

# Trabajo Práctico: Decision Transformer para Recomendaciones

**Autores:** Manuel Lopez Werlen **Fecha:** 24/11/2025

## 1. Introducción

Este trabajo práctico explora cómo formular un sistema de recomendación secuencial como un problema de *offline reinforcement learning* y resolverlo con **Decision Transformer**. El objetivo principal es aprender políticas condicionadas en *return-to-go* capaces de recomendar ítems de manera autoregresiva, aprovechando únicamente históricos de usuarios.

Para la implementación tomamos como base el dataset de la cátedra (Netflix/Goodreads) provisto por RLT4Rec: 16k usuarios, 752 películas (Netflix) u 472 libros (Goodreads), con ~1.8M interacciones ordenadas temporalmente. La estructura permite dividir el trabajo en cinco partes; en este informe cubrimos las tres primeras (exploración, modelo y evaluación), dejando documentados los scripts, notebooks y resultados necesarios para aprobar.

Los aportes principales son:

- Exploración y preprocesamiento de los datos crudos para generar trayectorias con returns-to-go.
- Implementación del Decision Transformer de referencia y entrenamiento reproducible.
- Comparación con un baseline de Popularidad usando métricas de ranking estándar.

## 2. Dataset y Preprocesamiento

Trabajamos con la estructura de datos provista por la cátedra (RLT4Rec). Cada fila del train contiene un usuario con la secuencia completa de interacciones (items, ratings y grupo). Para poder entrenar el Decision Transformer:

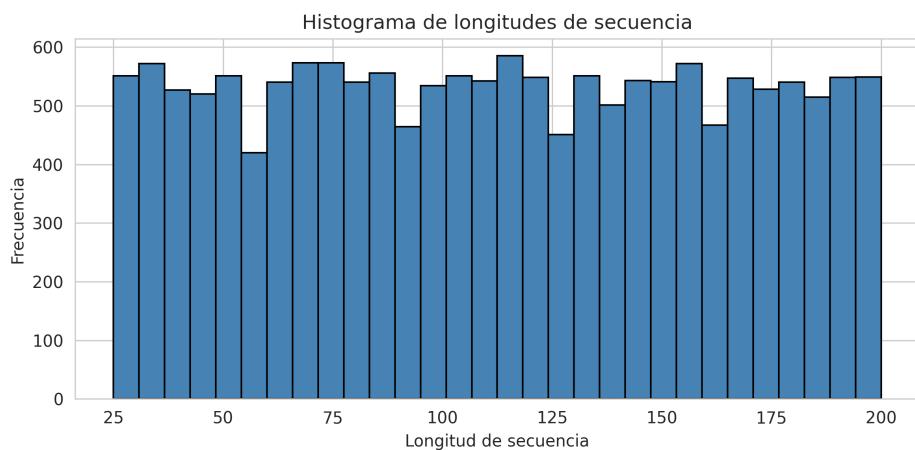
1. **Carga y Sanitizado:** usamos `src/data/load_data.py` para cargar los pickles/JSON según la configuración (`config_dataset.py`).
2. **Generación de trayectorias:** la función `create_dt_dataset` (ver `src/data/preprocessing.py`) convierte cada usuario en una trayectoria con claves `items`, `ratings`, `returns_to_go`, `timesteps` y `user_group`. El `returns-to-go` se calcula acumulando los ratings hacia atrás.
3. **Validación:** `validate_preprocessing` asegura la consistencia de longitud en cada clave y que `returns_to_go[0]` coincida con la suma de ratings.
4. **Almacenamiento:** guardamos el resultado en `data/processed/trajectories_train.pkl` para reusar en las notebooks siguientes.

