacciones se utilizan para el conjunto de entrenamiento, en la figura 4.1 se muestra un esquema de este método de partición de los datos. Este método es común en problemas de recomendación secuencial, ya que permite evaluar el rendimiento del sistema en la predicción de los siguientes ítems en la secuencia de interacciones de los usuarios.

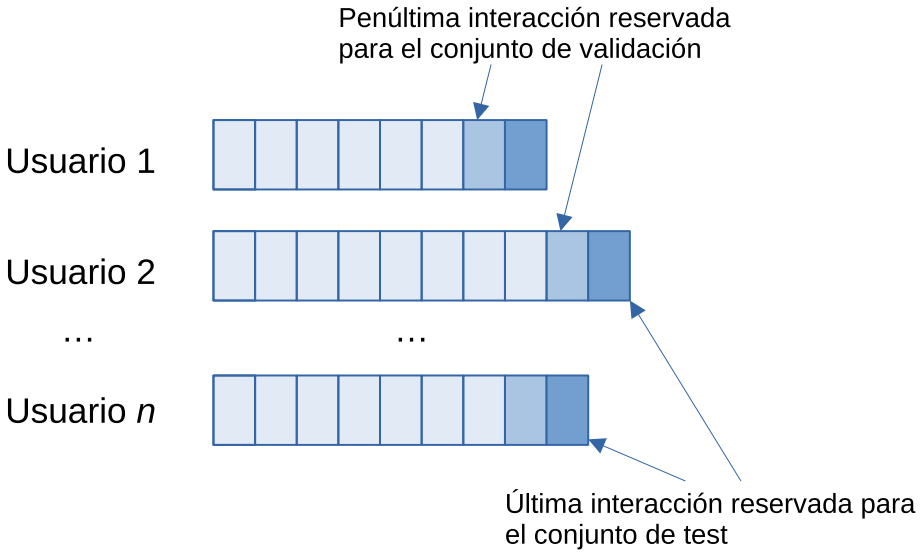

Figura 4.1: Partición de los conjuntos de datos en el caso de estudio.

Para evaluar el rendimiento de los distintos sistemas de recomendación, se emplean las métricas de *MRR@k*, *NDCG@k* y *Recall@k* con $k \in \{5, 10\}$, donde k es el número de ítems recomendados que se tienen en cuenta en la evaluación.

Estas métricas, explicadas en detalle en la sección 2.3.2, son comunes en problemas de recomendación y permiten evaluar la calidad de las recomendaciones generadas por los sistemas. En concreto, *MRR* mide la calidad de la recomendación en función de la posición del primer ítem relevante en la lista de recomendaciones, *NDCG* considera la relevancia de los ítems en función de su posición y *Recall* mide la proporción de ítems relevantes que han sido recomendados.

Para el *retriever* la métrica más relevante será la de *Recall*, ya que se busca recuperar la mayor cantidad de ítems relevante para su posterior ordenación. Por lo tanto, se guardará el modelo con el mejor *Recall@10* para su uso posterior junto con el *ranker*. Para el *ranker* se seleccionará el modelo con el mejor *NDCG@10*, ya que se busca ordenar los ítems de forma que los más relevantes aparezcan en las primeras posiciones.

Los modelos de referencia y el *retriever*, LRURec, se entrenan con los hiperparámetros

que aparecen en la tabla 4.1. Hay que tener en cuenta que los valores específicos usados para el *weight decay* y el *dropout rate* no se reportan en el artículo original, sino que se determinan mediante una búsqueda en rejilla sobre los valores indicados en la tabla[2]. También hay que indicar que la longitud máxima de las secuencias es de 200 para MovieLens-100k y de 50 para los otros conjuntos de datos, subdividiendo aquellas que excedan estos valores. Durante el entrenamiento, se toman las secuencias del conjunto de entrenamiento y se alimentan al modelo de *retrieval*.

Hiperparámetro	**Valor**
Optimizador	AdamW
Learning rate	0.001
Épocas máximas	500
Steps de validación	500
Early stopping	20
Weight decay	[0, 1e-2]
Dropout rate	[0.1, 0.2, 0.3, 0.4, 0.5]

Tabla 4.1: Hiperparámetros de los modelos de referencia y LRURec.

Mientras tanto, el *fine-tuning* del LLM base que servirá como *ranker* se realiza con los hiperparámetros que aparecen en la tabla 4.2. Para el entrenamiento se usan como máximo 20 ítems del historial de interacciones del usuario y se ordenan los top-20 ítems candidatos devueltos por el *retriever*. La longitud de los títulos de los ítems se trunca si excede los 32 tókenes.

El método empleado para hacer el *fine-tuning* es QLoRA con una cuantización de 4-bits, lo que permite entrenar el modelo en tarjetas gráficas de consumo general y reducir el consumo energético, al no tener que gastar horas cómputo en el entrenamiento de todos los pesos del modelo. Durante el entrenamiento, las secuencias del conjunto de entrenamiento se dividen en subsecuencias crecientes, empezando desde una longitud mínima de dos interacciones hasta la longitud completa del historial del usuario. Estas subsecuencias se alimentan al LLM, que se entrena para predecir el último ítem de dichas secuencias.

Hiperparámetro	**Valor**
Learning rate	1e-4
Épocas máximas	1
Steps de validación	100
Dimensión de LoRA	8
LoRA α	32
Dropout rate	0.05
Módulos objetivo	Q y V

Tabla 4.2: Hiperparámetros del *ranker* de LlamaRec.

En el artículo original, se compara el rendimiento de LlamaRec con otros métodos basados en LLM, como P5 [34], PALR [131], GPT4Rec [65], RecMind [119] y POD [67].

[2] `https://github.com/Yueeeeeeee/LlamaRec/issues/1#issuecomment-1949472178`

Sin embargo, para realizar esta comparación, tan solo se han tomado los datos de los resultados reportados por los autores de estos métodos sobre el conjunto de datos Beauty, compartido por todos ellos. Esto se debe a que en algunos casos el código de la implementación no está disponible; sin embargo, las comparaciones realizadas de esta forma pueden arrojar resultados imprecisos, ya que no se han seguido los mismos procedimientos experimentales. Por ejemplo, GPT4Rec usa como LLM base GPT-2 [90], un modelo publicado en 2019 y considerablemente inferior al modelo Llama 2 7B, que es el que se emplea en LlamaRec.

4.3. Selección del conjunto de datos

El dominio elegido para la evaluación de LlamaRec en otros contextos es el de la música, ya que no se ha evaluado en el trabajo original y es un dominio común en problemas de recomendación secuencial. Los sistemas de recomendación de música surgen como respuesta a la nueva forma de consumir música en la era digital, donde los usuarios tienen acceso a un catálogo de millones de canciones y artistas a través de plataformas de *streaming* como Spotify, Apple Music o YouTube Music [103]. En este contexto, los sistemas de recomendación secuencial juegan un papel fundamental en tareas como la recomendación de la siguiente canción o la continuación automática de una lista de reproducción.

Es importante resaltar que el dominio de la música presenta particularidades que lo diferencian significativamente de otros dominios tratados en el trabajo de LlamaRec. Una de las principales diferencias es la repetitividad del consumo de los ítems, puesto que los usuarios tienden a escuchar la misma canción muchas veces a lo largo del tiempo. Mientras tanto, en dominios como la compra de productos o el cine, los ítems suelen consumirse una sola vez o con una frecuencia mucho menor. Además, la secuencia de consumo de los ítems en la música ocurre en intervalos de tiempo muy pequeños, siendo normal que los usuarios escuchen varias canciones en una sesión continua en el tiempo. En contraste, el intervalo entre consumos de ítems en otros dominios suele ser mucho mayor. Estas diferencias en los patrones de consumo pueden tener una influencia notable en los sistemas de recomendación, ya que los modelos deben adaptarse a las características específicas de cada dominio para ser efectivos.

Además de poner a prueba el método de LlamaRec en un dominio diferente, se busca evaluar su capacidad para obtener buenos resultados en un conjunto de datos de mayor tamaño, con al menos 1M de interacciones. En este caso de estudio se empleará un enfoque basado en sesiones, para lo que se necesitará un conjunto de datos que contenga información sobre las sesiones de los usuarios, es decir, las secuencias de interacciones que realizan en un periodo de tiempo determinado. Si el conjunto de datos no contiene la información de las sesiones, estas se pueden obtener si las interacciones del conjunto contienen una marca temporal que permitan saber cuando se han producido.

Por lo tanto, los requisitos del conjunto de datos para este caso de estudio son los siguientes: (1) que pertenezca al dominio de la música, (2) que tenga al menos 1M de interacciones, (3) que contenga información sobre las sesiones de los usuarios **o** que se pueda obtener a partir de las marcas temporales de las interacciones.

Tras realizar una revisión de la literatura, se han encontrado cinco conjuntos de datos que cumplen con los requisitos mencionados, estos se han descrito en la tabla 4.3. De estos *datasets*, tres de ellos ya no están disponibles para su descarga (a fecha de la publicación de este trabajo) por problemas con la licencia de los datos u otros motivos, estos conjuntos están marcados con una cruz en la tabla.

Dataset	**Ítems**	**Usuarios**	**Sesiones**	**Interacciones**
LastFM-1K [15]	960K	1K	N/A	19M
30Music [114]	5.6M	40K	2.7M	31M
LFM-1B† [101]	32M	120K	N/A	1B
LFM-2B† [102]	50M	120K	N/A	2B
MSSD† [12]	3.7M	N/A	160M	3B

Tabla 4.3: Conjuntos de datos de música que cumplen los requisitos propuestos.

Debido a los recursos limitados y al tiempo disponible para la realización de este trabajo, se ha decidido emplear como base el conjunto de datos LastFM-1K, ya que es el más pequeño de los conjuntos de datos disponibles y permitirá realizar los experimentos de forma más rápida. Además, este conjunto de datos ha sido ampliamente utilizado en la literatura y se encuentra disponible para su descarga en la página web del autor[3].

LastFM-1K contiene información sobre los eventos de escucha de alrededor de 1000 usuarios de LastFM. Cada evento contiene el identificador del usuario, de la canción y del artista, junto con una marca temporal. De esta forma se representan los hábitos de escucha de los usuarios hasta el 5 de mayo de 2009.

Como el conjunto de LastFM-1K no incluye de forma explícita la información sobre las sesiones de escucha de los usuarios, será necesario utilizar la marca temporal de las interacciones para construirlas. En este caso, se considerará que una sesión termina cuando ha habido un lapso de tiempo superior a los 30 minutos desde la última interacción del usuario. Teniendo en cuenta que la duración media de una canción es de aproximadamente 3 minutos, un tiempo de 30 minutos entre interacciones se considera como un periodo de inactividad suficiente para considerar que una sesión ha finalizado.

Para obtener este conjunto de datos con las sesiones, primero se ordena el conjunto de interacciones por el identificador del usuario y por la marca temporal de las interacciones. Posteriormente, se construyen las sesiones de escucha de los usuarios siguiendo el criterio mencionado. De esta forma, se obtendrán las secuencias de interacciones de los usuarios que se utilizarán para entrenar y evaluar los sistemas de recomendación[4]. Además de la información de las sesiones, este conjunto contiene los metadatos de las canciones derivados de la API de Spotify.

En la tabla 4.4 se muestran las estadísticas del conjunto de datos LastFM-1K con sesiones después de ser procesado. También se ha realizado muestreos aleatorios del 10 % y del 1 % de las sesiones para comprobar el efecto del tamaño del conjunto de

[3] `http://ocelma.net/MusicRecommendationDataset/lastfm-1K.html`

[4] LastFM-1K con sesiones está disponible en: `https://archive.org/details/lastfm1k`

datos en el rendimiento de LlamaRec. Como referencia, el conjunto con el 1 % de las sesiones contiene un número de interacciones similar al de los conjuntos de datos usados en la publicación original.

Dataset	**Ítems**	**Usuar.**	**Ses.**	**Interac.**	$\overline{\textbf{Long.}}$	**Densidad**
LFM-1K	747K	992	920K	12.4M	15.68	1.81e-5
LFM-1K (10 %)	308K	969	92K	824K	15.55	2.91e-5
LFM-1K (1 %)	76K	861	9.2K	116K	15.60	1.67e-4

Tabla 4.4: Estadísticas del conjunto de datos LastFM-1K después de ser procesado

4.4. Selección de los LLM base

Para evaluar el impacto del modelo de lenguaje en el rendimiento de LlamaRec, es necesario realizar experimentos con distintos LLM entrenados siguiendo el mismo procedimiento que el modelo original. En este caso de estudio, se plantea la evaluación de distintos LLM que actualmente se encuentran en el estado del arte y que varían en tamaño y capacidad.

Debido a los recursos limitados y al tiempo disponible para la realización de este trabajo, los LLM seleccionados deben cumplir ciertos requisitos para poder ser entrenados y evaluados de forma eficiente. El requisito más importante a tener en cuenta es la disponibilidad y el tamaño del modelo, ya que es necesario tener acceso a los pesos del modelo preentrenado y que estos sean de un tamaño que permita su entrenamiento con los recursos disponibles.

Se ha contado con una tarjeta gráfica con 24GB de memoria desde el inicio del proyecto, por lo que se han de seleccionar modelos que no superen este límite de tamaño. Teniendo en cuenta que actualmente la mayoría de LLM se distribuyen con pesos de una precisión de 16 bits (lo que equivale a 2 bytes por parámetro), se puede asumir que cualquier modelo que supere los 12B de parámetros no podrá ser cargado en la memoria de la tarjeta gráfica para su posterior entrenamiento.

Los modelos a elegir, también deben estar soportados por la biblioteca de HuggingFace Transformers [124], al ser la biblioteca empleada en el trabajo original para el entrenamiento y evaluación del modelo. Afortunadamente, la mayoría de los modelos de lenguaje pertenecientes al estado del arte están disponibles en esta biblioteca.

Teniendo estos requisitos en cuenta, y basándose en la revisión de la literatura realizada en el capítulo 2, se han seleccionado los siguientes modelos de lenguaje para su evaluación en el caso de estudio:

- **Llama 3 8B**: Este modelo ha sido publicado por Meta [4] en abril de 2024, y es el sucesor del modelo Llama 2 7B que se emplea en el trabajo original. Al tener un tamaño ligeramente mayor al modelo original y ser considerablemente más moderno, se espera que pueda mejorar los resultados obtenidos por LlamaRec.
- **Mistral 7B**: Mistral es un modelo de lenguaje publicado por la empresa francesa Mistral AI [57] en septiembre de 2023. Este modelo tiene el mismo

tamaño que Llama 2 7B, pero obtiene mejores resultados tanto en *benchmarks* tradicionales como en evaluaciones humanas.

- **Phi 3 3.8B**: La familia de modelos de Phi 3 fue publicada por Microsoft [1] en abril de 2024. Se ha seleccionado el modelo más pequeño de la familia, de 3.8B de parámetros, para comprobar como se comporta un modelo de menor tamaño para este caso de uso. Hay que tener en cuenta que, a diferencia del resto de modelos, no se distribuye una versión fundacional del modelo, tan solo una versión *instruction tuned* para tareas conversacionales, lo que puede afectar a su rendimiento en tareas de recomendación.
- **Gemma 2B**: Esta familia de modelos ha sido publicada por Google [112] en febrero de 2024. Esta versión de 2B de parámetros será el modelo más pequeño que se evaluará en este caso de estudio. A pesar de su tamaño, en su evaluación obtiene resultados competitivos con Llama 2 7B, siendo este último más de 3 veces mayor en tamaño.

Todos estos modelos, excepto Mistral 7B, han sido publicados durante el propio desarrollo del trabajo, lo que evidencia la rápida evolución de los modelos de lenguaje en la actualidad. Incluso mientras se redactan estas líneas, ya se han publicado modelos más nuevos y que prometen mejorar los resultados obtenidos por algunos de los modelos seleccionados, como por ejemplo Gemma 2 [33], la iteración más reciente de la familia Gemma. Esto subraya el dinamismo de la investigación en este campo.

4.5. Descripción de los experimentos

Para lograr responder a las preguntas de investigación planteadas, se han diseñado una serie de experimentos que permitirán evaluar el rendimiento de LlamaRec en distintos dominios y con distintos modelos de lenguaje. Estos experimentos se han estructurado en tres fases, cada una de las cuales se centra en una de las preguntas de investigación planteadas.

