

# **Recommendation Unlearning**

**Chong Chen, Fei Sun, Min Zhang, Bolin Ding**

Lucas Vidal, Nicolás Herrera, Vicente Steidle  
IIC3666

# Contexto

En muchos casos, los sistemas necesitan olvidar datos de entrenamiento:

## 1. Privacidad

- Información sensible de usuarios puede ser filtrada por los modelos entrenados.
- Usuarios desean una herramienta que elimine el impacto de su información sensible en los modelos entrenados.

# Contexto

## 2. Utilidad

- Nueva información es recolectada de forma incremental para entrenar/refinar los modelos.
- Datos erróneos (*dirty data*) degradan gravemente el rendimiento de las recomendaciones.
- Además, las preferencias de los usuarios son dinámicas y cambiantes → se requiere eliminar ciertos datos para que el sistema pueda proporcionar recomendaciones más útiles.

# Problema de recomendación

¿Cómo podemos diseñar un sistema de recomendación capaz de **“desaprender”** (*unlearn*) eficientemente ciertos datos, por motivos de privacidad, utilidad o cambios en preferencias, **sin necesidad de reentrenar** todo el modelo y **sin degradar la calidad** de las recomendaciones?

# Problema de recomendación

**3 datasets reales:**

- Yelp2018
- MovieLens-1m
- MovieLens-10m

**3 modelos de recomendación representativos:**

- BPR
- WMF
- LightGCN

# Contribución

- **RecEraser:** el primero *framework* general para abordar el problema de *unlearning* en sistemas de recomendación.
- Se proponen **tres estrategias de partición de datos** y un **método de agregación adaptativa** basado en atención para mantener el rendimiento.
- Se realizan experimentos en **tres datasets reales y tres modelos de recomendación**, demostrando que RecEraser es eficiente y supera *frameworks* previos.

# Estado del arte y marco teórico

## **SISA (*Sharding Isolation Separation Aggregation*)**

- Tiene su origen en visión computacional y NLP.
- Divide los datos en fragmentos y entrena submodelos → no funciona bien en recomendación por la pérdida de información colaborativa.

