Trabajo de Fin de Grado - Grado en Ingeniería Informática.

Análisis Comparativo de Sistemas de Recomendación Secuenciales en la Plataforma Steaml

Lucas Rengifo Garcia

Tutor: Alejandro Bellogin Kouki

Ponente: Alejandro Bellogin Kouki

Universidad Autónoma de Madrid (UAM)

Tabla de contenidos

Tabla de contenidos

Introducción - Steam

Steam

- Plataforma de distribución digital creada por Valve que ofrece videojuegos, software y servicios de comunidad para jugadores, incluyendo actualizaciones, multijugador y funciones sociales.

Introducción - Steam

Steam

- Plataforma de distribución digital creada por Valve que ofrece videojuegos, software y servicios de comunidad para jugadores, incluyendo actualizaciones, multijugador y funciones sociales.
- En 2023, Steam alcanzó un récord de más de 33 millones de usuarios concurrentes, un aumento considerable respecto a años anteriores.
- Además, 14,500 títulos fueron lanzados en 2023, lo que representa un incremento de más de 2,000 juegos en comparación con 2022.

Introducción - Steam

Steam

- Plataforma de distribución digital creada por Valve que ofrece videojuegos, software y servicios de comunidad para jugadores, incluyendo actualizaciones, multijugador y funciones sociales.
- En 2023, Steam alcanzó un récord de más de 33 millones de usuarios concurrentes, un aumento considerable respecto a años anteriores.
- Además, 14,500 títulos fueron lanzados en 2023, lo que representa un incremento de más de 2,000 juegos en comparación con 2022.
- Aproximadamente un cuarto de esos nuevos lanzamientos provienen de desarrolladores que publicaron su primer juego en Steam.
- Esto demuestra la importancia que tiene la plataforma de contar con un buen recomendador que haga que los usuarios puedan encontrar sus productos óptimos.

Introducción - RecBole

RecBole

- RecBole es un marco de trabajo unificado diseñado para crear y evaluar sistemas de recomendación de manera eficiente, compatible con múltiples tipos de modelos de recomendación.
- RecBole es un proyecto desarrollado por RUCAIBox, un grupo de investigación en inteligencia artificial con sede en China.
- Basado en Python y PyTorch.

Introducción - RecBole

RecBole

- RecBole es un marco de trabajo unificado diseñado para crear y evaluar sistemas de recomendación de manera eficiente, compatible con múltiples tipos de modelos de recomendación.
- RecBole es un proyecto desarrollado por RUCAIBox, un grupo de investigación en inteligencia artificial con sede en China.
- Basado en Python y PyTorch.

Tipos de recomendadores

- General Recommendation.
- Sequential Recommendation.
- Context-aware Recommendation.
- Knowledge-based Recommendation.

Tabla de contenidos

Experimentacion - Métricas de Evaluación (1/2)

Precision: Mide la proporción de recomendaciones relevantes entre todas las recomendaciones realizadas.

$$\text{Precision} = \frac{|\{relevant\ \ items\} \cap \{recommended\ \ items\}|}{|\{recommended\ \ items\}|} \tag{1}$$

Experimentacion - Métricas de Evaluación (1/2)

Precision: Mide la proporción de recomendaciones relevantes entre todas las recomendaciones realizadas.

$$\text{Precision} = \frac{|\{relevant\ \ items\} \cap \{recommended\ \ items\}|}{|\{recommended\ \ items\}|} \tag{1}$$

Recall: Mide la proporción de elementos relevantes recomendados entre todos los elementos relevantes.

$$\text{Recall} = \frac{|\{relevant\ \ items\} \cap \{recommended\ \ items\}|}{|\{relevant\ \ items\}|} \tag{2}$$

Experimentacion - Métricas de Evaluación (1/2)

Precision: Mide la proporción de recomendaciones relevantes entre todas las recomendaciones realizadas.

$$\text{Precision} = \frac{|\{relevant\ \ items\} \cap \{recommended\ \ items\}|}{|\{recommended\ \ items\}|} \tag{1}$$

Recall: Mide la proporción de elementos relevantes recomendados entre todos los elementos relevantes.

$$\text{Recall} = \frac{|\{relevant\ \ items\} \cap \{recommended\ \ items\}|}{|\{relevant\ \ items\}|} \tag{2}$$

