PONTIFICIA
UNIVERSIDAD
CATÓLICA
DE CHILE

OFF-POLICY LEARNING IN TWO-STAGE RECOMMENDER SYSTEMS

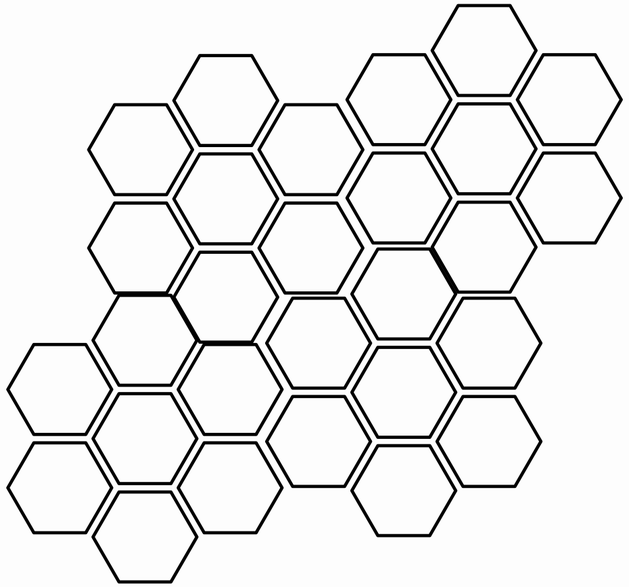
INTEGRANTES:
MANUEL CIFUENTES
DIEGO FERNÁNDEZ
JUAN MANUEL HERNÁNDEZ

ÍNDICE

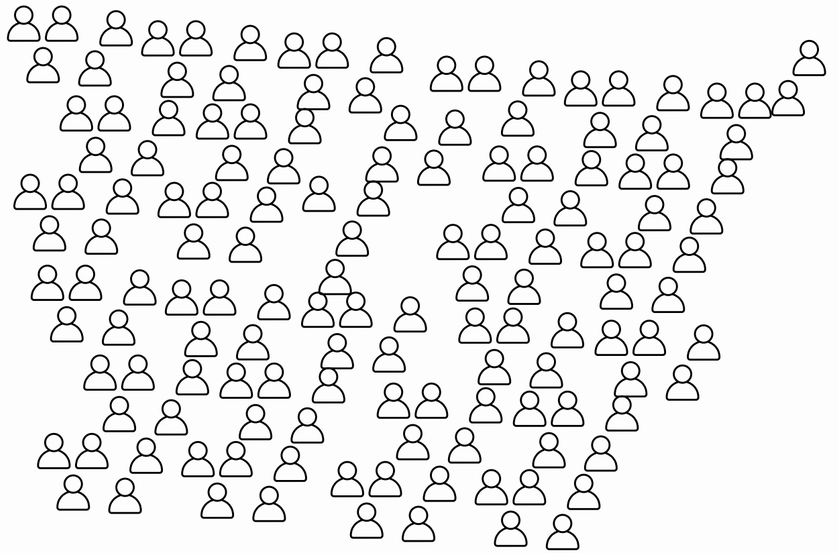
C O N T E X T O

Sistemas recomendados en la actualidad:

Tenemos millones
de items



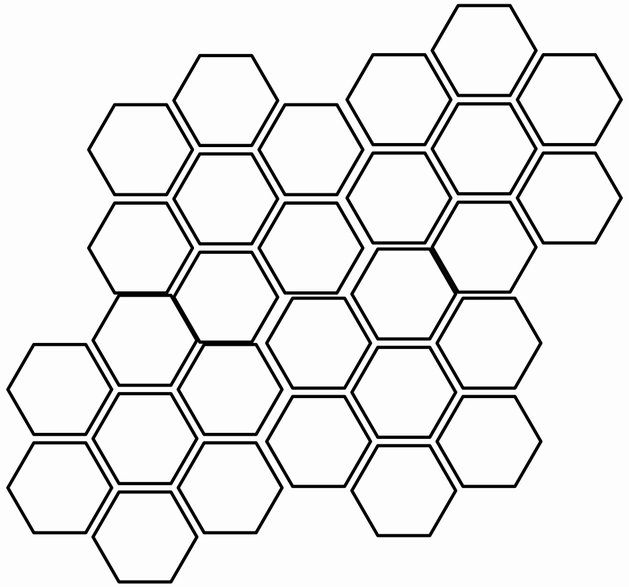
Tenemos billones
de usuarios



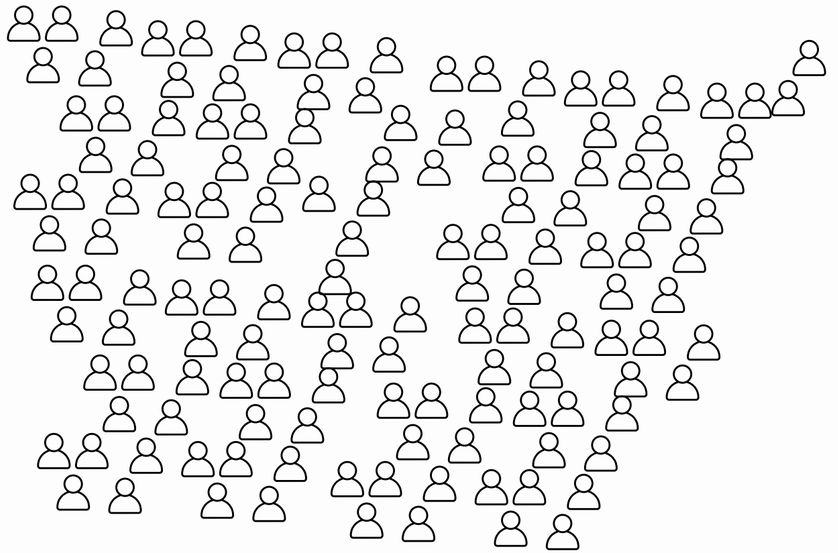
C O N T E X T O

Sistemas recomendados en la actualidad:

Tenemos millones
de items



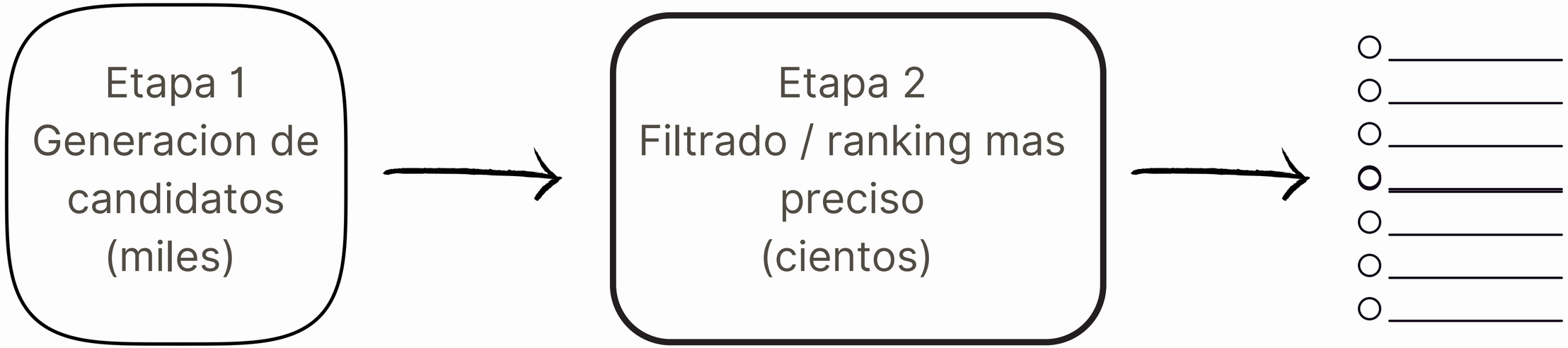
Tenemos billones
de usuarios



¿Cómo los relacionamos eficientemente?

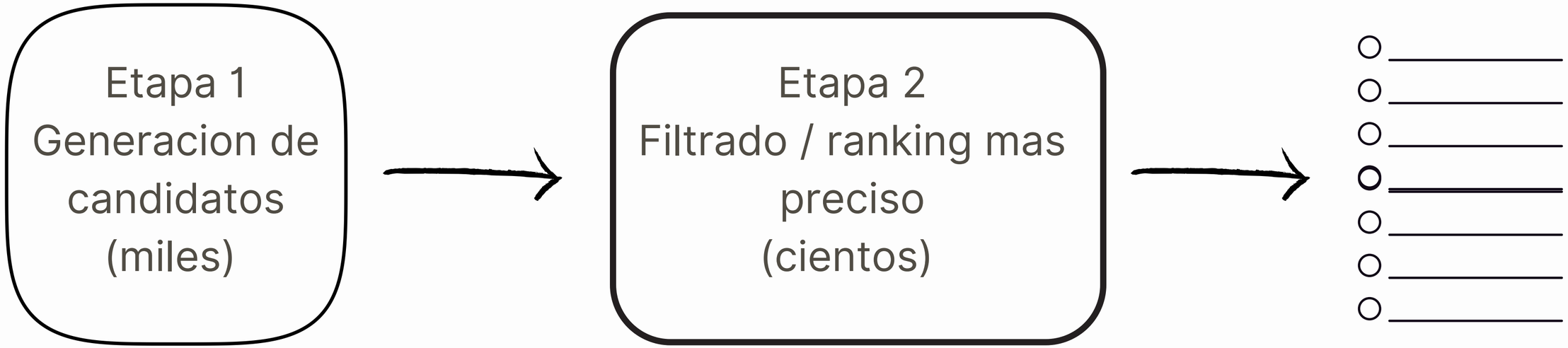
CONTEXTO

Posible solución... Sistema de recomendacion de dos etapas!!



