UNIVERSIDAD AUTONOMA DE MADRID

Escuela Politécnica Superior

22|23

Trabajo fin de grado

Estudio de algoritmos de aprendizaje profundo en predicción de compatibilidad de moda

Paula Polo Cabas

Escuela Politécnica Superior
Universidad Autónoma de Madrid
C\Francisco Tomás y Valiente nº 11

excelencia Campus Internacional UAM CSIC

www.uam.es

UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Estudio de algoritmos de aprendizaje profundo en predicción de compatibilidad de moda

Autor: Paula Polo Cabas
Tutor: Alejandro Bellogín Kouki

julio 2023

Paula Polo Cabas
Estudio de algoritmos de aprendizaje profundo en predicción de compatibilidad de moda

Paula Polo Cabas
C\ Francisco Tomás y Valiente Nº 11

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

Agradecimientos

En primer lugar quería agradecer a mi tutor de este Trabajo de Fin de Grado su paciencia, esfuerzo y dedicación. No ha sido un camino fácil, ni corto, pero hemos formado un buen equipo y ha sido un placer tenerle como mentor y guia en este proyecto.

Agradecer el apoyo a mis amigas, que jamás dudaron de mí, a mi novio, que me ha ayudado y alentado durante todo este proceso. Y por supuesto, a mi familia, especialmente a mis padres, que han hecho de mí la persona luchadora que soy hoy, sin ellos nada de esto hubiera sido posible.

Resumen

A raíz de la pandemia reciente, los consumidores se han vuelto más propensos a realizar las compras en línea, por lo que la industria de la moda ha sufrido una metamorfosis que ha obligado tanto a grandes como pequeñas empresas a transformarse digitalmente. La adaptación y la innovación son clave para que las marcas de moda puedan sobrevivir y prosperar en este nuevo entorno. Es por ello que buscan formas de mejorar la experiencia de unos clientes cada vez más exigentes, ayudándoles a encontrar los productos que necesita de una forma fácil y rápida. Es aquí donde entran en juego los sistemas de recomendación, que en este ámbito trabajan junto con los sistemas de predicción de compatibilidad para ofrecer opciones de artículos en el proceso de compra.

En este Trabajo de Fin de Grado trataremos los sistemas de predicción de compatibilidad visual, y el uso de Redes Convolucionales de Grafos que permiten tener en cuenta el contexto de cada producto dentro de la predicción. El campo de la moda es diferente con respecto a otros, ya que los productos tienen unas características visuales, e incluso subjetivas, que son imprescindibles a la hora de realizar la tarea de predicción de compatibilidad.

Para el desarrollo de este proyecto se ha llevado a cabo una investigación en la que se ha estudiado un algoritmo reciente que utiliza la información de productos y sus relaciones para determinar las mejores compatibilidades, incluyendo sus atributos visuales.

Como conclusión, se ha logrado constatar cómo el contexto, entendido como prendas compatibles, en el que estña enmarcado un artículo de moda, es sumamente importante para la mejora en la precisión de la predicción de compatibilidad entre este y otro artículo que no esté previamente relacionado.

Palabras clave

Sistemas de recomendación, Moda, Compatibilidad, Atuendo

Abstract

As a result of the recent pandemic, consumers have become more inclined to make online purchases, causing the fashion industry to undergo a metamorphosis that has forced both large and small companies to transform digitally. Adaptation and innovation are key for fashion brands to survive and thrive in this new environment. That is why they are seeking ways to enhance the experience of increasingly demanding customers, helping them find the products they need easily and quickly. This is where recommender systems come into play, working together with compatibility prediction systems to offer product options during the purchasing process.

In this Bachelor's Thesis, we will explore visual compatibility prediction systems and the use of Graph Convolutional Networks, which allow taking into account the context of each product within the prediction. The fashion field is different from others, since products have visual and even subjective characteristics that are crucial for compatibility prediction tasks.

For the development of this project, research has been conducted on a recent algorithm that uses product information and relationships to determine the best compatibilities, including their visual attributes.

In conclusion, it has been demonstrated that the context, understood as compatible garments, in which a fashion item is framed, is highly important for improving the accuracy of compatibility prediction with other unrelated items.

Keywords

Recommender systems, Fashion, Compatibility, Outfit

Índice

Listas

Lista de algoritmos

Lista de códigos

Lista de cuadros

Lista de ecuaciones

Lista de figuras

Lista de tablas

Lista de cuadros

1

Introducción

Es un hecho que, en los últimos años, nuestra forma de comprar ha cambiado por completo. La llegada de la COVID-19 aceleró aún más el proceso de transformación digital que nuestra sociedad estaba viviendo. Con el auge del comercio electrónico (*e-commerce*) se hizo necesario el desarrollo de nuevas maneras de ofrecer productos a los clientes a la hora de realizar su compra on-line. En ese contexto aparecen los sistemas de recomendación, que sustituyen a los dependientes que ayudan o recomiendan productos basándose en la intuición o en las características personales del cliente. Estas sugerencias personalizadas son fáciles de reproducir por un algoritmo siempre que se basen en premisas como edad, sexo, productos adquiridos previamente por otros clientes, etc., las cuales son objetivas y medibles. La dificultad aparece cuando la recomendación desea hacerse sobre algo subjetivo, como el estilo, el contexto, o la tendencia, ya que tales propiedades pueden variar de un individuo a otro y evolucionar con el tiempo. Esto es lo que ocurre en el ámbito de la moda, sobre todo a la hora de recomendar artículos compatibles o un *outfit* [1] completo. Por ello se usan algoritmos de predicción de compatibilidad en moda, los cuales son fundamentales en otras muchas aplicaciones de esta industria como el diseño de moda personalizado, la composición de outfits, o la previsión de tendencias.

1.1. Motivación

La moda juega un papel muy importante en la sociedad, ya que esta se usa como una forma de expresión, y se prevé que el mercado global de comercio electrónico de moda alcance un valor de más de 820 mil millones de dólares estadounidenses en 2023, e incluso podría alcanzar poco más de 1.2 billones de dólares estadounidenses para el año 2027 [2]. Esto demuestra la creciente demanda de compras de ropa en línea y motiva a las empresas a desarrollar sistemas de recomendación más avanzados. Se han realizado numerosas investigaciones utilizando técnicas computacionales para resolver problemas en la moda, especialmente en el comercio electrónico.

Una de las líneas de investigación más comunes se ha centrado en recomendar artículos de moda

[1] Un outfit es un conjunto de prendas de vestir y accesorios seleccionados para crear un atuendo completo y coordinado [1].

individuales a los consumidores en función de su historial de compras o navegación. Sin embargo, la recomendación de moda es única en comparación con otros dominios, no solo debido a su naturaleza visual, sino también porque el concepto de **compatibilidad** es crucial, más que en cualquier otro tipo de productos. Las personas suelen estar interesadas en comprar artículos que combinen bien entre sí y compongan un atuendo compatible. Tradicionalmente, los sistemas de recomendación de moda se basan en historiales de compras y clics conjuntos, y recomiendan artículos en función de la similitud y las reseñas de los usuarios, al igual que los sistemas de recomendación tradicionales [3]. La recomendación de outfits completos requiere ir más allá de recuperar artículos similares para desarrollar un modelo que comprenda la noción de "compatibilidad". La modelización de la compatibilidad es un desafío debido a que los aspectos semánticos que determinan lo que es compatible y coherente son extremadamente complejos, y muchos factores como el color, el corte, el patrón, la textura, el estilo, la cultura y el gusto personal desempeñan un papel en lo que las personas perciben como compatible y a la moda. Desarrollar un algoritmo de inteligencia artificial que pueda aprender la compatibilidad de los artículos mejoraría considerablemente la calidad de muchos sistemas de recomendación de moda.

Los algoritmos de predicción de compatibilidad en moda consisten en determinar si un conjunto de prendas combinan bien o son visualmente atractivos, y requiere comprender la estética de la moda, los estilos visuales, los contextos culturales y sociales, así como considerar las preferencias y tendencias individuales en la industria de la moda. En este trabajo, se estudiarán métodos recientes de la literatura donde se ha intentado integrar la compatibilidad en la recomendación de moda.

1.2. Objetivos

El objetivo principal de este trabajo es adquirir conocimientos diferentes a los estudiados durante el grado, analizando algoritmos de predicción de compatibilidad como mejora de los sistemas de recomendación en moda, ámbito que es muy diferente al resto debido a su subjetividad e importancia en el ámbito visual y por ende, más complejo. Por ello, será necesario explorar métodos que exploten algoritmos de aprendizaje profundo.

Uno de los algoritmos que hemos estudiado más a fondo se utilizan redes neuronales de grafos, en concreto Codificadores Automáticos de Grafos o GAE (del inglés, *Graph Auto-Encoders*), en los que las prendas de vestir son vértices, y las aristas conectan artículos compatibles. Este modelo permite aprovechar la información estructural y obtener mejores *embeddings* [2]. Utilizando estos métodos se hace posible la principal novedad de este proyecto: el uso del contexto como información necesaria a la hora de predecir lo que se considera "compatible", siendo en sí mismo un sesgo subjetivo del individuo y la tendencia del momento.

[2] Los embeddings de un grafo se pueden traducir como "incrustaciones" o "representaciones". Estas incrustaciones o representaciones son vectores numéricos que capturan las características y relaciones de los nodos en el grafo, lo que permite realizar diferentes tareas de análisis o predicción en este.

El uso del concepto de compatibilidad añade, de manera inherente, variación en las predicciones que reciben los usuarios, ya que cuando no se tiene en cuenta, la importancia de un producto se basa sólo en sus preferencias y en factores como la popularidad o la tendencia, por lo que, a igualdad de condiciones, la predicción no cambiará. No obstante, al crear o recomendar un outfit, necesariamente hay que tener en cuenta el resto de artículos que conforman el atuendo, por lo que dos productos que podrían tener un valor de compatibilidad muy alto de manera aislada, podría no ser una predicción tan interesante para un outfit determinado.

Finalmente, otro de los objetivos de este proyecto es realizar pruebas usando estos modelos para poder obtener conclusiones sobre cómo diferentes configuraciones de entrenamiento o de los datos pueden afectar al resultado de la predicción.

1.3. Organización de la memoria

El documento se encuentra dividido en cinco capítulos:

- Capítulo 1. Introducción: se presenta el problema sobre el que gira todo el trabajo, los objetivos a completar del proyecto y la estructura del documento.
- Capítulo 2. Estado del arte: se explica cómo funcionan en general los sistemas de recomendación, las métricas de evaluación utilizadas y se detalla el problema a resolver durante el trabajo.
- Capítulo 3. Análisis, diseño e implementación: contiene, por un lado, el diseño que se ha llevado a cabo durante el proyecto como puede ser el análisis de requisitos, su estructura general y su ciclo de vida. Por otro lado, se explica detalladamente la implementación del sistema.
- Capítulo 4. Pruebas y resultados: se detalla cómo se han realizado y cuáles son los experimentos que se han llevado a cabo para estudiar el rendimiento del recomendador implementado. Además, se muestran y analizan los resultados obtenidos.
- Capítulo 5. Conclusiones y trabajo futuro: se exponen las conclusiones finales del trabajo, así como las posibles mejoras que se pueden investigar como trabajo futuro.

2

Estado del arte

En este capítulo se exploran los conceptos fundamentales para el entendimiento de este trabajo. Se introduce el problema de predicción de compatibilidad como un medio para mejorar los sistemas de recomendación en moda, y desde un enfoque diferente al de otros modelos de aprendizaje métrico que se basan únicamente en comparaciones entre características de los artículos. En este caso se tendrá en cuenta el contexto, entendido como los productos que son compatibles con cada uno de los diferentes artículos, ya que esto se ha comprobado que facilita la tarea a los modelos. Además, se estudian las redes neuronales de grafos y los codificadores automáticos de grafos, en los cuales se basa el modelo que se presentará en el próximo capítulo. Antes de eso, se definen los sistemas de recomendación de manera genérica, ya que son la base de los sistemas más complejos que se usan en el dominio de la moda.