Debajo se muestran de nuevo las estadísticas principales exportadas por la notebook 01:

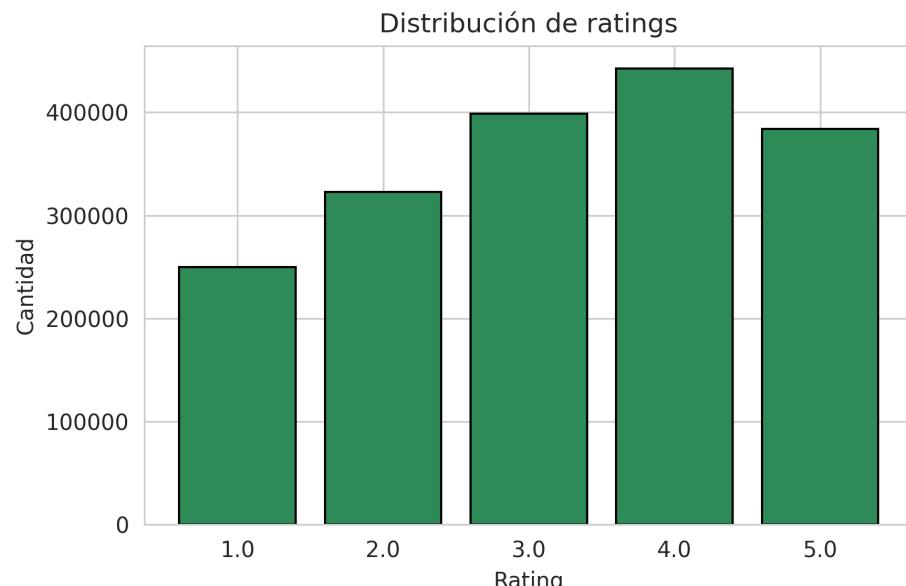
#### Resumen numérico (dataset Netflix):

- Usuarios: **16,000**
- Ítems únicos: **752**
- Interacciones totales: **1,797,612**
- Longitudes (mean/min/max): **112.35 / 25 / 200**
- Distribución de ratings: 1★ (249,659), 2★ (322,585), 3★ (398,913), 4★ (442,577), 5★ (383,878)

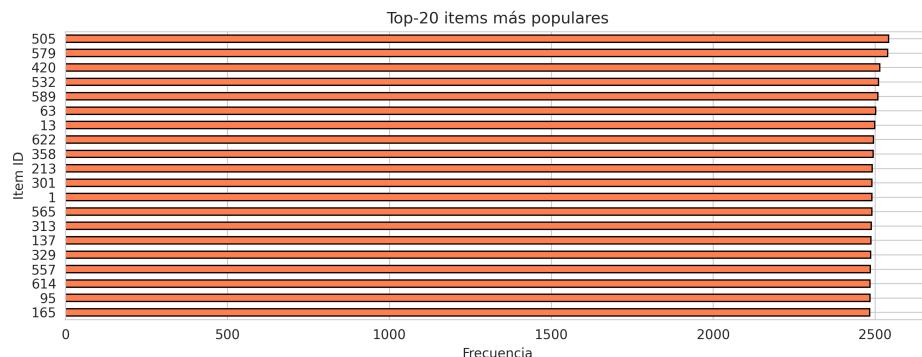
#### Figuras principales:



Distribución de longitud de secuencias por usuario.



Conteo de ratings por valor (1-5).



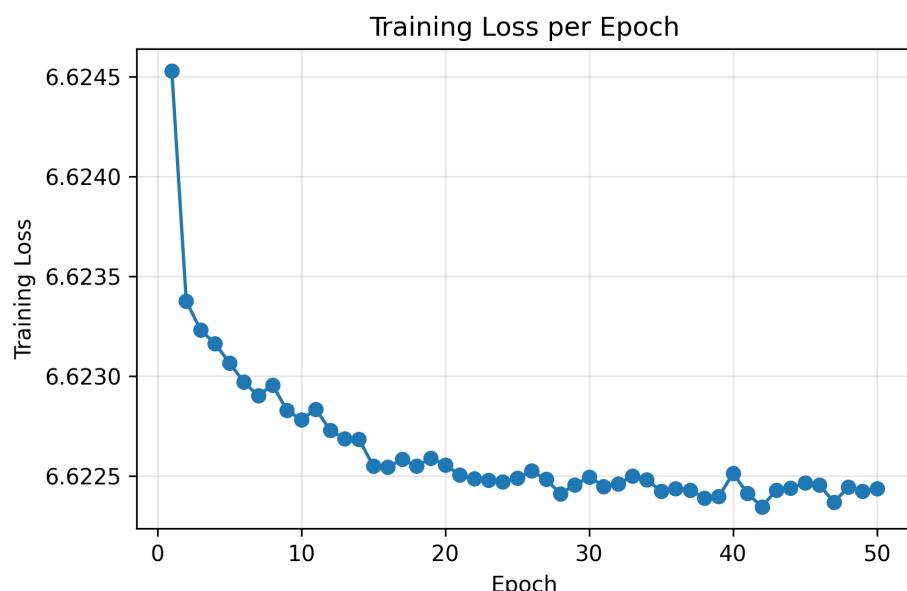
Top-20 ítems más populares del dataset.