CONTEXTO

Posible solución... Sistema de recomendacion de dos etapas!!



P R O B L E M A

1.-Sesgo en los Datos de Retroalimentación

- Feedback solamente en items previamente recomendados
- “The richs get richer”

2.- Limitaciones Computacionales

- Complejidad **O(|A|^k)** para calcular gradientes exactos
- Con millones de items (|A|) y k=50, computacionalmente intratable

3.- Interdependencia de Etapas

- Optimizar solo generación de candidatos → política sub-óptima del sistema
- Metodos existentes ignoran la interaccion entre etapas

"How do we correct biases when the system has two interdependent stages?"

CONTRIBUCIÓN

Método que explícitamente considera el modelo de ranking al entrenar el generador de candidatos, optimizando el rendimiento del sistema completo.

C O N T R I B U C I Ó N

- Algoritmo eficiente: Aproximación Monte Carlo que reduce complejidad de O(|A|^k) a computacionalmente factible
- Experimentos en MovieLens-1M y Wiki10-31K demuestran mejoras significativas
- Aplicabilidad práctica: Compatible con arquitecturas industriales existentes

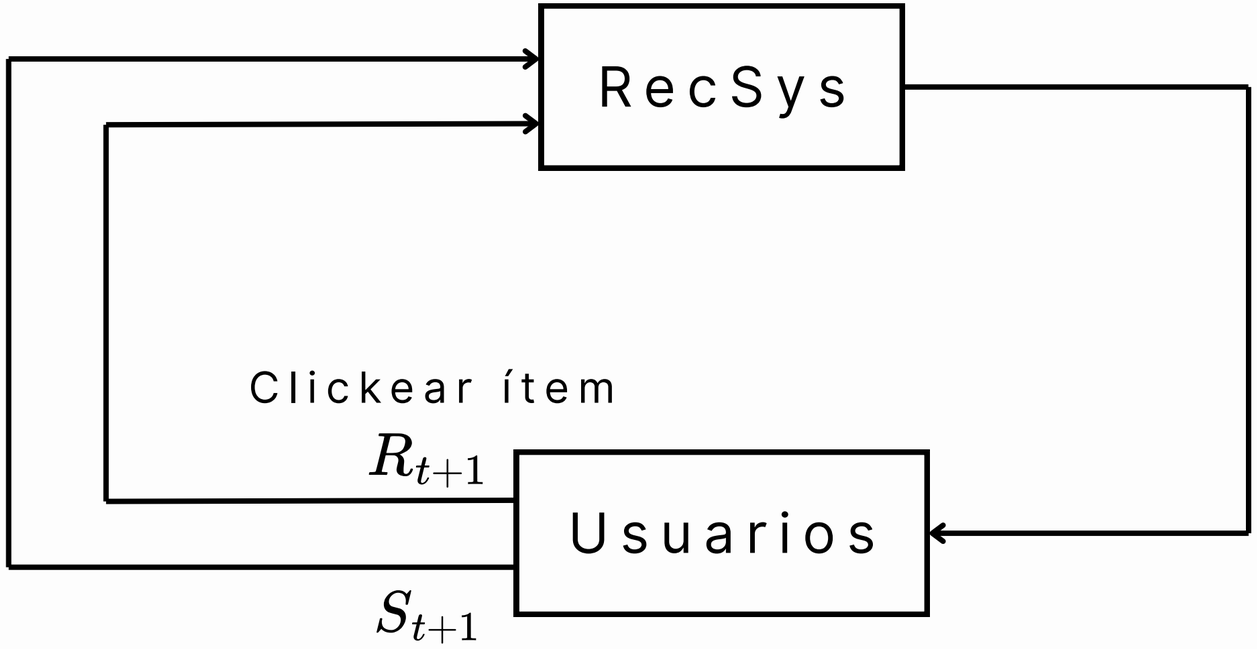
ESTADO DEL ARTE

Aplicaciones en RecSys, Swaminathan & Joachims, 2015

LinkedIn CaSMoS, Framework para selección de candidatos

Minmin Chen et. al 2019. Top-k off-policy correction for a REINFORCE recommender system.

APRENDIZAJE REFORZADO EN RECSYS



El objetivo del agente es que el usuario clickee los ítems recomendados

FORMAS DE APRENDIZAJE

La política del agente es cómo selecciona sus acciones:

$\beta(a|s)$ (behavior policy): cómo el agente selecciona sus acciones durante el entrenamiento

$\pi(a|s)$ (target policy): política que el agente quiere optimizar

On-policy learning → $\pi(a|s) = \beta(a|s)$

Off-policy learning → $\pi(a|s) \neq \beta(a|s)$

RECOMENDACIÓN COMO UN PROBLEMA DE RL

- Queremos entonces optimizar el valor de la recompensa promedio

$$V(\pi) = \mathbb{E}_{s\sim\rho(s), a\sim\pi(a|s)}\left[r(s,a)\right]$$

- Por simplicidad a esta ecuación se le denotará como $\mathbb{E}_{\pi}$

- Para poder modelar la política a optimizar, utilizamos una red neuronal. Es decir, la política $\pi_{\theta}(a|s)$

- Donde θ representa todos los pesos y sesgos de la red.

RECOMENDACIÓN COMO UN PROBLEMA DE RL

- Por lo tanto, se quiere optimizar $\mathbb{E}_{\pi_\theta}$, lo que se hace mediante el gradiente:

$$\nabla_\theta V(\pi_\theta) = \nabla_\theta \mathbb{E}_{a\sim\pi_\theta(a|s)}[r(s,a)]$$

- Que puede ser expresado como el gradiente REINFORCE:

$$\nabla_\theta V(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[r(s,a)\nabla_\theta \log \pi_\theta(a|s)\right]$$

RECOMENDACIÓN COMO UN PROBLEMA DE RL

- Como se está utilizando off-policy learning, se entrena con datos generados por la política $\beta(a|s)$
- Como esta política no es la optimizada, contiene sesgos.
- Los sesgos se corrigen con Inverse Propensity Scoring (IPS):

$$V(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[r(s,a)\right] = \mathbb{E}_{\beta}\left[\frac{\pi_\theta(a|s)}{\beta(a|s)} r(s,a)\right]$$

RECOMENDACIÓN COMO UN PROBLEMA DE RL

$$V(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[r(s,a)\right] = \mathbb{E}_{\beta}\left[\frac{\pi_\theta(a|s)}{\beta(a|s)} r(s,a)\right]$$

- A este término se le llama peso de importancia.

- Implica qué tanto debemos considerar la recompensa:
  - Si es mayor que 1, se le aumenta el peso de la recompensa
  - Si es igual a 1, se mantiene igual
  - Si es menor que 1, se reduce le peso de la recompensa