2.1. Sistemas de recomendación

Los sistemas de recomendación son herramientas y técnicas de software que proporcionan sugerencias de artículos que pueden ser de interés para un usuario [3]. Estos sistemas son especialmente útiles en webs donde existe una gran cantidad de productos entre los que elegir, y se basan en la información que tienen del usuario, identificando patrones y ofreciendo sugerencias personalizadas.

Se puede distinguir entre diferentes tipos de sistemas de recomendación, que varían dependiendo del algoritmo utilizado, es decir, de cómo se realiza la predicción de la utilidad de una recomendación. Para proporcionar una visión general inicial de los diferentes tipos de RS, usaremos una taxonomía proporcionada por [4] que se ha convertido en una forma clásica de distinguir y referirse a los sistemas de recomendación, donde se distingue entre seis clases diferentes de enfoques de recomendación:

- Basado en contenido. Se recomiendan artículos similares a los que le gustaron anteriormente al usuario. El sistema aprende a recomendar a través de dos fuentes: las características asociadas a los productos y las calificaciones que un usuario les ha dado. Se trata la recomendación como una tarea de clasificación específica de cada usuario y se aprende a clasificar si algo le gusta o no a un usuario basándose en las características del producto [5].

- Filtrado Colaborativo. Considerada como la técnica más usada y popular, es la forma original y más simple de recomendación, ya que genera recomendaciones basándose en otros usuarios que tuvieron gustos similares en el pasado. Los sistemas colaborativos identifican usuarios similares con un historial de valoraciones parecido al del usuario actual y generan recomendaciones basándose en estas relaciones [6].
- Basado en la comunidad. Este tipo de sistemas recomienda artículos basándose en las preferencias de los amigos de los usuarios. La recomendación se basa en las calificaciones proporcionadas por los amigos del usuario. De hecho, estos sistemas de recomendación siguen el auge de las redes sociales y permiten una adquisición sencilla y completa de datos relacionados con las relaciones sociales de los usuarios, algo muy importante ya que la evidencia sugiere que las personas tienden a confiar más en las recomendaciones de sus amigos que en las recomendaciones de individuos similares pero anónimos [7].
- Demográfico. Un sistema de recomendación demográfico proporciona recomendaciones basadas en el perfil demográfico del usuario. Los productos recomendados son creados para diferentes segmentos demográficos, combinando las calificaciones de los usuarios que pertenecen al mismo grupo.
- Basado en conocimiento. Estos sistemas recomiendan ítems teniendo en cuenta información específica sobre cómo ciertas características de estos satisfacen las necesidades y preferencias del usuario, así, utilizan el conocimiento especializado sobre un tema para determinar qué ítems serían útiles para el usuario, junto con otra información del dominio como podría ser información semántica o de otras bases de conocimiento [8].
- Sistemas de Recomendación Híbridos. Se basan en la combinación de las técnicas mencionadas anteriormente. Combinar dos técnicas permite aprovechar las ventajas de una, para suplir las desventajas de la otra [4].

A continuación, pasaremos a explicar en detalle el dominio en el que nos hemos centrado en este trabajo: la moda. Para ello, se usarán algoritmos de recomendación pertenecientes a varias de las clases descritas anteriormente (filtrado colaborativo, basado en contenido e híbridos, principalmente).

2.2. Sistemas de recomendación en moda

En el ámbito de la moda, la mayoría de los sistemas de recomendación disponibles utilizan la búsqueda por palabras clave, historiales de compras y calificaciones de los usuarios para recomendar artículos. Sin embargo, la recomendación de moda es única en comparación con otros dominios debido a la importancia de la apariencia visual de los artículos.

Algunas formas en que se aplica el aprendizaje profundo en la recomendación de moda incluyen:

- Recomendación basada en imágenes: Los modelos de aprendizaje profundo, como las Redes Neuronales Convolucionales (CNN), pueden extraer características visuales de imágenes de moda para recomendar prendas de vestir visualmente similares.
- Recomendación basada en texto: Técnicas de Procesamiento de Lenguaje Natural (PLN), como las Redes Neuronales Recurrentes (RNN) o los modelos Transformer, pueden procesar descripciones de texto de prendas de moda para comprender sus atributos y recomendar artículos basados en las preferencias del usuario.
- Filtrado colaborativo: Los modelos de aprendizaje profundo pueden analizar datos de interacción entre usuarios e items para predecir las preferencias del usuario y recomendar prendas de moda que se ajusten a sus gustos.
- Enfoques híbridos: Los modelos de aprendizaje profundo pueden combinarse con algoritmos de recomendación tradicionales, como métodos basados en contenido o filtrado colaborativo, para mejorar la calidad de las recomendaciones.

Para una revisión exhaustiva de cómo se han adaptado estos algoritmos a dicho dominio, se le recomienda al lector que revise los trabajos [9] y [10].

Por otro lado, el concepto de compatibilidad es más crucial que en cualquier otro tipo de productos, lo que requiere ir más allá de la recuperación de artículos similares, para desarrollar un modelo que comprenda la noción de "compatibilidad". Dado que el foco de nuestro trabajo está en el estudio de la compatibilidad y qué métodos se han propuesto para ello, pasaremos a explicarlo en más detalle a continuación.

2.2.1. Predicción de compatibilidad en la recomendación de moda

Predecir la compatibilidad en la moda se refiere a la tarea de determinar si un conjunto de prendas de moda combina bien entre sí. Esta tarea es fundamental para una variedad de aplicaciones en la industria, como el diseño de moda personalizado, la composición de conjuntos, la recomendación de artículos o la predicción de tendencias de moda. La compatibilidad es, por tanto, un concepto muy diferente a la similitud, ya que simplemente recuperar artículos que sean similares entre sí no es suficiente para formar conjuntos de moda. Muchas veces, dos elementos encajan perfectamente en un atuendo de moda, mientras que, al mirarlos individualmente, son visualmente muy diferentes.

Por ello, se han desarrollado técnicas que permitan predecir la compatibilidad de dos prendas. La mayor parte de estos presentan tres limitaciones principales:

1.– Solo pueden determinar la compatibilidad de un par de artículos y no funcionan en conjuntos de atuendos con un número arbitrario de elementos.

2.– Necesitan etiquetas de categoría (por ejemplo, camisa, zapatos) y atributos detallados

(por ejemplo, estampado floral, informal) para determinar la compatibilidad y no funcionarán si dicha información no está disponible.

3.– Requieren un orden fijo o un número fijo de elementos para determinar la compatibilidad de un atuendo. Por ejemplo, Han et al. [11] propusieron un método para el aprendizaje de la compatibilidad que requiere que los elementos en todos los atuendos estén ordenados cuidadosamente de arriba hacia abajo y luego los accesorios.

A la vez, es un problema desafiante debido a la naturaleza subjetiva y altamente compleja de la moda. La compatibilidad puede depender de varios factores, como el color, el estilo, el patrón, la textura, la forma y otros aspectos estéticos. Además, las preferencias de compatibilidad pueden variar entre diferentes personas y culturas.

Trabajos anteriores sobre el problema de la predicción de compatibilidad en la moda utilizan modelos que realizan principalmente comparaciones entre pares de elementos basadas en información de los elementos, como imágenes, categorías, descripciones, etc. Estos enfoques tienen la desventaja de que cada par de elementos considerado se trata de forma independiente, lo que hace que la predicción final se base en comparaciones entre las características de cada elemento de forma aislada. En un mecanismo de comparación que descarta el contexto, el modelo realiza la misma predicción para un par dado de prendas de vestir en cada ocasión. Por ejemplo, si el modelo está entrenado para hacer coincidir un estilo específico de camisa con un estilo específico de zapatos, hará consistentemente esta misma predicción en cada ocasión.

Sin embargo, dado que la compatibilidad es una medida subjetiva que puede cambiar con las tendencias y entre individuos, este comportamiento inflexible no siempre es deseable durante la evaluación. La compatibilidad entre la camisa y los zapatos mencionados no solo se define por las características de estos artículos en sí, sino que también está sesgada por las preferencias individuales y el sentido de la moda. Por lo tanto, se define el contexto de una prenda de vestir como el conjunto de artículos con los que es compatible [1]. Esta consideración proporciona al modelo algún conocimiento previo sobre lo que se considera “compatible”, algo que, en sí mismo, es un sesgo subjetivo del individuo y de la tendencia del momento.

La predicción de compatibilidad en moda puede abordarse utilizando técnicas de aprendizaje automático, como los codificadores automáticos de grafos. Estas técnicas buscan aprender patrones y relaciones en conjuntos de datos de moda, para poder hacer recomendaciones de combinaciones de productos que sean visualmente atractivas y consideradas compatibles por los usuarios.

2.2.2. Aprendizaje profundo en la recomendación de moda

El aprendizaje profundo en la recomendación de moda consiste en la aplicación de técnicas de aprendizaje profundo a fin de mejorar la precisión y efectividad de los sistemas de recomendación de

moda. El aprendizaje profundo es una subrama del aprendizaje automático que utiliza redes neuronales con múltiples capas para aprender automáticamente representaciones jerárquicas de los datos. En el contexto de la recomendación de moda, los modelos de aprendizaje profundo pueden procesar y analizar grandes cantidades de datos relacionados con la moda, como imágenes, descripciones de texto e interacciones de los usuarios, para comprender patrones, estilos y preferencias. El aprendizaje profundo ha demostrado resultados prometedores en la recomendación de moda, especialmente en la captura de patrones complejos y estéticas visuales.

Redes Neuronales en Grafos (GNN)

Las Redes Neuronales en Grafos, o *Graph Neural Networks* (GNN), son un tipo de modelo de aprendizaje automático diseñado específicamente para datos estructurados en forma de grafos. A diferencia de otros modelos que se centran en datos tabulares o secuenciales, las GNN se enfocan en capturar las relaciones y la estructura de los grafos. En un grafo, los datos se representan como nodos y las relaciones entre ellos se representan como aristas (edges). El funcionamiento de una GNN se basa en la propagación de información a través de las conexiones del grafo. Cada nodo recopila información de sus nodos vecinos, la combina con su propia información y actualiza su representación interna. Este proceso se repite en múltiples capas para capturar información de vecindarios cada vez más amplios [1].

Las GNN han demostrado ser eficaces en una amplia gama de aplicaciones, como el análisis de redes sociales, la recomendación de elementos, la clasificación de nodos y la predicción de enlaces en grafos. Son especialmente útiles cuando los datos tienen una estructura de grafo y las relaciones entre los elementos son importantes para el análisis y la predicción.

Las GNN utilizan técnicas como las capas de convolución de grafos, que generalizan las operaciones de convolución utilizadas en imágenes a dominios de grafos. Estas capas permiten extraer características relevantes de los nodos y capturar patrones y estructuras en el grafo. Además de las capas de convolución, las GNN también pueden incluir capas de agrupación, atención y otras operaciones para mejorar la capacidad de representación y el rendimiento del modelo.

Hay varios tipos de Graph Neural Networks (GNN) que se han desarrollado para abordar diferentes tareas y consideraciones en el procesamiento de datos de grafos. Algunos de los tipos más comunes de GNN son:

1.– Graph Convolutional Networks (GCN): Las GCN son una de las primeras y más populares arquitecturas de GNN. Utilizan operaciones de convolución en grafos para propagar y combinar información de los vecinos de cada nodo, generando representaciones en capas sucesivas.

2.– GraphSAGE: GraphSAGE (Graph Sample and Aggregated) es una arquitectura de GNN

que utiliza técnicas de muestreo y agregación para aprender representaciones de nodos en grafos grandes y escalables.

3.– Gated Graph Neural Networks (GGNN): Los GGNN utilizan mecanismos de puertas (gates) para controlar la propagación de información a lo largo de los nodos y aristas del grafo. Esto permite que los GGNN modelen relaciones de largo alcance en grafos.

