Recomendación de Videojuegos: Análisis comparativo de modelos

Rene Saavedra, Carlos Olguín, Geraldine Coli
Departamento de Ciencia de la Computación, Pontificia Universidad Católica de Chile, Santiago, Chile
{rsaavedrav, carlosfelipeolguin, gecoli}@uc.cl

Abstract

This work implements and compares video game recommendation models on Steam, evaluating techniques such as LightFM and LightGCN. Using a dataset that includes reviews, metadata, and user features, systems based on collaborative and content-based filtering were developed. Metrics such as precision, diversity, and novelty were applied, along with sensitivity analyses on hyperparameters and preprocessing. The results show that LightGCN significantly outperforms classic baselines, while LightFM faces structural limitations. [1]

1. Introducción

Los sistemas de recomendación buscan entregar sugerencias personalizadas utilizando datos de usuarios e ítems. En dominios como los videojuegos donde las preferencias son diversas y las interacciones altamente dispersas, el diseño de un sistema eficaz se vuelve especialmente desafiante.

Una estrategia efectiva es incorporar el *contexto usuario–ítem*, incluyendo atributos como géneros, estilos de juego, precios o comportamiento histórico del usuario. Esto permite enriquecer los modelos con señales semánticas y mitigar problemas clásicos como el *cold-start* (2).

Además, modelos basados en grafos como LightGCN (3), simplifican la arquitectura de redes de convolución al enfocarse en la propagación de representaciones latentes, mostrando mejoras consistentes en tareas de recomendación colaborativa densa.

1.1. Motivación

El catálogo dinámico y la alta esparsidad de interacciones en Steam impiden que métodos tradicionales basados sólo en historial funcionen adecuadamente. Por ello, surge la necesidad de comparar modelos avanzados con capacidad de integrar contenido contextual y representar relaciones implícitas más complejas.

2. Estado del Arte

Los sistemas de recomendación han evolucionado desde enfoques clásicos basados en filtrado colaborativo hasta modelos avanzados que incorporan redes neuronales profundas y representaciones en grafos. Esta sección resume las principales formas actuales de abordar la problemática:

Filtrado colaborativo y factorización de matrices: Técnicas como SVD o SVD++ descomponen la matriz usuario-ítem en factores latentes que capturan patrones de preferencia (1). Son efectivas pero sensibles a la esparsidad, lo que limita su aplicabilidad en escenarios como videojuegos.

Modelos basados en contenido e híbridos: Estos modelos utilizan atributos de los ítems (como género o precio) y/o características de usuarios para mejorar recomendaciones, especialmente en escenarios de *cold-start*. LightFM es un ejemplo híbrido que combina factorización de matrices con *embeddings* contextuales (5).

Tendencias actuales: La tendencia dominante es hacia enfoques híbridos que integran señales colaborativas, contenido, estructuras de grafo y secuencia. Estos métodos buscan un equilibrio entre rendimiento, interpretabilidad y adaptabilidad a distintos dominios.

3. Dataset

Antes de llegar a una fuente de información definitiva, se hicieron dos pivotes: uno basado en datos de recomendación de videojuegos de Amazon (2020) y luego uno de recomendación de artículos relacionados a videojuegos de Amazon (2023). No obstante, al final nos decantamos por utilizar el *dataset* **Game Recommendations on Steam**, el cual

[1] Repositorio disponible en: https://github.com/ReneSaavedraV/IIC3633-Proyecto/tree/main

contiene datos de usuarios, videojuegos y sus interacciones. Este conjunto está compuesto por múltiples tablas:

- `games`: metadatos de cada juego (nombre, fecha de lanzamiento, plataformas, precios, rating, etc.)
- `users`: identificadores de usuarios y métricas de actividad.
- `recommendations`: reseñas de juegos por usuarios con etiquetas como *is_recommended*, *hours_played*, y *funny/helpful*.
- `games_metadata`: descripciones y etiquetas (géneros, mecánicas) asociadas a los juegos.

Table 1. Resumen de las tablas del dataset original

Tabla	Descripción
games (50,872×13)	Información de videojuegos
users (14,306,064×3)	Datos básicos de usuarios
recommendations (41,154,794×8)	Interacciones usuario–juego
games_metadata (50,872×3)	Descripciones y etiquetas

3.1. Preprocesamiento y muestreo

Se eliminaron columnas redundantes y se normalizaron valores numéricos como *hours played* y *positive ratio*. Se aplicó codificación *one-hot* a etiquetas categóricas como géneros y plataformas.

