REPORTE FINAL — Decision Transformer para Recomendación de Películas (Netflix8)

1. Introducción

El objetivo de este trabajo es desarrollar un sistema de recomendación basado en aprendizaje por refuerzo siguiendo el enfoque del Decisión Transformer (DT). A diferencia de los métodos tradicionales (Collaborative Filtering, Popularity, o modelos puramente supervisados), el DT formula la recomendación como un problema de decisión secuencial, similar a un MDP (Markov Decision Process).
Los objetivos del trabajo fueron:

- Preparar el dataset en formato de trayectorias para un Decisión Transformer.
- Entrenar el DT y compararlo con baselines simples.
- Evaluar el impacto del conditioning en return-to-go.
- Analizar el desempeño en escenarios de cold-start.

2. Dataset y Preprocesamiento

El dataset Netflix8 está compuesto por un conjunto de interacciones usuario–ítem utilizadas para entrenar y evaluar modelos de recomendación secuencial. Su estructura principal incluye:
1.6 millones de interacciones usuario-película
16.000 usuarios únicos
752 ítems (películas disponibles en el catálogo)
Ratings en una escala discreta de 1 a 5
Además, cada usuario fue asignado a uno de 8 clusters (user groups) construidos a partir de sus preferencias promedio. Estos grupos permiten analizar patrones de comportamiento y evaluar si el modelo captura diferencias entre perfiles de usuarios.

Preprocesamiento realizado

1. Conversión a trayectorias usuario–item

Para cada usuario se construyó una trayectoria secuencial que incluye:

- items vistos en orden temporal
- ratings asociados a cada interacción
- returns_to_go, calculados hacia adelante
- timesteps normalizados
- user_group asignado según preferencias promedio

Este formato permite modelar el problema como una tarea de predicción autoregresiva.

2. Cálculo del Return-to-Go (RTG)

El retorno futuro acumulado para cada paso se definió como:
R^t=rt+rt+1+···+rT
Este valor se utiliza como señal de conditioning en el Decisión Transformer, permitiendo guiar la generación hacia secuencias de mayor recompensa.

3. Construcción de ventanas de contexto

Cada muestra del dataset se genera utilizando una ventana fija de:
context_length=20
Para secuencias más cortas:

- Se aplicó padding al inicio
- Se utilizó un masking correspondiente para evitar que el modelo atienda posiciones artificiales

Este esquema permite entrenar al modelo con batchings homogéneos sin perder información temporal.

3. Implementación

Detalles del entrenamiento

Los parámetros de entrenamiento utilizados fueron:

- Optimizer: AdamW
- Learning rate: 1e-4
  (se experimentó también con 3e-4)
- Weight decay: 1e-4
- Batch size: 64
- Épocas: 10
  (pruebas extendidas con 30 épocas)
- Tiempo de entrenamiento: ~12 minutos por corrida en una CPU/Mac

El entrenamiento fue estable y sin explosión del gradiente, incluso con secuencias largas.

Desafíos encontrados

Durante la implementación surgieron los siguientes puntos críticos:

- Máscara causal obligatoria (PyTorch 2.9):
  El uso del scaled_dot_product_attention exige la provisión explícita de la máscara; fue necesario ajustar el forward del modelo.
- Secuencias cortas y padding:
  Se requirió aplicar padding consistente al inicio y utilizar máscaras de atención para evitar que el modelo incorpore posiciones artificiales.
- Valor alto de loss (~6.62)

4. Resultados

Comparación de modelos

La siguiente tabla resume el desempeño de cada enfoque utilizando las métricas estándar de ranking:

Modelo	HR@10	NDCG@10	MRR
Decision Transformer	0.0137	0.0062	0.0097
Behavior Cloning	0.0140	0.0065	0.0101
Popularity	0.0123	0.0058	0.0096
Random	0.0137	0.0062	0.0097

Interpretación general

El rendimiento global es bajo debido a dos factores estructurales del dataset:

- Alta dispersión en las elecciones de ítems.
- 752 posibles clases, lo que reduce significativamente las probabilidades de acierto.

Aun así, se observan tendencias relevantes:

- Behavior Cloning (BC) obtiene el mejor desempeño, lo que indica que imitar secuencias reales es una estrategia sólida en este dataset.
- El Decision Transformer queda muy cerca de BC, mostrando que es capaz de capturar dependencias temporales incluso con un modelo pequeño.
- Popularity es sorprendentemente competitivo, sugiriendo que los usuarios consumen muchos ítems frecuentes.
- Random, aunque sirve solo como baseline inferior, obtiene valores similares a DT en HR@10 debido a la probabilidad extremadamente baja de acierto.

4.2. Efecto del Return Conditioning

Se evaluó el Decision Transformer utilizando distintos valores objetivo de retorno acumulado (R^):

Percentil	R^ objetivo
p25	213
p50	350
p75	497
p90	604
máximo	849

Resultado principal

Independientemente del return objetivo, el modelo produjo el mismo rendimiento:

- HR@10: 0.0137
- NDCG@10: 0.0062
- MRR: ≈ 0.0097

Interpretación

Estos resultados sugieren que:

- El return no aporta información diferenciadora en este dataset.
- Las trayectorias poseen poca estructura causal explícita entre los ratings.
- El DT termina comportándose esencialmente como un modelo autoregresivo clásico condicionado por historial, más que por objetivos de retorno.

5. Discusión

¿Qué aprendió el modelo?

- Captura razonablemente bien el orden temporal y patrones locales del historial.
- Aprende transiciones probables basadas en secuencias largas.
- El aprendizaje del Return-to-Go es débil debido a la baja estructura del reward.

¿Cuándo funciona mejor?

- Usuarios con trayectorias extensas y consistentes.
- Grupos donde las preferencias están claramente definidas.
- Secuencias donde los ratings muestran correlación temporal.

¿Cuándo falla?

- Usuarios con historiales muy variados o erráticos.
- Ítems poco frecuentes, difíciles de predecir por falta de datos.
- Situaciones de cold-start extremo.

Limitaciones observadas

- El Return-to-Go no mejoró el rendimiento.
- El modelo es pequeño para un catálogo de 752 ítems.
- Poca señal de recompensa explícita para explotar con conditioning.
- Dataset disperso, trayectorias muy heterogéneas.

Posibles mejoras

- Usar embeddings más grandes (dim=256 o 512).
- Añadir positional encoding tipo GPT.
- Entrenar más épocas.
- Incorporar features de ítems (género, año, popularidad).

6. Conclusiones

El Decisión Transformer ofrece un enfoque moderno que unifica ideas de modelos generativos, secuencias temporales y Reinforcement Learning aplicado a sistemas de recomendación.
En este trabajo:

- Se implementó un pipeline completo de preprocesamiento, modelado y evaluación.
- Se compararon DT, BC, Popularity y Random sobre un dataset real.
- Se analizaron los efectos del return conditioning y del cold-start por grupos.
- Se validó que el DT funciona correctamente, pero no supera a BC en este dataset.

Aun así, el DT aporta ventajas conceptuales importantes:

- Maneja naturalmente dependencias temporales.
- Permite conditioning flexible para diferentes objetivos.
- Escala hacia configuraciones de RL más complejas.

Trabajo futuro recomendado

- Expandir el tamaño del modelo.
- Enriquecer la señal de reward.
- Incorporar metadata de ítems.