# Estado del arte y marco teórico

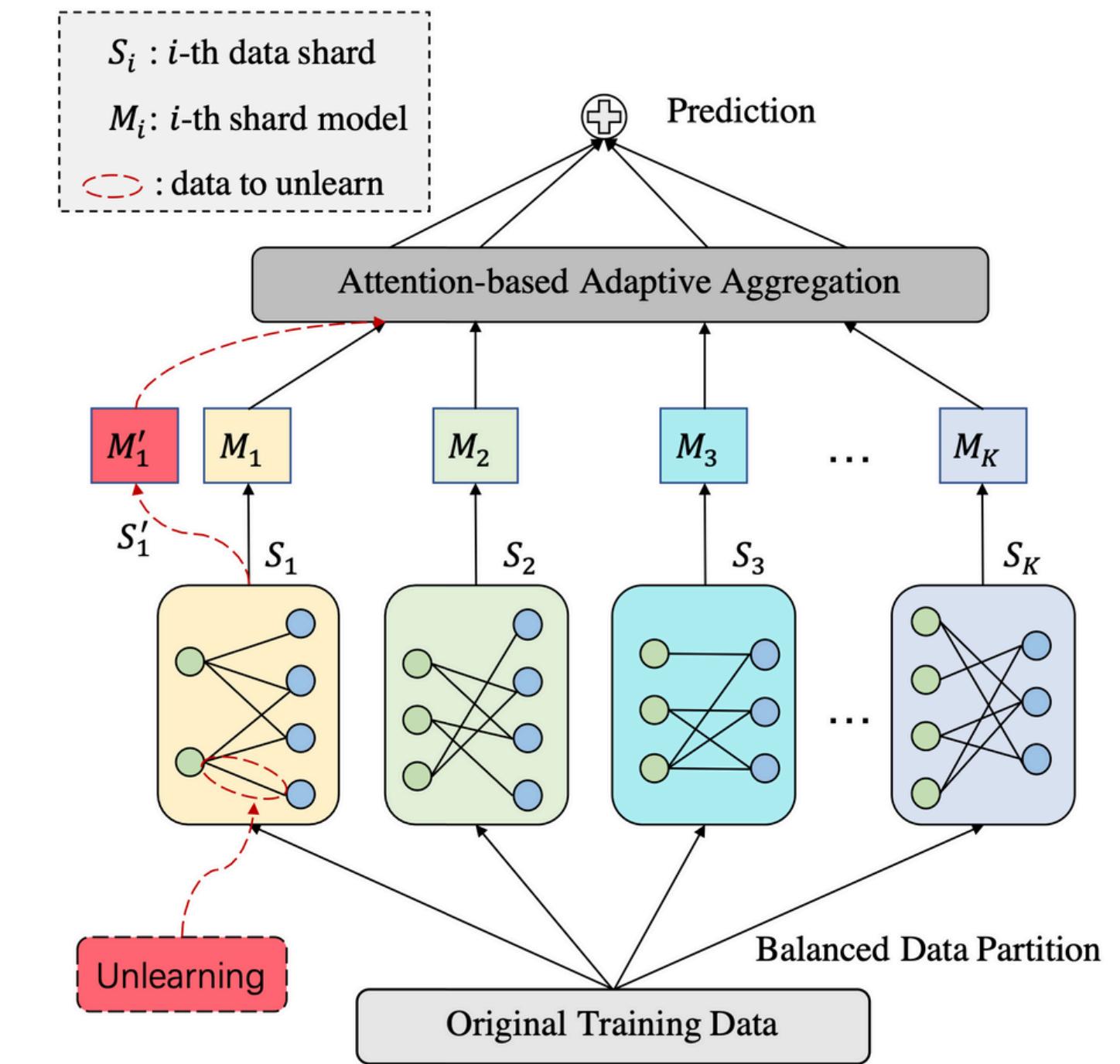
## GraphEraser

- Divide el set de entrenamiento (representado como un grafo) en particiones balanceadas.
- Entrena un submodelo GNN por cada fragmento y luego los agrupo usando una función de agregación *learning-based*.
- Ignora la información colaborativa global de usuarios e ítems.
- Utiliza una combinación estática de submodelos para todas las predicciones, impidiendo adaptaciones dinámicas para diferentes interacciones *user-item*.

# Solución

## Overview

- RecEraser consiste de 3 etapas:
  - a. Partitionar los datos de manera balanceada en fragmentos (*shards*).
  - b. Entrenar cada submodelo.
  - c. Agregar las predicciones utilizando *attention-based aggregation*.
- La partición de los datos está diseñada para preservar información colaborativa de las interacciones.
- Cuando se necesita “desaprender” una interacción, solo uno de los modelos debe ser entrenado nuevamente (el correspondiente al fragmento al cual la interacción pertenece).



# Solución

## Partición de los datos en shards

- Tres métodos:
  - a. User-based Balanced Partition (UBP).
  - b. Item-based Balanced Partition (IBP).
  - c. Interaction-based Balanced Partition (InBP).
- Basados en la similitud entre usuarios, ítems e interacciones.

# Solución

## Interaction-based Balanced Partition

- Ahora requerimos embeddings tanto para usuarios como ítems.
- Definimos aleatoriamente los anchors  $\mathbf{A} = \{a_1, \dots, a_K\}$
- Ahora cada anchor es un par ordenado  $a_i = (\bar{\mathbf{p}}_i, \bar{\mathbf{q}}_i)$
- Calculamos la distancia entre cada par (interacción, anchor):

$$\begin{aligned} \text{dist}(a_i, y_{uv}) &= \|\bar{\mathbf{p}}_i - \bar{\mathbf{p}}_u\|_2 \times \|\bar{\mathbf{q}}_i - \bar{\mathbf{q}}_v\|_2 \\ &= \sqrt{\sum_{j=1}^n (\bar{p}_{i,j} - \bar{p}_{u,j})^2} \times \sqrt{\sum_{j=1}^n (\bar{q}_{i,j} - \bar{q}_{v,j})^2} \end{aligned} \quad (3)$$

- Actualizamos los anchors con los nuevos shards, que ahora contienen interacciones  $y_{u,v}$ :

$$a_i = \left( \frac{\sum_{j \in S_i} \bar{\mathbf{p}}_j}{|S_i|}, \frac{\sum_{j \in S_i} \bar{\mathbf{q}}_j}{|S_i|} \right) \quad (4)$$