MRR (Mean Reciprocal Rank): Evalúa la calidad según la posición del primer elemento relevante.

$$\text{MRR} = \frac{1}{|U|} \sum_{i=1}^{|U|} \frac{1}{rank_i} \tag{3}$$

Experimentacion - Métricas de Evaluación (2/2)

NDCG (Normalized Discounted Cumulative Gain): Mide la calidad de las recomendaciones considerando la posición de los elementos relevantes.

$$\text{NDCG} = \frac{DCG}{IDCG}, \quad DCG = \sum_{i=1}^{|R|} \frac{rel_i}{\log_2(i+1)} \tag{4}$$

Experimentacion - Métricas de Evaluación (2/2)

NDCG (Normalized Discounted Cumulative Gain): Mide la calidad de las recomendaciones considerando la posición de los elementos relevantes.

$$\text{NDCG} = \frac{DCG}{IDCG}, \quad DCG = \sum_{i=1}^{|R|} \frac{rel_i}{\log_2(i+1)} \tag{4}$$

Hit Rate (Tasa de Aciertos): Proporción de usuarios con al menos un ítem relevante en la lista recomendada.

$$\text{Hit Rate} = \frac{|\{users\ with\ hits\}|}{|\{total\ users\}|} \tag{5}$$

Experimentación - Selección de algoritmos

- **GRU4REC** $\rightarrow$ Redes neuronales recurrentes (RNNs) para recomendaciones basadas en sesiones.

Experimentación - Selección de algoritmos

- **GRU4REC** $\rightarrow$ Redes neuronales recurrentes (RNNs) para recomendaciones basadas en sesiones.
- **FOSSIL** $\rightarrow$ Modelos de similitud + cadenas de Markov.

Experimentación - Selección de algoritmos

- **GRU4REC** $\rightarrow$ Redes neuronales recurrentes (RNNs) para recomendaciones basadas en sesiones.
- **FOSSIL** $\rightarrow$ Modelos de similitud + cadenas de Markov.
- **HRM** $\rightarrow$ Arquitectura jerárquica para capturar interacciones complejas entre ítems.

Experimentación - Selección de algoritmos

- **GRU4REC** $\rightarrow$ Redes neuronales recurrentes (RNNs) para recomendaciones basadas en sesiones.
- **FOSSIL** $\rightarrow$ Modelos de similitud + cadenas de Markov.
- **HRM** $\rightarrow$ Arquitectura jerárquica para capturar interacciones complejas entre ítems.
- **SRGNN** $\rightarrow$ Redes neuronales de grafos (GNN).

Experimentación - Selección de algoritmos

- **GRU4REC** $\rightarrow$ Redes neuronales recurrentes (RNNs) para recomendaciones basadas en sesiones.
- **FOSSIL** $\rightarrow$ Modelos de similitud + cadenas de Markov.
- **HRM** $\rightarrow$ Arquitectura jerárquica para capturar interacciones complejas entre ítems.
- **SRGNN** $\rightarrow$ Redes neuronales de grafos (GNN).
- **BERT4REC** $\rightarrow$ Transformadores bidireccionales profundos.

Experimentación - Selección de algoritmos

- **GRU4REC** $\rightarrow$ Redes neuronales recurrentes (RNNs) para recomendaciones basadas en sesiones.
- **FOSSIL** $\rightarrow$ Modelos de similitud + cadenas de Markov.
- **HRM** $\rightarrow$ Arquitectura jerárquica para capturar interacciones complejas entre ítems.
- **SRGNN** $\rightarrow$ Redes neuronales de grafos (GNN).
- **BERT4REC** $\rightarrow$ Transformadores bidireccionales profundos.
- **HGN** $\rightarrow$ Red jerárquica de compuertas.