4.5.1. Fase 1: Evaluación de los resultados originales

En esta fase se busca reproducir los resultados obtenidos por LlamaRec en el artículo original, utilizando los conjuntos de datos y configuraciones experimentales descritos por los autores. El objetivo es validar la efectividad del método propuesto y establecer una línea base confiable para las comparaciones subsecuentes.

Para ello, se emplea el código proporcionado por los autores de LlamaRec[5] y se intenta seguir los mismos procedimientos experimentales descritos en su publicación. Sin embargo, este código presenta algunas deficiencias que complican su completa reproducción, como la ausencia del código correspondiente a los modelos de referencia (*baselines*) y de un fichero de configuración con un versionado correcto de las dependencias del proyecto.

En otro de los repositorios de los autores[6], sí que aparece reflejado el código de los mismos modelos de referencia que se emplean en el artículo original, por lo que se

[5] `https://github.com/Yueeeeeeee/LlamaRec`
[6] `https://github.com/Yueeeeeeee/RecSys-Extraction-Attack`

ha decidido emplear este código para la evaluación de los modelos de referencia. Sin embargo, es importante mencionar que estas implementaciones no son las originales desarrolladas en sus respectivos trabajos. Estas implementaciones realizadas por terceros de otros sistemas de recomendación pueden tener efectos indeseados, derivando en resultados inferiores a los esperados, como han demostrado trabajos previos [46, 85].

Para solventar el problema relacionado con las dependencias del proyecto, se ha decidido crear un entorno virtual con Mamba[7] donde se han instalado las últimas versiones de las bibliotecas que se emplean a lo largo del código fuente. Esto no debería resultar en cambios significativos en los resultados, ya que el trabajo original es relativamente reciente y las bibliotecas involucradas no han sufrido cambios significativos desde entonces.

Tras aplicar estos cambios, se puede proceder a replicar los experimentos realizados en el trabajo original. Para ello, primero se entrenan los modelos de referencia y el *retriever* LRURec realizando una búsqueda en rejilla sobre los hiperparámetros de *dropout rate* y *weight decay*. Los valores finales de estos hiperparámetros se corresponderán con los que obtengan los mejores resultados en la métrica de *Recall@10*. Este entrenamiento se realiza sobre los tres conjuntos de datos mencionados en el trabajo original, MovieLens-100K, Amazon Beauty y Amazon Games.

Una vez entrenados los modelos de referencia y LRURec, se puede proceder a entrenar el *ranker* de LlamaRec con el modelo Llama 2 7B, usando los candidatos devueltos por LRURec en el paso anterior. El *fine-tuning* del *ranker* se realiza con los hiperparámetros configurados por defecto en el código original, y se entrena sobre los tres conjuntos de datos mencionados anteriormente.

Después de finalizar con esta fase, se espera obtener un conjunto de resultados que se desvíen mínimamente de los reportados en el trabajo original, reproduciendo de forma satisfactoria los resultados obtenidos por LlamaRec.

4.5.2. Fase 2: Evaluación en otros dominios

En esta fase se evalúa el rendimiento de LlamaRec en el dominio de la música, utilizando el conjunto de datos LastFM-1K con sesiones. El objetivo es comprobar si el método es efectivo en distintos contextos y si es capaz de generalizar su rendimiento a otros dominios de recomendación.

Para ello, hay que replicar el código de procesamiento de los conjuntos de datos ya usados por LlamaRec en el nuevo conjunto de datos. Esto supone implementar la descarga del conjunto total y el procesamiento de los datos correspondientes con las interacciones, por una parte, y de los metadatos de las canciones por otra. También se ha de implementar el método de muestreo aleatorio de las sesiones, para poder evaluar el efecto del tamaño del conjunto de datos en el rendimiento del sistema.

Una vez procesado el conjunto de datos, se puede proceder a entrenar los modelos de referencia y el *retriever* LRURec siguiendo el mismo procedimiento que en la fase anterior. Debido al tamaño de este conjunto de datos y el tiempo necesario para realizar el entrenamiento de los modelos, se ha decidido modificar los hiperparámetros

[7] `https://github.com/mamba-org/mamba`

para reducir este tiempo de entrenamiento. En concreto, se ha reducido la frecuencia de validación a una vez por época y el early stopping se ha fijado en 5 épocas. Además, en lugar de hacer una búsqueda en rejilla sobre los hiperparámetros de *dropout rate* y *weight decay*, se han fijado los valores indicados en la publicación de LRURec [134] para el conjunto de datos MovieLens-1M, que son 0.2 y 1e-2 respectivamente.

Mientras tanto, para el entrenamiento del LLM, se modificarán los hiperparámetros de tal forma que se puedan obtener resultados en un tiempo razonable. Esto puede suponer una reducción tanto en los pasos de entrenamiento, como en la frecuencia de validación y en el *early stopping*. Para evitar en la medida de lo posible el impacto de estos cambios en los resultados, se intentará entrenar el modelo durante el máximo número de pasos que permita el tiempo disponible.

En esta fase se esperan que LlamaRec demuestre un buen rendimiento en el dominio de la música, superando a los métodos de referencia con un margen comparable al observado en los conjuntos de datos iniciales

4.5.3. Fase 3: Evaluación con otros LLM

En esta fase se pretende evaluar el rendimiento de LlamaRec utilizando otros LLM del estado del arte, con el objetivo de explorar si el uso de modelos más modernos y con mejores resultados en *benchmarks* tradicionales puede mejorar los resultados del método original. De obtenerse esta mejora, se podría concluir que el rendimiento del sistema está directamente relacionado con la capacidad del LLM y puede tener margen de mejora a medida que aparezcan nuevos modelos. Para probar esta hipótesis, se emplearán los modelos seleccionados en la sección anterior, Llama 3 8B, Mistral 7B, Phi 3 3.8B y Gemma 2B.

El procedimiento a seguir para incorporar un nuevo tipo de LLM en LlamaRec es relativamente simple, tan solo hay que modificar la función de *forward* de la implementación del modelo para que la función de pérdida solo se ejecute si el modelo está entrenándose[8]. Si no se hace este cambio, el entrenamiento fallará durante la evaluación. Una vez realizados estos cambios para cada uno de los LLM, se puede proceder a su entrenamiento manteniendo los mismos hiperparámetros que en el método original, excepto para el *batch size*, que dependiendo del modelo, puede ser necesario reducirlo para evitar problemas de memoria.

Todos los LLM se prueban sobre el conjunto de datos MovieLens-100K, ya que es el conjunto de datos más pequeño y permite realizar los experimentos de forma más rápida. Además, los modelos que tengan el mejor rendimiento relativo a su tamaño se evalúan en el resto de conjuntos, incluyendo el de música, para comprobar que el rendimiento obtenido no es específico de un conjunto de datos.

Se espera que los resultados obtenidos en esta fase mejoren para los modelos más modernos y con mejores valores en *benchmarks* tradicionales, especialmente en el caso de Llama 3 8B, que es el modelo más grande y con los mejores resultados en distintas evaluaciones. También se espera que los modelos más pequeños, como Gemma 2B, obtengan resultados similares a los de Llama 2 7B, pero con un tiempo de entrenamiento menor.

[8] `https://github.com/GarciaLnk/LlamaRec/blob/b9d055d/model/llm.py#L112-L127`

Capítulo 5

Resultados

En este capítulo se presentan los resultados obtenidos en el desarrollo de este trabajo, con los que se pretende responder a las preguntas de investigación planteadas en el capítulo 4. Primero, se muestra una comparación de los resultados originales de LlamaRec [135] con los resultados obtenidos en la reproducción de los experimentos, con el fin de responder a la pregunta de investigación I. Posteriormente, se presentan los resultados obtenidos en el dominio de la música, respondiendo a la pregunta de investigación II. Para responder a la pregunta de investigación III, se presentan los resultados obtenidos al utilizar otros *LLM* en el sistema. Finalmente, se presenta una herramienta interactiva desarrollada para demostrar el funcionamiento del modelo como un sistema de recomendación de música, el nuevo dominio de aplicación propuesto en este trabajo.

5.1. Reproducción de los resultados originales

En esta sección se presentan los resultados obtenidos en la reproducción de los experimentos realizados en el trabajo original de LlamaRec. Para ello, se comparan los resultados originales con los resultados obtenidos en este trabajo. Esta comparación se realiza empleando las métricas de *MRR*, *NDCG* y *Recall* con $k \in \{5, 10\}$ para los conjuntos de datos ML-100k, Beauty y Games.

Tanto los resultados originales tomados del artículo de LlamaRec [135] como los resultados obtenidos en este trabajo se presentan en la tabla 5.2. En esta tabla, se observa que los resultados obtenidos en este trabajo son muy similares a los resultados originales en todas las métricas. Para obtener estos resultados, se han utilizado los hiperparámetros ya comentados en las tablas 4.1 y 4.2. Al realizar la optimización de hiperparámetros en los modelos de referencia y en LRURec, se han obtenido los valores de *weight decay* y *dropout rate* que se muestran en la tabla 5.1.

Para facilitar la visualización de las diferencias entre los resultados originales y los resultados obtenidos en este trabajo, se ha calculado la diferencia entre las métricas de ambos conjuntos de resultados. Estos resultados se presentan en la tabla 5.3. En esta tabla, los valores negativos indican que los resultados obtenidos son peores que los originales, mientras que los valores positivos indican que los resultados obtenidos son mejores.

	ML-100k				Beauty				Games			
	NARM	**BERT**	**SAS**	**LRU**	**NARM**	**BERT**	**SAS**	**LRU**	**NARM**	**BERT**	**SAS**	**LRU**
Weight decay	0	0.01	0.01	0	0.01	0	0.01	0	0	0	0.01	0.01
Dropout rate	0.3	0.1	0.4	0.5	0.3	0.3	0.5	0.5	0.5	0.3	0.4	0.5

Tabla 5.1: Valores de *weight decay* y *dropout rate* para los modelos de referencia y LRURec en los conjuntos de datos ML-100k, Beauty y Games.

	Datos originales					Datos obtenidos				
	ML-100k					ML-100k				
	NARM	**BERT**	**SAS**	**LRU**	**LlamaRec**	**NARM**	**BERT**	**SAS**	**LRU**	**LlamaRec**
M@5	0.0298	0.0188	0.0333	0.0390	**0.0440**	0.0329	0.0169	0.0378	0.0390	**0.0440**
N@5	0.0359	0.0232	0.0409	0.0468	**0.0543**	0.0414	0.0230	0.0451	0.0472	**0.0555**
R@5	0.0544	0.0368	0.0641	0.0705	**0.0852**	0.0673	0.0417	0.0673	0.0721	**0.0902**
M@10	0.0332	0.0244	0.0378	0.0491	**0.0529**	0.0398	0.0219	0.0430	0.0491	**0.0518**
N@10	0.0441	0.0366	0.0522	0.0705	**0.0759**	0.0574	0.0353	0.0579	0.0712	**0.0745**
R@10	0.0801	0.0785	0.0993	0.1426	**0.1524**	0.1154	0.0801	0.1074	0.1458	**0.1492**
	Beauty					**Beauty**				
	NARM	**BERT**	**SAS**	**LRU**	**LlamaRec**	**NARM**	**BERT**	**SAS**	**LRU**	**LlamaRec**
M@5	0.0289	0.0246	0.0336	0.0376	**0.0385**	0.0317	0.0240	0.0358	0.0389	**0.0400**
N@5	0.0342	0.0298	0.0397	0.0435	**0.0450**	0.0374	0.0292	0.0422	0.0451	**0.0466**
R@5	0.0503	0.0457	0.0582	0.0614	**0.0648**	0.0545	0.0451	0.0615	0.0639	**0.0664**
M@10	0.0321	0.0276	0.0371	0.0417	**0.0428**	0.0351	0.0274	0.0396	0.0425	**0.0437**
N@10	0.0420	0.0372	0.0481	0.0533	**0.0554**	0.0457	0.0376	0.0515	0.0538	**0.0554**
R@10	0.0746	0.0686	0.0844	0.0916	**0.0971**	0.0807	0.0711	0.0904	0.0910	**0.0939**
	Games					**Games**				
	NARM	**BERT**	**SAS**	**LRU**	**LlamaRec**	**NARM**	**BERT**	**SAS**	**LRU**	**LlamaRec**
M@5	0.0479	0.0422	0.0515	0.0533	**0.0600**	0.0471	0.0369	0.0507	0.0524	**0.0537**
N@5	0.0576	0.0512	0.0617	0.0640	**0.0714**	0.0560	0.0447	0.0604	0.0630	**0.0661**
R@5	0.0874	0.0788	0.0930	0.0966	**0.1061**	0.0832	0.0687	0.0900	0.0954	**0.1041**
M@10	0.0541	0.0478	0.0583	0.0598	**0.0671**	0.0533	0.0424	0.0571	0.0589	**0.0610**
N@10	0.0729	0.0649	0.0783	0.0800	**0.0887**	0.0712	0.0582	0.0760	0.0790	**0.0838**
R@10	0.1351	0.1214	0.1446	0.1463	**0.1599**	0.1302	0.1106	0.1386	0.1453	**0.1587**

Tabla 5.2: Resultados de rendimiento de LlamaRec y modelos de referencia en términos de *MRR* (M), *NDCG* (N) y *Recall* (R) con $k \in \{5, 10\}$ para los conjuntos de datos ML-100k, Beauty y Games. BERT, SAS y LRU son las abreviaciones de BERT4Rec, SASRec y LRURec, respectivamente.

Por lo general, se observa que la diferencia entre los resultados originales y los resultados obtenidos en este trabajo es muy pequeña, inferior a una centésima en la mayoría de los casos, lo que indica que los resultados de la reproducción de los experimentos son muy similares a los originales. La diferencia media de todas las métricas para los tres conjuntos de datos de 0.0007, mientras que la media de las diferencias en ML-100k es de 0.0038, en Beauty de 0.0017 y en Games de -0.0035. Estas diferencias son un orden inferior a la mejora proporcionada por LlamaRec respecto al *retriever*, que suele encontrarse en el orden de las centésimas.

En todos los casos, el modelo con mejor rendimiento es LlamaRec, seguido, también en todos los casos, por LRURec. Estos resultados validan los obtenidos por el trabajo original, sugiriendo que LlamaRec puede ordenar con efectividad los ítems candidatos que son de interés para el usuario, mejorando así el rendimiento de las recomendaciones a través del conocimiento del mundo que posee el LLM.

ML-100k					
	NARM	**BERT**	**SAS**	**LRU**	**LlamaRec**
M@5	0.0031	-0.0019	0.0045	0.0000	0.0000
N@5	0.0055	-0.0002	0.0042	0.0004	0.0012
R@5	0.0129	0.0049	0.0032	0.0016	0.0050
M@10	0.0066	-0.0025	0.0052	0.0000	-0.0011
N@10	0.0133	-0.0013	0.0057	0.0007	-0.0014
R@10	0.0353	0.0016	0.0081	0.0032	-0.0032
Beauty					
	NARM	**BERT**	**SAS**	**LRU**	**LlamaRec**
M@5	0.0028	-0.0006	0.0022	0.0013	0.0015
N@5	0.0032	-0.0006	0.0025	0.0016	0.0016
R@5	0.0042	-0.0006	0.0033	0.0025	0.0016
M@10	0.0030	-0.0002	0.0025	0.0008	0.0009
N@10	0.0037	0.0004	0.0034	0.0005	0.0000
R@10	0.0061	0.0025	0.0060	-0.0006	-0.0032
Games					
	NARM	**BERT**	**SAS**	**LRU**	**LlamaRec**
M@5	-0.0008	-0.0053	-0.0008	-0.0009	-0.0063
N@5	-0.0016	-0.0065	-0.0013	-0.0010	-0.0053
R@5	-0.0042	-0.0101	-0.0030	-0.0012	-0.0020
M@10	-0.0008	-0.0054	-0.0012	-0.0009	-0.0061
N@10	-0.0017	-0.0067	-0.0023	-0.0010	-0.0049
R@10	-0.0049	-0.0108	-0.0060	-0.0010	-0.0012

Tabla 5.3: Diferencia entre los resultados obtenidos y los originales en términos de diversas métricas para ML-100k, Beauty y Games. Los valores en rojo indican que los resultados obtenidos son peores que los originales, mientras que los valores en verde indican que los resultados obtenidos son mejores.