Table 2. Columnas originales por tabla del dataset

Tabla	Columnas
`games`	`app_id`, `title`, `date_release`, `win`, `mac`, `linux`, `rating`, `positive_ratio`, `user_reviews`, `price_final`, `price_original`, `discount`, `steam_deck`
`users`	`user_id`, `products`, `reviews`
`recommendations`	`user_id`, `app_id`, `is_recommended`, `hours`, `helpful`, `funny`, `review_id`, `date`
`games_metadata`	`app_id`, `description`, `tags`

Table 3. Columnas finales utilizadas tras preprocesamiento

Columna	Tipo	Descripción / Transformación
`user_id`	Categórica	ID único de usuario
`app_id`	Categórica	ID único de videojuego
`is_recommended`	Binaria	1 si el usuario recomienda el juego
`hours`	Numérica (escalada)	Horas jugadas (MinMaxScaler)
`positive_ratio`	Numérica (escalada)	Proporción de reseñas positivas
`price_final`	Numérica (escalada)	Precio final del juego
`rating_score`	Ordinal (codificada)	Conversión de rating textual a puntaje
`win`, `mac`, `linux`	Binarias	Disponibilidad por plataforma (1/0)
`tags_onehot`	Multihot (binaria)	Codificación de etiquetas (*genres*, *mechanics*)
`user_reviews_count`	Numérica (escalada)	Número de reseñas escritas por el usuario
`game_review_count`	Numérica (escalada)	Número de reseñas recibidas por el juego

4. Metodología

4.1. Preprocesamiento y filtrado

Partimos de un dataset de reseñas de Steam que incluye usuarios, juegos, interacciones (recomendaciones binarias), metadatos (plataforma, género, *rating*, número de reseñas, descripción) y etiquetas.

Aplicamos un muestreo del 10% por limitaciones de *hardware*, luego filtramos para manejar el *cold-start*, usando parámetros:

$$\text{MIN\_REVIEWS\_USER} = 150$$

$$\text{MIN\_REVIEWS\_ITEM} = 200$$

Creamos un dataframe conjunto que incluye solo usuarios y juegos con cantidad de interacciones mayor o iguales al umbral de reseñas, y eliminamos columnas irrelevantes. El resultado final fue de aproximadamente 6 000 interacciones entre 2 800 usuarios y 900 juegos.

4.2. Construcción de interacciones y features

Definimos interacciones estructuradas como tripletas (usuario, ítem, *is_recommended*). Además, generamos:

- **User-features**: promedio de horas jugadas y cantidad de reseñas, escaladas con MinMaxScaler.
- **Item-features**: cantidad de reseñas, *rating* convertido a escala numérica, ratio positivo, plataformas (win/mac/linux) y etiquetas *one-hot*. Todas escaladas.

4.3. Modelos evaluados

Se entrenaron y compararon tres métodos:

LightFM

- Arquitectura híbrida con *embeddings* por colaborativo y contenido.
- Hiperparámetros explorados: {*learning_rate* = 0.05, *no_components* = 128, *epochs* = 100}.
- Evaluación en *hold-out* (20% test), midiendo Precision@10, Recall@10, MAP@10, NDCG@10, diversidad, novedad y popularidad.

Baselines

- *"Most Popular"*: se recomienda a todos los usuarios el top-10 de juegos más populares (reseñas ≥ 4).
- *"Random"*: recomendaciones aleatorias sin reemplazo.

LightGCN

- Grafo bipartito usuario–ítem con filtros positivos.

- Red neuronal GNN personalizada ('torch_geometric') con 2–5 capas y dimensiones de embedding en {64,128,256}.
- Se entrenó mediante pérdida BPR sobre batches de tamaño 1 024, optimizando con Adam y tasas de learning en $\{0.001, 0.005, 0.01\}$ por 50 epochs.
- Evaluación mediante NDCG, MAP, diversidad y novedad, usando score matrix y cálculo directo de métricas.

4.4. Análisis de sensibilidad

Exploramos combinaciones de:

$$(\text{dim}_{\text{emb}} \in \{64, 128, 256\}),$$
$$(\text{layers} \in \{2, 3, 5\}),$$
$$(\text{lr} \in \{0.001, 0.005, 0.01\})$$

para LightGCN, evaluando pérdida de entrenamiento y desempeño en métricas de *ranking*. Además, una vez identificada la mejor combinación de hiperparámetros, analizamos la repercusión de un cambio en MIN_REVIEWS_USER ∈ {0, 50, 100, 150, 200} y MIN_REVIEWS_ITEM ∈ {0, 50, 100, 150, 200}.