4.– Graph Attention Networks (GAT): Los GAT se basan en mecanismos de atención para calcular la importancia relativa de los vecinos de cada nodo durante la propagación de información en el grafo. Esto permite que los GAT asignen diferentes pesos a diferentes vecinos en función de su relevancia para la tarea.

5.– Graph Autoencoders (GAE): Los GAE utilizan técnicas de autoencoder para aprender representaciones latentes de los nodos en el grafo. Esto permite que los GAE capturen las estructuras latentes y las relaciones entre los nodos.

Redes Convolucionales de Grafos (GCN)

Las Redes Convolucionales de Grafos, o Graph Convolutional Networks (GCN), son un tipo de arquitectura de redes neuronales profundas diseñadas específicamente para trabajar con datos estructurados en forma de grafo. Las GCN permiten realizar operaciones de convolución en los grafos, de manera similar a cómo las redes convolucionales tradicionales operan en imágenes. Estas son capaces de propagar información a través de los nodos y capturar patrones y relaciones complejas en los datos del grafo. Cada capa de una GCN combina información de los nodos vecinos para actualizar las representaciones de los nodos, y estas actualizaciones se propagan a través de múltiples capas para capturar características de alto nivel.

Las GCN han demostrado ser eficaces en tareas de aprendizaje automático y análisis de datos basados en grafos, como clasificación de nodos, predicción de enlaces y tareas de recomendación en redes sociales, biología computacional y otras áreas donde los datos se estructuran en forma de grafo.

En resumen, las Redes Convolucionales de Grafos son una arquitectura de redes neuronales diseñada para trabajar con datos estructurados en forma de grafo, permitiendo la propagación de información y el aprendizaje de patrones y relaciones complejas en los grafos.

Codificadores Automáticos de Grafos (GAE)

Los codificadores automáticos de Grafos (en ingles, Graph Auto-Encoders) es una técnica utilizada en el campo del aprendizaje automático y la inteligencia artificial. El objetivo principal de los codificadores automáticos de grafos es aprender una representación latente de los nodos del grafo que capture las características y relaciones importantes de los datos.

Para lograr esto, el framework utiliza una estructura de codificador automático, que consta de dos

componentes principales: un codificador (encoder) y un decodificador (decoder). El codificador mapea los nodos del grafo a un espacio de menor dimensionalidad, donde se capturan las características. El decodificador reconstruye los nodos a partir de las características, tratando de replicar fielmente los nodos originales.

Además, el codificador se puede implementar como un GCN, lo que permite arquitecturas más complejas e interesantes. De hecho, los GAE (Graph Auto-encoders) y GCN (Graph Convolutional Networks) son dos tipos de arquitecturas distintas dentro de las Graph Neural Networks (GNN) que comparten ciertas similitudes y pueden complementarse en algunos casos. Como se ha explicado previamente, las GCN son una arquitectura específica de GNN que utiliza convoluciones en grafos para propagar y combinar información entre los nodos del grafo. Su objetivo principal es aprender representaciones de nodos en función de la estructura del grafo y las características de los nodos y sus vecinos. Por otro lado, los GAE son una variante de las GNN que se basan en técnicas de codificador-decodificador para aprender representaciones latentes de los nodos en el grafo. En lugar de enfocarse exclusivamente en la propagación de información, los GAE buscan comprimir y reconstruir eficientemente los datos del grafo, lo que les permite capturar las estructuras latentes y las relaciones entre los nodos. Si bien los GAE y los GCN tienen objetivos diferentes y enfoques distintos, pueden complementarse en algunos casos. Por ejemplo, se pueden utilizar GCN para extraer características iniciales del grafo y luego utilizar GAE para aprender representaciones latentes más compactas y expresivas. Esta combinación puede ayudar a mejorar el rendimiento y la capacidad de generalización del modelo en ciertas tareas de procesamiento de grafos.

En general, el modelo de GAE es muy útil en diferentes problemas, como sistemas de recomendación, predicción de compatibilidad en moda y otros donde las relaciones entre los elementos son importantes.

2.3. Evaluación de sistemas de recomendación

Los sistemas de recomendación son algoritmos que, como tal, deben someterse a una evaluación, algo fundamental para evaluar la eficacia de los sistemas de recomendación [12]. Esto consiste en comprobar la correcta implementación del sistema, que los resultados son los esperados, y ver su efectividad. Además, esta evaluación es útil para comparar sistemas de recomendación, y determinar cuál es más efectivo. Hay múltiples formas de medir la calidad de la recomendación generada por un algoritmo, dependiendo de lo que se busque optimizar [13, 14]:

- Estudio de usuarios. Se mide la satisfacción del usuario mediante calificaciones explícitas. Los usuarios reciben recomendaciones generadas por diferentes sistemas de recomendación, las califican y, a través de la media, se obtiene información de cuál es considerado el mejor sistema.

- Online. Este mecanismo se realiza el tiempo real, aprovechando el uso real del sistema por parte de usuarios reales. Se basa en la tasa de aceptación como medida implícita de satisfacción del usuario, y se puede medir mediante la tasa de clics (CTR), la proporción de artículos descargados o comprados. Sin embargo, esta suposición no siempre es confiable ya que un usuario puede incluso comprar artículos, que finalmente no sean lo más acertados.
- Offline. Como su nombre indica, se utiliza información que se posee del usuario previa al uso del sistema de recomendación. Este método necesita obtener esta información por medio de lo que se conoce como dataset [1], que se divide en entrenamiento y test, de manera que usando los datos de entrenamiento, se puedan crear recomendaciones que predigan los datos de test. Existen distintas métricas usadas en este tipo de evaluación, algunas de las más comunes son:
    - Precisión: Mide la proporción de recomendaciones correctas sobre el total de recomendaciones realizadas.
    - Recall: Mide la proporción de recomendaciones correctas sobre el total de elementos relevantes en el conjunto de datos.
    - F1-score: Es una medida que combina la precisión y el recall para proporcionar una medida más equilibrada del rendimiento.
    - MAP (Mean Average Precision): Calcula el promedio de las precisiones en diferentes niveles de recall.
    - NDCG (Normalized Discounted Cumulative Gain): Mide la utilidad de las recomendaciones teniendo en cuenta la posición de los elementos relevantes en la lista de recomendaciones.
    - RMSE (Root Mean Squared Error): Se utiliza en sistemas de recomendación basados en calificaciones para medir la diferencia entre las calificaciones predichas y las reales.
    - MAE (Mean Absolute Error): Similar al RMSE, mide el promedio de las diferencias absolutas entre las calificaciones predichas y las reales.

2.3.1. Evalución de la recomendación de moda

En la tarea de recomendación de moda, aparecen tareas concretas de evaluación inherente a la distinta naturaleza del dominio y su complejidad, las cuales pueden tener metodologías de evaluación o incluso métricas específicas, como se puede ver a continuación:

[1] Un dataset es una colección de datos organizados y estructurados que se utiliza para realizar análisis, investigaciones o entrenar modelos en diversos campos.

Fill In The Blank (FITB)

La tarea de llenar el espacio en blanco consiste en elegir la prenda que mejor complemente un conjunto de atuendo dado, de entre un conjunto de posibles opciones. Se define una pregunta FITB para cada atuendo de prueba, y cada una consta de un conjunto de productos que forman un atuendo parcial y un conjunto de opciones posibles que incluye la respuesta correcta y productos elegidos al azar. FITB puede entenderse como un problema de predicción de bordes, donde el modelo primero genera la probabilidad de bordes entre pares de prendas para todos. Luego, el puntaje para cada una de las M opciones se calcula como la suma de las probabilidades de los bordes, y se selecciona la que tiene el puntaje más alto como la prenda que se agrega al conjunto de atuendo parcial. La tarea se evalúa midiendo si la prenda correcta fue seleccionada de la lista de opciones.

Predicción de compatibilidad de atuendos

En la tarea de predicción de compatibilidad de atuendos, el objetivo es producir un puntaje de compatibilidad de atuendo, que representa la compatibilidad general de las prendas que forman el conjunto de atuendo. Los puntajes cercanos a 1 representan atuendos compatibles, y los puntajes cercanos a 0 representan atuendos incompatibles. La tarea de predicción de compatibilidad de atuendos se evalúa utilizando el área bajo la curva ROC para los puntajes predichos.

Evaluación por tamaño del vecindario

Sea el k-vecindario del nodo i en nuestro grafo relacional el conjunto de k nodos visitados por un proceso de búsqueda en anchura, comenzando desde i. Para medir el efecto del tamaño de la estructura relacional alrededor de cada prenda, durante las pruebas permitimos que cada muestra de prueba contenga las prendas y sus k-vecindarios, y evaluamos nuestro modelo variando k. Así, cuando k = 0, no se utiliza información relacional, y el embedding de cada producto se basa solo en sus propias características. A medida que aumenta el valor de k, el embedding de las prendas comparadas estará condicionada a más vecinos.

2.4. Datasets

Para realizar las evaluaciones descritas en la sección anterior, como hemos explicado, es necesario utilizar datasets a no ser que se hagan evaluaciones online o a través de un estudio de usuarios. Algunos de los más utilizados en sistemas de recomendación de moda son [1]:

- Conjunto de datos Polyvore. El conjunto de datos Polyvore [15] es un conjunto de datos creado por los usuarios de un sitio web del mismo nombre; el sitio web permitía a sus miembros cargar fotos de prendas de moda y agruparlas en conjuntos. Contiene un total de 164,379

artículos que forman 21,899 conjuntos diferentes. El número máximo de artículos por conjunto es de 8, y el número promedio de artículos por conjunto es de 6.5. El grafo se crea conectando cada par de nodos que aparecen en el mismo conjunto con una arista. Se suele utilizar para entrenar y aplicar varias de las tareas descritas anteriormente.

El subconjunto para la tarea FITB contiene 3,076 preguntas y la tarea de compatibilidad de conjuntos tiene 3,076 conjuntos válidos y 4,000 conjuntos inválidos. En el conjunto de datos Polyvore original, las opciones incorrectas de FITB y los conjuntos inválidos se seleccionan al azar entre todos los productos restantes. El conjunto de datos de muestra propuesto por Vasileva et al. [16] es más desafiante: las opciones incorrectas en cada pregunta de la tarea FITB se muestrean de los artículos que tienen la misma categoría que la opción correcta; para la compatibilidad de conjuntos, los conjuntos se muestrean al azar de manera que cada artículo en un conjunto dado sea de una categoría distinta. Algunos autores han creado un conjunto más desafiante aún, donde se limita el tamaño de los conjuntos a 3 artículos seleccionados al azar. En este escenario, las tareas se vuelven más difíciles porque hay menos información disponible para el modelo.

- Conjunto de datos Fashion-Gen Outfits. Fashion-Gen [17] es un conjunto de datos de productos de moda recopilados de una plataforma en línea que vende productos de lujo de diseñadores independientes. Cada producto tiene imágenes, descripciones, atributos e información relacional. Las relaciones de Fashion-Gen están definidas por diseñadores profesionales y siguen un tema general, mientras que las relaciones de Polyvore son generadas por usuarios con diferentes gustos y nociones de compatibilidad.

  Es posible crear conjuntos de Fashion-Gen agrupando entre 3 y 5 productos que están conectados entre sí. De esta forma, en [1] los autores generan un conjunto de entrenamiento que consta de 60,159 conjuntos diferentes de las colecciones de 2015 a 2017, y un conjunto de validación y prueba con 2,683 y 3,104 conjuntos respectivamente, de la colección de 2014.

- Conjunto de datos de productos de Amazon. El conjunto de datos de productos de Amazon [18] contiene más de 180 millones de relaciones entre casi 6 millones de productos de diferentes categorías. Aquellos trabajos que se centran en la recomendación de moda, utilizan un subconjunto con los productos de ropa, llegando a explotar las categorías de Hombres y Mujeres de dichos productos.