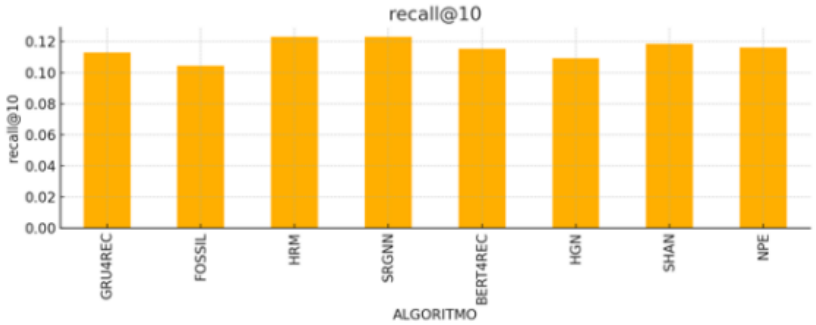
Experimentación - Selección de algoritmos

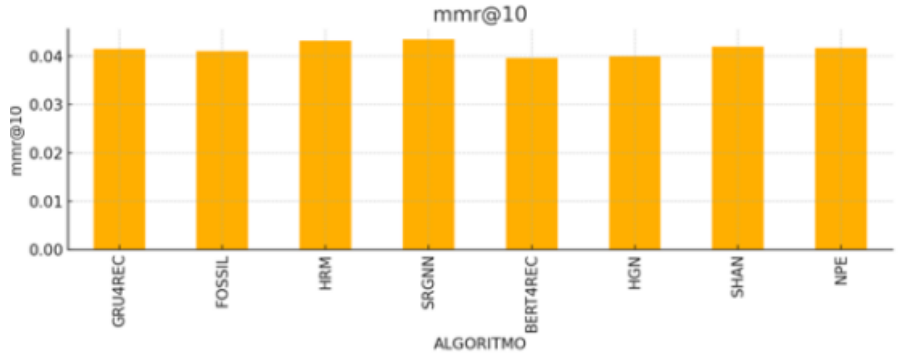
- **GRU4REC** $\rightarrow$ Redes neuronales recurrentes (RNNs) para recomendaciones basadas en sesiones.
- **FOSSIL** $\rightarrow$ Modelos de similitud + cadenas de Markov.
- **HRM** $\rightarrow$ Arquitectura jerárquica para capturar interacciones complejas entre ítems.
- **SRGNN** $\rightarrow$ Redes neuronales de grafos (GNN).
- **BERT4REC** $\rightarrow$ Transformadores bidireccionales profundos.
- **HGN** $\rightarrow$ Red jerárquica de compuertas.
- **SHAN** $\rightarrow$ Red auto-atentiva.

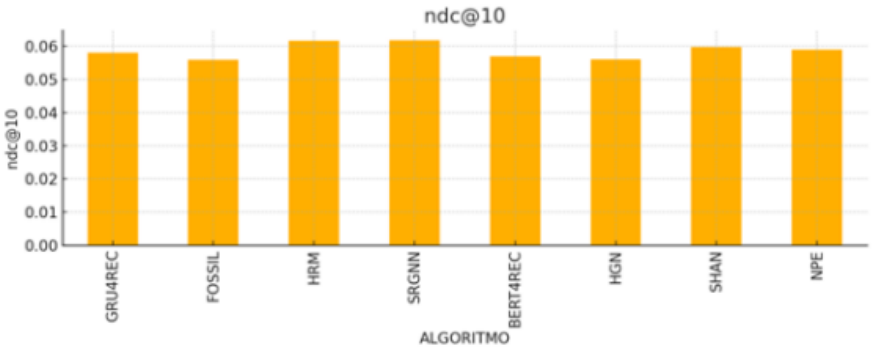
Experimentación - Selección de algoritmos

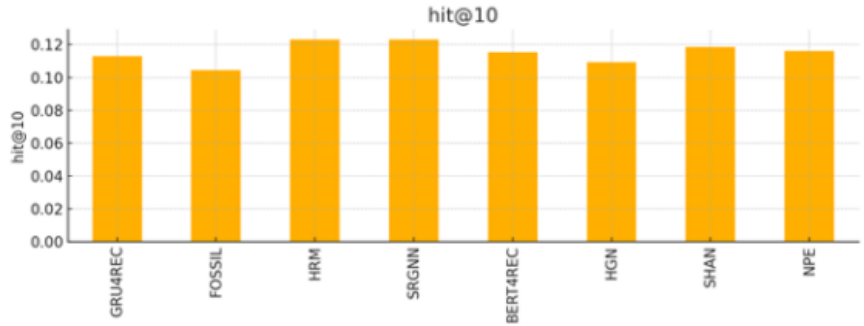
- **GRU4REC** → Redes neuronales recurrentes (RNNs) para recomendaciones basadas en sesiones.
- **FOSSIL** → Modelos de similitud + cadenas de Markov.
- **HRM** → Arquitectura jerárquica para capturar interacciones complejas entre ítems.
- **SRGNN** → Redes neuronales de grafos (GNN).
- **BERT4REC** → Transformadores bidireccionales profundos.
- **HGN** → Red jerárquica de compuertas.
- **SHAN** → Red auto-atentiva.
- **NPE** → Red neuronal para generar embeddings personalizados de usuarios e ítems.