Es importante tener en cuenta que el modelo de ranking (LlamaRec) solo mejora el rendimiento de las recomendaciones dentro del subconjunto recuperado por el modelo de recuperación (LRURec en este caso). Por lo tanto, el rendimiento de LlamaRec depende en cierta medida del rendimiento del modelo de recuperación.

Un aspecto a resaltar de ambos conjuntos de resultados es que el modelo BERT4Rec rinde por debajo de SASRec en todos los casos. Este resultado es interesante, ya que BERT4Rec es un modelo más reciente y en su publicación original [108] se muestra que supera a SASRec en todos los conjuntos de datos utilizados en ese trabajo. Esto puede indicar que la implementación de BERT4Rec en LlamaRec no es la más adecuada, problema que ya ha surgido en otras implementaciones de este modelo [85]. Para obtener una conclusión más precisa sobre este aspecto, sería necesario realizar un estudio más detallado de la implementación tanto de BERT4Rec como del resto de modelos de referencia, pero este aspecto queda fuera del alcance de este trabajo.

En definitiva, los resultados obtenidos a partir de la reproducción de los experimentos realizados en el trabajo original de LlamaRec, y basándose en el código publicado

por sus autores, son muy similares a los resultados originales. Esto permite realizar comparaciones válidas con otros dominios de aplicación, como el de la música, y con otros LLM base distintos a Llama 2 7B.

5.2. Resultados en el dominio de la música

En este apartado se presentan y analizan los resultados obtenidos al aplicar el enfoque de LlamaRec en el dominio de la música. Este análisis es fundamental para evaluar la capacidad del modelo para generalizar y proporcionar recomendaciones precisas en un contexto diferente al de los conjuntos de datos originales. Para ello, se han utilizado el conjunto de datos de LastFM-1k con sesiones, descrito en la sección 4.3. Los resultados obtenidos en este conjunto de datos se comparan con los resultados obtenidos en los conjuntos de datos ML-100k, Beauty y Games.

Para realizar una recomendación basada en sesiones, el proceso es análogo al que se sigue cuando se tienen secuencias basadas en el usuario, ya que se considera que cada sesión es un usuario distinto y anonimizado. Es decir, solo se tienen en cuenta las relaciones entre las canciones dentro de una misma sesión, y no entre sesiones distintas, aunque pertenezcan al mismo usuario. Realizar esta división de secuencias es particularmente importante en el dominio de la música, ya que el contexto del usuario puede afectar en gran medida a las canciones que escucha en una sesión. Por ejemplo, una sesión de reproducción en la que el usuario esté escuchando música para meditar, será muy distinta a otra sesión en la que el mismo usuario esté escuchando música antes de salir de fiesta.

Dada la gran diferencia en tamaño entre el conjunto de datos de LastFM-1k, que contiene 12.4M de interacciones, y los conjuntos de datos originales, que como máximo alcanzan las 200K interacciones, se realizan las pruebas sobre el conjunto íntegro de LastFM-1k y sobre dos subconjuntos con el 10 % y 1 % de las sesiones seleccionadas aleatoriamente, cada uno con 824K y 116K interacciones, respectivamente.

El tiempo de entrenamiento de los modelos con un conjunto de datos de este tamaño se ve incrementado considerablemente, por lo que se ha decidido no realizar una optimización de hiperparámetros para este conjunto. En su lugar, se han utilizado los valores configurados por defecto para el *weight decay* y *dropout rate* en los modelos de referencia y en LRURec, iguales a 0.01 y 0.2, respectivamente. Además, para agilizar el entrenamiento del *retriever* en el conjunto con el 100 % y 10 % de sesiones, se ha cambiado la estrategia de evaluación para realizarla una vez por época, en lugar de cada 500 iteraciones, y se ha reducido el *early stopping* de 20 a 5.

Tras el entrenamiento del *retriever*, LRURec, se ha observado que el rendimiento del modelo en la tarea de recomendación depende en enorme medida del tamaño del conjunto de datos. Al ser entrenado sobre el conjunto de datos completo, el modelo obtiene un *Recall@10* en el conjunto de test del 0.3529, mientras que en el conjunto con el 1 % de las sesiones, el *Recall@10* es de 0.1998.

Estos resultados indican que el rendimiento del modelo de recuperación disminuye a medida que se reduce el tamaño del conjunto de datos, lo cual es de esperar, ya que el modelo puede aprender mejor los patrones de las secuencias de interacciones con un conjunto de datos más grande.

En la figura 5.1 se muestra el valor de las métricas de *Recall*, *MRR* y *NDCG* en los distintos tamaños del conjunto de datos y con diferentes valores de k. Se puede observar que hay una diferencia considerable en el rendimiento al reducir el tamaño del conjunto de datos, especialmente al pasar del 10 % al 1 % de las sesiones. La mejora en el rendimiento con el aumento de k se mantiene en todos los casos.

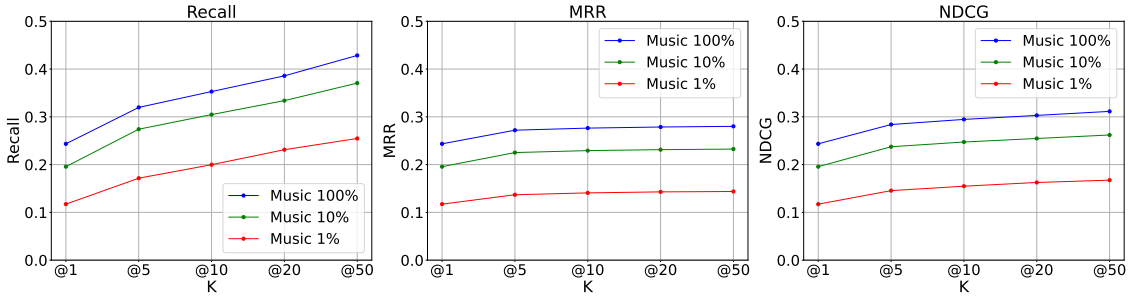

Figura 5.1: Métricas de LRURec en los distintos tamaños del conjunto de música.

A la hora de realizar el entrenamiento sobre el LLM para que aprenda la tarea de recomendación, los problemas derivados del gran tamaño del conjunto de datos de música se multiplican debido al tiempo de entrenamiento que requiere el modelo. Para el conjunto de datos de Beauty, con casi 200K interacciones, el entrenamiento de Llama 2 7B en una sola GPU tarda aproximadamente 36 horas en completarse, alcanzando el *early stop* después de casi 50K iteraciones.

Mientras tanto, el conjunto de datos de LastFM-1k con sesiones, con 60 veces más interacciones, tardaría varios meses en completar el entrenamiento con los mismos ajustes, algo inviable para este trabajo. Las opciones para reducir el tiempo de entrenamiento manteniendo el conjunto de datos, y sin modificar el método de entrenamiento, son limitadas.

Por lo tanto, se ha decidido limitar el entrenamiento del LLM sobre el conjunto de datos de música a un máximo de 200K iteraciones, sin realizar etapas de validación durante el entrenamiento para acelerar el proceso. Esta limitación se ha impuesto tanto para el conjunto de datos completo como para el subconjunto con el 10 % de sesiones, mientras que para el subconjunto con el 1 % de las sesiones se ha seguido el mismo procedimiento que con los conjuntos de referencia.

Los resultados obtenidos tras este proceso se desvían de los esperados, y se muestran en la tabla 5.4. En esta tabla, se observa que el rendimiento de LlamaRec en el conjunto de datos de música con el 100 % y 10 % de las sesiones es inferior al de los métodos de referencia sobre casi todas las métricas, siendo además SASRec superior a LRURec en muchos casos. Por otro lado, en el conjunto de datos con el 1 % de las sesiones, LlamaRec obtiene resultados muy parejos a los de LRURec, siendo mejor en algunas métricas y peor en otras.

En la figura 5.2 se muestra una comparación directa entre los resultados de LRURec y LlamaRec en los conjuntos de datos de música de distinto tamaño. Se puede observar que la diferencia de rendimiento entre los dos modelos es bastante notable para el conjunto de datos completo y el de 10 % de las sesiones, mientras que para el conjunto de 1 % de las sesiones, la diferencia es mucho menor.

	Music (100 %)				
	NARM	**BERT**	**SAS**	**LRU**	**LlamaRec**
M@5	0.1652	0.1947	0.2692	**0.2720**	0.1043
N@5	0.1747	0.2141	0.2820	**0.2839**	0.1272
R@5	0.2031	0.2726	**0.3204**	0.3197	0.1977
M@10	0.1690	0.2003	0.2737	**0.2764**	0.1194
N@10	0.1838	0.2277	0.2928	**0.2946**	0.1641
R@10	0.2313	0.3143	**0.3538**	0.3529	0.3118
	Music (10 %)				
	NARM	**BERT**	**SAS**	**LRU**	**LlamaRec**
M@5	0.2012	0.1554	**0.2374**	0.2252	0.0834
N@5	0.2104	0.1728	**0.2408**	0.2374	0.1053
R@5	0.2382	0.2254	0.2712	**0.2741**	0.1730
M@10	0.2047	0.1605	**0.2343**	0.2293	0.0979
N@10	0.2190	0.1850	**0.2495**	0.2473	0.1406
R@10	0.2647	0.2625	0.2980	**0.3047**	0.2824
	Music (1 %)				
	NARM	**BERT**	**SAS**	**LRU**	**LlamaRec**
M@5	0.1091	0.0889	0.1251	**0.1369**	0.1320
N@5	0.1147	0.0995	0.1303	0.1455	**0.1461**
R@5	0.1318	0.1318	0.1459	0.1715	**0.1883**
M@10	0.1108	0.0914	0.1270	**0.1408**	0.1350
N@10	0.1189	0.1057	0.1351	**0.1548**	0.1535
R@10	0.1449	0.1506	0.1611	0.1998	**0.2113**

Tabla 5.4: Resultados de rendimiento para los modelos NARM, BERT4Rec, SASRec, LRURec y LlamaRec en el dominio de la música, con diferentes tamaños del conjunto.

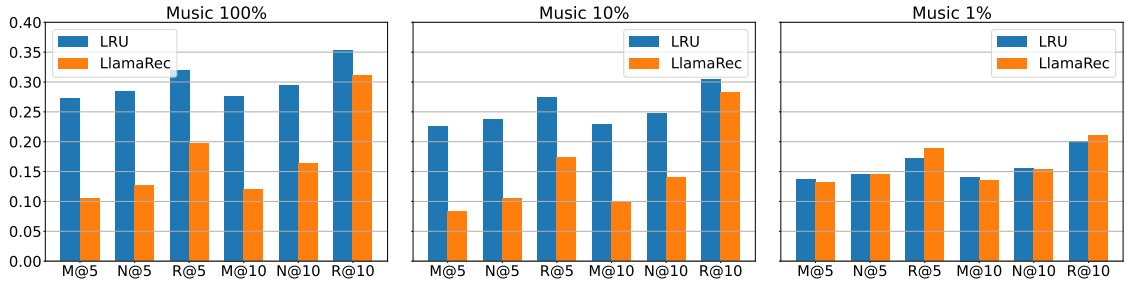

Figura 5.2: Comparación de las métricas de LRURec y LlamaRec en los conjuntos de música con distintos tamaños.

El resultado obtenido en el conjunto de datos con el 1 % de las sesiones, de una dimensión similar a los conjuntos de referencia, sugiere que LlamaRec tiene potencial para mejorar el rendimiento de las recomendaciones en el dominio de la música. Aunque la mejora sea casi inapreciable en este caso, es posible que, con un LLM base distinto y un ajuste de hiperparámetros, el modelo pueda superar en todas las métricas a LRURec en este dominio.

Los malos resultados obtenidos en los conjuntos de datos más grandes pueden explicarse por dos motivos distintos. El primero, que el modelo no haya sido capaz de aprender correctamente debido a la falta de tiempo de entrenamiento, para solventar este problema, sería necesario entrenar el modelo durante más tiempo, o cambiar la estrategia de entrenamiento para que sea más eficiente. El segundo motivo, que el modelo de lenguaje tenga un límite en su capacidad para mejorar el rendimiento de un *retriever* que de por sí ya obtiene resultados muy buenos.

Este último caso supondría un límite en la utilidad de este enfoque, siendo más útil en situaciones de *cold start* donde el sistema dispone de pocos datos sobre las interacciones del usuario con los ítems. Mientras que en situaciones de donde el sistema ya dispone de suficientes datos para realizar recomendaciones, los métodos tradicionales de recomendación pueden ser más efectivos. Esto es algo que ya se ha observado en otros trabajos sobre sistemas de recomendación basados en LLM, como el de Kim et al. [60].

5.3. Resultados con otros LLM base

A continuación, se exploran los resultados obtenidos al implementar otros LLM como base en el enfoque de LlamaRec. El objetivo es determinar si el rendimiento de LlamaRec puede mejorar al emplear otros LLM en lugar del modelo utilizado inicialmente. Los modelos seleccionados para esta comparación son Llama 3 8B, Mistral 7B, Phi 3 3.8B y Gemma 2B, descritos con anterioridad en la sección 4.4.

Todos estos modelos se probarán sobre el conjunto de datos de ML-100k, al ser el más pequeño de los conjuntos de referencia y, por lo tanto, el más rápido de entrenar. Los modelos con mejor rendimiento respecto a su tamaño, que como se verá más adelante son Llama 3 8B y Gemma 2B, se probarán también en el resto de conjuntos de datos, incluyendo el de música.

Para cada uno de estos modelos, se ha seguido el mismo procedimiento que con Llama 2 7B, utilizando el mismo *retriever* preentrenado (LRURec), y realizando un proceso de *fine-tuning* para la tarea de recomendación sobre los distintos conjuntos de datos. El entrenamiento de los modelos se ha realizado con los mismos hiperparámetros que en el caso de Llama 2 7B, con la excepción del *batch size*, que se ha reducido a 8 para Llama 3 8B y Gemma 2B debido a limitaciones en la memoria de la GPU.

Los resultados obtenidos en el conjunto de datos de ML-100k se muestran en la tabla 5.5. En esta tabla, se observa que Llama 3 8B obtiene los mejores resultados en la mayoría de las métricas, seguido de cerca por Gemma 2B. Exceptuando Phi 3, que obtiene los peores resultados, llegando a ser peor que LRURec en algunas métricas, el resto de modelos superan a la implementación original de LlamaRec, basada en Llama 2 7B. Dados los buenos resultados de Llama 3 8B y Gemma 2B se entrenarán también en los conjuntos de datos de Beauty, Games y música.

Llama la atención que Phi 3 3.8B obtenga resultados tan bajos en comparación con los demás modelos, especialmente teniendo en cuenta que en *benchmarks* tradicionales, Phi 3 obtiene muy buenos resultados. Esto puede deberse a que el modelo es una versión que ya ha sido *instruction tuned* para tareas de asistente de chat, mientras que el resto de modelos son versiones fundacionales.

	ML-100k					
	LRU	**LlamaRec**	**Llama3Rec**	**GemmaRec**	**MistralRec**	**Phi3Rec**
M@5	0.0390	0.0440	**0.0624**	0.0624	0.0559	0.0413
N@5	0.0472	0.0555	**0.0745**	0.0717	0.0676	0.0525
R@5	0.0721	0.0902	**0.1115**	0.1000	0.1033	0.0869
M@10	0.0491	0.0518	0.0677	**0.0680**	0.0620	0.0488
N@10	0.0712	0.0745	**0.0876**	0.0857	0.0827	0.0706
R@10	0.1458	0.1492	**0.1525**	0.1443	0.1508	0.1426

Tabla 5.5: Resultados de rendimiento de los distintos LLM en el conjunto de datos ML-100k. LlamaRec se corresponde con Llama 2 7B, Llama3Rec con Llama 3 8B, GemmaRec con Gemma 2B, MistralRec con Mistral 7B y Phi3Rec con Phi 3 3.8B.

Para probar esta teoría, se ha decidido usar como *ranker* la versión de Gemma 2B que ha sido sometida a un proceso de *instruction tuning* para tareas de asistente de chat, Gemma-IT 2B [112]. Esta versión ha sido posteriormente entrenada para la tarea de ordenación de ítems siguiendo el mismo proceso que para la versión fundacional. En la figura 5.3 se muestran los resultados obtenidos por Gemma-IT 2B en comparación con Gemma 2B. Se puede observar que Gemma-IT 2B obtiene resultados peores que Gemma 2B en todas las métricas, lo que sugiere que el proceso previo de *instruction tuning*, que incluye tareas no relacionadas con la tarea de recomendación, afecta negativamente al rendimiento del modelo en esta tarea.