  Con estos datos, algunos autores han sido capaces de capturar 4 tipos de relaciones entre los artículos: (1) usuarios que vieron A también vieron B; (2) usuarios que vieron A compraron B; (3) usuarios que compraron A también compraron B; y (4) usuarios que compraron A y B simultáneamente. Para los dos últimos casos, se asume que el par de artículos A y B son compatibles (para la tarea de compatibilidad) y se evalúa el modelo en función de esta suposición. Como explicaremos más tarde, en este proyecto haremos esta misma distinción al usar este conjunto de datos.

3

Análisis, diseño e implementación

En este capítulo, se detallará la estructura, ciclo de vida y requisitos del proyecto desarrollado, en concreto, todo lo relativo al sistema de predicción de compatibilidad. Además, se explica detalladamente la implementación de cada fase y módulo del sistema.

3.1. Análisis y diseño

Esta sección recoge el análisis de requisitos, la estructura del proyecto, las tecnologías utilizadas y el proceso llevado a cabo.

3.1.1. Estructura general

Para entender mejor el diseño seguido en el proyecto, la Figura 3.1 describe los módulos en los que se divide este proyecto.

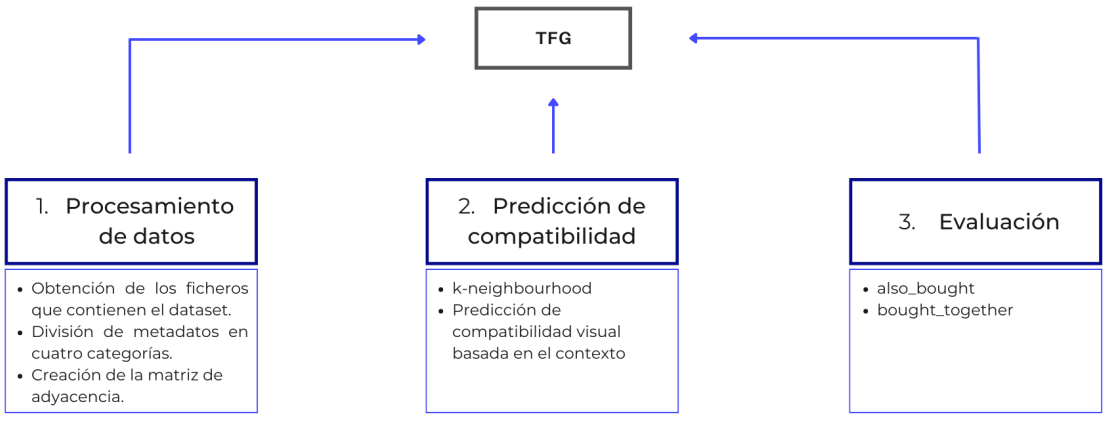

Figura 3.1: Diagrama con los diferentes módulos del proyecto y su relación.

Como se observa en la figura, la estructura del proyecto consta de 3 módulos, los cuales se mencionan en orden de implementación, y cuya funcionalidad se detallará más adelante:

1.– Módulo de procesamiento de datos. Recibe un archivo en formato JSON y un fichero binario, y se encarga de tratarlos para poder obtener la información que será usada posteriormente por los módulos de predicción de compatibilidad y evaluación. Es decir, procesa los datos del dataset de forma que sean legibles por las siguientes partes del proyecto. A priori, este módulo está pensado para admitir los datasets descritos en la Sección 2.4.

2.– Módulo de predicción de compatibilidad. Partiendo de los datos procesados del anterior módulo, este permite entrenar el modelo para poder resolver el problema de predicción de compatibilidad. Su tarea es realizar las predicciones solicitadas a través del algoritmo usado, de manera que se pueda evaluar en el siguiente módulo. Esta parte de la implementación se basa en lo explicado en la Sección 2.2.1.

3.– Módulo de evaluación. Módulo necesario para evaluar las predicciones obtenidas para cada uno de los casos entrenados. Aquí se implementarán algunas de las métricas comentadas en la Sección 2.3.1. De esta manera, se comparan los resultados de las predicciones obtenidas por el segundo módulo, teniendo en cuenta los diferentes parámetros que se hayan elegido para cada una de las ejecuciones del algoritmo, con la información real sobre la compatibilidad de los artículos.

3.1.2. Ciclo de vida

El ciclo de vida del proyecto se basa en un modelo en cascada retroalimentada. Se ha tomado la decisión de seguir el ciclo de vida en cascada debido a la presencia de los distintos módulos comentados anteriormente, los cuales necesitan del correcto funcionamiento de su antecesor para poder ser implementados. Además, este ciclo debía ser retroalimentado, ya que en ciertos puntos de bloqueo hubo que replantear partes del proyecto.

1.– Análisis. Se realiza una investigación sobre el estado del arte de los sistemas de recomendación en moda, herramientas, algoritmos y conjuntos de datos. También se definen los requisitos funcionales de la aplicación.

2.– Diseño. Basándose en el análisis y requisitos definidos, en esta fase se toman decisiones sobre la estructura del proyecto y los algoritmos a implementar.

3.– Implementación. Se lleva a cabo la puesta en marcha del proyecto, se realizan los cambios necesarios para su correcto funcionamiento. A lo largo de esta fase, y debido a imprevistos en el código y librerías, fue necesario volver a etapas anteriores para poder hacer algunos ajustes, e incluso en una primera iteración del ciclo de vida, volver al análisis y reestructurar de nuevo todo el proyecto.

4.– Pruebas. Ejecución del sistema implementado y obtención de datos necesarios para el siguiente paso.

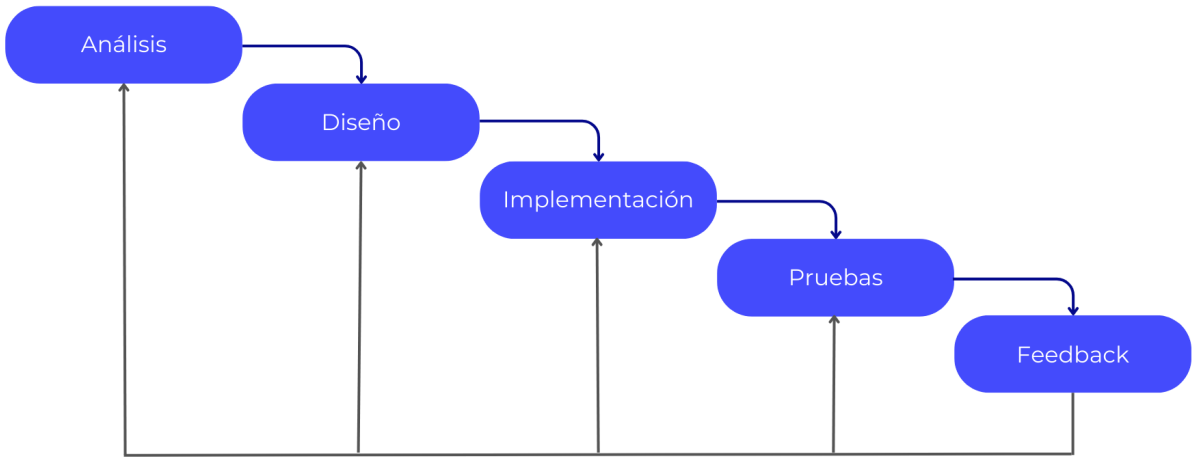

Figura 3.2: Diagrama descriptivo del ciclo de vida de este proyecto en particular.

5.– Feedback. Revisión de los resultados obtenidos y ajuste de parámetros según las pruebas, de cara a la realización de mejoras.

3.2. Requisitos

En esta sección se definen los distintos requisitos de este trabajo de investigación. Para la implementación de este proyecto se ha usado el lenguaje Python por las siguientes razones:

- Lenguaje no familiar. No tuve oportunidad de trabajar de una manera extensa con Python durante la carrera, por lo que su uso me ha parecido una buena forma de aprender algo más en este proyecto.
- Es un lenguaje de alto nivel.
- Tiene una gran cantidad de librerías necesarias para trabajar con él.
- Es adecuado para el uso en la plataforma de desarrollo utilizada (Google Colab).

3.2.1. Requisitos funcionales

General

RF-1.– Todo el almacenamiento debe ser mediante ficheros, para permitir retomar la ejecución desde cualquier módulo del sistema.

RF-2.– El sistema debe notificar de cualquier error y exponer la causa, así como informar del estado en el que se encuentra el algoritmo. Como se especifica en el requisito anterior, los errores también se

deben guardar en fichero (logs).

RF-3.– El sistema debe poder ejecutarse con la librería Tensorflow mayor o igual que 1.0.0

RF-4.– El sistema debe poder ejecutarse sobre una GPU

Procesado

RF-5.– El conjunto de datos de entrada debe estar en formato JSON.

RF-6.– Los datos deben permitir el trabajo con GNNs.

RF-7.– La aplicación debe permitir el uso de un dataset de Amazon.

RF-8.– El conjunto de datos de salida debe generarse en un fichero Pickle.

Predicción

RF-9.– Los tiempos de ejecución deben ser guardados.

RF-10.– El sistema no generará gráficas a partir de los datos obtenidos, solamente ficheros de texto.

RF-11.– El sistema admitirá pruebas con distintos parámetros en las ejecuciones.

RF-12.– El modelo de predicción de compatibilidad debe tener en cuenta el resto de artículos de un outfit.

RF-13.– Las pruebas deben realizarse en cuadernos de Colab para facilitar su lectura.

RF-14.– Todos los ficheros Notebook que faciliten las distintas pruebas y ejecuciones serán accesibles vía web, sin necesidad de instalar nada localmente.

Evaluación

RF-15.– La aplicación debe usar métricas de evaluación para saber cómo de buena es la recomendación.

3.2.2. Requisitos no funcionales

RNF-1.– El sistema estará documentado en español.

RNF-2.– El lenguaje usado será Python.

RNF-3.– El tiempo de predicción de compatibilidad de una subcategoría del dataset no debe ser mayor a 12 horas.

RNF-4.– El tiempo de evaluación del modelo no debe ser mayor a 10 minutos.

3.3. Implementación

En esta sección se tratará detalladamente tanto el flujo completo del proyecto, desde la obtención del dataset hasta la extracción de datos a partir de las evaluaciones del algoritmo.

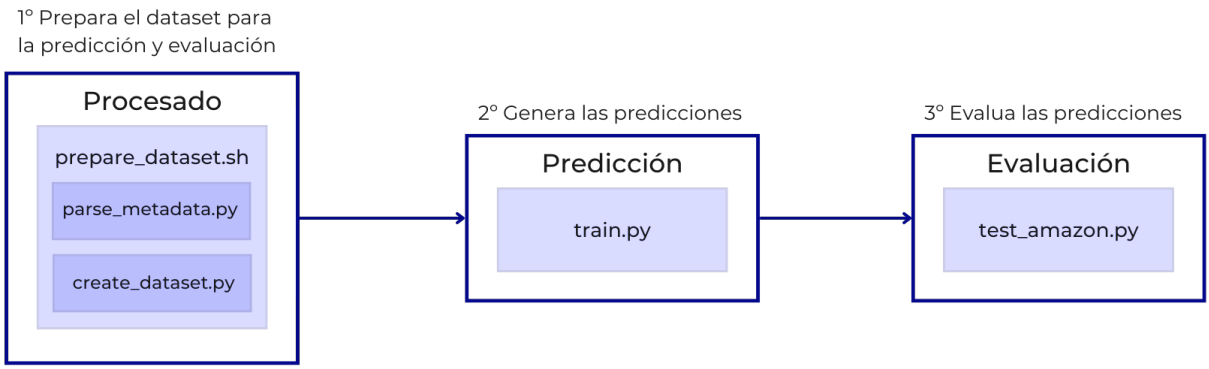

Figura 3.3: Diagrama del flujo de ejecución del proyecto.