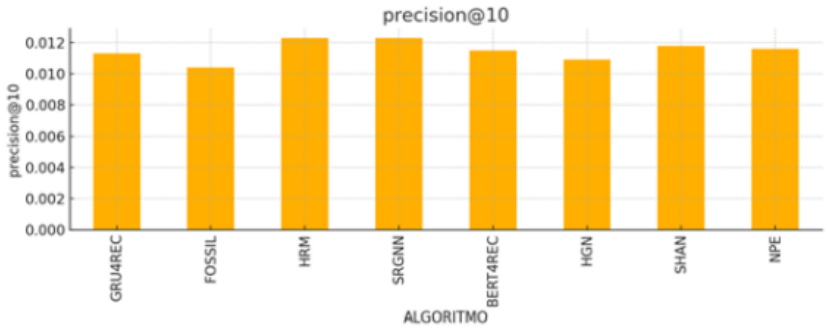
Experimentacion - Resultados de Evaluación











Experimentación - SRGNN en RecBole

SRGNN (Session-based Recommendation using Graph Neural Networks) es un algoritmo utilizado para recomendaciones basadas en sesiones. Su lógica se basa en los siguientes puntos:

- **Redes Neuronales de Grafos (GNN):** SRGNN emplea GNN para modelar la secuencia de interacciones de un usuario en una sesión como un grafo dirigido. Cada nodo representa un ítem y las aristas denotan las transiciones entre ítems.

Experimentación - SRGNN en RecBole

SRGNN (Session-based Recommendation using Graph Neural Networks) es un algoritmo utilizado para recomendaciones basadas en sesiones. Su lógica se basa en los siguientes puntos:

- **Redes Neuronales de Grafos (GNN):** SRGNN emplea GNN para modelar la secuencia de interacciones de un usuario en una sesión como un grafo dirigido. Cada nodo representa un ítem y las aristas denotan las transiciones entre ítems.
- **Representación de la Sesión:** Convierte las interacciones de la sesión en embeddings de nodos utilizando la propagación de mensajes a través de los grafos.

Experimentación - SRGNN en RecBole

SRGNN (Session-based Recommendation using Graph Neural Networks) es un algoritmo utilizado para recomendaciones basadas en sesiones. Su lógica se basa en los siguientes puntos:

- **Redes Neuronales de Grafos (GNN):** SRGNN emplea GNN para modelar la secuencia de interacciones de un usuario en una sesión como un grafo dirigido. Cada nodo representa un ítem y las aristas denotan las transiciones entre ítems.
- **Representación de la Sesión:** Convierte las interacciones de la sesión en embeddings de nodos utilizando la propagación de mensajes a través de los grafos.
- **Actualización de los Embeddings:** Utiliza el algoritmo de propagación para actualizar los embeddings de los ítems en función de sus vecinos en el grafo.

Experimentación - SRGNN en RecBole

SRGNN (Session-based Recommendation using Graph Neural Networks) es un algoritmo utilizado para recomendaciones basadas en sesiones. Su lógica se basa en los siguientes puntos:

- **Redes Neuronales de Grafos (GNN):** SRGNN emplea GNN para modelar la secuencia de interacciones de un usuario en una sesión como un grafo dirigido. Cada nodo representa un ítem y las aristas denotan las transiciones entre ítems.
- **Representación de la Sesión:** Convierte las interacciones de la sesión en embeddings de nodos utilizando la propagación de mensajes a través de los grafos.
- **Actualización de los Embeddings:** Utiliza el algoritmo de propagación para actualizar los embeddings de los ítems en función de sus vecinos en el grafo.
- **Atención para la Sesión Completa:** Se aplica un mecanismo de atención que permite priorizar las interacciones más recientes dentro de una sesión, asignando pesos más altos a las transiciones recientes.