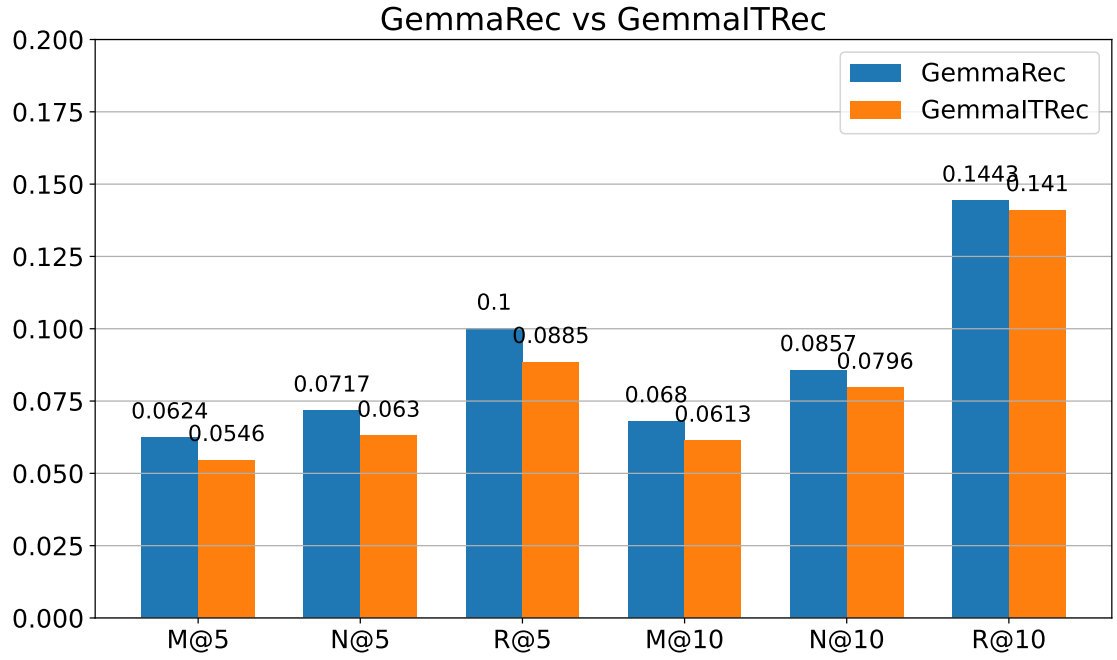

Figura 5.3: Comparación del rendimiento de Gemma 2B y Gemma-IT 2B (*instruction tuned*) en el conjunto de datos ML-100k.

La mejora obtenida por Llama 3 8B y Gemma 2B en comparación con Llama 2 7B es muy significativa. Si se pone en términos relativos, la mejora de LlamaRec con Llama 2 respecto a LRURec es del 4.62 % para NDCG@10. Mientras que la mejora de LlamaRec con Llama 3 es del 23.05 % y con Gemma es del 20.37 %. Esto supone

una mejora mucho mayor en el rendimiento de las recomendaciones, lo que sugiere que el enfoque de LlamaRec no solo es efectivo con otros LLM, sino que escala bien con modelos con mayores capacidades.

En la tabla 5.6 se muestra la mejora relativa de rendimiento de LlamaRec con Llama 2, Llama 3 y Gemma sobre LRURec en el conjunto de datos ML-100k. Es destacable que la métrica de Recall@10 es la que menos mejora en todos los casos. Esto puede deberse a que el *ranker* está limitado por el rendimiento del *retriever*, y, por lo tanto, no puede incluir en el top 10 de recomendaciones a más ítems relevantes al no haber sido recuperados para su ordenación.

	LlamaRec	**Llama3Rec**	**GemmaRec**
M@5	12.79 %	59.88 %	59.81 %
N@5	17.66 %	57.87 %	51.82 %
R@5	25.03 %	54.58 %	38.67 %
M@10	5.59 %	38.10 %	38.71 %
N@10	4.62 %	23.05 %	20.37 %
R@10	2.30 %	4.54 %	-1.08 %

Tabla 5.6: Mejora relativa de rendimiento de LlamaRec sobre LRURec con Llama 2, Llama 3 y Gemma en el conjunto de datos ML-100k.

	ML-100k				**Music (100 %)**			
	LRU	**LlamaRec**	**Llama3Rec**	**GemmaRec**	**LRU**	**LlamaRec**	**Llama3Rec**	**GemmaRec**
M@5	0.0390	0.0440	**0.0624**	0.0624	**0.2720**	0.1043	0.1470	0.1346
N@5	0.0472	0.0555	**0.0745**	0.0717	**0.2839**	0.1272	0.1735	0.1591
R@5	0.0721	0.0902	**0.1115**	0.1000	**0.3197**	0.1977	0.2541	0.2338
M@10	0.0491	0.0518	0.0677	**0.0680**	**0.2764**	0.1194	0.1576	0.1462
N@10	0.0712	0.0745	**0.0876**	0.0857	**0.2946**	0.1641	0.1989	0.1872
R@10	0.1458	0.1492	**0.1525**	0.1443	**0.3529**	0.3118	0.3319	0.3201
	Beauty				**Music (10 %)**			
	LRU	**LlamaRec**	**Llama3Rec**	**GemmaRec**	**LRU**	**LlamaRec**	**Llama3Rec**	**GemmaRec**
M@5	0.0389	0.0400	**0.0462**	0.0429	**0.2252**	0.0834	0.1261	0.1137
N@5	0.0451	0.0466	**0.0534**	0.0498	**0.2374**	0.1053	0.1516	0.1372
R@5	0.0639	0.0664	**0.0753**	0.0708	**0.2741**	0.1730	0.2294	0.2091
M@10	0.0425	0.0437	**0.0496**	0.0466	**0.2293**	0.0979	0.1361	0.1247
N@10	0.0538	0.0554	**0.0616**	0.0589	**0.2473**	0.1406	0.1754	0.1637
R@10	0.0910	0.0939	**0.1007**	0.0990	**0.3047**	0.2824	0.3025	0.2907
	Games				**Music (1 %)**			
	LRU	**LlamaRec**	**Llama3Rec**	**GemmaRec**	**LRU**	**LlamaRec**	**Llama3Rec**	**GemmaRec**
M@5	0.0524	0.0537	**0.0667**	0.0543	0.1369	0.1320	**0.1486**	0.1449
N@5	0.0630	0.0661	**0.0791**	0.0658	0.1455	0.1461	**0.1589**	0.1540
R@5	0.0954	0.1041	**0.1168**	0.1006	0.1715	0.1883	**0.1899**	0.1815
M@10	0.0589	0.0610	**0.0733**	0.0614	0.1408	0.1350	**0.1522**	0.1480
N@10	0.0790	0.0838	**0.0953**	0.0831	0.1548	0.1535	**0.1674**	0.1614
R@10	0.1453	0.1587	**0.1674**	0.1545	0.1998	0.2113	**0.2160**	0.2040

Tabla 5.7: Resultados de rendimiento para los modelos LRU, LlamaRec con Llama 2, Llama 3 y Gemma en los conjuntos de datos ML-100k, Beauty, Games y LastFM-1k con sesiones (con diferentes tamaños).

Al aplicar Llama 3 8B y Gemma 2B en el resto de conjuntos, las mejoras son generalizadas, como se puede ver en la tabla 5.7. En todos los casos, Llama 3 supera

a Llama 2, mientras que Gemma en la mayoría de los casos está por debajo de Llama 3, pero por encima de Llama 2. Esto confirma las mejoras observadas en el conjunto de ML-100k. En la figura 5.4 se puede ver una comparación del resultado de la métrica NDCG@10 para los modelos basados en los distintos LLM.

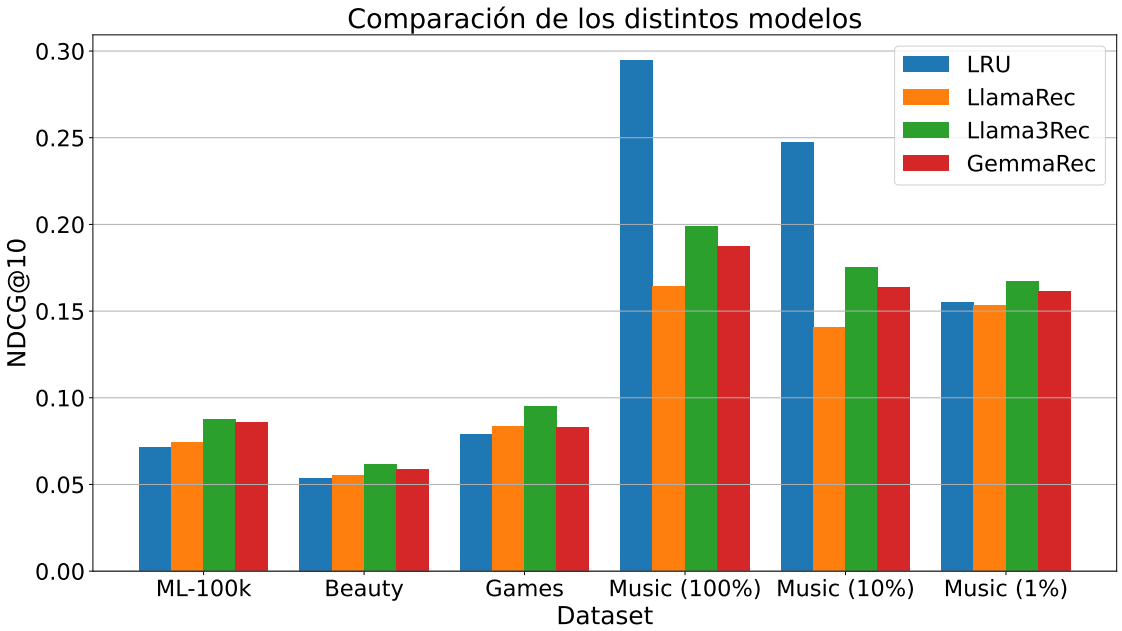

Figura 5.4: Comparación de los resultados de NDCG@10 para Llama 2, Llama 3 y Gemma en los distintos conjuntos de datos.

En el caso del conjunto de datos del dominio de la música, para el conjunto con el 1 % de las sesiones, tanto Llama 3 como Gemma son capaces de mejorar el rendimiento de LRURec en todas las métricas, siendo Llama 3 el que obtiene los mejores resultados. Esto confirma que el enfoque de LlamaRec es efectivo en este dominio, y que la elección del LLM base es crucial para obtener buenos resultados. En los conjuntos de mayor tamaño, se observa una mejora en el rendimiento, pero aun así, los resultados siguen siendo inferiores a los obtenidos por LRURec.

5.4. Herramienta interactiva del sistema

En esta sección se describe el desarrollo y las funcionalidades de la herramienta desarrollada para demostrar el funcionamiento del enfoque de LlamaRec como un sistema de recomendación de música. Esta herramienta ha sido diseñada como una prueba de concepto para mostrar los resultados de un sistema basado en LlamaRec en un entorno más interactivo, donde los usuarios pueden introducir su historial de reproducción y recibir recomendaciones personalizadas. La herramienta se ha desarrollado utilizando la biblioteca Streamlit y se ha creado un cuaderno de Jupyter[1] para que cualquier usuario pueda desplegar la herramienta de forma gratuita en un entorno de Google Colab.

Para ejecutar esta herramienta, es necesario proporcionar un diccionario del conjunto de datos que mapee los identificadores de las canciones a sus metadatos corres-

[1] `https://github.com/GarciaLnk/LlamaRec/blob/main/demo_nb.ipynb`

pondientes, que incluyen información como el nombre de la canción, el nombre del artista y el identificador de Spotify. La herramienta también necesita acceso a un modelo *retriever* preentrenado y a un LLM que haya sido sujeto a un proceso de *fine-tuning* para la tarea de recomendación de música. Para facilitar la ejecución de la herramienta, ya se proporcionan estos archivos en un archivo comprimido[2] que contiene una versión preentrenada de LRURec y de Llama 3, así como el diccionario con los metadatos de las canciones del conjunto de datos de LastFM-1k con sesiones.

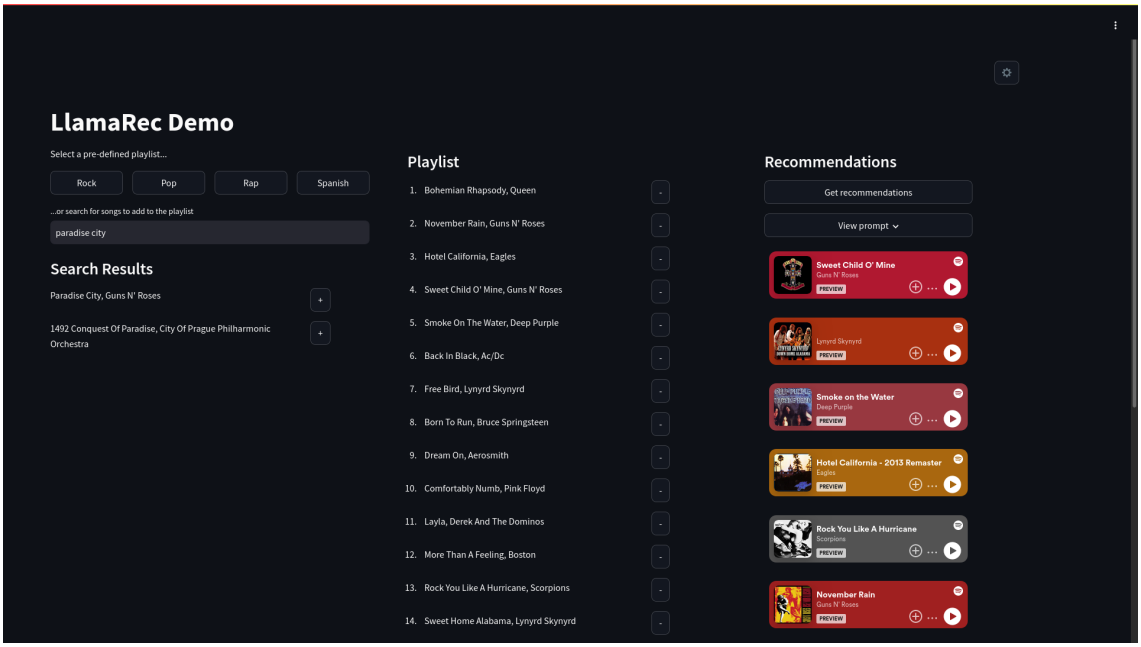

Figura 5.5: Captura de pantalla de la herramienta basada en LlamaRec.

Al inicializar la herramienta, además de cargarse los modelos en memoria, se crea un índice con los títulos de las canciones y los nombres de los artistas para permitir la búsqueda de canciones por su nombre. Estas canciones se muestran en una lista en la interfaz de usuario, donde el usuario puede añadirlas a su historial de reproducción. Además, también hay cuatro *playlists* predefinidas que el usuario puede seleccionar para establecer como su historial de reproducción. Una vez que el usuario ha establecido su historial de reproducción, puede hacer clic en el botón de obtener recomendación para obtener una lista de canciones recomendadas, que pueden reproducirse directamente en la página web o en Spotify. En la figura 5.5 se muestra una captura de pantalla de esta herramienta.

Una vez se obtiene la lista de recomendaciones, se puede consultar el *prompt* de entrada que se ha empleado para que el LLM ordene las canciones recuperadas por el *retriever*. En la figura 5.6 se muestra un ejemplo de un *prompt* que se ha utilizado para obtener las recomendaciones. Este *prompt* contiene una instrucción para el modelo: *"Given user history in chronological order, recommend an item from the candidate pool with its index letter"*, ("Dado el historial del usuario en orden cronológico, recomienda un elemento de la lista de candidatos con su letra de índice"). Esta instrucción está seguida de una lista de canciones que forman parte del historial de reproducción del usuario, y por último, se muestra la lista de canciones

[2] `https://files.garcialnk.com/llamarec-demo.zip`

recuperadas por el *retriever*, identificadas por una letra que se corresponde con el índice de la lista.