RECOMENDACIÓN COMO UN PROBLEMA DE RL

- Reemplazando el IPS en el gradiente

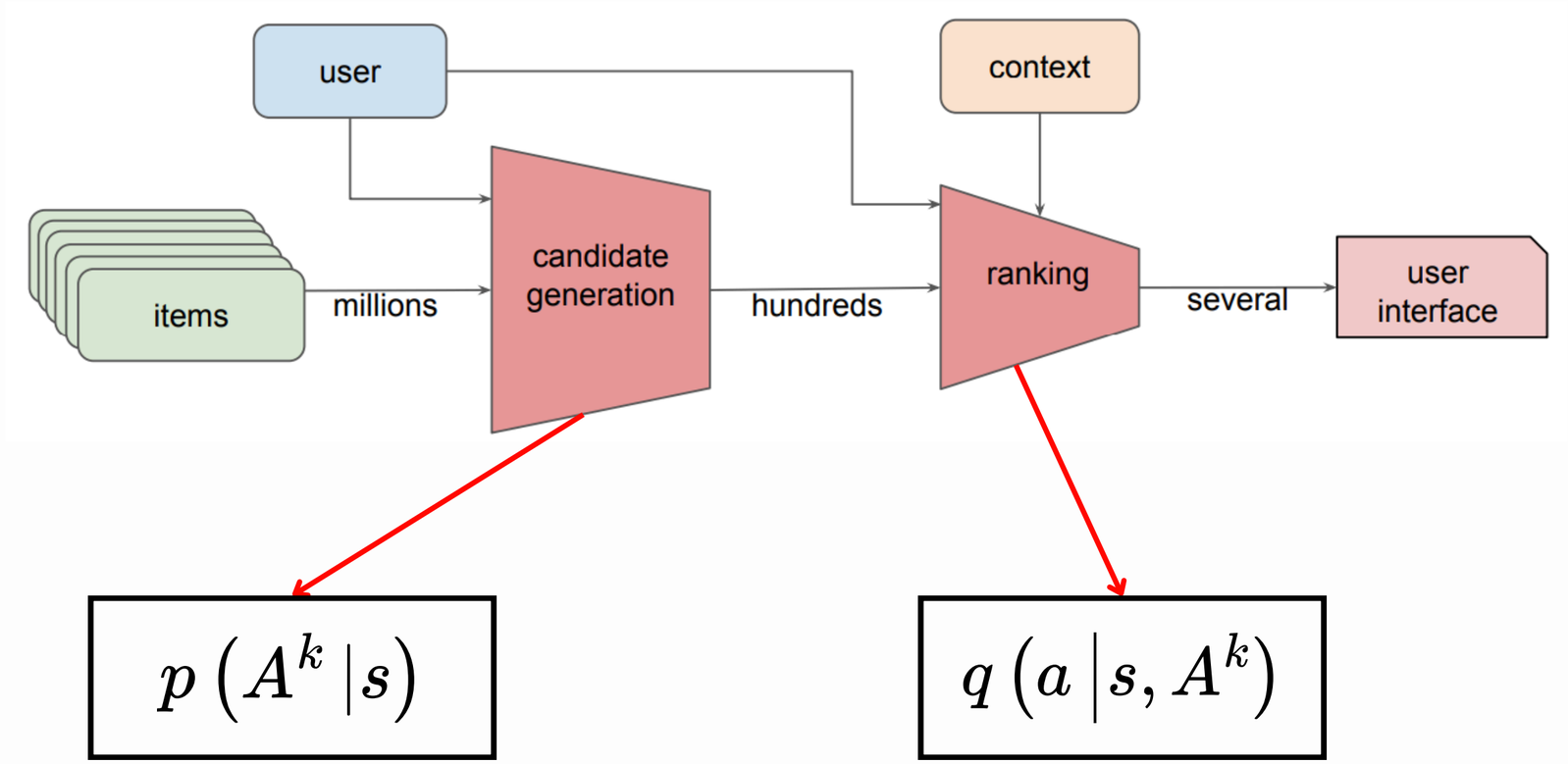
$$\nabla_\theta V(\pi_\theta) = \mathbb{E}_\beta \left[\frac{\pi_\theta(a|s)}{\beta(a|s)} r(s,a) \nabla_\theta \log \pi_\theta(a|s) \right]$$

- Este gradiente se entrena con el dataset $D = \{(s_i, a_i, r_i)\}_{i=1}^{n}$ generado por β
- Entonces, se puede escribir el gradiente como:

$$\nabla_\theta \hat{V}(\pi_\theta) = \frac{1}{n} \sum_{i=1}^{n} \frac{\pi_\theta(a_i|s_i)}{\beta(a_i|s_i)} r_i \nabla_\theta \log \pi_\theta(a_i|s_i)$$

RECOMENDACIÓN EN DOS ETAPAS

- Se tiene un modelo que genera candidatos y uno que los rankea.
- Sea A^k un set de ítems con tamaño k



RECOMENDACIÓN EN DOS ETAPAS

- Entonces, la política objetivo se puede escribir como:

$$\pi_\theta(a|s) = \sum_{A^k} p_\theta(A^k|s) q(a|s, A^k)$$

- En la ecuación se puede notar que la política óptima depende de:
  - $\rho_\theta\left(A^k|s\right)$ (modelo de generación de candidatos).
  - $q\left(a\left|s, A^k\right.\right)$ (modelo de ranking)
- Por lo tanto, si solo se optimiza uno sin tener en cuenta el otro, se puede obtener una estimación off-policy sesgada.

COMPLEJIDAD DEL GRADIENTE

- Reemplazando la política en el gradiente

$$\frac{1}{n}\sum_{i=1}^{n}\frac{\sum_{A^k} q(a_i|s_i,A^k)\nabla_\theta p_\theta(A^k|s_i)}{\beta(a_i|s_i)}r_i$$

- Se debe iterar por todos los posibles sets de candidatos con tamaño k, es decir, una complejidad de $O\left(|A|^k\right)$
- A contiene alrededor de millones o billones de datos
- No es viable computacionalmente

APROXIMACIÓN MEDIANTE MUESTREO

$$\frac{1}{n}\sum_{i=1}^{n}\frac{\sum_{A^k} q(a_i|s_i,A^k)\nabla_\theta p_\theta(A^k|s_i)}{\beta(a_i|s_i)}r_i$$

- Se puede notar que el gradiente de la política objetivo en la iteración i es:

$$\nabla_\theta \pi_\theta(a_i|s_i) = \sum_{A^k} q(a_i|s_i,A^k)\nabla_\theta p_\theta(A^k|s_i)$$

- Desarrollando y reescribiendo:

$$= \mathbb{E}_{A^k\sim p_\theta(A^k|s_i)}\left[q(a_i|s_i,A^k)\nabla_\theta \log p_\theta(A^k|s_i)\right]$$

APROXIMACIÓN MEDIANTE MUESTREO

$$\nabla_\theta \pi_\theta(a_i|s_i) = \mathbb{E}_{A^k \sim p_\theta(A^k|s_i)} \left[q(a_i|s_i, A^k) \nabla_\theta \log p_\theta(A^k|s_i) \right]$$

- En vez de iterar sobre todos los posibles conjuntos, se obtiene un estimador insesgado sampleando $A^k \sim p_\theta(A^k|s_i)$
- Lo que implica que se puede aproximar la esperanza con una muestra
- Pero, hay una ineficiencia:

$$q(a|s, A^k) = 0, \text{ if } a \notin A^k$$

MUESTREO CON REEMPLAZO

- Asumiendo que el set de candidatos es generado con muestreo con reemplazo:

$$p_\theta(A^k|s) = \prod_{j=1}^{k} p_\theta(A_j^k|s)$$

- Al reemplazar esto en la ecuación del gradiente en la iteración i:

$$\nabla_\theta \pi_\theta(a_i|s_i) = p_\theta(a_i|s_i)\, \mathbb{E}_{A^{k-1}} \left[q(a_i|s_i, \{a_i\} \cup A^{k-1}) \cdot \left(\nabla_\theta \log p_\theta(a_i|s_i) + \nabla_\theta \log p_\theta(A^{k-1}|s_i)\right)\right]$$

EJEMPLO: APLICACIÓN EN UN RECOMENDADOR SOFTMAX

- Se usa la función softmax en el output del recomendador
- Al definir una recompensa binaria, como:

$$r(s,a) = \begin{cases} 1, \text{ if } a \text{ is clicked given state } s, \\ 0, \text{ else,} \end{cases}$$

- El objetivo de aprendizaje de dos etapas se convierte en una versión ponderada de la pérdida de cross-entropy, usando los pesos de importancia.

EJEMPLO: APLICACIÓN EN UN RECOMENDADOR SOFTMAX

$$J_{\text{CE}}(\theta) = -\frac{1}{n}\sum_{i=1}^{n} r(s_i, a_i)\log p_\theta(a_i|s_i) \qquad J_{\text{1-IPS}}(\theta) = -\frac{1}{n}\sum_{i=1}^{n} \frac{\text{sg}(p_\theta(a_i|s_i))}{\beta(a_i|s_i)} r(s_i, a_i)\log p_\theta(a_i|s_i)$$

- Al considerar dos etapas, se expresa el IPS loss como:

$$J_{\text{2-IPS}}(\theta) = -\frac{1}{n}\sum_{i=1}^{n} \frac{\text{sg}(p_\theta(a_i|s_i))}{\beta(a_i|s_i)} r(s_i, a_i) h(\theta)$$

- Donde h es el gradiente del muestreo con reemplazo:

$$h(\theta) = \sum_{A^{k-1}} \Big[\text{sg}(p_\theta(A^{k-1}|s_i)) q(a_i|s_i, \{a_i\} \cup A^{k-1}) \Big(\log p_\theta(a_i|s_i) + \log p_\theta(A^{k-1}|s_i) \Big)\Big]$$

EJEMPLO: APLICACIÓN EN UN RECOMENDADOR SOFTMAX

$$h(\theta) = \sum_{A^{k-1}} \Big[\mathrm{sg}(p_\theta(A^{k-1}|s_i)) q(a_i|s_i, \{a_i\} \cup A^{k-1}) \Big(\log p_\theta(a_i|s_i) + \log p_\theta(A^{k-1}|s_i) \Big) \Big]$$

- Aproximando debido al muestro, considerando τ como el largo de la muestra:

$$h(\theta) \simeq \frac{1}{\tau} \sum_{A^{k-1} \sim p_\theta} \Big[q(a_i|s_i, \{a_i\} \cup A^{k-1}) \cdot \Big(\log p_\theta(a_i|s_i) + \log p_\theta(A^{k-1}|s_i) \Big) \Big]$$

EJEMPLO: APLICACIÓN EN UN RECOMENDADOR SOFTMAX

$$h(\theta) \simeq \frac{1}{\tau} \sum_{A^{k-1} \sim p_\theta} \left[q(a_i|s_i, \{a_i\} \cup A^{k-1}) \cdot \left(\log p_\theta(a_i|s_i) + \log p_\theta(A^{k-1}|s_i) \right) \right]$$