Experimentación - SRGNN en RecBole

SRGNN (Session-based Recommendation using Graph Neural Networks) es un algoritmo utilizado para recomendaciones basadas en sesiones. Su lógica se basa en los siguientes puntos:

- **Redes Neuronales de Grafos (GNN):** SRGNN emplea GNN para modelar la secuencia de interacciones de un usuario en una sesión como un grafo dirigido. Cada nodo representa un ítem y las aristas denotan las transiciones entre ítems.
- **Representación de la Sesión:** Convierte las interacciones de la sesión en embeddings de nodos utilizando la propagación de mensajes a través de los grafos.
- **Actualización de los Embeddings:** Utiliza el algoritmo de propagación para actualizar los embeddings de los ítems en función de sus vecinos en el grafo.
- **Atención para la Sesión Completa:** Se aplica un mecanismo de atención que permite priorizar las interacciones más recientes dentro de una sesión, asignando pesos más altos a las transiciones recientes.

Experimentación -Hiperparametros SRGNN

- **HyperTuning Descartado:** Aunque inicialmente se intentó usar la herramienta HyperTuning de RecBole, se abandonó por la falta de ejemplos y la escasa documentación. También presentaba errores internos de la librería.

Experimentación -Hiperparametros SRGNN

- **HyperTuning Descartado:** Aunque inicialmente se intentó usar la herramienta HyperTuning de RecBole, se abandonó por la falta de ejemplos y la escasa documentación. También presentaba errores internos de la librería.
- **Automatización de Pruebas:** Se creó una función para generar archivos de configuración mediante argumentos, facilitando la automatización de las pruebas de hiperparámetros.

Experimentación -Hiperparametros SRGNN

- **HyperTuning Descartado:** Aunque inicialmente se intentó usar la herramienta HyperTuning de RecBole, se abandonó por la falta de ejemplos y la escasa documentación. También presentaba errores internos de la librería.
- **Automatización de Pruebas:** Se creó una función para generar archivos de configuración mediante argumentos, facilitando la automatización de las pruebas de hiperparámetros.
- **Hiperparámetros Probados:**
  - **Tamaño de embedding:** 32, 64 y 128.
  - **Tasa de aprendizaje:** 0.01, 0.001 y 0.0001.
  - **Step:** 1, 2, 3 y 4 (capas en la GNN).

Experimentación -Hiperparametros SRGNN

- **HyperTuning Descartado:** Aunque inicialmente se intentó usar la herramienta HyperTuning de RecBole, se abandonó por la falta de ejemplos y la escasa documentación. También presentaba errores internos de la librería.
- **Automatización de Pruebas:** Se creó una función para generar archivos de configuración mediante argumentos, facilitando la automatización de las pruebas de hiperparámetros.
- **Hiperparámetros Probados:**
  - **Tamaño de embedding:** 32, 64 y 128.
  - **Tasa de aprendizaje:** 0.01, 0.001 y 0.0001.
  - **Step:** 1, 2, 3 y 4 (capas en la GNN).
- **Resultados:** Se realizaron 36 combinaciones de pruebas, con una duración de 62 horas. Algunas pruebas debieron repetirse debido a pantallazos azules de Windows causados por la alta demanda de recursos.

Experimentación - Hiperparámetros SRGNN

Conclusiones:

- **Embedding Size:** Tamaños más grandes proporcionaron mejores resultados, siendo el de 32 el menos efectivo.
- **Learning Rate:** El valor óptimo fue 0.001, equilibrando velocidad de convergencia y capacidad de exploración.
- **Step:** El valor óptimo fue 1, evitando sobreajuste y ruido innecesario.

Experimentación - Hiperparámetros SRGNN

Conclusiones:

- **Embedding Size:** Tamaños más grandes proporcionaron mejores resultados, siendo el de 32 el menos efectivo.
- **Learning Rate:** El valor óptimo fue 0.001, equilibrando velocidad de convergencia y capacidad de exploración.
- **Step:** El valor óptimo fue 1, evitando sobreajuste y ruido innecesario.