3.3.1. Flujo del proyecto

Una representación del flujo de ejecución del sistema puede verse en la Figura 3.3, la cual está compuesta por los módulos descritos en la Sección 3.1.1.

Más concretamente, el módulo de Procesamiento es el que se encarga de generar los ficheros de datos necesarios para el módulo de Predicción, el cual es posteriormente analizado por el módulo de Evaluación, donde se obtienen los resultados de las predicciones, permitiendo realizar comparaciones entre los modelos entrenados.

Un ejemplo de ejecución de este flujo se muestra en la Figura 3.4.

3.3.2. Módulo de procesamiento de datos

En primer lugar, hablaremos del punto de partida de todo el trabajo: los datos. Este proyecto se ha realizado en torno al dataset *Clothing, Shoes and Jewelry* de Amazon. Este dataset está orientado al estudio de los algoritmos de recomendación de moda, como se introdujo en la Sección 2.4. Este dataset fue publicado por Amazon por primera vez en 2014, y actualizado en 2018; es este último el que usaremos para este proyecto.

```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

```
%cd gdrive/My Drive/TFG
```

```
/content/gdrive/My Drive/TFG
```

```
%cd visual-compatibility/data/amazon/
```

```
/content/gdrive/MyDrive/TFG/visual-compatibility/data/amazon
```

```
!bash prepare_dataset.sh
```

```
%cd ..
```

```
/content/gdrive/MyDrive/TFG/visual-compatibility/data
```

```
%cd ..
```

```
/content/gdrive/MyDrive/TFG/visual-compatibility
```

```
!python train.py -d amazon -e 500 -lr 0.001 -deg 10 -hi 350 350 350 -ws True -amzd Women_also_bought
```

```
!python test_amazon.py -k 5 -lf logs/Women_also_bought_1 -amzd Women_also_bought
```

Figura 3.4: Diagrama con ejemplo de ejecución desde un notebook.

En este apartado nos centraremos en la forma en la que hemos explotado estos datos, desde su descarga hasta su uso en la implementación del algoritmo basado en GNN.

Dataset de Amazon

A pesar de existir un subconjunto más pequeño de este dataset (como se puede ver en su página [1]), este no contiene información de las imágenes de los artículos, lo cual es crítico en este proyecto. Además, al ser información reducida no se pueden obtener métricas ni conclusiones tan exactas para nuestras pruebas como sería necesario.

Al ser de un tamaño considerable, se divide en categorías de productos. En el caso de este proyecto, al basarse en el campo de la moda, se tendrá en cuenta solo la categoría *Clothing, Shoes and Jewelry*, que es realmente un "subdataset" del principal de todo el e-commerce. Cada categoría contiene reseñas (*reviews*) y metadatos de productos de Amazon, así como los atributos visuales de las imágenes de cada producto.

Para nuestra implementación solo usaremos el fichero de metadatos, y el de características visuales. Ambos deben ser descargados (habiendo solicitado previamente su uso a los autores) y ubicados en la ruta correspondiente.

[1] https://jmcauley.ucsd.edu/data/amazon/, accedido por última vez en Julio de 2023.

```
{
  "asin": "0000031852",
  "title": "Girls Ballet Tutu Zebra Hot Pink",
  "price": 3.17,
  "imUrl": "http://ecx.images-
amazon.com/images/I/51fAmVkTbyL._SY300_.jpg",
  "related":
  {
    "also_bought": ["B00JHONN1S", "B002BZX8Z6", "B00D2K1M3O",
"0000031909", "B00613WDTQ", "B00D0WDS9A", "B00D0GCI8S", "0000031895",
"B003AVKOP2", "B003AVEU6G", "B003IEDM9Q", "B002R0FA24", "B00D23MC6W",
"B00D2K0PA0", "B00538F5OK", "B00CEV86I6", "B002R0FABA", "B00D10CLVW",
"B003AVNY6I", "B002GZGI4E", "B001T9NUFS", "B002R0F7FE", "B00E1YRI4C",
"B008UBQZKU", "B00D103F8U", "B007R2RM8W"],
    "also_viewed": ["B002BZX8Z6", "B00JHONN1S", "B008F0SU0Y",
"B00D23MC6W", "B00AFDOPDA", "B00E1YRI4C", "B002GZGI4E", "B003AVKOP2",
"B00D9C1WBM", "B00CEV8366", "B00CEUX0D8", "B0079ME3KU", "B00CEUWY8K",
"B004FOEEHC", "0000031895", "B00BC4GY9Y", "B003XRKA7A", "B00K18LKX2",
"B00EM7KAG6", "B00AMQ17JA", "B00D9C32NI", "B002C3Y6WG", "B00JLL4L5Y",
"B003AVNY6I", "B008UBQZKU", "B00D0WDS9A", "B00613WDTQ", "B00538F5OK",
"B005C4Y4F6", "B004LHZ1NY", "B00CPHX76U", "B00CEUWUZC", "B00IJVASUE",
"B00GOR07RE", "B00J2GTM0W", "B00JHNSNSM", "B003IEDM9Q", "B00CYBU84G",
"B008VV8NSQ", "B00CYBULSO", "B00I2UHSZA", "B005F50FXC", "B007LCQI3S",
"B00DP68AVW", "B009RXWNSI", "B003AVEU6G", "B00HSOJB9M", "B00EHAGZNA",
"B0046W9T8C", "B00E79VW6Q", "B00D10CLVW", "B00B0AVO54", "B00E95LC8Q",
"B00GOR92SO", "B007ZN5Y56", "B00AL2569W", "B00B608000", "B008F0SMUC",
"B00BFXLZ8M"],
    "bought_together": ["B002BZX8Z6"]
  },
  "salesRank": {"Toys & Games": 211836},
  "brand": "Coxlures",
  "categories": [["Sports & Outdoors", "Other Sports", "Dance"]]
}
```


Figura 3.5: Ejemplo de uno de los artículos contenidos en los metadatos, en formato JSON. Captura obtenida de la misma página de donde se puede obtener el dataset y equivalente al dataset usado en este proyecto.

- **Metadatos** de los productos (ver Figura 3.5).
  - asin. ID del producto
  - title. Nombre del producto
  - price. Precio del producto en dólares
  - imUrl. URL de la imagen del producto
  - related. Productos relacionados (también comprado, también visto, comprados juntos, comprado después de visto)
  - salesRank. Información sobre el ranking de ventas
  - brand. Nombre de la marca
  - categories. Lista de las categorias a las que pertenece el producto
- **Atributos visuales** Las características visuales de cada imagen de producto se obtienen utilizando una CNN profunda. Estas se almacenan en un fichero en formato binario, el cual consta de 10 caracteres (ID del producto), seguido de 4096 números decimales que se repiten para cada producto.

Procesamiento

Para procesar los datos, se usan una serie de scripts en Python, cuya funcionalidad se describe a continuación.

prepare_dataset.sh Este programa será el encargado de comprobar si las rutas de los archivos procesados de metadatos y dataset están presentes, y en caso contrario ejecuta los scripts de parse_metadata.py y create_dataset.py, que generarán los ficheros necesarios para la implementación y funcionamiento de este sistema.

parse_metadata.py Este programa espera encontrar en la ruta indicada el fichero JSON comprimido que corresponde a los metadatos del dataset.

Tal y como recomiendan los autores del dataset [19, 20], una de las opciones de tratamiento de los datos es como objetos de diccionario de Python. De esta forma tan simple se pueden leer y tratar los datos para su posterior uso en el sistema (ver Código 3.1). Una vez parseados los datos, se guardan en el fichero **metadata.pkl**, cuya información se usa en el resto de scripts.

create_dataset.py Este script usa el fichero JSON comprimido con los metadatos utilizados también en el programa anterior, y un archivo binario que corresponde a los atributos visuales. La función de este script es procesar de nuevo los metadatos, esta vez guardándolos en 4 diccionarios distintos: Men_also_bought, Women_also_bought, Women_bought_together y Men_bought_together. De esta forma, se generan 4 subconjuntos de datos, y a partir de este momento son considerados como datasets independientes que parten de un conjunto de datos mayor, el dataset de la categoría *Clothing,*

Código 3.1: Código para parsear los metadatos de los ficheros

```
def parse(path):
  g = gzip.open(path, 'r')
  for l in g:
    yield eval(l)

data_path = '../'
meta_file = os.path.join(data_path, 'meta_Clothing_Shoes_and_Jewelry.json.gz')
metadata = {}
print('Parsing metadata...')
for i,l in enumerate(parse(meta_file)):
    metadata[l['asin']] = l

metadata_path = os.path.join(data_path, 'metadata.pkl')
with open(metadata_path, 'wb') as f:
    pkl.dump(metadata, f)
```

Shoes and Jewelry de Amazon. Posteriormente, se obtienen del fichero de características visuales de las imágenes los atributos que se irán asignando a cada producto.

Por lo tanto, ambos ficheros se complementan para dar lugar a un grafo que se utilizará para entrenar la red neuronal (ver Sección 2.2.2), en el que sabiendo que cada nodo es un producto, los atributos visuales obtenidos de cada imagen actuarán como embeddings y donde, para los productos que tienen una relación de compra entre ellos, se crea una arista entre ambos.

3.3.3. Módulo de predicción de compatibilidad

El modelo implementado realiza la predicción de compatibilidad basándose en el marco de un Codificador Automático de Grafos, descritos en la Sección 2.2.2. La parte del codificador calcula embeddings de artículos según sus conexiones y características propias, mientras que el decodificador utiliza estos embeddings para calcular la compatibilidad entre pares de elementos. Al condicionar estos embeddings de los productos en los vecinos, la información de estilo contenida en la representación es más robusta, lo que produce predicciones de compatibilidad más precisas.

Entrenamiento de los modelos

El entrenamiento del modelo se lleva a cabo con los datos anteriormente procesados.

train.py Este script es el encargado del entrenamiento de los modelos. Se deben entrenar cada uno de los diccionarios (o datasets) comentados en el apartado de procesamiento.

El modelo se entrena para predecir la compatibilidad entre los productos. Teniendo el grafo de productos (nodos) comentado antes, este se encuentra representado como una matriz de adyacencia. Para el entrenamiento se eliminan aleatoriamente un conjunto de aristas (relaciones), dando así lugar a una matriz de adyacencia incompleta. Al conjunto de aristas eliminadas las llamaremos aristas positivas, ya que representan pares de nodos que sabemos que estaban conectados. Por otro lado, se escoge aleatoriamente un conjunto de aristas de nodos no conectados o no compatibles, que serán las aristas negativas.

Cada modelo es entrenado para predecir tanto las aristas positivas como las negativas. De esta forma, el algoritmo (es decir, la combinación del encoder-decoder explicado en el modelo GAE de la Sección 2.2.2) irá aprendiendo para minimizar una función de pérdida basada en la entropía cruzada (*cross entropy*) entre las aristas predichas y sus valores verdaderos. Estos valores son 1 para las aristas entre nodos que son compatibles (aristas positivas), y 0 para los que no (aristas negativas). La entropía cruzada o entropía cruzada binaria es una medida utilizada comúnmente en el contexto del aprendizaje automático y la teoría de la información para evaluar la diferencia entre dos distribuciones de probabilidad, como la distribución de probabilidad predicha por un modelo y la distribución de probabilidad real o verdadera.

Código 3.2: Ejemplo de ejecución para uno de los conjuntos de datos