- Esta ecuación es uno de los mayores aportes del paper.
- Consideremos el caso de que a_i es mal rankeado:

$$q(a_i|s_i, \{a_i\} \cup A^{k-1}) \ll 1$$

  - Lo que implica que el gradiente será pequeño
  - El modelo generador de candidatos aprende a ignorar ítems que el modelo de ranking no va a valorar

EJEMPLO: APLICACIÓN EN UN RECOMENDADOR SOFTMAX

$$h(\theta) \simeq \frac{1}{\tau} \sum_{A^{k-1} \sim p_\theta} \left[q(a_i | s_i, \{a_i\} \cup A^{k-1}) \cdot \left(\log p_\theta(a_i | s_i) + \log p_\theta(A^{k-1} | s_i) \right) \right]$$

- Consideremos este segundo ejemplo:

- Un ítem a_i fue clickeado
- Existe un falso positivo a_j que el usuario no clickeó, pero el ranking le da más score que a a_i

$$q(a_i | s_i, \{a_i\} \cup A^{k-1}) \ll 1$$

  - Aunque a_j esté en A^{k-1}, no tiene recompensa directa porque no es clickeado, por lo tanto el gradiente es bajo
  - El modelo aprende a evitar generar a_j como candidato

ALGORITMO TWO STAGE IPS LOSS

$D = \{(s_i, a_i, r_i)\}_{i=1}^{n}$

$\beta(a|s)$

k

T

$\rho_\theta(a|s)$

$q(a|s, A^k)$

$m = |\mathcal{A}|$

$\approx J_{2-IPS}(\theta)$

EXPERIMENTOS

SETTING DE EXPERIMENTACIÓN

Se comparan 3 funciones de pérdida:

- Cross-entropy
- One-stage IPS (1-IPS)
- Two-stage IPS (2-IPS)

En 2 set de datos:

- MovieLens-1M (ratings)
- Wiki10-31K(labels)

MOVIELENS-1M

Dataset de usuario-película con ratings del 1 al 5:

- Información de usuario (edad, ocupación, zipcode)
- Información de película (título, género)

No se sabe nada sobre las películas que el usuario no evalúo.

PROBLEMA: DATOS NO ESTÁN PREPARADOS PARA UN AGENTE RL:

- Es necesario procesar los datos
- Datos incompletos
- Ratings son no binarios

APLICACIÓN DE RL A RECOMENDACIÓN

Transformación de dataset de aprendizaje supervisado a uno de tipo bandit.

Sea un dataset con **m** items (etiquetas):

$$\mathcal{D}_{\text{full}} = \{(\mathbf{x}^i, \mathbf{y}^i)\}_{i=1}^{n}$$

$\mathbf{x}^i$: es el vector de features del usuario (elemento) i

$\mathbf{y}^i$: es el vector de items (etiquetas), donde cada entrada tiene la forma:

y_a^i con $a \in \{1, 2, \cdots, m\}$ es decir $(y_1^i, y_2^i, \ldots, y_m^i)$

cada entrada es un valor de 0 ó 1 dependiendo de si el item es relevante para el usuario i. (Información completa de etiquetado).

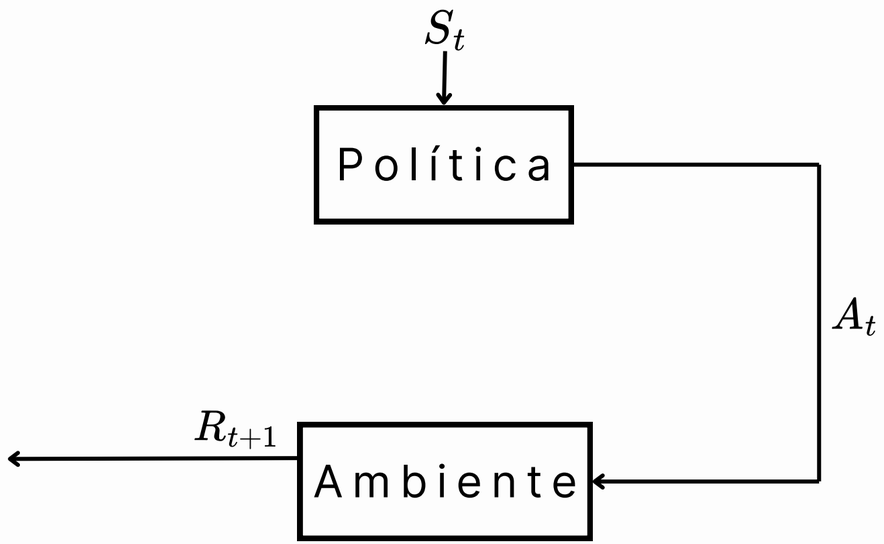
APLICACIÓN DE RL A RECOMENDACIÓN

Con lo anterior y con una política $\beta(a|\mathbf{x})$ se puede generar un dataset para RL:

$$\mathcal{D}_{\text{bandit}} = \{(\mathbf{x}^j, a^j, p^j, r^j)\}_{j=1}^{l}$$

El procedimiento es:

- La política escoge una película a partir del usuario (contexto).
- Se registra:
  - El usuario.
  - La película (acción del agente).
  - La probabilidad de elección.
  - La recompensa: $r^j = \mathbf{y}^j_{a^j}$



¿QUÉ HACER CON LA PELÍCULAS CON LAS QUE EL USUARIO NO HA INTERACTUADO?

NO SABEMOS LA RECOMPENSA EN ESTOS CASOS

SOLUCIÓN: SIMULAR RECOMPENSAS

Es por esto que para este set de datos, es necesario simular la interacción con las películas que el usuario no ha visto.

Se entrena un modelo, que recibe un par usuario acción y predice 1 ó 0.

Con este modelo se predicen todos las interacciones de los pares usuario-película.

Con estos datos se obtiene el dataset full antes mostrado

$$\mathcal{D}_{\text{full}} = \{(\mathbf{x}^i, \mathbf{y}^i)\}_{i=1}^{n}$$

Luego se procede como se explicó anteriormente para obtener:

$$\mathcal{D}_{\text{bandit}} = \{(\mathbf{x}^j, a^j, p^j, r^j)\}_{j=1}^{l}$$

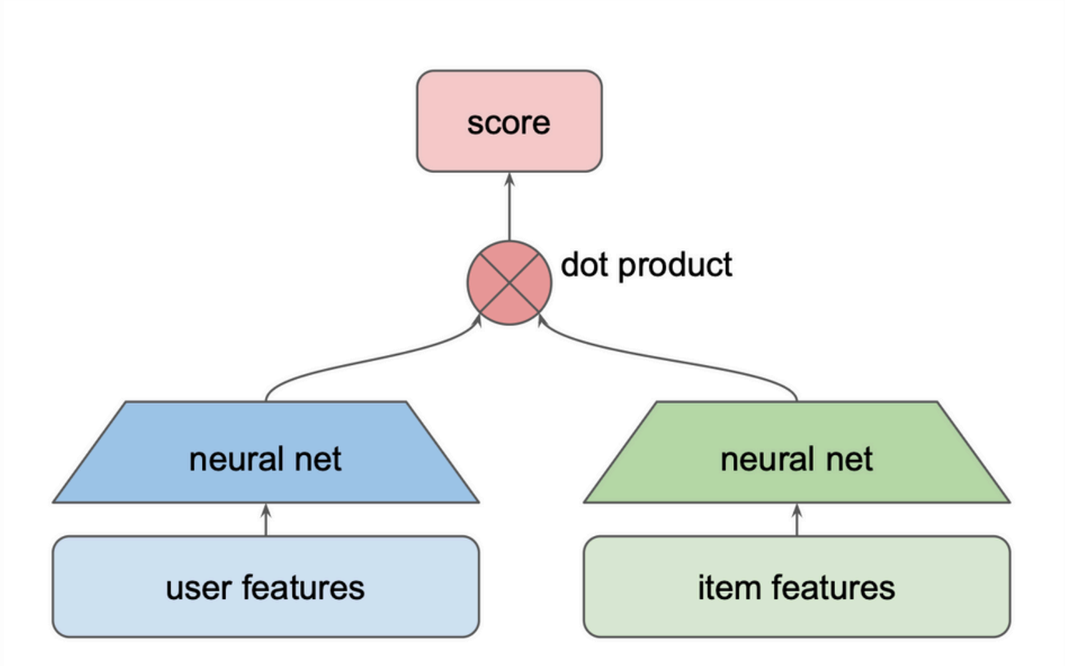
PROCEDIMIENTO MOVIELENS-1M

TWO-STAGE RECOMMENDER INFERENCIA

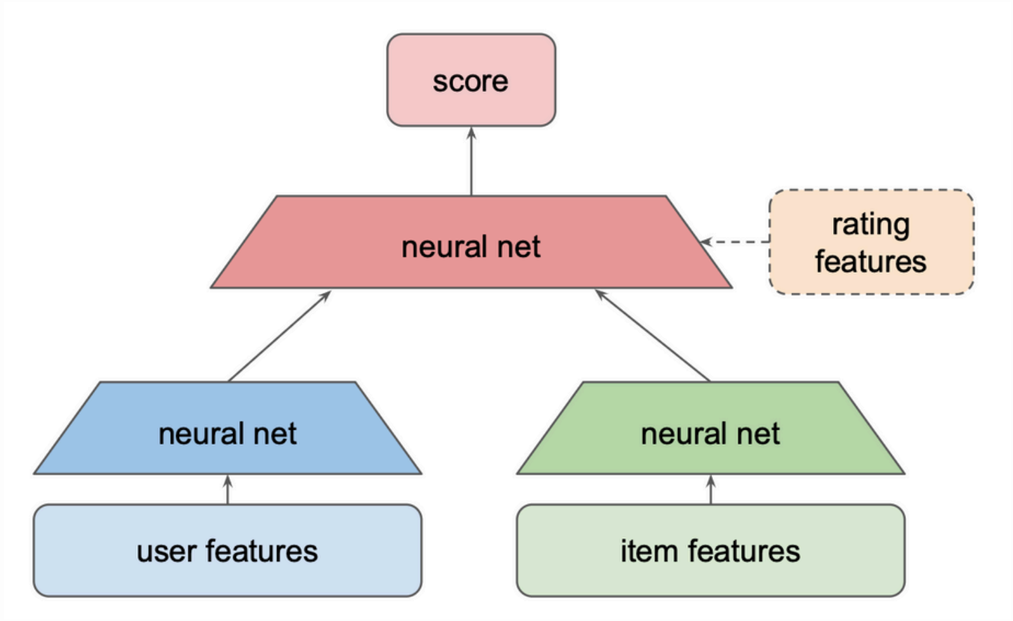
Simulator

Policy

ARQUITECTURA DE REDES



Two-Tower model



Other model

Candidate

Behaviour policy

Ranker

Simulator

FLUJO DE ENTRENAMIENTO 2-IPS

Se aplica un threshold para hacer binario el rating (>3)

Dataset original

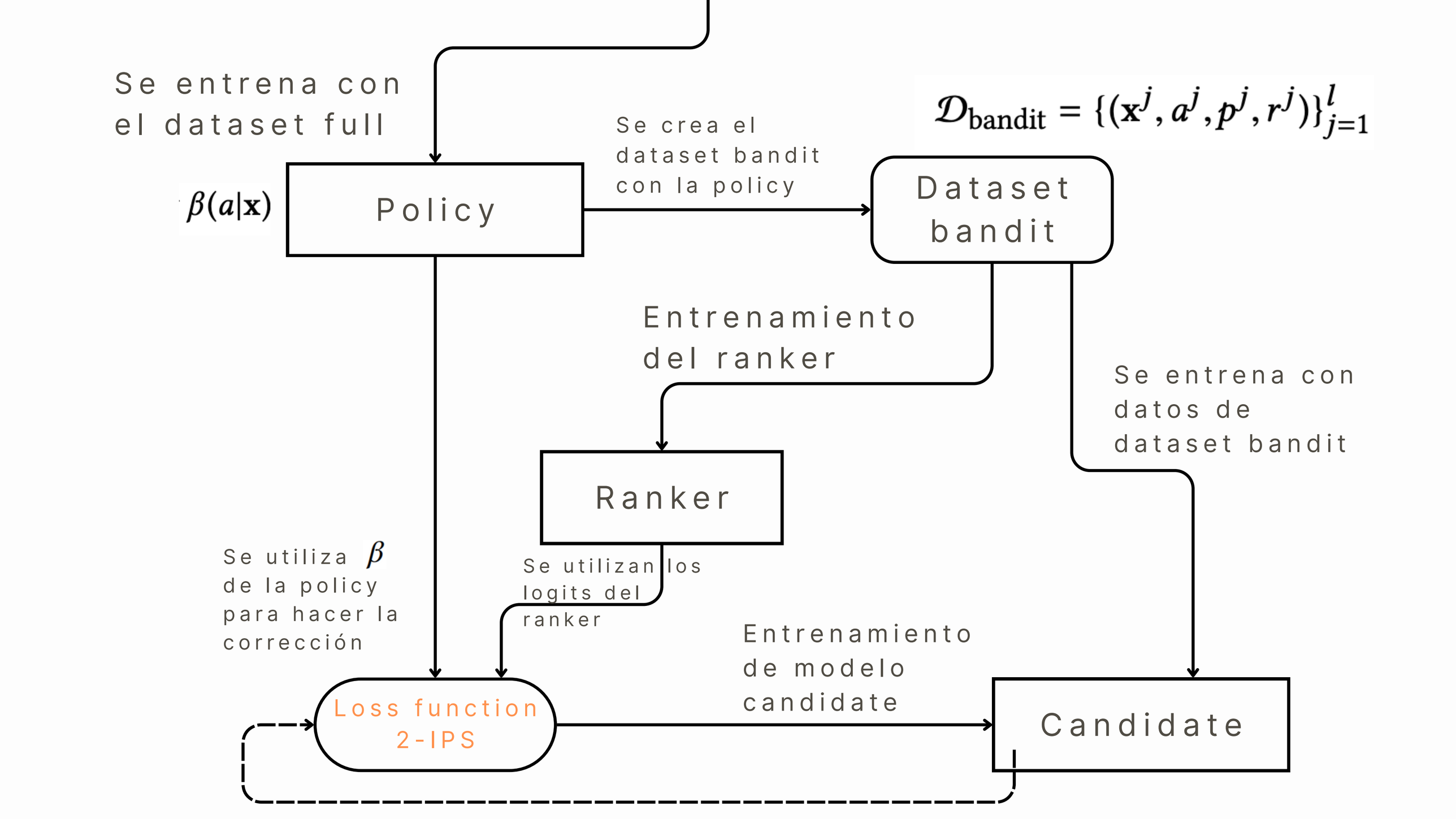
Con estos datos se entrena el simulador

Simulator

Genera ratings (1 ó 0) para todos los pares user-item

Dataset fulll

$$\mathcal{D}_{\text{full}} = \{(\mathbf{x}^i, \mathbf{y}^i)\}_{i=1}^{n}$$


Se entrena con el dataset full
$\beta(a|\mathbf{x})$
Policy
Se crea el dataset bandit con la policy
$\mathcal{D}_{\text{bandit}} = \{(\mathbf{x}^j, a^j, p^j, r^j)\}_{j=1}^{l}$
Dataset bandit
Entrenamiento del ranker
Ranker
Se entrena con datos de dataset bandit
Se utiliza β de la policy para hacer la corrección
Se utilizan los logits del ranker
Entrenamiento de modelo candidate
Loss function 2-IPS
Candidate

RESULTADOS MOVIELENS-1M

MOVIELENS-1M DATASET

Se entrenaron 20 modelos, two-stage evalúa la recomendación del ranker (top-1)

k = 30

Table 2: Ranking metrics (%) of two-stage evaluation on MovieLens-1M. The number after ± indicates the standard error of the mean over 20 runs. The percentage in indicates the relative improvement over the Cross-Entropy method.

	Precision@5	Precision@10	Recall@5	Recall@10	NDCG@5	NDCG@10
Cross-Entropy	95.8 ± 0.1 (+0%)	93.9 ± 0.0 (+0%)	2.2 ± 0.0 (+0%)	3.5 ± 0.0 (+0%)	96.4 ± 0.1 (+0%)	95.0 ± 0.1 (+0%)
1-IPS	95.1 ± 0.2 (-1%)	93.3 ± 0.2 (-1%)	2.0 ± 0.1 (-8%)	3.4 ± 0.1 (-3%)	95.7 ± 0.3 (-1%)	94.3 ± 0.2 (-1%)
2-IPS	**98.1** ± 0.1 (+2%)	**96.8** ± 0.1 (+3%)	**2.9** ± 0.0 (+33%)	**4.9** ± 0.0 (+39%)	**98.3** ± 0.1 (+2%)	**97.5** ± 0.1 (+3%)

- Precision: Items relevantes dentro de los primeros n
- Recall: Proporción de relevantes recuperados y relevantes existentes.
- NDCG: Calidad del ranking. Considera orden.

MOVIELENS-1M DATASET

One-stage evalúa la recomendación del candidate (top-k)

k = 30

Table 3: Ranking metrics (%) of one-stage evaluation on MovieLens-1M. Notations are the same as Table 2.

	Precision@5	Precision@10	Recall@5	Recall@10	NDCG@5	NDCG@10
Cross-Entropy	86.3 ± 0.1 (+0%)	86.4 ± 0.1 (+0%)	1.2 ± 0.0 (+0%)	2.4 ± 0.0 (+0%)	86.3 ± 0.1 (+0%)	86.4 ± 0.1 (+0%)
1-IPS	90.1 ± 0.4 (+4%)	88.9 ± 0.3 (+3%)	1.5 ± 0.0 (+27%)	2.8 ± 0.1 (+17%)	90.3 ± 0.4 (+5%)	89.4 ± 0.3 (+3%)
2-IPS	**95.7** ± 0.2 (+11%)	**95.3** ± 0.2 (+10%)	**2.3** ± 0.1 (+102%)	**4.5** ± 0.1 (+89%)	**95.4** ± 0.3 (+11%)	**95.3** ± 0.2 (+10%)

PROCEDIMIENTO WIKI10-31K

WIKI10-31K

Dataset de artículos de wikipedia con multi-etiqueta:

- Información del artículo (Bag of words)

Se trata este problema como uno de recomendación, con:

- Usuarios: artículos.
- Ítems: etiquetas.