128	0.001	1	0.1254	0.0452	0.0638	0.1254	0.0125
64	0.001	4	0.1247	0.0444	0.0629	0.1247	0.0125
64	0.001	1	0.1234	0.044	0.0623	0.1234	0.0123

Experimentación - HRM en RecBole

- **HRM (Hierarchical Representation Model):**
  - El algoritmo HRM está diseñado para capturar interacciones complejas entre ítems a través de una representación jerárquica.
  - Utiliza una estructura de capas que permite modelar relaciones de alto orden entre ítems, lo que facilita la identificación de patrones en bases de datos con alta dispersión.
  - La estructura jerárquica representa interacciones a varios niveles:
    - **Nivel bajo:** Captura las interacciones directas entre ítems (similar a modelos tradicionales de recomendación).
    - **Nivel alto:** Modela interacciones más profundas entre ítems relacionados indirectamente, basándose en las relaciones de ítems previos.
  - El parámetro clave **high_order** controla la profundidad de las interacciones observadas. Valores más altos permiten capturar dependencias más complejas, mientras que valores más bajos se limitan a interacciones recientes.

Experimentación - HRM en RecBole

Funcionamiento:

- HRM divide los ítems en grupos jerárquicos basados en interacciones previas del usuario.
- Luego, modela estas interacciones de forma secuencial, construyendo una representación jerárquica que refleja tanto la relación inmediata entre ítems como las conexiones más profundas entre grupos de ítems.
- Utiliza una función de pérdida que ajusta las representaciones jerárquicas, maximizando la predicción de ítems relevantes.

Experimentación - HRM en RecBole

Funcionamiento:

- HRM divide los ítems en grupos jerárquicos basados en interacciones previas del usuario.
- Luego, modela estas interacciones de forma secuencial, construyendo una representación jerárquica que refleja tanto la relación inmediata entre ítems como las conexiones más profundas entre grupos de ítems.
- Utiliza una función de pérdida que ajusta las representaciones jerárquicas, maximizando la predicción de ítems relevantes.

Aplicaciones en bases de datos dispersas:

- Al capturar interacciones de alto orden, HRM es eficaz para manejar bases de datos con alta dispersión, donde las relaciones entre ítems no son evidentes.
- Esto permite al modelo generalizar mejor y adaptarse a las características de los datos sin sobreajustar.

Experimentación - HRM - Hiperparámetros

Elección de Hiperparámetros:

Se añadió **reproducibility=False** al fichero de configuración, lo que permitió reducir el tiempo de entrenamiento a 1800 minutos (30 horas).

- **Parámetro clave:** Tras estudiar la documentación de RecBole y realizar pruebas, se determinó que el hiperparámetro más relevante para HRM es **high_order**.
- **Importancia de high_order:** Influye en la capacidad del modelo para capturar dependencias de alto orden, adaptarse a datos dispersos, y afecta el equilibrio entre complejidad y generalización, así como la efectividad de las representaciones jerárquicas.
- **Otros parámetros:** Se utilizó la misma configuración de **embedding_size** y **learning_rate** que en SRGNN, mientras que **high_order** se ajustó con valores de 1 a 4.

Experimentación - HRM - Estudio de Hiperparámetros

Conclusiones:

- **Embedding_size:** El tamaño óptimo fue 64, mientras que 128 también funcionó bien, pero tendió a añadir ruido. El tamaño de 32 fue insuficiente, produciendo los peores resultados.
- **Learning_rate:** Al igual que en SRGNN, la tasa de aprendizaje de 0.001 fue la más efectiva.
- **High_order:** El peor valor fue 1, que prácticamente ignoró las dependencias anteriores. Los valores más altos de 4 funcionaron bien, pero 2 mostró un rendimiento constante y competitivo.

64	0.001	4	0.1246	0.0438	0.0625	0.1246	0.0125
64	0.001	2	0.1238	0.0437	0.0622	0.1238	0.0124
64	0.001	3	0.1237	0.0436	0.0621	0.1237	0.0124

Experimentacion - SHAN en RecBole

- **SHAN (Self-Attentive Neural Network):**
  - SHAN utiliza una arquitectura auto-atentiva, lo que le permite ponderar las interacciones pasadas de los usuarios según su importancia.
  - El mecanismo de auto-atención ajusta dinámicamente el peso de las interacciones previas, priorizando aquellas más relevantes para la recomendación actual.