---

**Algorithm 2** Interaction-based Balanced Partition algorithm (InBP)

---

**Require:** Pre-trained user embeddings  $\bar{\mathbf{P}} = \{\bar{\mathbf{p}}_1, \bar{\mathbf{p}}_2, \dots, \bar{\mathbf{p}}_m\}$ ; item embeddings  $\bar{\mathbf{Q}} = \{\bar{\mathbf{q}}_1, \bar{\mathbf{q}}_2, \dots, \bar{\mathbf{q}}_n\}$ ; user-item interactions  $\mathbf{Y}$ ; number of shards  $K$ ; maximum number of each shard  $t$

**Ensure:** Shards  $\mathbf{S} = \{S_1, S_2, \dots, S_K\}$

- 1: Randomly select  $K$  anchors  $\mathbf{A} = \{a_1, a_2, \dots, a_K\}$  from  $\mathbf{Y}$
- 2: **while** Stopping criteria is not met **do**
- 3:   **for** each  $a_i$  in  $\mathbf{A}$  **do**
- 4:     **for** each  $y_{uv}$  in  $\mathbf{Y}$  **do**
- 5:       Calculate  $E(a_i, y_{uv}) = \text{dist}(a_i, y_{uv})$  (Eq.(3))
- 6:     **end for**
- 7:   **end for**
- 8:   Sort  $E$  in ascending order to get  $E_s$
- 9:   Empty  $\mathbf{S}$
- 10:   **for** each  $a_i$  and  $y_{uv}$  in  $E_s$  **do**
- 11:     **if**  $|S_i| < t$  and  $y_{uv}$  has not been assigned **then**
- 12:        $S_i \leftarrow S_i \cup y_{uv}$
- 13:     **end if**
- 14:   **end for**
- 15:   Updating  $\mathbf{A}$  by Eq.(4)
- 16: **end while**
- 17: **return**  $\mathbf{S}$

---

# Solución

## Entrenamiento

- Entrenamos cada submodelo sobre el shard correspondiente.

## Agregación de predicciones usando attention

- Queremos agregar las predicciones de cada submodelo
- Las representaciones internas de usuarios e ítems aprendidas por cada submodelos pueden estar en espacios distintos.
- Los transferimos a un mismo espacio.

# Solución

## Agregación de predicciones usando attention

- Para cada  $\mathbf{P}^i, \mathbf{Q}^i$  aprendido por el i-ésimo submodelo, definimos

$$\mathbf{P}_{tr}^i = \mathbf{W}^i \mathbf{P}^i + \mathbf{b}^i; \quad \mathbf{Q}_{tr}^i = \mathbf{W}^i \mathbf{Q}^i + \mathbf{b}^i \quad (5)$$

donde  $\mathbf{W}^i \in \mathbb{R}^{d \times d}$ ,  $\mathbf{b}^i \in \mathbb{R}^d$  representan la matriz y vector de transferencia.

- Ahora combinamos estos embeddings (ya transferidos) de cada submodelo para obtener los embeddings agregados:

$$\mathbf{p}_u = \sum_{i=1}^K \alpha_{i,u} \mathbf{p}_{tr,u}^i; \quad \mathbf{q}_v = \sum_{i=1}^K \beta_{i,v} \mathbf{q}_{tr,v}^i$$

# Solución

## Agregación de predicciones usando attention

- Los parámetros de atención  $\alpha_{i,u}, \beta_{i,v}$  vienen dados por

$$\alpha_{i,u}^* = \mathbf{h}_1^T \sigma(\mathbf{W}_1 \mathbf{p}_{tr,u}^i + \mathbf{b}_1); \quad \alpha_{i,u} = \frac{\exp(\alpha_{i,u}^*)}{\sum_{j=1}^K \exp(\alpha_{j,u}^*)}$$

$$\beta_{i,v}^* = \mathbf{h}_2^T \sigma(\mathbf{W}_2 \mathbf{q}_{tr,v}^i + \mathbf{b}_2); \quad \beta_{i,v} = \frac{\exp(\beta_{i,v}^*)}{\sum_{j=1}^K \exp(\beta_{j,v}^*)}$$

donde  $\mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^{k \times d}$ ,  $\mathbf{b}_1, \mathbf{h}_1, \mathbf{b}_2, \mathbf{h}_2 \in \mathbb{R}^k$  son los parámetros de atención de usuarios (1) e ítems (2).