```
1  import time
2  for deg in [1,5,10]:
3    for lr in [0.001,0.005,0.01]:
4      for hi in ["50 50 50", "350 350 350"]:
5        print('Starting...',lr,deg,hi,'Men_bought_together')
6        before = time.time()
7        python train.py -d amazon -e 500 -lr lr -deg deg -hi hi -amzd
             Women_bought_together
8        after = time.time()
9        print('Ended:',after-before,'seconds',lr,deg,hi,'Men_bought_together')
```

Este entrenamiento se puede ejecutar como se indica en el Código 3.2. Como se puede observar, el programa que lleva a cabo el entrenamiento de los modelos acepta distintos argumentos de entrada, los cuales permiten realizar diferentes ejecuciones basadas en contextos muy distintos:

- -d DATASET. Permite seleccionar distintos datasets de entrada
- -lr LEARNING_RATE. Permite ajustar la tasa de aprendizaje. Una tasa de aprendizaje alta

permitirá mayor rapidez, pero puede hacer que el proceso de optimización sea más inestable. Por otro lado, una tasa de aprendizaje baja hará que el proceso de entrenamiento sea más lento, pero puede ayudar a obtener una convergencia más estable y precisa

- -wd WEIGHT_DECAY. Decaimiento de pesos
- -e EPOCHS. Número de épocas de entrenamiento
- -hi HIDDEN. Un argumento por capa que define el número de unidades ocultas en esa capa. Aumentar el número de unidades ocultas puede permitir que el modelo capture representaciones más complejas y ricas en información, pero también puede aumentar el coste computacional y el riesgo de sobreajuste
- -do DROPOUT. Fracción de desactivación
- -deg DEGREE. Grado de convolución o tamaño del vecindario utilizado alrededor de cada nodo
- -sdir SUMMARIES_DIR. Directorio para guardar resúmenes de TensorFlow
- -sup_do SUPPORT_DROPOUT. Aplica dropout a las matrices de soporte, eliminando todas las conexiones de algunos nodos
- -ws WRITE_SUMMARY. Opción para activar la escritura de resúmenes.
- -amzd AMZ_DATA. Opción que solo tiene sentido en el dataset de Amazon. Permite el entrenamiento de los distintos modelos disponibles ('Men_also_bought', 'Women_also_bought', 'Women_bought_together', 'Men_bought_together')

Tras cada ejecución, se generan ficheros de log que muestran información sobre la misma. Un ejemplo de uno de estos ficheros es el siguiente, donde se repite la información de configuración y se muestra los mejores valores obtenidos en validación y en entrenamiento:


```
{"dataset": "amazon", "learning_rate": 0.001, "weight_decay": 0.0,
"epochs": 500, "hidden": [50, 50, 50], "dropout": 0.5, "degree": 5,
"summaries_dir": "logs/", "support_dropout": 0.15,
"write_summary": true, "batch_norm": true,
"amz_data": "Women_bought_together", "best_val_score": 0.8378134369850159,
"best_epoch": 491, "best_epoch_train_score": 0.9096487164497375,
"best_train_score": 0.9104092717170715, "seed": 1685099115}
```


Profundidad del vecindario

Aprovechar la información de contexto es la base de todo este proyecto, y como ya hemos comentado, permite el cálculo de embeddings muy detallados sobre las relaciones de los nodos. Esta información está controlada por el parámetro "-d" en la ejecución del entrenamiento, que representa la profundidad del vecindario que se considera durante el mismo. El vecindario en la profundidad S del

nodo *i*, es el conjunto de nodos a una distancia (número de aristas recorridas) *S* o menos desde *i*. En los experimentos llevados a cabo se cambiará el parámetro -d para poder realizar una variedad de entrenamientos en los modelos.

Sabiendo cómo se hace el entrenamiento, es hora de explicar cómo se llevará a cabo la evaluación de los resultados obtenidos.

3.3.4. Módulo de evaluación

Como se describe en la Sección 2.4, los metadatos contienen información sobre productos relacionados, distinguidos en 4 tipos:

1.– Usuarios que vieron A también vieron B

2.– Usuarios que vieron A compraron B

3.– Usuarios que compraron A también compraron B

4.– Usuarios que compraron A y B simultáneamente

Al estar investigando la compatibilidad entre los productos A y B, se asume que esto puede derivarse de los dos últimos casos, por lo tanto, la evaluación de este modelo se realiza con respecto a estas relaciones obtenidas de cada producto.

Tras realizar el entrenamiento de un modelo para cada tipo de relación, se pasa a evaluar las predicciones obtenidas.

A la hora de ejecutar el test, proporcionamos al modelo variaciones en el contexto mediante distintos parámetros:

- -k. Valor usado como el número máximo de vecinos que se considerará en la evaluación.
- -lf. Ubicación del log con las predicciones del modelo que se quiere evaluar
- -amzd. Dataset o grafo usado para evaluar el modelo. Como antes, una de las opciones entre ('Men_also_bought', 'Women_also_bought', 'Women_bought_together', 'Men_bought_together') y que debería coincidir con el subconjunto usado para entrenar el modelo.

Código 3.3: Ejemplo de evaluación para uno de los conjuntos de datos

```
1 python test_amazon.py -k 5 -lf logs/9 -amzd Women_bought_together
```

El argumento más importante de este script (ver Código 3.3) es -k, ya que permite realizar diversas evaluaciones sobre un mismo modelo entrenado, obteniendo conclusiones más valiosas.

Es posible también una evaluación de un modelo sobre un dataset que no es el que le corresponde, por ejemplo, un modelo entrenado con “ropa de mujer comprada junta” se puede evaluar con el dataset de “ropa de hombre comprada junta”. Esta configuración de género cruzado nos permite evaluar cómo el modelo se adapta a cambios en el contexto, a diferencia de un enfoque básico que lo ignora completamente.

Además, evaluar usando k=0 simularía una situación donde no se usa el contexto o la información de los vecinos, y cuanto mayor sea la k y más información de vecindario utilice un modelo, más robusto es al cambio de dominio. Esto ocurre porque cuando el modelo depende del contexto, puede adaptarse mejor a estilos o tipos de ropa no vistos previamente. Esto lo confirmaremos en el siguiente capítulo con los experimentos.

4

Pruebas y resultados

Este capítulo describe el entorno en el cual se han implementado y realizado los experimentos del sistema de predicción de compatibilidad en moda descrito a lo largo del documento. Además, se analizan los resultados obtenidos en las diferentes ejecuciones. Se comentarán también los problemas que hemos ido encontrando, y cómo se han ido resolviendo.

4.1. Entorno de pruebas

Tras estudiar el uso del entorno local para la ejecución de las pruebas, se descartó y decidió usar Google Colab, aunque por lo complejo de los experimentos, se tuvo que mejorar el entorno al nivel de pago Pro. De esta forma se conseguía acceso a una GPU, necesaria para la implementación de los algoritmos de predicción de compatibilidad, y una memoria RAM de hasta 83GB.

Una de las razones para elegir este entorno fue la facilidad para el trabajo de experimentación que permiten los notebooks de Jupyter, que posibilitan crear un documento legible que represente los pasos para llevar a cabo las ejecuciones de las pruebas. Además, las librerías necesarias para la implementación de este proyecto se encontraban ya disponibles en el entorno de Google Colab, que a la vez hace posible ejecutar los experimentos desde cualquier dispositivo, con el único requisito de tener conexión a internet.

Recurso	Características
S.O.	Ubuntu 22.04.2 LTS
Versión Python	3.10.6
CPU	Intel(R) Xeon(R)
GPU	NVIDIA T4/P100
RAM	83GB

Tabla 4.1: Especificaciones técnicas del entorno de pruebas Google Colab Pro.

No obstante, esta herramienta presenta algunos inconvenientes, en cualquiera de sus versiones (Pro o Pro+). Por ejemplo, la compartición de recursos, el límite de tiempo de ejecución en las sesiones

y el límite en la RAM. Estas características han influido desfavorablemente en el desarrollo del proyecto, sin embargo, no se ha descartado este entorno ya que cumple con los requisitos funcionales del proyecto, a diferencia del entorno local.

La manera en la que se han abordado estos problemas han sido: enlazando los notebooks con una cuenta de Google Drive, para que se pueda leer y guardar la información cada vez que se inicia la sesión, en lugar de perder la información cuando se desconecta el servidor; dividir las ejecuciones en pasos más pequeños para que no superen el límite de tiempo, contratar el nivel Pro para poder aumentar la memoria, y modificar el parámetro -e en los entrenamientos, reduciendo el número de épocas que permitiera que la ejecución finalizara sin problemas de tiempo o memoria.

4.2. Experimentos

En esta sección, se analizan los resultados obtenidos en el proyecto desarrollado. Para ello se compara el modelo estudiado usando varios subconjuntos de datos y analizando métricas de evaluación estándar en la literatura.

4.2.1. Datos usados en los experimentos

Como ya comentamos en la Sección 3.3.2 el dataset utilizado es el de Amazon, a pesar de que como se comentó en la Sección 3.1.1, en un principio el sistema podría admitir otros datasets. Sin embargo, problemas a la hora de extraer las características visuales del dataset Polyvore, han hecho inviable la ejecución con este conjunto de datos. El error en concreto es el siguiente:

```
RuntimeError: output with shape [1, 224, 224] doesn't match
the broadcast shape [3, 224, 224]
```

Este fallo en la ejecución se debe a que la primera dimensión del tensor, que representa el color, es 1 (imagen en escala de grises) en lugar de 3 (imagen RGB). Teniendo esto en cuenta, se realizaron algunos cambios en el código para intentar subsanar el error, identificando las imágenes corruptas (no fallaba para todas las imágenes procesadas). Aun así, no se consiguió ni localizar la imagen (las formas que se usaron para buscar, no obtenían ningún resultado) ni arreglar ese problema cambiando el código (en concreto, el cambio modificado, pero que sigue dando fallo en algunas imágenes, se puede ver en el Código 4.1).

Una vez se descartó el conjunto de datos Polyvore, se realizaron pruebas con el dataset de Amazon. En este caso, también hubo problemas al ejecutar el código pero se arreglaron usando el módulo de compatibilidad de TensorFlow (`tensorflow.compat.v1`) que permite ejecutar en la versión presente en Google Colab (TensorFlow 2.x), el código del proyecto (TensorFlow 1.x). Además, como parte

Código 4.1: Código modificado en extract_features.py para intentar arreglar problema con Polyvore.

```
1  im = skimage.io.imread(image_path)
2  if len(im.shape) == 2:
3      im = gray2rgb(im)
4  if im.shape[2] == 4:
5      im = rgba2rgb(im)
6
7  if im.shape[2] == 1:
8      print('Antes:' + image_path)
9
10 im = resize(im, (256, 256))
11 im = img_as_ubyte(im)
12
13 if im.shape[2] == 1:
14     print('Despues' + image_path)
```

de estas modificaciones para conseguir compatibilidad, había que deshabilitar un tipo de ejecución (incluir el código `tf.compat.v1.diable_eager_execution()`).

Además, como se puede ver en el Código 4.2, también se tuvo que arreglar un problema al indexar matrices, ya que las variables eran de tipo double, y eso no está permitido, por lo que hay que hacer un casting explícito a entero (líneas 126-127, 140-141).

Una vez se pudo ejecutar por completo el dataset de Amazon, pasó a ser procesado por el módulo de procesamiento, donde se generan 4 subdatasets que formarán los modelos a entrenar y evaluar. Estos son:

- Women bought together. Contiene la matriz de adyacencia con la información sobre artículos de moda categorizados como “de mujer”, utilizando el atributo de cada producto denominado ’bought_together’ (ver Figura 3.5). De esta manera, se aprovechan las relaciones de los productos que se han *comprado juntos*.
- Men bought together. Contiene la matriz de adyacencia con la información sobre artículos de moda categorizados como “de hombre”, y sus relaciones de *comprado junto*, como en el caso anterior.
- Women also bought. Contiene la matriz de adyacencia con la información sobre artículos de moda categorizados como “de mujer”, en esta caso utilizando el atributo de cada producto denominado ’also_bought’; de esta manera se utilizan las relaciones de *también comprado*

Código 4.2: Código modificado en test_amazon.py para intentar arreglar problema de índices.