Experimentacion - SHAN en RecBole

- **SHAN (Self-Attentive Neural Network):**
  - SHAN utiliza una arquitectura auto-atentiva, lo que le permite ponderar las interacciones pasadas de los usuarios según su importancia.
  - El mecanismo de auto-atención ajusta dinámicamente el peso de las interacciones previas, priorizando aquellas más relevantes para la recomendación actual.
- **Funcionamiento:**
  - SHAN divide las interacciones de los usuarios en dos partes: las interacciones recientes y las de largo plazo.
  - Mediante auto-atención, el modelo pondera de manera diferente las interacciones a corto y largo plazo, enfocándose en las más importantes para hacer predicciones.
  - Usa capas auto-atentivas para mejorar la precisión de las recomendaciones, ajustando el grado de influencia de las interacciones pasadas.

Experimentacion - SHAN - Hiperparametros

Eficacia en bases de datos dispersas:

- SHAN sobresale en datos dispersos al capturar relaciones relevantes entre ítems, incluso cuando las interacciones son infrecuentes.
- Al aplicar atención diferenciada a las interacciones más útiles, evita el sobreajuste y mejora la generalización.
- **Parámetros probados:**
  - **Short Item Length:** Se probó el tamaño de la ventana de historial del usuario, con valores de 1 a 4.

Experimentacion - SHAN - Hiperparametros

Eficacia en bases de datos dispersas:

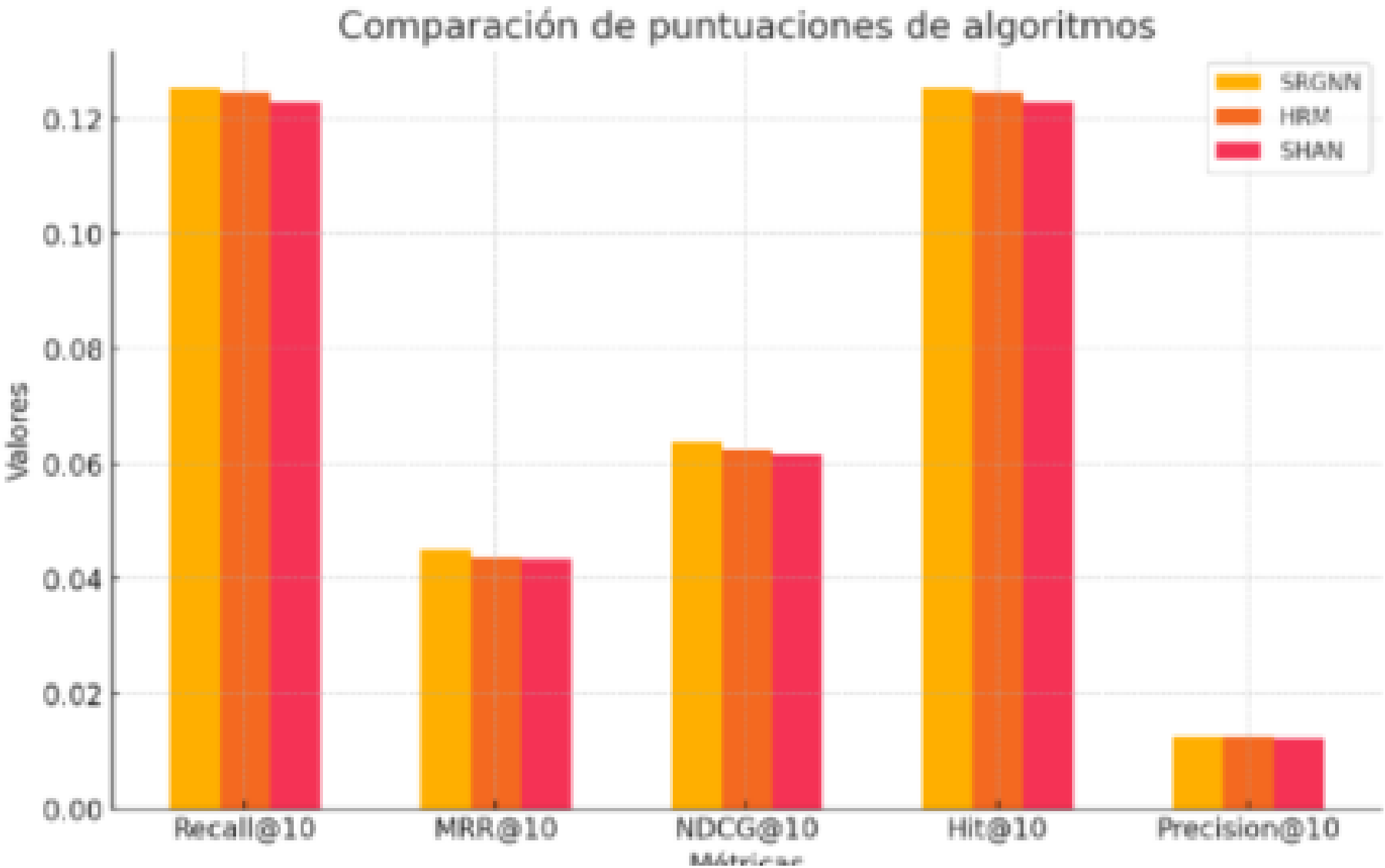
- SHAN sobresale en datos dispersos al capturar relaciones relevantes entre ítems, incluso cuando las interacciones son infrecuentes.
- Al aplicar atención diferenciada a las interacciones más útiles, evita el sobreajuste y mejora la generalización.
- **Parámetros probados:**
  - **Short Item Length:** Se probó el tamaño de la ventana de historial del usuario, con valores de 1 a 4.
- **Resultados:**
  - **Embedding Size:** Un tamaño de 128 proporcionó los mejores resultados, permitiendo al modelo encontrar patrones más complejos en una base de datos dispersa como la de Steam.
  - **Learning Rate:** El valor óptimo fue 0.0001, lo que permitió profundizar en el descenso de gradiente sin converger rápidamente en un mínimo local.
  - **Short Item Length:** Los valores 2, 3 y 4 capturaron suficiente contexto de las interacciones, mientras que valores más bajos (como 1) mostraron variabilidad en sus resultados.