Lastimosamente, para LightFM, la forma de codificar el texto causó que el tiempo de entrenamiento y recomendación rondara los 40 minutos, por lo que no se pudo hacer un análisis de sensibilidad respeto a sus hiperparámetros dado los cambios de *dataset* y el tiempo acotado de experimentación.

4.5. Métricas de evaluación

Para evaluación usamos:

$$\text{Precision@10, Recall@10, MAP@10, NDCG@10,}$$
$$\text{Diversidad} = 1 - \frac{1}{|U|K(K-1)} \sum_{u,i \neq j} \text{sim}(i,j),$$
$$\text{Novedad} = \frac{1}{|U|K} \sum -\log_2 \left(\frac{\#\text{interacciones con } i}{\#\text{usuarios}} \right).$$

Estas métricas evaluaron precisión y capacidad de descubrimiento de ítems no triviales.

5. Resultados

LightGCN

- Se variaron los hiperparámetros *embeddings (D), layers (L) y learning rate (LR)* para encontrar la configuración óptima. La tabla siguiente refleja los resultados, destacando los mejores valores en cada métrica:

Table 4. Comparación de métricas para LightGCN

Configuración	NDCG@10	MAP@10	Diversidad	Novedad
D256_L5_LR0.01	**0.7802**	**0.7098**	0.9520	10.7191
D64_L2_LR0.001	0.2988	0.2502	0.9073	11.2608
D128_L3_LR0.005	0.6007	0.5235	0.9410	10.9912
D256_L2_LR0.005	0.6640	0.5961	**0.9645**	10.8838
D128_L5_LR0.001	0.5563	0.4762	0.9420	**11.2620**

Dada la diferencia mínima entre diversidad y novedad, y su claro mejor desempeño en NDCG@10 y MAP@10, se selecciona la configuración **D256_L5_LR0.01** como la más adecuada para los análisis posteriores.

- La variación en los resultados de *coverage* y *precision-recall* en cuanto al tamaño de la lista de recomendación K también fue evaluada:

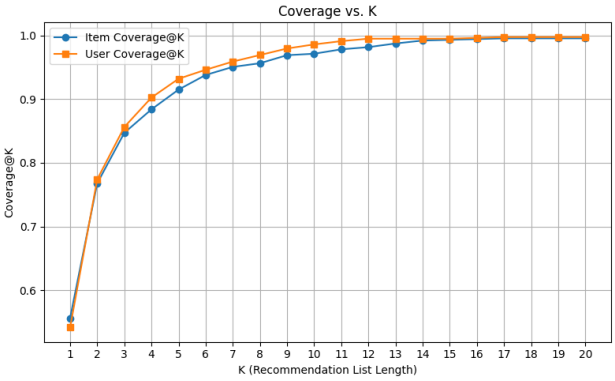

Figure 1. Análisis de cobertura vs. tamaño de lista K

Coverage@K, tanto por ítems como por usuarios, se estabiliza rápidamente, alcanzando más del 95% para $K \geq 10$, lo que evidencia una buena capacidad del modelo para generar recomendaciones diversas sin sacrificar calidad.

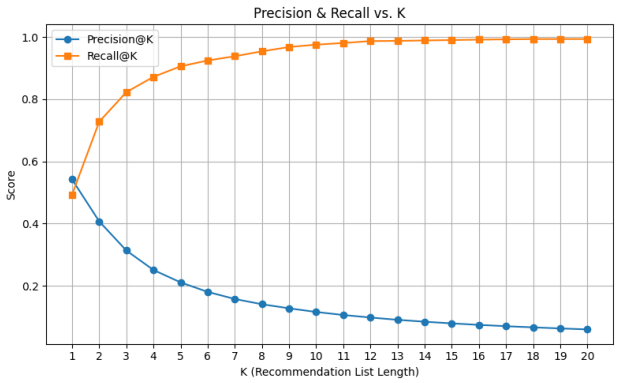

Figure 2. Precisión y recall vs. tamaño de lista K

Precision@K disminuye a medida que crece K, mientras que Recall@K aumenta, lo cual es un comportamiento esperado en sistemas de recomendación y refleja el *trade-off* inherente entre precisión y cobertura.