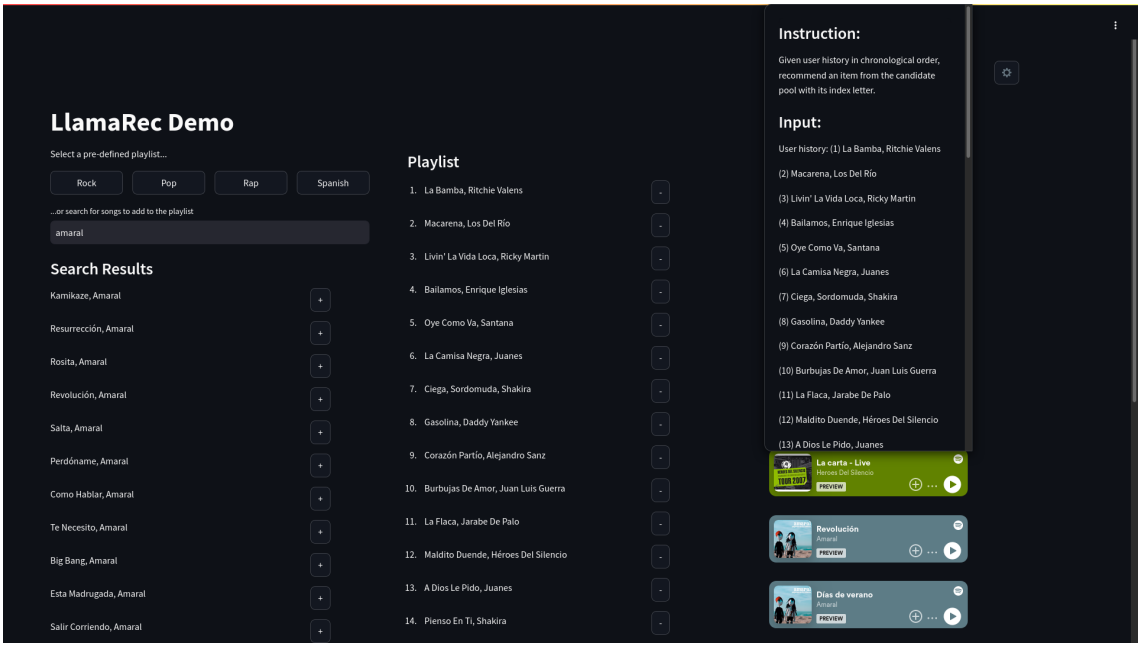

Figura 5.6: Ejemplo de un *prompt* utilizado por el LLM.

La instrucción proporcionada al modelo se puede modificar en los ajustes del usuario, sin embargo, hay que tener en consideración que el LLM ha sido entrenado solo con la instrucción proporcionada por defecto. Por lo tanto, si se cambia la instrucción, el modelo puede no ser capaz de generar recomendaciones precisas, y probablemente no será capaz de seguir adecuadamente las indicaciones del usuario. Además de la instrucción, también se pueden modificar otros parámetros, como el número de canciones a recuperar por el *retriever* y el número de canciones a recomendar por el *ranker*. En la figura 5.7 se muestra la interfaz de usuario donde se pueden modificar estos parámetros.

La herramienta desarrollada ofrece resultados adecuados, por ejemplo, cuando se selecciona una *playlist* de rock, se obtienen recomendaciones de canciones de rock, y cuando se selecciona una *playlist* de música en español, se obtienen recomendaciones de canciones en español. Un posible uso de esta herramienta sería en un estudio de usuario, donde los usuarios podrían evaluar de forma sencilla la calidad de las recomendaciones obtenidas dada una lista con sus canciones favoritas. Distintos usuarios podrían interactuar con distintas versiones de la herramienta, donde se podrían modificar los modelos usados para evaluar su impacto en la calidad de las recomendaciones. De esta forma podría obtenerse información adicional el enfoque de LlamaRec en un entorno más realista.

El código de esta herramienta está disponible en GitHub[3] y además de poder ejecutarse en un entorno de Google Colab, también se puede desplegar en cualquier otro entorno con una gráfica dedicada. Los requisitos técnicos para ejecutar la herramienta dependerán en gran parte del LLM subyacente y de la longitud de la lista de reproducción del usuario (que está limitada por defecto a 50). En general, si

[3] `https://github.com/GarciaLnk/LlamaRec/tree/main/demo`

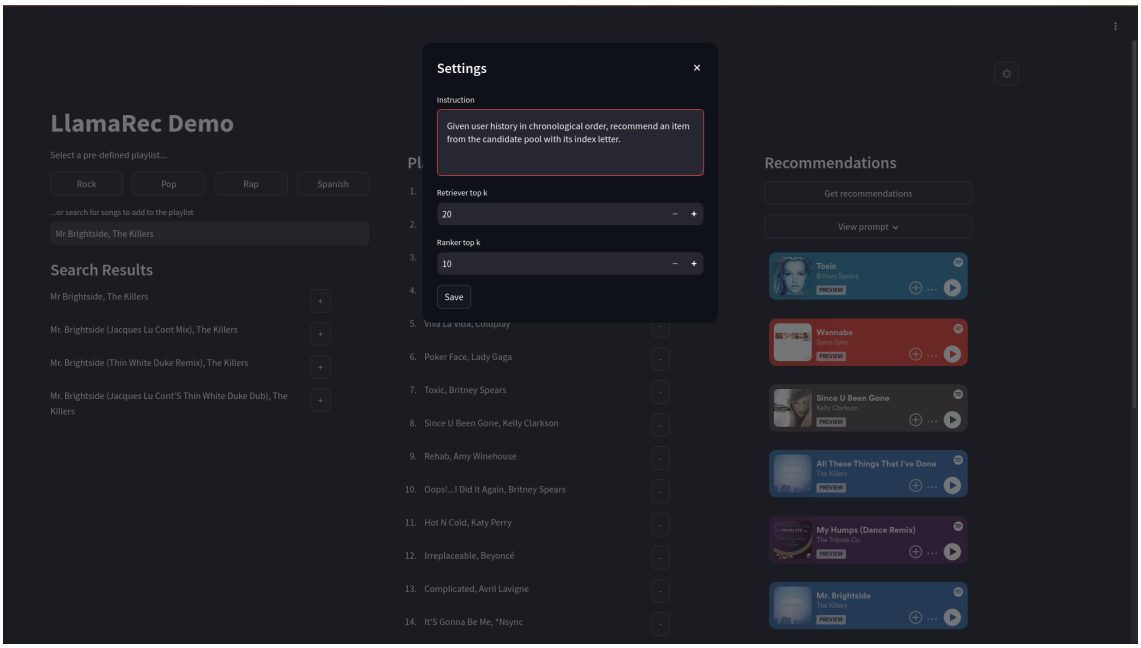

Figura 5.7: Ajustes de la herramienta.

se emplea Llama 3 8B como LLM base, se recomienda disponer de una GPU con al menos 8 GB de memoria VRAM, mientras que si se usa Gemma 2B, una GPU con 4 GB de memoria VRAM sería suficiente. En cualquier caso, se recomienda ejecutar la herramienta en un entorno con una GPU moderna para obtener resultados en un tiempo razonable. En una 3090 Ti, la herramienta tarda apenas un segundo en devolver las recomendaciones.

Capítulo 6

Conclusiones

En este trabajo, se ha explorado el potencial de los grandes modelos de lenguaje (*LLM*) en el campo de los sistemas de recomendación, con el objetivo de determinar si estos modelos pueden mejorar el rendimiento de los métodos tradicionales en la generación de recomendaciones personalizadas. En este capítulo, se presentan las conclusiones obtenidas a partir de la revisión de la literatura y del estudio de reproducibilidad del método LlamaRec, así como las limitaciones del estudio y las líneas futuras de investigación que se derivan de este trabajo.

Primero, se ha realizado una revisión de la literatura para identificar los enfoques más relevantes en los sistemas de recomendación y en los LLM. Esto ha permitido conocer el estado del arte en ambos campos y comprender cómo se pueden aplicar los LLM en los sistemas de recomendación para mejorar los mismos. Se ha observado un crecimiento significativo durante los últimos años en el desarrollo de LLM y en su aplicación en los sistemas de recomendación.

Posteriormente, se ha reproducido y evaluado uno de los métodos identificados en la revisión de la literatura, LlamaRec. Se ha conseguido reproducir con éxito los resultados originales del método, respondiendo a positivamente a la pregunta I, aunque se han identificado algunas deficiencias en la implementación original que han dificultado su reproducibilidad. No obstante, estos problemas han sido solucionados y se espera que pueda ayudar a futuros investigadores que quieran utilizar este método.

A continuación, se ha evaluado el enfoque de LlamaRec en un dominio de recomendación diferente a los ya evaluados, el de la música. El gran tamaño del conjunto de datos seleccionado ha supuesto un problema para su entrenamiento siguiendo el método original, obteniendo resultados inferiores a los de referencia. Sin embargo, al reducir los datos a un subconjunto más pequeño, el enfoque de LlamaRec ha demostrado ser efectivo para generar recomendaciones en el dominio de la música, lo que resuelve la pregunta II.

Además, se han aplicado distintos LLM del estado del arte para evaluar si estos modelos pueden mejorar los resultados del método original, y su mejora relativa respecto al *retriever*. Se ha observado que el rendimiento del método se ve significativamente mejorado al utilizar modelos de lenguaje más avanzados, como Llama 3 o Gemma, llegando a una mejora de hasta el 60 % sobre LRURec. Esto sugiere que el enfoque de LlamaRec puede beneficiarse del desarrollo y publicación de LLM con

mayores capacidades, lo que responde a la pregunta III.

También se ha desarrollado una herramienta interactiva que permite hacer pruebas de la aplicación práctica del método LlamaRec en el dominio de la música. Esta herramienta puede permitir explorar las recomendaciones generadas por el método en un entorno controlado y facilitar la evaluación de su rendimiento en diferentes escenarios de uso con usuarios reales.

En conclusión, se considera que este trabajo ha cumplido con éxito los objetivos planteados para el mismo, respondiendo también las preguntas de investigación planteadas, y ha contribuido a avanzar en la comprensión de cómo los LLM pueden mejorar los sistemas de recomendación. Se ha demostrado que los enfoques como LlamaRec son efectivos en múltiples dominios y que su rendimiento puede mejorarse aún más con la incorporación de modelos de lenguaje más avanzados. Sin embargo, también se han identificado varias limitaciones que podrían ser objeto de futuras investigaciones.

6.1. Limitaciones del estudio

A lo largo de este trabajo, se han identificado varias limitaciones que podrían haber afectado a los resultados obtenidos y que podrían ser objeto de futuras investigaciones. Algunas de las limitaciones más relevantes son:

- **Limitaciones en la reproducibilidad de los métodos de referencia**: La implementación de los métodos de referencia utilizados en el trabajo original no se incluía en el repositorio de código original. Sin embargo, a partir de otro repositorio del mismo autor se ha podido obtener la implementación de estos métodos. Aun así, no se ha explorado la correcta implementación de estos métodos, lo que podría haber afectado a los resultados obtenidos.
- **Dependencia de recursos computacionales significativos**: El entrenamiento de LlamaRec requiere de una cantidad significativa de recursos computacionales, lo que puede limitar su aplicabilidad en entornos con recursos limitados. En entornos de producción con grandes volúmenes de datos, el entrenamiento de LlamaRec podría ser prohibitivamente costoso, además de requerir un tiempo considerable para entrenarlo con nuevos datos. Esto pone en duda la utilidad de este enfoque, ya que las mejoras son relativamente pequeñas teniendo en cuenta este coste computacional.
- **Baja eficacia en conjuntos de datos de gran tamaño**: El rendimiento de LlamaRec en conjuntos de datos de gran tamaño ha sido inferior al esperado, lo que sugiere que el método podría no ser adecuado para aplicaciones con grandes volúmenes de datos. Esto puede estar causado por las limitaciones de tiempo y recursos a la hora de hacer *fine-tuning* del LLM, o porque el sistema de recomendación tradicional escala mucho mejor que el LLM en conjuntos de datos grandes.

Estas limitaciones sugieren que hay oportunidades para futuras investigaciones que podrían abordar estos problemas y avanzar en el desarrollo de sistemas de recomendación basados en LLM.

6.2. Líneas futuras de investigación

Basándose en las conclusiones obtenidas en este trabajo, se proponen varias líneas futuras de investigación que podrían abordar las limitaciones identificadas y avanzar en el desarrollo de sistemas de recomendación basados en LLM. Algunas de las líneas futuras de investigación más relevantes son:

- **Establecimiento de mejores prácticas para la reproducibilidad**: Se propone el desarrollo de un marco de trabajo que permita la reproducción de métodos de recomendación basados en LLM de forma más sencilla y eficiente. Esto podría incluir la publicación de implementaciones de referencia de los modelos más utilizados, la documentación detallada de los mismos y la publicación de los conjuntos de datos más populares para poder comparar los resultados de diferentes estudios de forma más sencilla, manteniendo la rigurosidad científica.
- **Prueba de otros modelos tradicionales de recomendación**: Se podría evaluar el rendimiento y la mejora del enfoque de LlamaRec al utilizar otros modelos tradicionales de recomendación en la fase de recuperación de candidatos. Esto podría permitir identificar si existen otros modelos distintos a LRURec que puedan mejorar el rendimiento del método, así como determinar el grado de mejora al incorporar un *ranker* sobre estos modelos. Ya que esta mejora del *ranker* depende directamente del *retriever* y existe un gran número de sistemas de recomendación secuencial que no se han evaluado en este trabajo.
- **Mejora de la eficiencia del entrenamiento de LlamaRec**: Sería apropiado investigar métodos para mejorar la eficiencia del entrenamiento de LlamaRec, reduciendo el tiempo y los recursos necesarios para entrenar el LLM. Esto podría incluir el uso de técnicas de destilación de conocimiento (transfiriendo el conocimiento de un modelo grande a uno más pequeño), el uso de arquitecturas más eficientes o la optimización de las secuencias de datos usadas para el entrenamiento.
- **Exploración de conjuntos de datos más grandes y LLM más complejos**: Empleando infraestructuras más avanzadas, se podría explorar la viabilidad de estos métodos en conjuntos de datos más grandes, así como el rendimiento al utilizar LLM con un mayor número de parámetros. Esto podría determinar con mayor certidumbre las limitaciones de estos métodos y las posibilidades de escalabilidad en aplicaciones reales.
- **Inclusión de información adicional para el LLM**: Se sugiere investigar la inclusión de información adicional en el entrenamiento del LLM, como más metadatos de los ítems, información contextual sobre el usuario, o instrucciones en lenguaje natural para guiar la generación de recomendaciones. Esto podría mejorar la capacidad del LLM para generar recomendaciones personalizadas y adaptadas a las preferencias del usuario, lo que podría mejorar su experiencia.

Estas líneas futuras de investigación podrían contribuir a mejorar la eficiencia de los sistemas de recomendación basados en LLM, y a avanzar el desarrollo de métodos más reproducibles, escalables y efectivos en este campo.

Lista de acrónimos

AP Del inglés, Average Precision, Precisión Media. 28

AUC Del inglés, Area Under the ROC Curve, Área Bajo la Curva ROC. 28

CNN Del inglés, Convolutional Neural Network, Red Neuronal Convolucional. 21

CoT Del inglés, Chain-of-Thoughts, Cadena de Pensamientos. 38, 39, 42

CTR Del inglés, Click-Through Rate, Tasa de Clics. 26

GCN Del inglés, Graph Convolutional Network, Red Convolucional basada en Grafos. 18

GNN Del inglés, Graph Neural Network, Red Neuronal de Grafos. 18, 21

ICL Del inglés, In-Context Learning, Aprendizaje en Contexto. 37, 39

LLM Del inglés, Large Language Model, Gran Modelo de Lenguaje. 8, 11, 29, 30, 42, 49, 53, 63, 76

LSTM Del inglés, Long Short-Term Memory, Memoria a Corto y Largo Plazo. 31

MAE Del inglés, Mean Absolute Error, Error Absoluto Medio. 27

MLP Del inglés, Multilayer Perceptron, Perceptrón Multicapa. 21, 30

MRR Del inglés, Mean Reciprocal Rank, Rango Recíproco Medio. 28, 55, 63, 64, 67

NDCG Del inglés, Normalized Discounted Cumulative Gain, Ganancia Acumulada Descontada Normalizada. 55, 63, 64, 67

NLP Del inglés, Natural Language Processing, Procesamiento de Lenguaje Natural. 29, 30

NPDM Del inglés, Normalized Distance based Performance Measure, Medida de Rendimiento basada en la Distancia Normalizada. 28

PEFT Del inglés, Parameter Efficient Fine-Tuning, Ajuste Fino Eficiente de Parámetros. 7, 35, 47, 51

PPO Del inglés, Proximal Policy Optimization, Optimización de Políticas Próximas. 34

PTQ Del inglés, Post-Training Quantization, Cuantización Post-Entrenamiento. 36

QAT Del inglés, Quantization-Aware Training, Entrenamiento Consciente de la Cuantización. 36

RLHF Del inglés, Reinforcement Learning from Human Feedback, Aprendizaje por Refuerzo a partir de Retroalimentación Humana. 32, 34

RMSE Del inglés, Root Mean Square Error, Raíz del Error Cuadrático Medio. 27

RNN Del inglés, Recurrent Neural Network, Red Neuronal Recurrente. 21, 23, 31, 32, 54

RPM Del inglés, Revenue Per Thousands Impressions, Ingresos Por cada Mil impresiones. 26

RQ Del inglés, Research Question, Pregunta de Investigación. 53

RS Del inglés, Recommender System, Sistema de Recomendación. 11, 24

SRS Del inglés, Sequential Recommender System, Sistema de Recomendación Secuencial. 19

Bibliografía

[1] Marah Abdin et al. *Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone*. 23 de mayo de 2024. DOI: 10.48550/arXiv.2404.14219. arXiv: 2404.14219 [cs]. URL: http://arxiv.org/abs/2404.14219. Prepublicado (vid. págs. 41, 60).