```
123         available_adj = dp.full_valid_adj + dp.full_train_adj
124         available_adj = available_adj.tolil()
125         for r,c in zip(test_r_indices, test_c_indices):
126             r = int(r)
127             c = int(c)
128             available_adj[r,c] = 0
129             available_adj[c,r] = 0
130         available_adj = available_adj.tocsr()
131         available_adj.eliminate_zeros()
132
133         G = Graph(available_adj)
134         get_edges_func = G.run_K_BFS
135
136         new_adj = sp.csr_matrix(full_adj.shape)
137         new_adj = new_adj.tolil()
138         for r,c in zip(test_r_indices, test_c_indices):
139             before = time.time()
140             r = int(r)
141             c = int(c)
142             if K > 0: #expand the edges
143                 nodes_to_expand = [r,c]
144                 for node in nodes_to_expand:
145                     edges = get_edges_func(node, K)
146                     for edge in edges:
147                         i, j = edge
148                         new_adj[i, j] = 1
149                         new_adj[j, i] = 1
150
151         new_adj = new_adj.tocsr()
```

al construir la matriz.

- Men also bought. Contiene la matriz de adyacencia con la información sobre artículos de moda categorizados como “de hombre”, y sus relaciones de *también comprado*.

Si revisamos la Figura 3.5, podemos observar que los conjuntos creados a partir del atributo ’bought_together’, al exigir una relación más fuerte entre los productos, contiene menos información. En los siguientes experimentos analizaremos cómo afectan estos distintos tamaños de información disponible a los modelos.

4.2.2. Comparativa de resultados

Para poder llegar a conclusiones significativas, y diferentes a las realizadas previamente sobre este dataset, se han hecho las siguientes variaciones sobre los argumentos de train.py y test_amazon.py:

train.py

- -e. El número de épocas se fija a 500, de tal forma que se realiza un buen entrenamiento, con unos tiempos y un consumo de memoria adecuados.
- -amzd. Es el primer parámetro en variar. Para cada opción de dataset (“Men_also_bought”, “Women_also_bought”, “Women_bought_together”, “Men_bought_together”) se realizan 18 iteraciones, combinando todos los demás parámetros.
- -deg. El grado varía entre 1, 5 y 10.
- -lr. La tasa de aprendizaje se prueba con 0.001, 0.005 y 0.01.
- -hi. El número de unidades ocultas varía entre [50, 50, 50] y [350, 350, 350].

test_amazon.py

- -k. La profundidad del vecindario varía entre 0, 3 y 10.
- -amzd. Variamos este parámetro, bien para hacer coincidir la predicción de un dataset con su evaluación, o para evaluar una predicción de un género con un dataset del género diferente (siendo consistente con el tipo de relación entre productos).

Efecto de grado

Como se puede observar en la Tabla 4.2 para grados de vecindad más pequeños se consiguen mejores predicciones para los distintos datasets (en negrita). En este caso, en el marco de evaluación k=0, o sea, sin tener en cuenta el contexto, se consiguen mejores resultados en modelos entrenados con profundidad de vecinos. Los resultados con contexto se analizan después.

Podemos concluir por tanto, que un grado de vecindad alto en el entrenamiento, no ayuda a mejorar

Grado	WBT	MBT
1	56.933	**73.302**
5	**60.459**	64.522
10	51.028	65.150

Tabla 4.2: Efecto de grado con bajo learning rate (0.001), fijando la arquitectura ([50, 50, 50]), donde WBT indica *Women bought together* y MBT *Men bought together*. Nótese que la métrica va de 0 a 100, donde un valor más alto es mejor (marcado en negrita).

Grado	WBT	MBT
1	**77.706**	**79.134**
5	71.341	78.134

Tabla 4.3: Efecto de grado con alto learning rate (0.01), resto de valores y notación como Tabla 4.2.

la precisión en la predicción, ya que se crearán *embeddings* con una gran cantidad de información, que lejos de ayudar, introducen ruido en los cálculos de compatibilidad. Esta situación no mejora al aumentar el learning rate (Tabla 4.3).

Efecto de arquitectura y learning rate

En este experimento vamos a estudiar cómo influyen en los resultados la arquitectura elegida y el valor del learning rate. Para ello, la Tabla 4.4 muestra los resultados obtenidos tomando el grado que ganaba más veces en el experimento anterior, es decir, grado 1.

Observamos que, al considerar el *learning rate* más bajo (lo que implica pasos más pequeños hacia la convergencia) se obtienen peores resultados. Sin embargo, según se van eligiendo valores mayores, se ve claramente que se obtiene un acierto mayor. Esto es debido a la relación tan importante entre el lr y el número de épocas. Tener un número de épocas "reducido" implica que el algoritmo no tiene tantas opciones de recorrer todos los posibles cambios para obtener una predicción óptima y, por lo tanto, un mayor lr permite cambios más grandes en los parámetros dentro del algoritmo de entrenamiento, por lo que llega a probar mayor variedad de valores, obteniendo así una mejor predicción.

Por otro lado, el parámetro hi o arquitectura de las capas, no produce grandes variaciones sobre el modelo. Parece que con menos capas funciona lo suficientemente bien, aunque en los experimentos completos que hemos obtenido, observamos que al aumentar el grado de entrenamiento esta diferencia no está tan clara, y se pueden conseguir mejoras con respecto a grados más bajos.

Por último, podemos concluir que en la relación de *bought_together*, el dataset de hombres obtiene mejores resultados en general que el de mujeres, indicando que son un subconjunto de la población donde es más fácil acertar (al menos en este dominio). Como veremos en el resto de experimentos, esta tendencia se sigue cumpliendo.

Datos:	Women bought together		Men bought together	
Arquitectura:	[50,50,50]	[350,350,350]	[50,50,50]	[350,350,350]
0.001	56.933	58.926	73.302	67.252
0.005	69.830	61.450	78.006	70.674
0.01	**77.706**	**66.278**	**79.134**	**73.122**

Tabla 4.4: Efecto de arquitectura y learning rate, tomando degree = 1. Nótese que el mejor valor para un dataset se marca con un subrayado.

Efecto de k

Los experimentos incluidos hasta ahora han utilizado k=0, es decir, no se ha usado el contexto del producto para la predicción. Sin embargo, al igual que reportan los autores de [1], la precisión del algoritmo mejora claramente al aumentar los valores de k.

Como podemos ver en la Tabla 4.5, donde hemos incluido los otros dos datasets que no se habían analizado hasta ahora (Women also bough o WAB, Men also bought o MAB), incluso tomando valores de learning rate diferentes, se obtienen mejores resultados con un k mayor. Esto se debe a que al dar la oportunidad de acceder a un mayor número de vecinos del producto (contexto), sea cual sea el grado de profundidad usado en el entrenamiento, se consigue una exactitud mayor en la tarea de predicción. Esto se puede corroborar con los resultados de la Tabla 4.6, donde se reporta para dos de estos datasets, los resultados con el mejor learning rate (0.01) y un grado mayor (5).

Por último, y al igual que comentamos antes, los datasets con información de hombres son más fáciles de acertar en general (excepto los casos de $k > 0$ de la Tabla 4.6). Una hipótesis que no hemos podido contrastar y que sería interesante hacer en el futuro es que quizá este colectivo interactúa con conjuntos más estables y homogéneos de productos, haciendo que la tarea de predicción sea más sencilla al haber menos ruido.

Learning rate:	0.001				0.01			
Datos:	WBT	MBT	WAB	MAB	WBT	MBT	WAB	MAB
0	56.933	73.302	51.776	55.231	77.706	79.134	55.710	64.600
3	88.086	90.579	85.225	89.022	89.059	91.079	86.884	91.840
10	**89.059**	**91.079**	**93.447**	**94.775**	**93.115**	**91.989**	**94.422**	**95.697**

Tabla 4.5: Efecto de k en la evaluación, tomando degree = 1 y arquitectura [50, 50, 50].

Eficiencia (tiempo de entrenamiento)

Como podemos observar en el gráfico de la Figura 4.1, el grado de profundidad en el entrenamiento (d) es el parámetro que más afecta a los tiempos de ejecución que tan importantes han sido en este proyecto. La tasa de aprendizaje también afecta algo a los tiempos, como comentábamos en su des-

k	WBT	MBT
0	71.341	78.134
3	92.967	92.553
10	**93.286**	**92.566**

Tabla 4.6: Efecto de k en la evaluación, tomando degree = 5, arquitectura [50, 50, 50] y lr = 0.01.

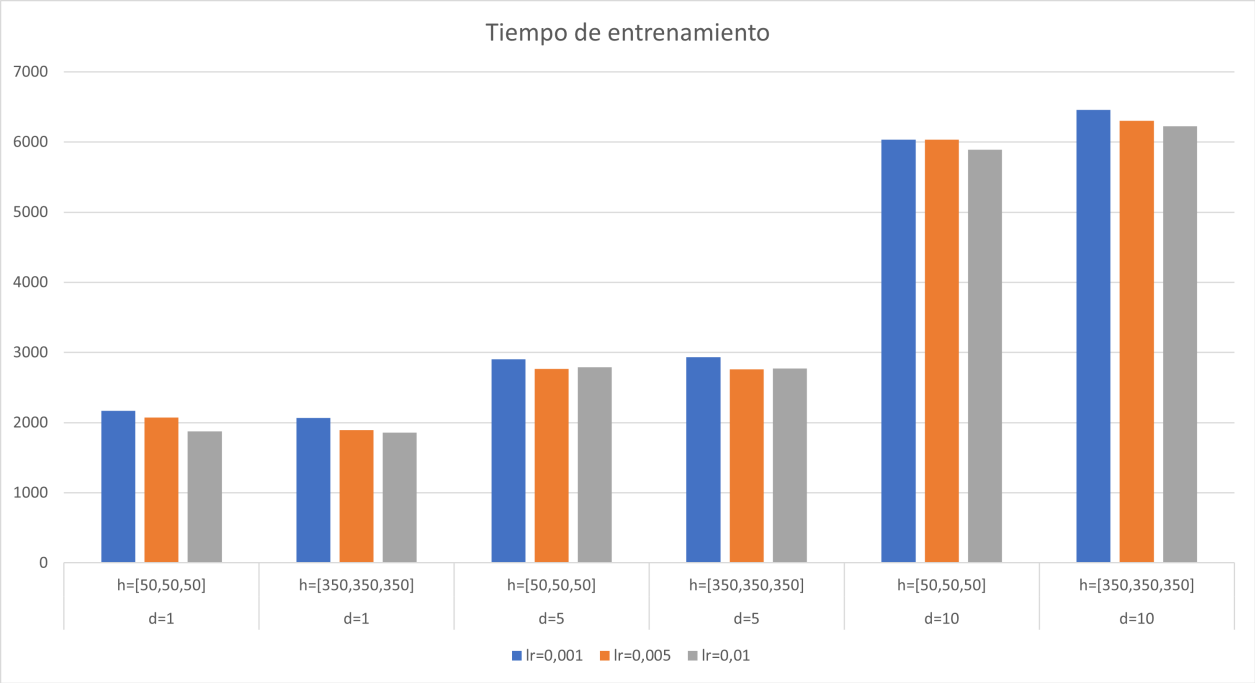

Figura 4.1: Comparación de tiempos de entrenamiento para datos *Women bought together*.

cripción en la Sección 3.3.3, encajando con la teoría: cuanto menor es esta, mayor coste de tiempo en la ejecución. Sin embargo, como hemos comprobado antes, en estos experimentos con menos épocas, es mucho más conveniente usar lr mayores, ya que ahorran tiempo y obtienen mejores resultados.

Evaluaciones de género cruzado

Como ya comentamos en la Sección 3.3.4, en este proyecto se permite mezclar los conjuntos de entrenamiento y evaluación de manera que no correspondan a los mismos datasets. Para comprobar esto, en la Tabla 4.7 hemos probado los dos casos más extremos: entrenar con datos de mujeres y evaluar con hombres y viceversa (en los dos casos, usando los datos de *bought together*). Estos valores hay que compararlos con los obtenidos en la Tabla 4.5, donde cada entrenamiento se evaluó con su mismo conjunto de datos.

Teniendo esto en cuenta, observamos que, como es de esperar, estos resultados son más bajos que los anteriores, sin embargo, cuanta más información de contexto se utiliza, el modelo es más robusto, ya que se puede adaptar mejor a estilos o tipos de ropa desconocidos [1].

Además, en línea con resultados anteriores, cuando se evalúan a los hombres es cuando mejores resultados se obtienen, evidenciando un sesgo claro hacia este tipo de usuarios.

k	Women $\rightarrow$ Men	Men $\rightarrow$ Women
0	68.021	61.272
3	80.710	73.304
10	81.530	73.630

Tabla 4.7: Evaluación de género cruzado, usando d=1, lr=0.01 y arquitectura [50, 50, 50]. En cada columna, para el caso A $\rightarrow$ B, se ha entrenado con A y se evalúa con B.

5

Conclusiones y trabajo futuro

En esta sección se exponen las conclusiones a las que se han llegado a lo largo del proyecto. Además, se hace un breve resumen de los posibles trabajos a realizar en el futuro.

5.1. Conclusiones