- Además, se evaluó la sensibilidad del modelo a los umbrales de filtrado de usuarios e ítems. El siguiente mapa de calor evidencia que el desempeño del modelo no es independiente de estos valores:

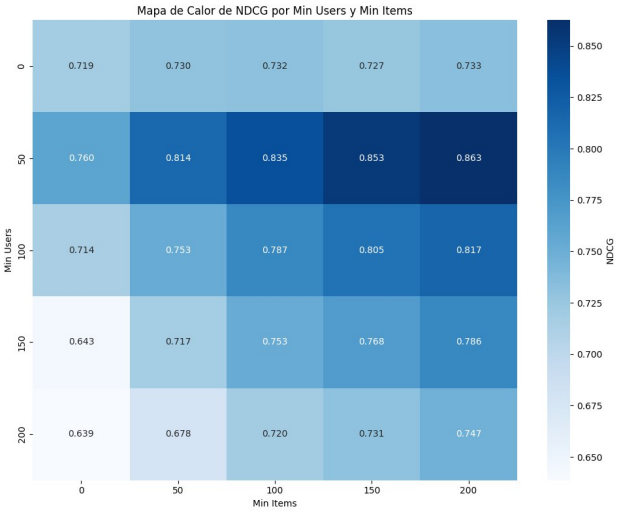

Figure 3. Impacto de MIN_REVIEWS_USER vs. MIN_REVIEWS_ITEM en cuanto a NDCG@10

La métrica NDCG@10 alcanza su punto óptimo cuando se aplica un filtrado moderado (MIN_REVIEWS_USER = 50, MIN_REVIEWS_ITEM = 200), indicando un equilibrio entre calidad y cantidad de interacciones.

Usuario	Ítems vistos	Recomendaciones
9268011	– Grim Fandango Remastered – Defend your life: TD	– Defend your life: TD – Grim Fandango Remastered – Astroflux – Armikrog – Deadly Premonition: The director's cut

Table 5. Ejemplo de recomendaciones personalizadas para un usuario.

- Ejemplo de recomendación:

  Las recomendaciones preservan la coherencia temática con las preferencias del usuario, combinando títulos de aventura (Armikrog, Deadly Premonition) y juegos casuales de estilo retro (Astroflux), alineados con el historial de interacción.

Comparación con otros modelos

- Ocupando las métricas antes mencionadas (NDGC, MAP, Diversidad y Novedad), se evaluaron los modelos *Most Popular, Random*, LightGCN y LightFM, manteniendo el umbral de MIN_REVIEWS_USER = 150 y MIN_REVIEWS_ITEM = 200.

Table 6. Comparación entre modelos

Modelo	NDCG	MAP	Diversidad	Novedad
Popular	0.051	0.034	0.894	**11.872**
Random	0.005	0.004	**0.994**	10.720
LightGCN	**0.7802**	**0.7098**	0.9520	10.7191
LightFM	0.0055	0.0027	0.0493	1.0493

Rendimiento deficiente de LightFM El modelo LightFM presentó un rendimiento notoriamente inferior. Esto puede explicarse por varias razones:

- **Representación binaria con alta esparsidad**: aunque se incluyeron interacciones negativas implícitas (valor 0), la matriz usuario–ítem sigue siendo extremadamente dispersa, lo que limita la capacidad del modelo para aprender sin técnicas de muestreo negativo explícito.
- **Desbalance entre *features***: mientras los ítems incluyen muchas variables (*tags*, *ratings*, plataformas), los usuarios fueron representados por sólo 2–3 atributos numéricos, lo que reduce la riqueza semántica de los *embeddings* de usuario.
- **Codificación *multihot* de tags**: esta codificación llevó a una alta dimensionalidad en las *features* de ítems. Sin reducción de dimensionalidad, esto genera ruido en lugar de señal útil.

– **Volumen de datos limitado**: el *dataset* final contiene apenas 6 000 interacciones. Esto es insuficiente para modelos como LightFM, que requieren mayor volumen para estabilizar el entrenamiento con *embeddings* grandes.

Discusión Los resultados muestran que LightGCN no solo supera ampliamente a los *baselines*, sino que también demuestra mayor robustez frente a la esparsidad y variaciones de filtrado. Si bien modelos como Random y Popular obtienen mayor diversidad o novedad, esto ocurre a costa de la relevancia.

También se observa un leve *trade-off* entre precisión y diversidad/novedad; las configuraciones que maximizan NDCG@10 tienden a recuperar ítems más populares, mientras que aquellas que priorizan diversidad entregan listas más variadas pero menos específicas. Este comportamiento es común en sistemas de recomendación y resalta la importancia de elegir métricas alineadas con los objetivos del dominio.