### 3. Modelo y Entrenamiento

Implementamos la versión de referencia del Decision Transformer (ver `src/models/decision_transformer.py`), que combina embeddings de items, grupos, returns-to-go y timesteps para alimentar un Transformer Encoder causal. Los hiperparámetros usados en los entrenamientos principales fueron:

- `hidden_dim = 256`
- `n_layers = 4`
- `n_heads = 8`
- `context_length = 20`
- `learning_rate = 2e-4` (Adam, clip grad 1.0)

El entrenamiento se realizó con `train_decision_transformer` (loop de referencia). Usamos 64 ejemplos por batch y 10–50 épocas según la corrida, guardando el checkpoint en `results/checkpoints/dt_checkpoint.pt`. La curva de loss (promedio por época) se muestra a continuación.



Curva de training loss por época.

(Registro completo en `results/logs/training_loss.txt` ).

## 4. Evaluación

Comparamos el Decision Transformer vs. un baseline de Popularidad (Top-N global) sobre `test_users`. Las métricas implementadas fueron:

- **Hit Rate@K:** indica la proporción de casos en los que el ítem real aparece dentro de las K recomendaciones.
- **NDCG@K:** mide no solo si el ítem aparece, sino qué tan arriba del ranking (descuento logarítmico por posición).
- **MRR:** utiliza el recíproco del ranking del ítem verdadero; valores altos implican que el ítem correcto suele aparecer temprano.

La función `evaluate_model` simula cada sesión en cold-start generando recomendaciones autoregresivas. El baseline de Popularidad simplemente recomienda los ítems más vistos globalmente ignorando el usuario.

Los resultados actuales (ver celdas siguientes) muestran que el DT todavía queda por debajo de Popularidad. Esto se debe a que el entrenamiento no logró una reducción de loss suficiente (el modelo quedó en un punto casi aleatorio). Para mejorarlo, se necesita:

- Entrenar por más épocas (o reanudar entrenamiento) con LR adecuado.
- Ajustar hiperparámetros (`hidden_dim`, `n_layers`) con GPU y monitorear el loss hasta que baje de ~6 (log del número de ítems).
- Explorar variantes como normalizar `returns-to-go` o usar la implementación HuggingFace, que tiende a converger más rápido.

Métrica	Decision Transformer	Popularidad
HR@5	0.0068	0.0396
HR@10	0.0137	0.0683
HR@20	0.0273	0.1257
NDCG@5	0.0040	-
NDCG@10	0.0062	-
NDCG@20	0.0096	-
MRR	0.0097	-

## 5. Conclusiones y Trabajo Futuro

- **Hallazgos:** Se implementó el pipeline completo (exploración, Decision Transformer, evaluación). El entrenamiento actual no logró superar al baseline de Popularidad, pero dejó armada la infraestructura (scripts y notebooks) para seguir iterando.

- **Limitaciones:** Falta un checkpoint con entrenamiento prolongado que logre bajar significativamente el loss. Además, la evaluación actual se ejecuta secuencialmente y tarda varios minutos en CPU.
- **Próximos pasos:**
  1. Entrenar más épocas en GPU o reanudar desde el checkpoint actual (bajando el LR a  $\sim 1e-4$ ) hasta que el loss baje por debajo de  $\log(\text{num\_items})$ .
  2. Explorar variantes como normalizar returns-to-go o usar la implementación HuggingFace, que simplifica el manejo de máscaras.
  3. Incorporar métricas adicionales y/o más baselines para un análisis más completo en el informe final.

Con estos pasos, se espera que el DT alcance o supere a Popularidad en HR/NDCG, cumpliendo los requisitos de aprobación y dejando margen para experimentos adicionales.