[2] Charu C. Aggarwal. «An Introduction to Recommender Systems». En: *Recommender Systems: The Textbook*. Ed. por Charu C. Aggarwal. Cham: Springer International Publishing, 2016, págs. 1-28. ISBN: 978-3-319-29659-3. DOI: 10.1007/978-3-319-29659-3_1. URL: https://doi.org/10.1007/978-3-319-29659-3_1 (vid. pág. 14).

[3] Charu C. Aggarwal. «Evaluating Recommender Systems». En: *Recommender Systems: The Textbook*. Ed. por Charu C. Aggarwal. Cham: Springer International Publishing, 2016, págs. 225-254. ISBN: 978-3-319-29659-3. DOI: 10.1007/978-3-319-29659-3_7. URL: https://doi.org/10.1007/978-3-319-29659-3_7 (vid. pág. 24).

[4] AI@Meta. «Llama 3 Model Card». En: (2024). URL: https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md (vid. págs. 41, 59).

[5] Vito Walter Anelli et al. «Top-N Recommendation Algorithms: A Quest for the State-of-the-Art». En: *Proceedings of the 30th ACM Conference on User Modeling, Adaptation and Personalization*. 4 de jul. de 2022, págs. 121-131. DOI: 10.1145/3503252.3531292. arXiv: 2203.01155 [cs]. URL: http://arxiv.org/abs/2203.01155 (vid. pág. 16).

[6] Anthropic. *Introducing Claude 3.5 Sonnet*. 21 de jun. de 2024. URL: https://www.anthropic.com/news/claude-3-5-sonnet (vid. pág. 41).

[7] Donald Bamber. «The Area above the Ordinal Dominance Graph and the Area below the Receiver Operating Characteristic Graph». En: *Journal of Mathematical Psychology* 12.4 (1 de nov. de 1975), págs. 387-415. ISSN: 0022-2496. DOI: 10.1016/0022-2496(75)90001-2. URL: https://www.sciencedirect.com/science/article/pii/0022249675900012 (vid. pág. 28).

[8] Keqin Bao et al. «TALLRec: An Effective and Efficient Tuning Framework to Align Large Language Model with Recommendation». En: *Proceedings of the 17th ACM Conference on Recommender Systems*. 14 de sep. de 2023, págs. 1007-1014. DOI: 10.1145/3604915.3608857. arXiv: 2305.00447 [cs]. URL: http://arxiv.org/abs/2305.00447 (vid. págs. 43, 45).

[9] Yoshua Bengio et al. «A Neural Probabilistic Language Model». En: *J. Mach. Learn. Res.* 3 (null 1 de mar. de 2003), págs. 1137-1155. ISSN: 1532-4435 (vid. pág. 30).

[10] Daniel Billsus y Michael J. Pazzani. «Learning Collaborative Information Filters». En: *Proceedings of the Fifteenth International Conference on Machine Learning*. ICML '98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 24 de jul. de 1998, págs. 46-54. ISBN: 978-1-55860-556-5 (vid. pág. 18).

[11] Vadim Borisov et al. *Language Models Are Realistic Tabular Data Generators*. 22 de abr. de 2023. DOI: 10.48550/arXiv.2210.06280. arXiv: 2210.06280 [cs]. URL: http://arxiv.org/abs/2210.06280. Prepublicado (vid. pág. 44).

[12] Brian Brost, Rishabh Mehrotra y Tristan Jehan. «The Music Streaming Sessions Dataset». En: *The World Wide Web Conference*. WWW '19. New York, NY, USA: Association for Computing Machinery, 13 de mayo de 2019, págs. 2594-2600. ISBN: 978-1-4503-6674-8. DOI: 10.1145/3308558.3313641. URL: https://doi.org/10.1145/3308558.3313641 (vid. pág. 58).

[13] Tom B. Brown et al. *Language Models Are Few-Shot Learners*. 22 de jul. de 2020. DOI: 10.48550/arXiv.2005.14165. arXiv: 2005.14165 [cs]. URL: http://arxiv.org/abs/2005.14165. Prepublicado (vid. págs. 31, 37).

[14] Robin Burke. «Hybrid Web Recommender Systems». En: *The Adaptive Web: Methods and Strategies of Web Personalization*. Ed. por Peter Brusilovsky, Alfred Kobsa y Wolfgang Nejdl. Berlin, Heidelberg: Springer, 2007, págs. 377-408. ISBN: 978-3-540-72079-9. DOI: 10.1007/978-3-540-72079-9_12. URL: https://doi.org/10.1007/978-3-540-72079-9_12 (vid. pág. 12).

[15] Òscar Celma. «Music Recommendation». En: *Music Recommendation and Discovery: The Long Tail, Long Fail, and Long Play in the Digital Music Space*. Ed. por Òscar Celma. Berlin, Heidelberg: Springer, 2010, págs. 43-85. ISBN: 978-3-642-13287-2. DOI: 10.1007/978-3-642-13287-2_3. URL: https://doi.org/10.1007/978-3-642-13287-2_3 (vid. pág. 58).

[16] Yupeng Chang et al. *A Survey on Evaluation of Large Language Models*. 28 de dic. de 2023. DOI: 10.48550/arXiv.2307.03109. arXiv: 2307.03109 [cs]. URL: http://arxiv.org/abs/2307.03109. Prepublicado (vid. pág. 38).

[17] Mark Chen et al. *Evaluating Large Language Models Trained on Code*. 14 de jul. de 2021. DOI: 10.48550/arXiv.2107.03374. arXiv: 2107.03374 [cs]. URL: http://arxiv.org/abs/2107.03374. Prepublicado (vid. págs. 39, 41).

[18] Wei-Lin Chiang et al. *Chatbot Arena: An Open Platform for Evaluating LLMs by Human Preference*. 6 de mar. de 2024. DOI: 10.48550/arXiv.2403.04132. arXiv: 2403.04132 [cs]. URL: http://arxiv.org/abs/2403.04132. Prepublicado (vid. págs. 40, 41).

[19] Paul F Christiano et al. «Deep Reinforcement Learning from Human Preferences». En: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017. URL: https://papers.nips.cc/paper_files/paper/2017/hash/d5e2c0adad503c91f91df240d0cd4e49-Abstract.html (vid. pág. 34).

[20] Hyung Won Chung et al. *Scaling Instruction-Finetuned Language Models*. 6 de dic. de 2022. DOI: 10.48550/arXiv.2210.11416. arXiv: 2210.11416 [cs]. URL: http://arxiv.org/abs/2210.11416. Prepublicado (vid. pág. 33).

[21] Karl Cobbe et al. *Training Verifiers to Solve Math Word Problems*. 17 de nov. de 2021. DOI: 10.48550/arXiv.2110.14168. arXiv: 2110.14168 [cs]. URL: http://arxiv.org/abs/2110.14168. Prepublicado (vid. pág. 41).

[22] Paul Covington, Jay Adams y Emre Sargin. «Deep Neural Networks for YouTube Recommendations». En: *Proceedings of the 10th ACM Conference on Recommender Systems*. RecSys '16. New York, NY, USA: Association for Computing Machinery, 7 de sep. de 2016, págs. 191-198. ISBN: 978-1-4503-4035-9. DOI: 10.1145/2959100.2959190. URL: https://doi.org/10.1145/2959100.2959190 (vid. pág. 50).

[23] Tim Dettmers et al. *LLM.Int8(): 8-Bit Matrix Multiplication for Transformers at Scale*. 10 de nov. de 2022. DOI: 10.48550/arXiv.2208.07339. arXiv: 2208.07339 [cs]. URL: http://arxiv.org/abs/2208.07339. Prepublicado (vid. pág. 36).

[24] Tim Dettmers et al. *QLoRA: Efficient Finetuning of Quantized LLMs*. 23 de mayo de 2023. DOI: 10.48550/arXiv.2305.14314. arXiv: 2305.14314 [cs]. URL: http://arxiv.org/abs/2305.14314. Prepublicado (vid. pág. 36).

[25] Jacob Devlin et al. «BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding». En: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. NAACL-HLT 2019. Ed. por Jill Burstein, Christy Doran y Thamar Solorio. Minneapolis, Minnesota: Association for Computational Linguistics, jun. de 2019, págs. 4171-4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423 (vid. págs. 31, 44).

[26] Ning Ding et al. *OpenPrompt: An Open-source Framework for Prompt-learning*. 2 de nov. de 2021. DOI: 10.48550/arXiv.2111.01998. arXiv: 2111.01998 [cs]. URL: http://arxiv.org/abs/2111.01998. Prepublicado (vid. pág. 51).

[27] Yiran Ding et al. *LongRoPE: Extending LLM Context Window Beyond 2 Million Tokens*. 21 de feb. de 2024. DOI: 10.48550/arXiv.2402.13753. arXiv: 2402.13753 [cs]. URL: http://arxiv.org/abs/2402.13753. Prepublicado (vid. pág. 48).

[28] Xinyu Du et al. *Frequency Enhanced Hybrid Attention Network for Sequential Recommendation*. 17 de mayo de 2023. DOI: 10.48550/arXiv.2304.09184. arXiv: 2304.09184 [cs]. URL: http://arxiv.org/abs/2304.09184. Prepublicado (vid. pág. 23).

[29] Yann Dubois et al. *Length-Controlled AlpacaEval: A Simple Way to Debias Automatic Evaluators*. 5 de abr. de 2024. DOI: 10.48550/arXiv.2404.04475. arXiv: 2404.04475 [cs, stat]. URL: http://arxiv.org/abs/2404.04475. Prepublicado (vid. págs. 39, 41).

[30] Hui Fang et al. *Deep Learning for Sequential Recommendation: Algorithms, Influential Factors, and Evaluations*. 10 de oct. de 2020. DOI: 10.48550/arXiv.1905.01997. arXiv: 1905.01997 [cs]. URL: http://arxiv.org/abs/1905.01997. Prepublicado (vid. pág. 22).

[31] Elias Frantar et al. *GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers*. 22 de mar. de 2023. DOI: 10.48550/arXiv.2210.17323. arXiv: 2210.17323 [cs]. URL: http://arxiv.org/abs/2210.17323. Prepublicado (vid. pág. 36).

[32] Yunfan Gao et al. *Chat-REC: Towards Interactive and Explainable LLMs-Augmented Recommender System*. 3 de abr. de 2023. DOI: 10.48550/arXiv.2303.14524. arXiv: 2303.14524 [cs]. URL: http://arxiv.org/abs/2303.14524. Prepublicado (vid. págs. 43, 44, 47).

[33] Gemma Team, Google DeepMind. *Gemma 2: Improving Open Language Models at a Practical Size*. Google, 27 de jun. de 2024. URL: https://storage.googleapis.com/deepmind-media/gemma/gemma-2-report.pdf (vid. págs. 41, 60).

[34] Shijie Geng et al. «Recommendation as Language Processing (RLP): A Unified Pretrain, Personalized Prompt & Predict Paradigm (P5)». En: *Proceedings of the 16th ACM Conference on Recommender Systems*. RecSys '22. New York, NY, USA: Association for Computing Machinery, 13 de sep. de 2022, págs. 299-315. ISBN: 978-1-4503-9278-5. DOI: 10.1145/3523227.3546767. URL: https://doi.org/10.1145/3523227.3546767 (vid. págs. 43-48, 56).

[35] David Goldberg et al. «Using Collaborative Filtering to Weave an Information Tapestry». En: *Communications of the ACM* 35.12 (1 de dic. de 1992), págs. 61-70. ISSN: 0001-0782. DOI: 10.1145/138859.138867. URL: https://dl.acm.org/doi/10.1145/138859.138867 (vid. pág. 11).

[36] Albert Gu y Tri Dao. *Mamba: Linear-Time Sequence Modeling with Selective State Spaces*. 31 de mayo de 2024. DOI: 10.48550/arXiv.2312.00752. arXiv: 2312.00752 [cs]. URL: http://arxiv.org/abs/2312.00752. Prepublicado (vid. pág. 32).

[37] Asela Gunawardana, Guy Shani y Sivan Yogev. «Evaluating Recommender Systems». En: *Recommender Systems Handbook*. Ed. por Francesco Ricci, Lior Rokach y Bracha Shapira. New York, NY: Springer US, 2022, págs. 547-601. ISBN: 978-1-07-162197-4. DOI: 10.1007/978-1-0716-2197-4_15. URL: https://doi.org/10.1007/978-1-0716-2197-4_15 (vid. págs. 24, 26).

[38] Ihsan Gunes et al. «Shilling Attacks against Recommender Systems: A Comprehensive Survey». En: *Artificial Intelligence Review* 42.4 (1 de dic. de 2014), págs. 767-799. ISSN: 1573-7462. DOI: 10.1007/s10462-012-9364-9. URL: https://doi.org/10.1007/s10462-012-9364-9 (vid. pág. 15).

[39] Shibo Hao et al. *Reasoning with Language Model Is Planning with World Model*. 23 de oct. de 2023. DOI: 10.48550/arXiv.2305.14992. arXiv: 2305.14992 [cs]. URL: http://arxiv.org/abs/2305.14992. Prepublicado (vid. pág. 38).

[40] F. Maxwell Harper y Joseph A. Konstan. «The MovieLens Datasets: History and Context». En: *ACM Transactions on Interactive Intelligent Systems* 5.4 (22 de dic. de 2015), 19:1-19:19. ISSN: 2160-6455. DOI: 10.1145/2827872. URL: https://doi.org/10.1145/2827872 (vid. págs. 17, 51, 52, 54).

[41] Ruining He y Julian McAuley. *Fusing Similarity Models with Markov Chains for Sparse Sequential Recommendation*. 28 de sep. de 2016. DOI: 10.48550/arXiv.1609.09152. arXiv: 1609.09152 [cs]. URL: http://arxiv.org/abs/1609.09152. Prepublicado (vid. pág. 22).

[42] Xiangnan He et al. *LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation*. 7 de jul. de 2020. DOI: 10.48550/arXiv.2002.02126. arXiv: 2002.02126 [cs]. URL: http://arxiv.org/abs/2002.02126. Prepublicado (vid. pág. 18).

[43] Xiangnan He et al. «Neural Collaborative Filtering». En: *Proceedings of the 26th International Conference on World Wide Web*. WWW '17. Republic y Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 3 de abr. de 2017, págs. 173-182. ISBN: 978-1-4503-4913-0. DOI: 10.1145/3038912.3052569. URL: https://doi.org/10.1145/3038912.3052569 (vid. pág. 18).

[44] Dan Hendrycks et al. *Measuring Massive Multitask Language Understanding*. 12 de ene. de 2021. DOI: 10.48550/arXiv.2009.03300. arXiv: 2009.03300 [cs]. URL: http://arxiv.org/abs/2009.03300. Prepublicado (vid. págs. 39, 41).

[45] Dan Hendrycks et al. *Measuring Mathematical Problem Solving With the MATH Dataset*. 8 de nov. de 2021. DOI: 10.48550/arXiv.2103.03874. arXiv: 2103.03874 [cs]. URL: http://arxiv.org/abs/2103.03874. Prepublicado (vid. pág. 39).