6. Conclusiones

Este trabajo exploró comparativamente el desempeño de modelos de recomendación aplicados al dominio de videojuegos, destacando los beneficios de arquitecturas basadas en grafos sobre métodos híbridos tradicionales.

Durante el proceso se enfrentaron diversos desafíos prácticos: desde la necesidad de pivotear el *dataset* original (Amazon Games) por baja calidad estructural, hasta los altos tiempos de entrenamiento de LightFM debido a la codificación *multihot* de etiquetas y gran cantidad de *features* dispersas. Esta situación impidió realizar un *tuning* exhaustivo de sus hiperparámetros, en contraste con LightGCN, que se adaptó mejor al entorno computacional disponible.

Nuestros resultados indican que LightGCN logra un desempeño robusto y generalizable en escenarios de alta esparsidad, superando significativamente a los *baselines* clásicos y a LightFM en métricas clave como NDCG, MAP y diversidad. Además, se evidenció la sensibilidad del modelo al preprocesamiento, especialmente al ajustar los umbrales de filtrado de usuarios e ítems.

En conjunto, este estudio proporciona evidencia empírica clara para preferir arquitecturas basadas en grafos en dominios complejos como los videojuegos, al tiempo que resalta la importancia de decisiones prácticas -como codificación y muestreo- en el éxito final de un sistema de recomendación.

Además, se resalta una vez más la problemática no menor de enfrentarse al clásico desafío del *cold-start*.

6.1. Recomendaciones para trabajos futuros

Si bien este estudio se enfocó en la comparación de modelos clásicos avanzados (LightFM y LightGCN) sobre un subconjunto del *dataset* de Steam, se identifican varias oportunidades para mejorar la calidad de las recomendaciones y el proceso experimental:

- **Reducción de dimensionalidad**: las etiquetas de los juegos fueron codificadas con esquema *multihot*, lo que generó una matriz dispersa de alta dimensión. Aplicar técnicas como PCA o TruncatedSVD sobre vectores TF-IDF permitiría comprimir esta información y evitar el sobreajuste en modelos como LightFM.
- **Codificación semántica de texto**: campos como la `description` o los `tags` podrían aprovecharse mejor mediante *embeddings* preentrenados (ej. BERT) o representaciones aprendidas de forma supervisada, capturando similitud semántica entre juegos.
- **Modelos más expresivos**: sería interesante explorar arquitecturas modernas como DeepFM, que combina interacciones cruzadas y redes profundas para capturar patrones de comportamiento más complejos. Estas alternativas podrían superar tanto a LightFM como a LightGCN en escenarios híbridos y dispersos.
- **Augmentación de datos e interacciones negativas**: el muestreo negativo controlado o técnicas contrastivas podrían mejorar la robustez de modelos como LightFM, que dependen fuertemente de una buena separación entre clases.
- **Expansión del dataset**: entrenar sobre una mayor proporción del *corpus* original permitiría estabilizar métricas y extraer mejores patrones latentes, especialmente si se optimiza el preprocesamiento para escalar a millones de interacciones.

Estas mejoras no sólo podrían aumentar la precisión de las recomendaciones, sino también su diversidad, novedad y utilidad en entornos reales.

Referencias

[1] Koren, Y., Bell, R., & Volinsky, C. (2009). *Matrix factorization techniques for recommender systems*. IEEE Computer, 42(8), 30-37. https://doi.org/10.1109/MC.2009.263

[2] Rahman, Md., Liu, W., & Peng, H. (2022). *A Survey of Cold-Start Problems in Recommender Systems: Challenges, Strategies, and Trends*. Expert Systems with Applications, 198, 116834. https://doi.org/10.1016/j.eswa.2022.116834

[3] Wang, X., He, X., Cao, Y., Liu, M., & Chua, T.S. (2020). *LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation*. In SIGIR '20. https://doi.org/10.1145/3397271.3401063

[4] Shani, G., & Gunawardana, A. (2011). *Evaluating Recommendation Systems*. En Ricci et al. (eds.), Recommender Systems Handbook. Springer. https://doi.org/10.1007/978-0-387-85820-3_8

[5] Kula, M. (2015). *Metadata Embeddings for User and Item Cold-start Recommendations*. In RecSys '15 Workshop on New Trends for Content-Based Recommender Systems (CBRecSys).