RESULTADOS WIKI10

WIKI10-31K DATASET

Table 4: Top-1 precisions (%) of two-stage evaluation and one-stage evaluation on Wiki10-31K. Notations are the same as Table 2.

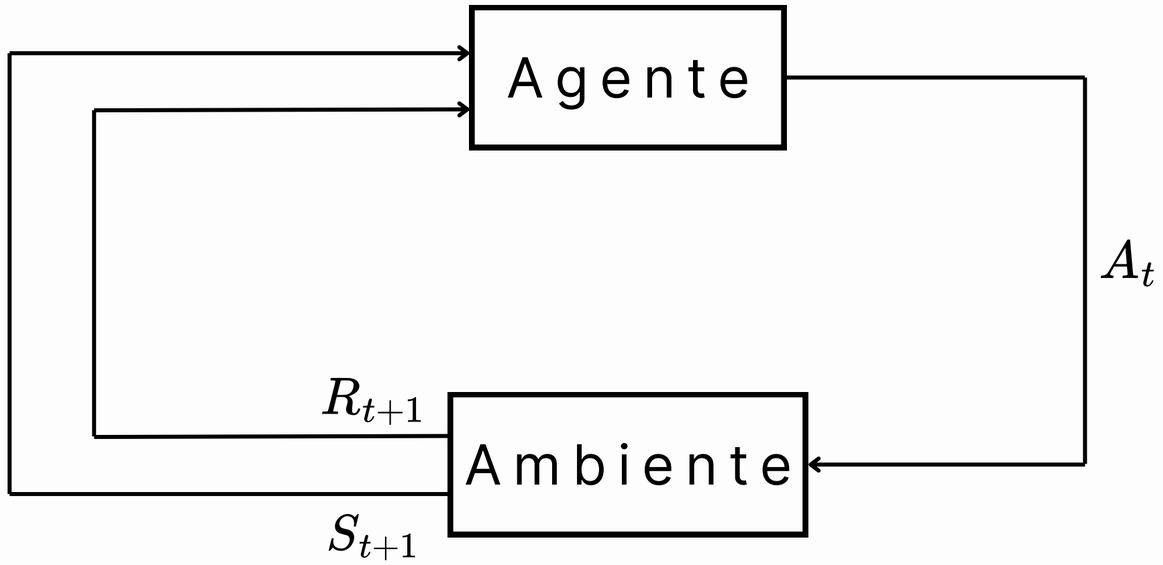
	One-stage Eval	Two-stage Eval
Cross-Entropy	19.3 ± 0.4 (+0%)	38.8 ± 0.3 (+0%)
1-IPS	20.8 ± 0.4 (+8%)	39.6 ± 0.3 (+2%)
2-IPS	**21.7** ± 0.4 (+12%)	**40.5** ± 0.3 (+4%)

REFERENCIAS

- Ma, J., Zhao, Z., Yi, X., Yang, J., Chen, M., Tang, J., Hong, L., & Chi, E. (2020). Off-policy Learning in Two-stage Recommender Systems. In Proceedings of The Web Conference 2020 (WWW '20). https://doi.org/10.1145/3366423.3380130
- Jiaqi Ma, GitHub: https://github.com/jiaqima/Off-Policy-2-Stage
- Dataset MovieLens-1M: https://grouplens.org/datasets/movielens/1m/
- Dataset Wiki10-31K: http://manikvarma.org/downloads/XC/XMLRepository.html#wiki10
- Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H Chi. 2019. Top-k off-policy correction for a REINFORCE recommender system. In Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining. ACM, 456–464.
- Adith Swaminathan and Thorsten Joachims. 2015. Batch learning from logged bandit feedback through counterfactual risk minimization. Journal of Machine Learning Research 16, 1 (2015), 1731–1755.

ANEXOS

APRENDIZAJE REFORZADO

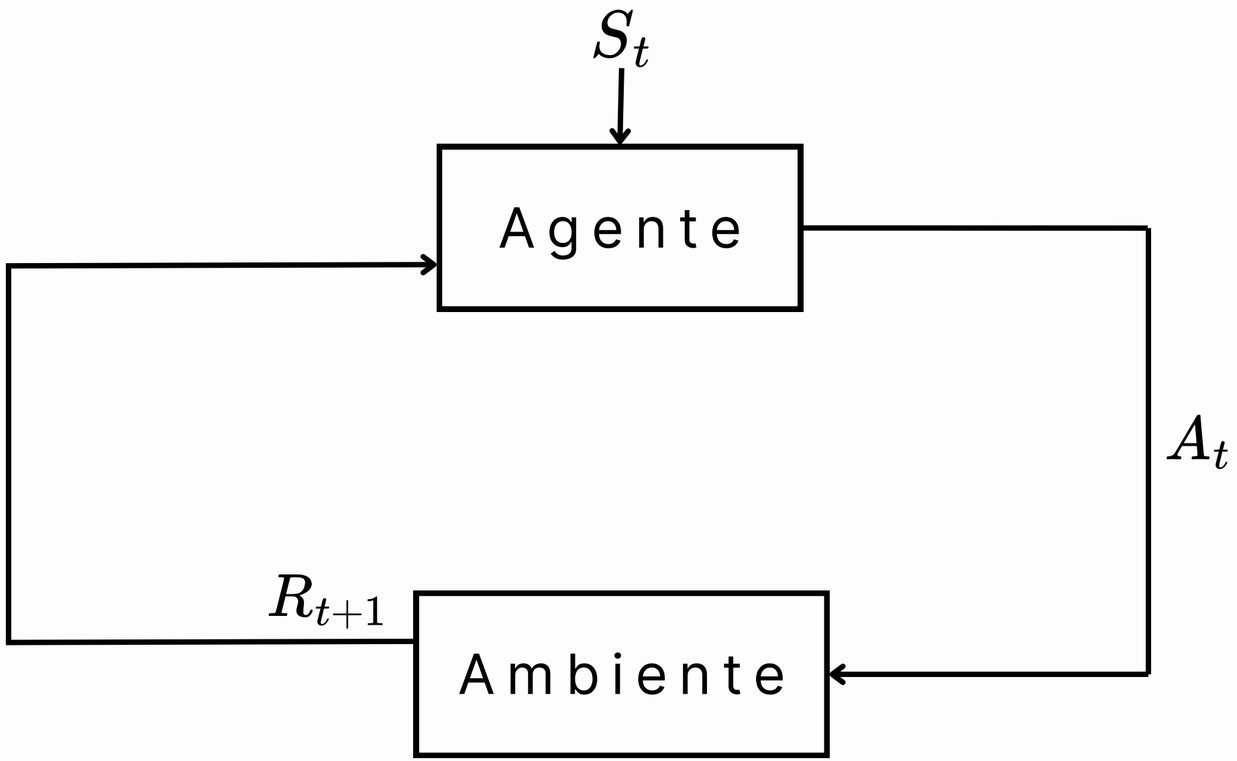


Como resultado se producen episodios:

$$S_0, A_0, R_1, S_{1,} A_{1,...}$$

El objetivo del agente es maximizar la recompensa

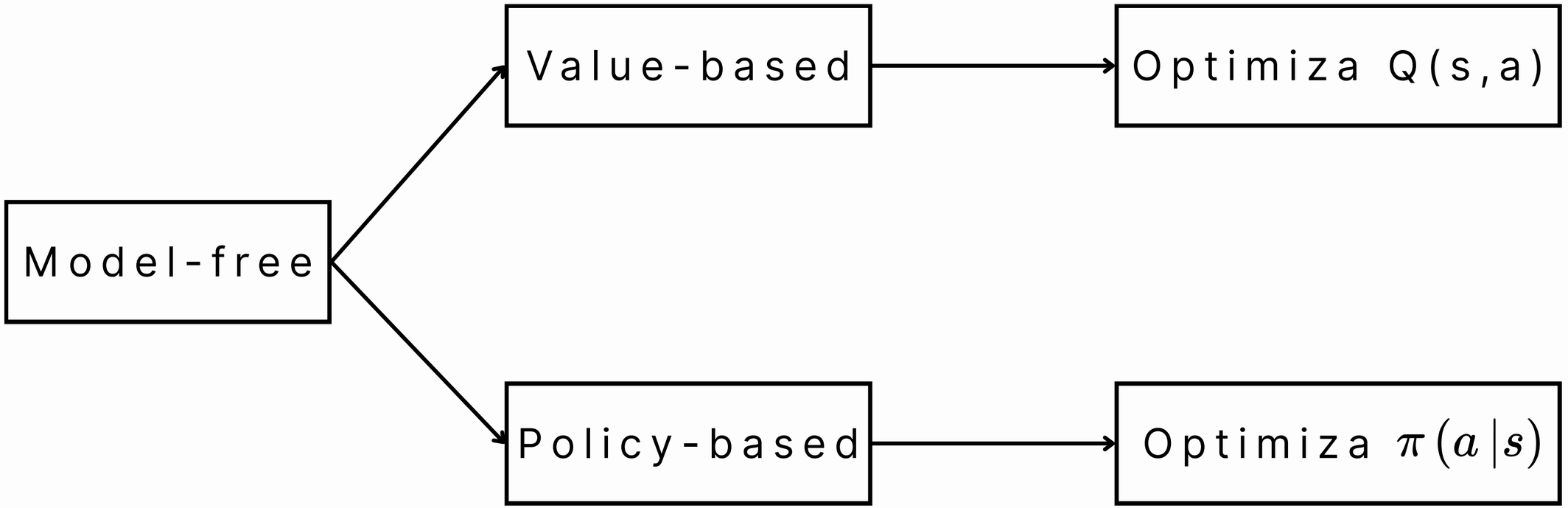
CONTEXTUAL BANDITS



Como resultado se producen episodios de una sola acción:

$$S_0, A_0, R_1$$

¿CÓMO APRENDE EL AGENTE?