La realización de este trabajo fue desde el principio muy vocacional, teniendo un gran interés en aprender más sobre aplicaciones tecnológicas a un mundo tan artístico. La moda es una forma de expresión y manifestación cultural, basada en las tendencias, y en gustos subjetivos y únicos de cada persona, y donde la compatibilidad de un outfit varía con el paso del tiempo y la localización geográfica. Es por eso, que desde un inicio se conocía la complejidad que implicarían los algoritmos usados en recomendación de moda. No obstante, se quería estudiar algo más novedoso, de ahí que surgiera la posibilidad de analizar el problema de predecir o recomendar según la compatibilidad en un outfit. Es una tarea que debe tener en cuenta más información que los clics de un usuario en los productos, o las calificaciones que estos puedan tener, donde ni siquiera funciona de manera óptima la obtención de compatibilidad entre un par de artículos.

Intentando abordar estas limitaciones con técnicas recientes de recomendación y procesamiento de datos, se investigó un sistema que necesita basarse en las características visuales de un artículo. Este sistema, además, tiene en cuenta el resto de prendas del atuendo (outfit) como el contexto en que se enmarcaba el producto a comparar o recomendar.

Personalmente, ha sido un proceso largo y complicado hasta encontrar una forma de implementar y probar todo lo estudiado, ya que debido a inconvenientes con el entorno, o con las versiones de los distintos códigos explorados en los que se quería basar el proyecto, tuvimos que replantear el trabajo en distintos puntos del mismo.

Sin embargo, los resultados han sido positivos, como se ha mostrado en el último capítulo. Ahí, hemos confirmado que cuanto mayor es el contexto, mejores resultados se obtienen. Esto, unido a otra variable importante, la tasa de aprendizaje, enmarcada siempre en las circunstancias de nuestro proyecto con pocos recursos (lo que se traduce en un número de épocas manejable), demuestra que

un mayor *lr* funciona mejor, a pesar de las premisas teóricas previas a la investigación. Por otro lado, se concluye que para el dataset usado en los experimentos (Amazon), realizar predicciones de compatibilidad entre dos artículos de la categoría masculina, tiene siempre una mayor tasa de éxito que para la femenina.

Aunque no se ha conseguido proponer un método novedoso como solución a este problema, se han sentado las bases sobre las tecnologías que funcionan y parecen fiables de cara a resolver esta tarea, así como los conjuntos de datos y formas de evaluar de las que se puedan partir en el futuro.

5.2. Trabajo futuro

A continuación, detallaremos algunos de los aspectos que, tras mucha investigación, consideramos que puede ser interesante tener en cuenta de cara a próximos trabajos, pero que por falta de tiempo y/o recursos no hemos podido entrar en detalle en esta ocasión, a pesar de que nos habría gustado intentarlo.

Por un lado, y partiendo del estudio realizado en este trabajo, sería interesante construir una herramienta completa o *framework* orientada a la tarea de predicción de compatibilidad, que permitiera comparar una amplia variedad de técnicas, a poder ser con más de un conjunto de datos. Si bien en este trabajo, debido a las complicaciones técnicas y de recursos ya mencionadas, sólo se ha podido probar y analizar un modelo, convendría comparar las opciones disponibles en la literatura entre sí, analizando si todos se comportan igual frente a características de los datos, como la densidad, o cómo de sensibles son a los métodos utilizados para conseguir los *embeddings* de las características visuales de las imágenes.

Por otro lado, y siguiendo las tendencias actuales donde los sesgos y el aprendizaje ético son cada vez más importantes [21], se podría hacer un estudio sobre los sesgos que tienen los algoritmos tradicionales de recomendación en moda, y en concreto los que utilizan la compatibilidad de los atuendos. Estos sesgos, como ocurre en los sistemas de recomendación tradicionales [22], se pueden estudiar desde el punto de vista del usuario que recibe la recomendación (si los hombres o las mujeres reciben un tipo de recomendación de manera injustificada) o del producto (si hay productos que se recomiendan siempre o nunca, independientemente de su calidad o características). Para poder realizar estos estudios, sin embargo, es necesaria información de los usuarios que, habitualmente, es privada, como su sexo o su edad, por lo que quizá requiere la obtención de un conjunto de datos específicamente para este análisis.

Por último, y teniendo en cuenta algo que hemos echado en falta en los artículos que hemos leído al respecto, faltan estudios donde la evaluación se realice directamente con usuarios, es decir, a base de encuestas o preguntas, y que no consistan únicamente en evaluaciones offline, que son las más habituales en los campos del Aprendizaje Autómatico y los Sistemas de Recomendación por

su facilidad para ejecutar y repetir experimentos. Consideramos que estos tipos de estudios pueden arrojar algo de luz al problema de la compatibilidad, evitando sesgos inherentes de los conjuntos de datos o de cómo se han construido.

Bibliografía

[1] G. Cucurull, P. Taslakian, and D. Vázquez, "Context-aware visual compatibility prediction," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pp. 12617–12626, Computer Vision Foundation / IEEE, 2019.

[2] S. R. Department, "Fashion e-commerce worldwide - statistics & facts." https://www.statista.com/topics/9288/fashion-e-commerce-worldwide/#topicOverview, Marzo 2023. Visitado última vez en Julio 2023.

[3] F. Ricci, L. Rokach, and B. Shapira, "Recommender systems: Techniques, applications, and challenges," in *Recommender Systems Handbook* (F. Ricci, L. Rokach, and B. Shapira, eds.), pp. 1–35, Springer US, 2022.

[4] R. D. Burke, "Hybrid web recommender systems," in *The Adaptive Web, Methods and Strategies of Web Personalization* (P. Brusilovsky, A. Kobsa, and W. Nejdl, eds.), vol. 4321 of *Lecture Notes in Computer Science*, pp. 377–408, Springer, 2007.

[5] C. Musto, M. de Gemmis, P. Lops, F. Narducci, and G. Semeraro, "Semantics and content-based recommendations," in *Recommender Systems Handbook* (F. Ricci, L. Rokach, and B. Shapira, eds.), pp. 251–298, Springer US, 2022.

[6] A. N. Nikolakopoulos, X. Ning, C. Desrosiers, and G. Karypis, "Trust your neighbors: A comprehensive survey of neighborhood-based methods for recommender systems," in *Recommender Systems Handbook* (F. Ricci, L. Rokach, and B. Shapira, eds.), pp. 39–89, Springer US, 2022.

[7] R. R. Sinha and K. Swearingen, "Comparing recommendations made by online systems and friends," in *Proceedings of the Second DELOS Network of Excellence Workshop on Personalisation and Recommender Systems in Digital Libraries, DELOS 2001, Dublin, Ireland, June 18-20, 2001* (A. F. Smeaton and J. Callan, eds.), vol. 01/W03 of *ERCIM Workshop Proceedings*, ERCIM, 2001.

[8] C. C. Aggarwal, *Recommender Systems - The Textbook*. Springer, 2016.

[9] S. Jaradat, N. Dokoohaki, H. J. C. Pampín, and R. Shirvany, "Fashion recommender systems," in *Recommender Systems Handbook* (F. Ricci, L. Rokach, and B. Shapira, eds.), pp. 1015–1055, Springer US, 2022.

[10] Y. Deldjoo, F. Nazary, A. Ramisa, J. J. McAuley, G. Pellegrini, A. Bellogín, and T. D. Noia, "A review of modern fashion recommender systems," *CoRR*, vol. abs/2202.02757, 2022.

[11] X. Han, Z. Wu, Y. Jiang, and L. S. Davis, "Learning fashion compatibility with bidirectional lstms," in *Proceedings of the 2017 ACM on Multimedia Conference, MM 2017, Mountain View, CA, USA, October 23-27, 2017* (Q. Liu, R. Lienhart, H. Wang, S. K. Chen, S. Boll, Y. P. Chen, G. Friedland, J. Li, and S. Yan, eds.), pp. 1078–1086, ACM, 2017.

[12] J. Beel and S. Langer, “A comparison of offline evaluations, online evaluations, and user studies in the context of research-paper recommender systems,” in *Research and Advanced Technology for Digital Libraries - 19th International Conference on Theory and Practice of Digital Libraries, TPDL 2015, Poznań, Poland, September 14-18, 2015. Proceedings* (S. Kapidakis, C. Mazurek, and M. Werla, eds.), vol. 9316 of *Lecture Notes in Computer Science*, pp. 153–168, Springer, 2015.

[13] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. Riedl, “Evaluating collaborative filtering recommender systems,” *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 5–53, 2004.

[14] A. Gunawardana, G. Shani, and S. Yogev, “Evaluating recommender systems,” in *Recommender Systems Handbook* (F. Ricci, L. Rokach, and B. Shapira, eds.), pp. 547–601, Springer US, 2022.

[15] X. Han, Z. Wu, Y. Jiang, and L. S. Davis, “Learning fashion compatibility with bidirectional lstms,” in *Proceedings of the 2017 ACM on Multimedia Conference, MM 2017, Mountain View, CA, USA, October 23-27, 2017* (Q. Liu, R. Lienhart, H. Wang, S. K. Chen, S. Boll, Y. P. Chen, G. Friedland, J. Li, and S. Yan, eds.), pp. 1078–1086, ACM, 2017.

[16] M. I. Vasileva, B. A. Plummer, K. Dusad, S. Rajpal, R. Kumar, and D. A. Forsyth, “Learning type-aware embeddings for fashion compatibility,” *CoRR*, vol. abs/1803.09196, 2018.

[17] N. Rostamzadeh, S. Hosseini, T. Boquet, W. Stokowiec, Y. Zhang, C. Jauvin, and C. Pal, “Fashion-gen: The generative fashion dataset and challenge,” *CoRR*, vol. abs/1806.08317, 2018.

[18] J. J. McAuley, C. Targett, Q. Shi, and A. van den Hengel, “Image-based recommendations on styles and substitutes,” *CoRR*, vol. abs/1506.04757, 2015.

[19] J. J. McAuley, C. Targett, Q. Shi, and A. van den Hengel, “Image-based recommendations on styles and substitutes,” in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015* (R. Baeza-Yates, M. Lalmas, A. Moffat, and B. A. Ribeiro-Neto, eds.), pp. 43–52, ACM, 2015.

[20] R. He and J. J. McAuley, “Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering,” in *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016* (J. Bourdeau, J. Hendler, R. Nkambou, I. Horrocks, and B. Y. Zhao, eds.), pp. 507–517, ACM, 2016.

[21] T. D. Noia, N. Tintarev, P. Fatourou, and M. Schedl, “Recommender systems under european AI regulations,” *Commun. ACM*, vol. 65, no. 4, pp. 69–73, 2022.

[22] M. D. Ekstrand, A. Das, R. Burke, and F. Diaz, “Fairness in recommender systems,” in *Recommender Systems Handbook* (F. Ricci, L. Rokach, and B. Shapira, eds.), pp. 679–707, Springer US, 2022.


UAM
UNIVERSIDAD AUTONOMA
DE MADRID