[46] Balázs Hidasi y Ádám Tibor Czapp. «The Effect of Third Party Implementations on Reproducibility». En: *Proceedings of the 17th ACM Conference on Recommender Systems*. RecSys '23. New York, NY, USA: Association for Computing Machinery, 14 de sep. de 2023, págs. 272-282. DOI: 10.1145/3604915.3609487. URL: https://doi.org/10.1145/3604915.3609487 (vid. pág. 61).

[47] Balázs Hidasi et al. *Session-Based Recommendations with Recurrent Neural Networks*. 29 de mar. de 2016. DOI: 10.48550/arXiv.1511.06939. arXiv: 1511.06939 [cs]. URL: http://arxiv.org/abs/1511.06939. Prepublicado (vid. pág. 22).

[48] Sepp Hochreiter y Jürgen Schmidhuber. «Long Short-Term Memory». En: *Neural Comput.* 9.8 (1 de nov. de 1997), págs. 1735-1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735 (vid. pág. 31).

[49] Jiwoo Hong, Noah Lee y James Thorne. *ORPO: Monolithic Preference Optimization without Reference Model*. 14 de mar. de 2024. DOI: 10.48550/arXiv.2403.07691. arXiv: 2403.07691 [cs]. URL: http://arxiv.org/abs/2403.07691. Prepublicado (vid. pág. 35).

[50] Yupeng Hou et al. *CORE: Simple and Effective Session-based Recommendation within Consistent Representation Space*. 23 de abr. de 2022. DOI: 10.48550/arXiv.2204.11067. arXiv: 2204.11067 [cs]. URL: http://arxiv.org/abs/2204.11067. Prepublicado (vid. pág. 23).

[51] Neil Houlsby et al. *Parameter-Efficient Transfer Learning for NLP*. 13 de jun. de 2019. DOI: 10.48550/arXiv.1902.00751. arXiv: 1902.00751 [cs, stat]. URL: http://arxiv.org/abs/1902.00751. Prepublicado (vid. pág. 35).

[52] Edward J. Hu et al. *LoRA: Low-Rank Adaptation of Large Language Models*. 16 de oct. de 2021. DOI: 10.48550/arXiv.2106.09685. arXiv: 2106.09685 [cs]. URL: http://arxiv.org/abs/2106.09685. Prepublicado (vid. pág. 35).

[53] Yifan Hu, Yehuda Koren y Chris Volinsky. «Collaborative Filtering for Implicit Feedback Datasets». En: *2008 Eighth IEEE International Conference on Data Mining*. 2008 Eighth IEEE International Conference on Data Mining. Dic. de 2008, págs. 263-272. DOI: 10.1109/ICDM.2008.22. URL: https://ieeexplore.ieee.org/document/4781121 (vid. pág. 18).

[54] Wenyue Hua et al. «Tutorial on Large Language Models for Recommendation». En: *Proceedings of the 17th ACM Conference on Recommender Systems*. RecSys '23. New York, NY, USA: Association for Computing Machinery, 14 de sep. de 2023, págs. 1281-1283. DOI: 10.1145/3604915.3609494. URL: https://doi.org/10.1145/3604915.3609494 (vid. pág. 43).

[55] Wenyue Hua et al. «UP5: Unbiased Foundation Model for Fairness-aware Recommendation». En: *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*. EACL 2024. Ed. por Yvette Graham y Matthew Purver. St. Julian's, Malta: Association for Computational Linguistics, mar. de 2024, págs. 1899-1912. URL: https://aclanthology.org/2024.eacl-long.114 (vid. pág. 48).

[56] Dietmar Jannach, Massimo Quadrana y Paolo Cremonesi. «Session-Based Recommender Systems». En: *Recommender Systems Handbook*. Ed. por Francesco Ricci, Lior Rokach y Bracha Shapira. New York, NY: Springer US, 2022, págs. 301-334. ISBN: 978-1-07-162197-4. DOI: 10.1007/978-1-0716-2197-4_8. URL: https://doi.org/10.1007/978-1-0716-2197-4_8 (vid. pág. 19).

[57] Albert Q. Jiang et al. *Mistral 7B*. 10 de oct. de 2023. DOI: 10.48550/arXiv.2310.06825. arXiv: 2310.06825 [cs]. URL: http://arxiv.org/abs/2310.06825. Prepublicado (vid. págs. 41, 59).

[58] Wang-Cheng Kang y Julian McAuley. «Self-Attentive Sequential Recommendation». En: 2018 IEEE International Conference on Data Mining (ICDM). IEEE Computer Society, 1 de nov. de 2018, págs. 197-206. ISBN: 978-1-5386-9159-5. DOI: 10.1109/ICDM.2018.00035. URL: https://www.computer.org/csdl/proceedings-article/icdm/2018/08594844/17D45Xq6dBh (vid. págs. 23, 54).

[59] Jared Kaplan et al. *Scaling Laws for Neural Language Models*. 22 de ene. de 2020. DOI: 10.48550/arXiv.2001.08361. arXiv: 2001.08361 [cs, stat]. URL: http://arxiv.org/abs/2001.08361. Prepublicado (vid. pág. 31).

[60] Sein Kim et al. *Large Language Models Meet Collaborative Filtering: An Efficient All-round LLM-based Recommender System*. 1 de jun. de 2024. DOI: 10.48550/arXiv.2404.11343. arXiv: 2404.11343 [cs]. URL: http://arxiv.org/abs/2404.11343. Prepublicado (vid. pág. 69).

[61] Takeshi Kojima et al. *Large Language Models Are Zero-Shot Reasoners*. 29 de ene. de 2023. DOI: 10.48550/arXiv.2205.11916. arXiv: 2205.11916 [cs]. URL: http://arxiv.org/abs/2205.11916. Prepublicado (vid. pág. 38).

[62] Maxime Labonne. *Introduction to Weight Quantization*. Medium. 7 de jul. de 2023. URL: https://towardsdatascience.com/introduction-to-weight-quantization-2494701b9c0c (vid. pág. 36).

[63] Brian Lester, Rami Al-Rfou y Noah Constant. *The Power of Scale for Parameter-Efficient Prompt Tuning*. 2 de sep. de 2021. DOI: 10.48550/arXiv.2104.08691. arXiv: 2104.08691 [cs]. URL: http://arxiv.org/abs/2104.08691. Prepublicado (vid. pág. 35).

[64] Jing Li et al. «Neural Attentive Session-based Recommendation». En: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. CIKM '17. New York, NY, USA: Association for Computing Machinery, 6 de nov. de 2017, págs. 1419-1428. ISBN: 978-1-4503-4918-5. DOI: 10.1145/3132847.3132926. URL: https://doi.org/10.1145/3132847.3132926 (vid. págs. 23, 54).

[65] Jinming Li et al. *GPT4Rec: A Generative Framework for Personalized Recommendation and User Interests Interpretation*. 7 de abr. de 2023. DOI: 10.48550/arXiv.2304.03879. arXiv: 2304.03879 [cs]. URL: http://arxiv.org/abs/2304.03879. Prepublicado (vid. págs. 43, 47, 56).

[66] Lei Li, Yongfeng Zhang y Li Chen. *Personalized Prompt Learning for Explainable Recommendation*. 13 de ene. de 2023. DOI: 10.48550/arXiv.2202.07371. arXiv: 2202.07371 [cs]. URL: http://arxiv.org/abs/2202.07371. Prepublicado (vid. pág. 42).

[67] Lei Li, Yongfeng Zhang y Li Chen. «Prompt Distillation for Efficient LLM-based Recommendation». En: *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. CIKM '23. New York, NY, USA: Association for Computing Machinery, 21 de oct. de 2023, págs. 1348-1357. DOI: 10.1145/3583780.3615017. URL: https://doi.org/10.1145/3583780.3615017 (vid. págs. 43, 47, 56).

[68] Xiang Lisa Li y Percy Liang. «Prefix-Tuning: Optimizing Continuous Prompts for Generation». En: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. ACL-IJCNLP 2021. Ed. por Chengqing Zong et al. Online: Association for Computational Linguistics, ago. de 2021, págs. 4582-4597. DOI: 10.18653/v1/2021.acl-long.353. URL: https://aclanthology.org/2021.acl-long.353 (vid. pág. 35).

[69] Dawen Liang et al. «Variational Autoencoders for Collaborative Filtering». En: *Proceedings of the 2018 World Wide Web Conference*. WWW '18. Republic y Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 23 de abr. de 2018, págs. 689-698. ISBN: 978-1-4503-5639-8. DOI: 10.1145/3178876.3186150. URL: https://dl.acm.org/doi/10.1145/3178876.3186150 (vid. pág. 18).

[70] Ji Lin et al. *AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration*. 23 de abr. de 2024. DOI: 10.48550/arXiv.2306.00978. arXiv: 2306.00978 [cs]. URL: http://arxiv.org/abs/2306.00978. Prepublicado (vid. pág. 36).

[71] Jianghao Lin et al. *How Can Recommender Systems Benefit from Large Language Models: A Survey*. 2 de feb. de 2024. DOI: 10.48550/arXiv.2306.05817. arXiv: 2306.05817 [cs]. URL: http://arxiv.org/abs/2306.05817. Prepublicado (vid. págs. 44, 47).

[72] Zihan Lin et al. «Improving Graph Collaborative Filtering with Neighborhood-enriched Contrastive Learning». En: *Proceedings of the ACM Web Conference 2022*. 25 de abr. de 2022, págs. 2320-2329. DOI: 10.1145/3485447.3512104. arXiv: 2202.06200 [cs]. URL: http://arxiv.org/abs/2202.06200 (vid. pág. 18).

[73] G. Linden, B. Smith y J. York. «Amazon.Com Recommendations: Item-to-Item Collaborative Filtering». En: *IEEE Internet Computing* 7.1 (ene. de 2003), págs. 76-80. ISSN: 1941-0131. DOI: 10.1109/MIC.2003.1167344. URL: https://ieeexplore.ieee.org/document/1167344 (vid. pág. 17).

[74] Vivian Liu y Lydia B Chilton. «Design Guidelines for Prompt Engineering Text-to-Image Generative Models». En: *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. CHI '22. New York, NY, USA: Association for Computing Machinery, 29 de abr. de 2022, págs. 1-23. ISBN: 978-1-4503-9157-3. DOI: 10.1145/3491102.3501825. URL: https://doi.org/10.1145/3491102.3501825 (vid. pág. 37).

[75] Zechun Liu et al. *LLM-QAT: Data-Free Quantization Aware Training for Large Language Models*. 29 de mayo de 2023. DOI: 10.48550/arXiv.2305.17888. arXiv: 2305.17888 [cs]. URL: http://arxiv.org/abs/2305.17888. Prepublicado (vid. pág. 36).

[76] Shervin Minaee et al. *Large Language Models: A Survey*. 20 de feb. de 2024. DOI: 10.48550/arXiv.2402.06196. arXiv: 2402.06196 [cs]. URL: http://arxiv.org/abs/2402.06196. Prepublicado (vid. pág. 30).

[77] Jianmo Ni, Jiacheng Li y Julian McAuley. «Justifying Recommendations Using Distantly-Labeled Reviews and Fine-Grained Aspects». En: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. EMNLP-IJCNLP 2019. Ed. por Kentaro Inui et al. Hong Kong, China: Association for Computational Linguistics, nov. de 2019, págs. 188-197. DOI: 10.18653/v1/D19-1018. URL: https://aclanthology.org/D19-1018 (vid. págs. 51, 54).

[78] Xia Ning y George Karypis. «SLIM: Sparse Linear Methods for Top-N Recommender Systems». En: *2011 IEEE 11th International Conference on Data Mining*. 2011 IEEE 11th International Conference on Data Mining. Dic. de 2011, págs. 497-506. DOI: 10.1109/ICDM.2011.134. URL: https://ieeexplore.ieee.org/document/6137254 (vid. pág. 17).

[79] OpenAI. *Hello GPT-4o*. 13 de mayo de 2024. URL: https://openai.com/index/hello-gpt-4o/ (vid. pág. 41).

[80] OpenAI et al. *GPT-4 Technical Report*. Ver. 6. 2023. DOI: 10.48550/ARXIV.2303.08774. URL: https://arxiv.org/abs/2303.08774. Prepublicado (vid. págs. 30, 34).

[81] Antonio Orvieto et al. *Resurrecting Recurrent Neural Networks for Long Sequences*. 11 de mar. de 2023. DOI: 10.48550/arXiv.2303.06349. arXiv: 2303.06349 [cs]. URL: http://arxiv.org/abs/2303.06349. Prepublicado (vid. pág. 51).

[82] Long Ouyang et al. *Training Language Models to Follow Instructions with Human Feedback*. 4 de mar. de 2022. DOI: 10.48550/arXiv.2203.02155. arXiv: 2203.02155 [cs]. URL: http://arxiv.org/abs/2203.02155. Prepublicado (vid. págs. 33, 34).

[83] Bibek Paudel et al. «Updatable, Accurate, Diverse, and Scalable Recommendations for Interactive Applications». En: *ACM Transactions on Interactive Intelligent Systems* 7.1 (19 de dic. de 2016), 1:1-1:34. ISSN: 2160-6455. DOI: 10.1145/2955101. URL: https://doi.org/10.1145/2955101 (vid. pág. 17).

[84] Bo Peng et al. *RWKV: Reinventing RNNs for the Transformer Era*. 10 de dic. de 2023. DOI: 10.48550/arXiv.2305.13048. arXiv: 2305.13048 [cs]. URL: http://arxiv.org/abs/2305.13048. Prepublicado (vid. pág. 32).

[85] Aleksandr Petrov y Craig Macdonald. «A Systematic Review and Replicability Study of BERT4Rec for Sequential Recommendation». En: *Proceedings of the 16th ACM Conference on Recommender Systems*. RecSys '22. New York, NY, USA: Association for Computing Machinery, 13 de sep. de 2022, págs. 436-447. ISBN: 978-1-4503-9278-5. DOI: 10.1145/3523227.3548487. URL: https://doi.org/10.1145/3523227.3548487 (vid. págs. 61, 65).

[86] Zhaopeng Qiu et al. «U-BERT: Pre-training User Representations for Improved Recommendation». En: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.5 (5 18 de mayo de 2021), págs. 4320-4327. ISSN: 2374-3468. DOI: 10.1609/aaai.v35i5.16557. URL: https://ojs.aaai.org/index.php/AAAI/article/view/16557 (vid. pág. 44).

[87] Massimo Quadrana, Paolo Cremonesi y Dietmar Jannach. *Sequence-Aware Recommender Systems*. 23 de feb. de 2018. DOI: 10.48550/arXiv.1802.08452. arXiv: 1802.08452 [cs]. URL: http://arxiv.org/abs/1802.08452. Prepublicado (vid. págs. 19, 20).

[88] Massimo Quadrana et al. «Personalizing Session-based Recommendations with Hierarchical Recurrent Neural Networks». En: *Proceedings of the Eleventh ACM Conference on Recommender Systems*. RecSys '17. New York, NY, USA: Association for Computing Machinery, 27 de ago. de 2017, págs. 130-137. ISBN: 978-1-4503-4652-8. DOI: 10.1145/3109859.3109896. URL: https://doi.org/10.1145/3109859.3109896 (vid. pág. 22).

[89] Qwen. *Hello Qwen2*. Qwen. 7 de jun. de 2024. URL: http://qwenlm.github.io/blog/qwen2/ (vid. pág. 41).

[90] Alec Radford et al. «Language Models Are Unsupervised Multitask Learners». En: 2019. URL: https://www.semanticscholar.org/paper/Language-Models-are-Unsupervised-Multitask-Learners-Radford-Wu/9405cc0d6169988371b2755e573cc28650d14dfe (vid. pág. 57).

[91] Rafael Rafailov et al. *Direct Preference Optimization: Your Language Model Is Secretly a Reward Model*. Ver. 2. 2023. DOI: 10.48550/ARXIV.2305.18290. URL: https://arxiv.org/abs/2305.18290. Prepublicado (vid. págs. 34, 35).

[92] Colin Raffel et al. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. 19 de sep. de 2023. DOI: 10.48550/arXiv.1910.10683. arXiv: 1910.10683 [cs, stat]. URL: http://arxiv.org/abs/1910.10683. Prepublicado (vid. pág. 32).