Policy-based suele ser más estable con NN y es la elección del paper

OFF-POLICY LEARNING

- On-policy learning necesita que el agente interactúe con el ambiente
- En RecSys esto no es lo ideal ya que podría ser caro, riesgoso o incluso ilegal
- Para esto, se ocupa off-policy learning, que puede aprender de datos seleccionados por políticas distintas a la que busca optimizar

RECOMENDACIÓN COMO UN PROBLEMA DE RL

- Se define la notación:
  - S: espacio de usuario-estado, donde cada estado describe a un usuario acompañado por su contexto
  - A: set de posibles ítem

RECOMENDACIÓN COMO UN PROBLEMA DE RL

- Queremos entonces optimizar el valor de la recompensa promedio

$$V(\pi) = \mathbb{E}_{s_0 \sim \rho(s), a_t \sim \pi(a|s_t), s_{t+1} \sim P(s|s_t, a_t)} \left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t) \right]$$

*s corresponde a cada usuario-estado
*a corresponde a cada ítem

- Como en contextual Bandits solo hay un paso por episodio:

$$V(\pi) = \mathbb{E}_{s \sim \rho(s), a \sim \pi(a|s)} \left[r(s, a) \right]$$

- Por simplicidad a esta ecuación se le denotará como $\mathbb{E}_\pi$

RECOMENDACIÓN COMO UN PROBLEMA DE RL

$$V(\pi) = \sum_{s} \rho(s) \sum_{a} \pi(a \mid s) \cdot r(s, a)$$

Donde:

- $\rho(s)$: distribución inicial de usuarios
- $\pi(a|s)$: probabilidad de recomendar el ítem a al usuario s
- $r(s, a)$: recompensa de recomendar ítem a al usuario s

Por simplicidad a esta ecuación se le denotará como $\mathbb{E}_{\pi}$

RECOMENDACIÓN EN DOS ETAPAS

$$\pi_\theta(a|s) = \sum_{A^k} p_\theta(A^k|s) q(a|s, A^k)$$

Política objetivo

$$\nabla_\theta \hat{V}(\pi_\theta) = \frac{1}{n} \sum_{i=1}^{n} \frac{\pi_\theta(a_i|s_i)}{\beta(a_i|s_i)} r_i \nabla_\theta \log \pi_\theta(a_i|s_i)$$

Gradiente

$$\frac{1}{n} \sum_{i=1}^{n} \frac{\sum_{A^k} q(a_i|s_i, A^k) \nabla_\theta p_\theta(A^k|s_i)}{\beta(a_i|s_i)} r_i$$

Reemplazando

TRUCOS DE REDUCCIÓN DE VARIANZA

- Los métodos de gradiente off-policy en general tienen una varianza alta debido al peso de importancia

$$\omega(s,a) = \frac{\pi(a|s)}{\beta(a|s)}$$

- Primero que todo, se recorta el resultado:

$$\hat{\omega}_{c_1,c_2}(s,a) = \max\left\{\min\{\omega(s,a), c_1\}, c_2\right\}$$

- Después, se normaliza:

$$\hat{\omega}_n(s,a) = \frac{\omega(s,a)}{\frac{1}{n}\sum_{(s',a')\sim\beta}\omega(s',a')}$$

TRUCOS DE REDUCCIÓN DE VARIANZA

- Además, se agrega un hiperparámetro $\alpha \in [0, 1)$ al gradiente:

$$\nabla_\theta \log p_\theta(a_i|s_i) + \nabla_\theta \log p_\theta(A^{k-1}|s_i)$$

- Resultando en:

$$\nabla_\theta \log p_\theta(a_i|s_i) + \alpha \cdot \nabla_\theta \log p_\theta(A^{k-1}|s_i)$$

ALGORITMO TWO STAGE IPS LOSS

$J \leftarrow 0;$

for $i = 1, \cdots, n$ **do**

  $J_i \leftarrow 0;$

  for $j = 1, \cdots, n$ **do**

    $\tilde{p}(a_j|s_i) \leftarrow \text{stop\_gradient}(p_\theta(a_j|s_i));$

  end

  $\omega(a_i|s_i) \leftarrow \frac{\tilde{p}(a_i|s_i)}{\beta(a_i|s_i)};$

Inicialización de la pérdida

ALGORITMO TWO STAGE IPS LOSS

$J \leftarrow 0;$

for $i = 1, \cdots, n$ **do**

  $J_i \leftarrow 0;$

  for $j = 1, \cdots, n$ **do**

    $\tilde{p}(a_j|s_i) \leftarrow \text{stop\_gradient}(p_\theta(a_j|s_i));$

  end

  $\omega(a_i|s_i) \leftarrow \frac{\tilde{p}(a_i|s_i)}{\beta(a_i|s_i)};$

Se itera a través de todos los usuarios

ALGORITMO TWO STAGE IPS LOSS

$J \leftarrow 0;$

for $i = 1, \cdots, n$ **do**

  $J_i \leftarrow 0;$

  for $j = 1, \cdots, n$ **do**

    $\tilde{p}(a_j|s_i) \leftarrow \text{stop\_gradient}(p_\theta(a_j|s_i));$

  end

  $\omega(a_i|s_i) \leftarrow \frac{\tilde{p}(a_i|s_i)}{\beta(a_i|s_i)};$

Se inicializa la pérdida de la iteración actual

ALGORITMO TWO STAGE IPS LOSS

$J \leftarrow 0;$

for $i = 1, \cdots, n$ **do**

  $J_i \leftarrow 0;$

  for $j = 1, \cdots, n$ **do**

    $\tilde{p}(a_j|s_i) \leftarrow \text{stop\_gradient}(p_\theta(a_j|s_i));$

  end

  $\omega(a_i|s_i) \leftarrow \frac{\tilde{p}(a_i|s_i)}{\beta(a_i|s_i)};$

Se pausa la política para no propagar la actualización

ALGORITMO TWO STAGE IPS LOSS

$J \leftarrow 0;$

for $i = 1, \cdots, n$ **do**

  $J_i \leftarrow 0;$

  for $j = 1, \cdots, n$ **do**

    $\tilde{p}(a_j|s_i) \leftarrow \text{stop\_gradient}(p_\theta(a_j|s_i));$

  end

  $\omega(a_i|s_i) \leftarrow \frac{\tilde{p}(a_i|s_i)}{\beta(a_i|s_i)};$

Se actualiza el peso de importancia

ALGORITMO TWO STAGE IPS LOSS

for $t = 1, \cdots, \tau$ **do**
  $A_t \leftarrow \{a_i\}$;
  for $v = 1, \cdots, k-1$ **do**
    Sample $a_v \sim \tilde{p}$;
    $A_t \leftarrow A_t \cup \{a_v\}$;
  end

Se itera para generar las T muestras

ALGORITMO TWO STAGE IPS LOSS

for $t = 1, \cdots, \tau$ **do**
  $A_t \leftarrow \{a_i\}$;
  for $v = 1, \cdots, k-1$ **do**
    Sample $a_v \sim \tilde{p}$;
    $A_t \leftarrow A_t \cup \{a_v\}$;
  end

Se incluye el elemento a_i para evitar gradientes nulos

ALGORITMO TWO STAGE IPS LOSS

for $t = 1, \cdots, \tau$ **do**
  $A_t \leftarrow \{a_i\}$;
  for $v = 1, \cdots, k-1$ **do**
    Sample $a_v \sim \tilde{p}$;
    $A_t \leftarrow A_t \cup \{a_v\}$;
  end

Se muestrean los elementos

ALGORITMO TWO STAGE IPS LOSS

$A_t \leftarrow \text{unique}(A_t)$;
$l_t \leftarrow 0$;
for $a_v \in A_t$ **do**
  $l_t \leftarrow l_t + \log p_\theta(a_v|s_i)$;
end