Tabla de contenidos

Resultados Finales

- Los experimentos realizados muestran que el desempeño de los modelos secuenciales depende de los parámetros elegidos y de la base de datos utilizada.
- **SRGNN** se adapta bien a bases de datos dispersas debido a su capacidad para encontrar patrones ocultos mediante el uso de redes neuronales de grafos.
- **HRM** se beneficia de su estructura jerárquica, permitiendo capturar dependencias complejas y trabajar con datos limitados.
- **SHAN**, con su mecanismo de atención, se enfoca en interacciones relevantes y recientes, mostrando alta adaptabilidad.
- Aunque los algoritmos obtienen resultados similares, **SRGNN** se destaca como el mejor algoritmo para este problema en particular.

Grafica de resultados



Conclusiones

- Los sistemas de recomendación son cada vez más importantes para personalizar la experiencia en la web, impactando positivamente tanto a consumidores como a negocios.
- Este proyecto estudió el enfoque de recomendación secuencial aplicado a la base de datos de Steam, utilizando algoritmos diseñados para predecir la disposición de ítems en función de secuencias de clics.
- La librería **RecBole** fue una herramienta clave, aunque su falta de documentación fue un obstáculo importante, requiriendo pruebas exhaustivas para ajustar modelos y datos.
- Los recomendadores secuenciales demostraron ser efectivos con bases de datos dispersas, especialmente aquellos que requieren pocas interacciones para mejorar predicciones.
- **SRGNN** fue el modelo que mejor se adaptó a este problema específico, mientras que otros modelos orientados a secuencias largas no alcanzaron resultados prometedores.

Trabajo Futuro

- Una mejora sería explorar otros tipos de recomendadores para identificar cuáles se adaptan mejor al problema, permitiendo comparativas reales entre los diferentes modelos.
- Realizar un estudio exhaustivo de hiperparámetros para cada modelo podría mejorar significativamente el desempeño de aquellos que no mostraron los mejores resultados por defecto.
- Otro aspecto interesante seria probar estos recomendadores con otra base de datos mucho menos dispersa con usuarios que cuenten con muchas mas interacciones para que realmente se puedan apreciar sus interacciones como la idea que se tenia en mente de secuencia de clicks.
- Migrar el sistema a la nube sería otra posible ampliación del proyecto. Aunque tendría un coste importante por la carga de procesamiento, permitiría distribuir la herramienta como producto y facilitar pruebas sobre recomendadores.

Tabla de contenidos

Demo app web

Demo aplicacion web