[93] Steffen Rendle, Christoph Freudenthaler y Lars Schmidt-Thieme. «Factorizing Personalized Markov Chains for Next-Basket Recommendation». En: *Proceedings of the 19th International Conference on World Wide Web*. WWW '10. New York, NY, USA: Association for Computing Machinery, 26 de abr. de 2010, págs. 811-820. ISBN: 978-1-60558-799-8. DOI: 10.1145/1772690.1772773. URL: https://doi.org/10.1145/1772690.1772773 (vid. pág. 22).

[94] Steffen Rendle et al. «BPR: Bayesian Personalized Ranking from Implicit Feedback». En: *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. UAI '09. Arlington, Virginia, USA: AUAI Press, 18 de jun. de 2009, págs. 452-461. ISBN: 978-0-9749039-5-8 (vid. pág. 18).

[95] Steffen Rendle et al. «Neural Collaborative Filtering vs. Matrix Factorization Revisited». En: *Proceedings of the 14th ACM Conference on Recommender Systems*. RecSys '20. New York, NY, USA: Association for Computing Machinery, 22 de sep. de 2020, págs. 240-248. ISBN: 978-1-4503-7583-2. DOI: 10.1145/3383313.3412488. URL: https://dl.acm.org/doi/10.1145/3383313.3412488 (vid. pág. 18).

[96] Paul Resnick y Hal R. Varian. «Recommender Systems». En: *Communications of the ACM* 40.3 (1 de mar. de 1997), págs. 56-58. ISSN: 0001-0782. DOI: 10.1145/245108.245121. URL: https://dl.acm.org/doi/10.1145/245108.245121 (vid. pág. 11).

[97] Paul Resnick et al. «GroupLens: An Open Architecture for Collaborative Filtering of Netnews». En: *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*. CSCW '94. New York, NY, USA: Association for Computing Machinery, 22 de oct. de 1994, págs. 175-186. ISBN: 978-0-89791-689-9. DOI: 10.1145/192844.192905. URL: https://dl.acm.org/doi/10.1145/192844.192905 (vid. pág. 17).

[98] Francesco Ricci, Lior Rokach y Bracha Shapira. «Recommender Systems: Techniques, Applications, and Challenges». En: *Recommender Systems Handbook*. Ed. por Francesco Ricci, Lior Rokach y Bracha Shapira. New York, NY: Springer US, 2022, págs. 1-35. ISBN: 978-1-07-162197-4. DOI: 10.1007/978-1-0716-2197-4_1. URL: https://doi.org/10.1007/978-1-0716-2197-4_1 (vid. págs. 11, 12).

[99] Deepjyoti Roy y Mala Dutta. «A Systematic Review and Research Perspective on Recommender Systems». En: *Journal of Big Data* 9.1 (1 dic. de 2022), págs. 1-36. ISSN: 2196-1115. DOI: 10.1186/s40537-022-00592-5. URL: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-022-00592-5 (vid. pág. 15).

[100] Badrul Sarwar et al. «Item-Based Collaborative Filtering Recommendation Algorithms». En: *Proceedings of the 10th International Conference on World Wide Web*. WWW '01. New York, NY, USA: Association for Computing Machinery, 1 de abr. de 2001, págs. 285-295. ISBN: 978-1-58113-348-6. DOI: 10.1145/371920.372071. URL: https://doi.org/10.1145/371920.372071 (vid. pág. 17).

[101] Markus Schedl. «The LFM-1b Dataset for Music Retrieval and Recommendation». En: *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*. ICMR '16. New York, NY, USA: Association for Computing Machinery, 6 de jun. de 2016, págs. 103-110. ISBN: 978-1-4503-4359-6. DOI: 10.1145/2911996.2912004. URL: https://doi.org/10.1145/2911996.2912004 (vid. pág. 58).

[102] Markus Schedl et al. «LFM-2b: A Dataset of Enriched Music Listening Events for Recommender Systems Research and Fairness Analysis». En: *Proceedings of the 2022 Conference on Human Information Interaction and Retrieval*. CHIIR '22. New York, NY, USA: Association for Computing Machinery, 14 de mar. de 2022, págs. 337-341. ISBN: 978-1-4503-9186-3. DOI: 10.1145/3498366.3505791. URL: https://doi.org/10.1145/3498366.3505791 (vid. pág. 58).

[103] Markus Schedl et al. «Music Recommendation Systems: Techniques, Use Cases, and Challenges». En: *Recommender Systems Handbook*. Ed. por Francesco Ricci, Lior Rokach y Bracha Shapira. New York, NY: Springer US, 2022, págs. 927-971. ISBN: 978-1-07-162197-4. DOI: 10.1007/978-1-0716-2197-4_24. URL: https://doi.org/10.1007/978-1-0716-2197-4_24 (vid. pág. 57).

[104] John Schulman et al. *Proximal Policy Optimization Algorithms*. Ver. 2. 2017. DOI: 10.48550/ARXIV.1707.06347. URL: https://arxiv.org/abs/1707.06347. Prepublicado (vid. pág. 34).

[105] Karan Singhal et al. «Large Language Models Encode Clinical Knowledge». En: *Nature* 620.7972 (ago. de 2023), págs. 172-180. ISSN: 1476-4687. DOI: 10.1038/s41586-023-06291-2. URL: https://www.nature.com/articles/s41586-023-06291-2 (vid. pág. 33).

[106] Aarohi Srivastava et al. *Beyond the Imitation Game: Quantifying and Extrapolating the Capabilities of Language Models*. 12 de jun. de 2023. DOI: 10.48550/arXiv.2206.04615. arXiv: 2206.04615 [cs, stat]. URL: http://arxiv.org/abs/2206.04615. Prepublicado (vid. págs. 39, 41).

[107] Harald Steck. «Embarrassingly Shallow Autoencoders for Sparse Data». En: *The World Wide Web Conference*. WWW '19. New York, NY, USA: Association for Computing Machinery, 13 de mayo de 2019, págs. 3251-3257. ISBN: 978-1-4503-6674-8. DOI: 10.1145/3308558.3313710. URL: https://doi.org/10.1145/3308558.3313710 (vid. pág. 17).

[108] Fei Sun et al. «BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer». En: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. CIKM '19. New York, NY, USA: Association for Computing Machinery, 3 de nov. de 2019, págs. 1441-1450. ISBN: 978-1-4503-6976-3. DOI: 10.1145/3357384.335

7895. URL: https://doi.org/10.1145/3357384.3357895 (vid. págs. 23, 54, 65).

[109] Jiaxi Tang y Ke Wang. «Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding». En: *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. WSDM '18. New York, NY, USA: Association for Computing Machinery, 2 de feb. de 2018, págs. 565-573. ISBN: 978-1-4503-5581-0. DOI: 10.1145/3159652.3159656. URL: https://doi.org/10.1145/3159652.3159656 (vid. pág. 23).

[110] Wilson L. Taylor. «Cloze Procedure: A New Tool for Measuring Readability». En: *Journalism Quarterly* 30.4 (1 de sep. de 1953), págs. 415-433. ISSN: 0022-5533. DOI: 10.1177/107769905303000401. URL: https://doi.org/10.1177/107769905303000401 (vid. pág. 23).

[111] Gemini Team et al. *Gemini 1.5: Unlocking Multimodal Understanding across Millions of Tokens of Context*. 14 de jun. de 2024. DOI: 10.48550/arXiv.2403.05530. arXiv: 2403.05530 [cs]. URL: http://arxiv.org/abs/2403.05530. Prepublicado (vid. pág. 41).

[112] Gemma Team et al. *Gemma: Open Models Based on Gemini Research and Technology*. 16 de abr. de 2024. DOI: 10.48550/arXiv.2403.08295. arXiv: 2403.08295 [cs]. URL: http://arxiv.org/abs/2403.08295. Prepublicado (vid. págs. 41, 60, 70).

[113] Hugo Touvron et al. *Llama 2: Open Foundation and Fine-Tuned Chat Models*. 19 de jul. de 2023. DOI: 10.48550/arXiv.2307.09288. arXiv: 2307.09288 [cs]. URL: http://arxiv.org/abs/2307.09288. Prepublicado (vid. pág. 41).

[114] R. Turrin et al. «30Music Listening and Playlists Dataset». En: RecSys Posters. 2015. URL: https://www.semanticscholar.org/paper/30Music-Listening-and-Playlists-Dataset-Turrin-Quadrana/b6bb738a5fab294e9b4c8b6d152be32c7f01d154 (vid. pág. 58).

[115] Ashish Vaswani et al. «Attention Is All You Need». En: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017. URL: https://papers.nips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html (vid. págs. 23, 31).

[116] Pengfei Wang et al. «Learning Hierarchical Representation Model for NextBasket Recommendation». En: *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '15. New York, NY, USA: Association for Computing Machinery, 9 de ago. de 2015, págs. 403-412. ISBN: 978-1-4503-3621-5. DOI: 10.1145/2766462.2767694. URL: https://doi.org/10.1145/2766462.2767694 (vid. pág. 24).

[117] Shoujin Wang et al. «Sequential Recommender Systems: Challenges, Progress and Prospects». En: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. Ago. de 2019, págs. 6332-6338. DOI: 10.24963/ijcai.2019/883. arXiv: 2001.04830 [cs]. URL: http://arxiv.org/abs/2001.04830 (vid. pág. 19).

[118] Wenjie Wang et al. *Diffusion Recommender Model*. 17 de abr. de 2023. DOI: 10.48550/arXiv.2304.04971. arXiv: 2304.04971 [cs]. URL: http://arxiv.org/abs/2304.04971. Prepublicado (vid. pág. 19).

[119] Yancheng Wang et al. *RecMind: Large Language Model Powered Agent For Recommendation*. 20 de mar. de 2024. DOI: 10.48550/arXiv.2308.14296. arXiv: 2308.14296 [cs]. URL: http://arxiv.org/abs/2308.14296. Prepublicado (vid. págs. 43, 46, 47, 56).

[120] Yizhong Wang et al. *Self-Instruct: Aligning Language Models with Self-Generated Instructions*. 25 de mayo de 2023. DOI: 10.48550/arXiv.2212.10560. arXiv: 2212.10560 [cs]. URL: http://arxiv.org/abs/2212.10560. Prepublicado (vid. pág. 33).

[121] Jason Wei et al. *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. 10 de ene. de 2023. DOI: 10.48550/arXiv.2201.11903. arXiv: 2201.11903 [cs]. URL: http://arxiv.org/abs/2201.11903. Prepublicado (vid. pág. 38).

[122] Jason Wei et al. *Emergent Abilities of Large Language Models*. 26 de oct. de 2022. DOI: 10.48550/arXiv.2206.07682. arXiv: 2206.07682 [cs]. URL: http://arxiv.org/abs/2206.07682. Prepublicado (vid. pág. 31).

[123] Jason Wei et al. *Finetuned Language Models Are Zero-Shot Learners*. 8 de feb. de 2022. DOI: 10.48550/arXiv.2109.01652. arXiv: 2109.01652 [cs]. URL: http://arxiv.org/abs/2109.01652. Prepublicado (vid. págs. 32, 33).

[124] Thomas Wolf et al. *HuggingFace's Transformers: State-of-the-art Natural Language Processing*. 13 de jul. de 2020. DOI: 10.48550/arXiv.1910.03771. arXiv: 1910.03771 [cs]. URL: http://arxiv.org/abs/1910.03771. Prepublicado (vid. pág. 59).

[125] Cameron R. Wolfe. *Understanding and Using Supervised Fine-Tuning (SFT) for Language Models*. Deep (Learning) Focus. 11 de sep. de 2023. URL: https://cameronrwolfe.substack.com/p/understanding-and-using-supervised (vid. pág. 32).

[126] Jiancan Wu et al. «Self-Supervised Graph Learning for Recommendation». En: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 11 de jul. de 2021, págs. 726-735. DOI: 10.1145/3404835.3462862. arXiv: 2010.10783 [cs]. URL: http://arxiv.org/abs/2010.10783 (vid. pág. 18).

[127] Likang Wu et al. *A Survey on Large Language Models for Recommendation*. 18 de ago. de 2023. DOI: 10.48550/arXiv.2305.19860. arXiv: 2305.19860 [cs]. URL: http://arxiv.org/abs/2305.19860. Prepublicado (vid. págs. 43, 46).

[128] Shu Wu et al. «Session-Based Recommendation with Graph Neural Networks». En: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01 (17 de jul. de 2019), págs. 346-353. ISSN: 2374-3468, 2159-5399. DOI: 10.1609/aaai.v33i01.3301346. arXiv: 1811.00855 [cs, stat]. URL: http://arxiv.org/abs/1811.00855 (vid. pág. 24).

[129] Yunjia Xi et al. *Towards Open-World Recommendation with Knowledge Augmentation from Large Language Models*. 4 de dic. de 2023. DOI: 10.48550/arXiv.2306.10933. arXiv: 2306.10933 [cs]. URL: http://arxiv.org/abs/2306.10933. Prepublicado (vid. pág. 44).

[130] Mingming Xu, Fangai Liu y Weizhi Xu. «A Survey on Sequential Recommendation». En: *2019 6th International Conference on Information Science and Control Engineering (ICISCE)*. 2019 6th International Conference on Information Science and Control Engineering (ICISCE). Dic. de 2019, págs. 106-111. DOI: 10.1109/ICISCE48695.2019.00031. URL: https://ieeexplore.ieee.org/document/9107897 (vid. pág. 20).

[131] Fan Yang et al. *PALR: Personalization Aware LLMs for Recommendation*. 7 de jun. de 2023. DOI: 10.48550/arXiv.2305.07622. arXiv: 2305.07622 [cs]. URL: http://arxiv.org/abs/2305.07622. Prepublicado (vid. págs. 43, 47, 56).

[132] Jingfeng Yang et al. «Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond». En: *ACM Trans. Knowl. Discov. Data* 18.6 (26 de abr. de 2024), 160:1-160:32. ISSN: 1556-4681. DOI: 10.1145/3649506. URL: https://doi.org/10.1145/3649506 (vid. págs. 40, 41).

[133] Fajie Yuan et al. «A Simple Convolutional Generative Network for Next Item Recommendation». En: *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. WSDM '19. New York, NY, USA: Association for Computing Machinery, 30 de ene. de 2019, págs. 582-590. ISBN: 978-1-4503-5940-5. DOI: 10.1145/3289600.3290975. URL: https://doi.org/10.1145/3289600.3290975 (vid. pág. 23).

[134] Zhenrui Yue et al. *Linear Recurrent Units for Sequential Recommendation*. 8 de nov. de 2023. DOI: 10.48550/arXiv.2310.02367. arXiv: 2310.02367 [cs]. URL: http://arxiv.org/abs/2310.02367. Prepublicado (vid. págs. 45, 51, 55, 62).

[135] Zhenrui Yue et al. *LlamaRec: Two-Stage Recommendation Using Large Language Models for Ranking*. 25 de oct. de 2023. DOI: 10.48550/arXiv.2311.02089. arXiv: 2311.02089 [cs]. URL: http://arxiv.org/abs/2311.02089. Prepublicado (vid. págs. 8, 43, 45, 49, 50, 53, 63).

[136] Jizhi Zhang et al. «Is ChatGPT Fair for Recommendation? Evaluating Fairness in Large Language Model Recommendation». En: *Proceedings of the 17th ACM Conference on Recommender Systems*. RecSys '23. New York, NY, USA: Association for Computing Machinery, 14 de sep. de 2023, págs. 993-999. DOI: 10.1145/3604915.3608860. URL: https://doi.org/10.1145/3604915.3608860 (vid. pág. 48).

[137] Wayne Xin Zhao et al. *A Survey of Large Language Models*. 24 de nov. de 2023. DOI: 10.48550/arXiv.2303.18223. arXiv: 2303.18223 [cs]. URL: http://arxiv.org/abs/2303.18223. Prepublicado (vid. págs. 32, 35, 38).

[138] Lianmin Zheng et al. *Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena*. 23 de dic. de 2023. DOI: 10.48550/arXiv.2306.05685. arXiv: 2306.05685 [cs]. URL: http://arxiv.org/abs/2306.05685. Prepublicado (vid. pág. 39).

[139] Kun Zhou et al. «S^3-Rec: Self-Supervised Learning for Sequential Recommendation with Mutual Information Maximization». En: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 19 de oct. de 2020, págs. 1893-1902. DOI: 10.1145/3340531.3411954. arXiv: 2008.07873 [cs]. URL: http://arxiv.org/abs/2008.07873 (vid. pág. 23).