Se calculan las log-probabilidades

$$\sum_{A^k} \log p_\theta(a_i|s_i)$$

ALGORITMO TWO STAGE IPS LOSS

end

$J_i \leftarrow J_i + q(a_i|s, A_t) \cdot l_t;$

end

$J \leftarrow J - r_i \cdot \omega(a_i|s_i) \cdot \frac{1}{\tau} J_i;$

Se actualiza la pérdida según el ranking de la muestra generada

ALGORITMO TWO STAGE IPS LOSS

end

$J_i \leftarrow J_i + q(a_i|s, A_t) \cdot l_t;$

end

$J \leftarrow J - r_i \cdot \omega(a_i|s_i) \cdot \frac{1}{\tau} J_i;$

Se actualiza la pérdida total con el valor de la muestra (s_i, a_i, r_i) y con el promedio de la pérdida de las muestras

ALGORITMO TWO STAGE IPS LOSS

end

$J \leftarrow \frac{1}{n} J;$

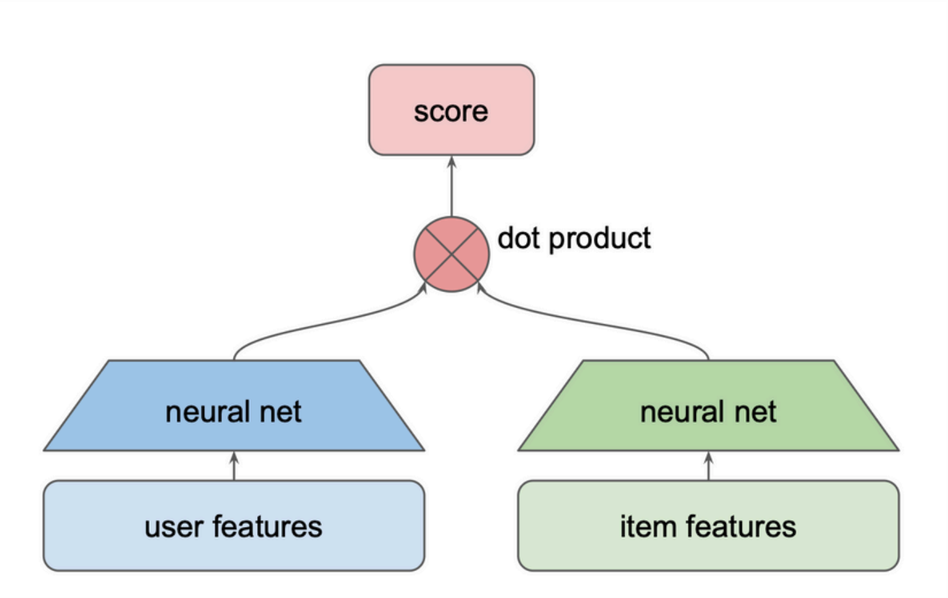
Return $J;$

Se calcula el promedio de la pérdida y se retorna

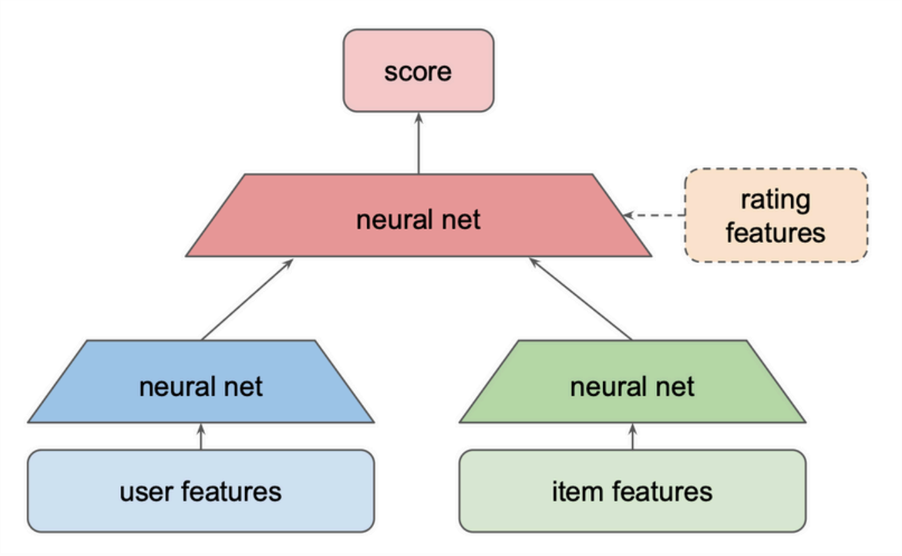
- 1-IPS and 2-IPS, we apply self-normalization and weight capping with c1 = 10,c2 = 0.01. For 2-IPS, we set $\varphi = 1e \simeq 2$ (see Eq. 8) and the candidate set sample size $\tau = 100$

ARQUITECTURA DE REDES

El entrenamiento es el mismo, sólo que este dataset sí tenía etiquetado completo por lo que no es necesario simular

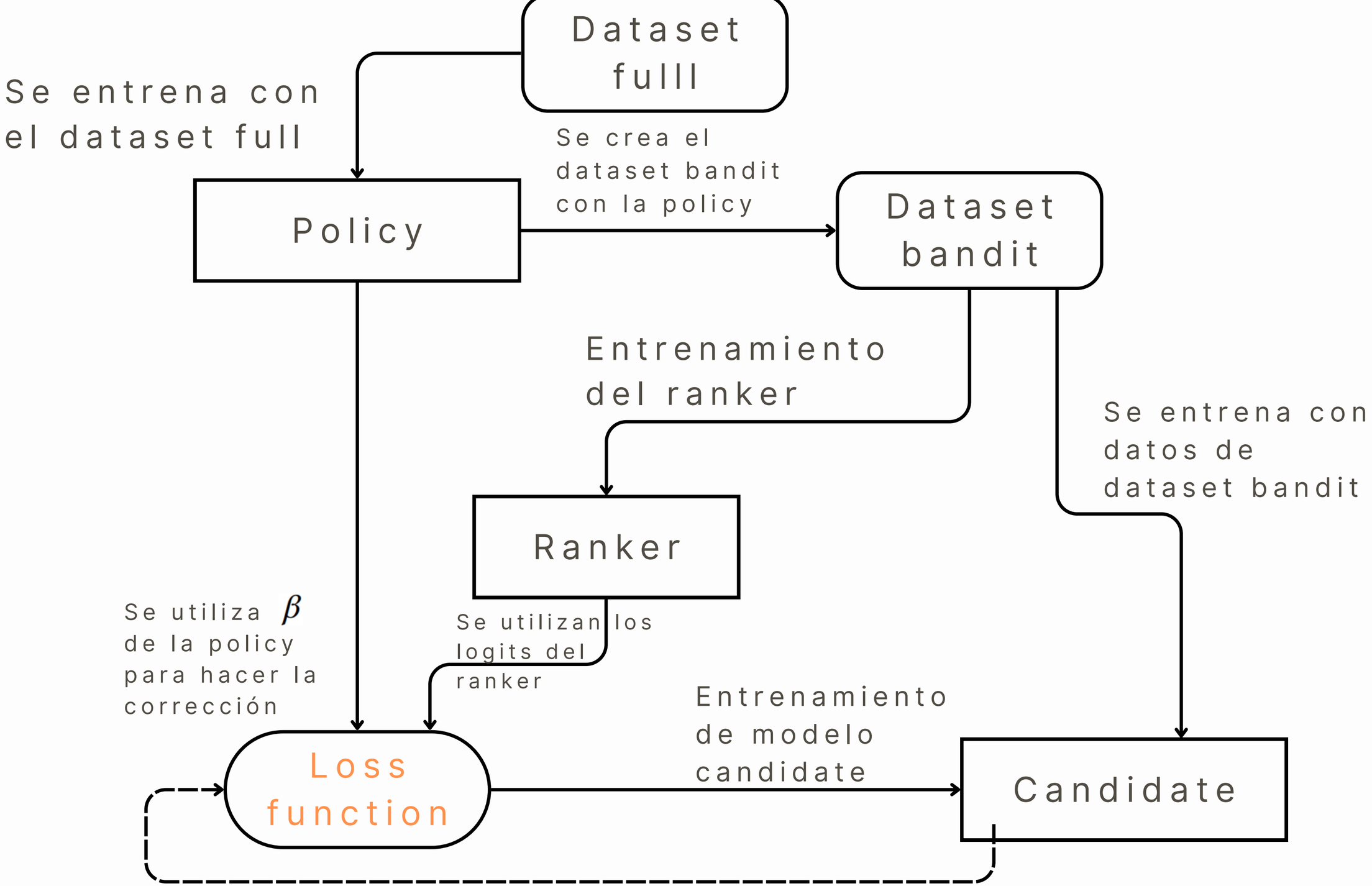


Two-Tower model



Other model

Candidate Behaviour policy Ranker Simulator


Dataset fulll
Se entrena con el dataset full
Policy
Se crea el dataset bandit con la policy
Dataset bandit
Entrenamiento del ranker
Se entrena con datos de dataset bandit
Ranker
Se utiliza β de la policy para hacer la corrección
Se utilizan los logits del ranker
Entrenamiento de modelo candidate
Loss function
Candidate

MOVIELENS-1M

- 1 Millón de ratings, 3900 películas, 6040 usuarios.

WIKI10-31K

- 31000 etiquetas, 20762 artículos

TWO-STAGE RECOMMENDER INFERENCIA

Simulator

Policy