- Finalmente, definimos  $\Theta$  como todos los parámetros de la red (incluyendo los de la transferencia), y el problema de optimización

$$\min_{\Theta} \mathcal{L}(\mathbf{P}, \mathbf{Q}, \mathbf{Y}) + \lambda \|\Theta\|_2^2 \quad (8)$$

# Evaluación

## Objetivos evaluación

- Buscar tener una precisión comparable a un entrenamiento completo (*retrain*).
- Superar a métodos previos de unlearning como SISA y GraphEraser.

# Evaluación

## Dataset

- Uso de dataset públicamente disponibles (Yelp2018 y MovieLens)
- Pre-procesados para convertirlo a *feedback* implícito
- 80% de interacciones para training set, resto para testing. 10% del training usado como validación para ajustar hiperparámetros.

**Table 2: Statistical details of the evaluation datasets.**

Dataset	#User	#Item	#Interaction	Density
<b>Yelp2018</b>	31,668	38,048	1,561,406	0.13%
<b>MovieLens-1m</b>	6,940	3,706	1,000,209	3.89%
<b>MovieLens-10m</b>	71,567	10,681	10,000,054	1.31%

# Evaluación (métodos comparados)

## Modelos de recomendación

- BPR (Bayesian Personalized Ranking)
- WMF (Weighted Matrix Factorization)
- LightGCN

## Métodos de unlearning

- *Retrain* (baseline)
- SISA (partición aleatoria + agregación por promedio)
- GraphEraser (particionado por grafos + pesos estáticos)
- RecEraser (partición basada en interacciones + attention-based aggregation)

# Evaluación (desempeño)

## Desempeño en Recomendación Top-N

- RecEraser supera consistentemente a SISA y GraphEraser.
- Comparable a *retrain* en datasets densos
- Resultados relativamente mejores en Yelp2018 (dataset más disperso)

**Table 3: Comparison of different unlearning methods for recommendation unlearning. The proposed InBP method is used for data partition of RecEraser in this Table. For efficient unlearning methods SISA, GraphEraser, and our RecEraser, best results are highlighted in bold and the superscript \*\* indicates  $p < 0.01$  for the paired t-test of RecEraser vs. SISA and GraphEraser.**

Yelp2018	BPR				WMF				LightGCN			
	Retrain	SISA	GraphEraser	RecEraser	Retrain	SISA	GraphEraser	RecEraser	Retrain	SISA	GraphEraser	RecEraser
Recall@10	0.0302	0.0151	0.0224	<b>0.0249**</b>	0.0368	0.0235	0.0250	<b>0.0335**</b>	0.0367	0.0221	0.0278	<b>0.0328**</b>
Recall@20	0.0521	0.0272	0.0397	<b>0.0439**</b>	0.0619	0.0401	0.0431	<b>0.0572**</b>	0.0637	0.0392	0.0479	<b>0.0568**</b>
Recall@50	0.1028	0.0566	0.0798	<b>0.0889**</b>	0.1189	0.0785	0.0825	<b>0.1105**</b>	0.1216	0.0768	0.0924	<b>0.1112**</b>
NDCG@10	0.0344	0.0181	0.0254	<b>0.0282**</b>	0.0422	0.0278	0.0295	<b>0.0385**</b>	0.0424	0.0259	0.0319	<b>0.0377**</b>
NDCG@20	0.0423	0.0224	0.0319	<b>0.0353**</b>	0.0512	0.0336	0.0360	<b>0.0471**</b>	0.0524	0.0321	0.0393	<b>0.0465**</b>
NDCG@50	0.0612	0.0334	0.0468	<b>0.0523**</b>	0.0723	0.0478	0.0508	<b>0.0669**</b>	0.0735	0.0461	0.0557	<b>0.0669**</b>

# Evaluación (desempeño)

**Table 3: Comparison of different unlearning methods for recommendation unlearning. The proposed InBP method is used for data partition of RecEraser in this Table. For efficient unlearning methods SISA, GraphEraser, and our RecEraser, best results are highlighted in bold and the superscript \*\* indicates  $p < 0.01$  for the paired t-test of RecEraser vs. SISA and GraphEraser.**

Movielens-1m	BPR				WMF				LightGCN			
	Retrain	SISA	GraphEraser	RecEraser	Retrain	SISA	GraphEraser	RecEraser	Retrain	SISA	GraphEraser	RecEraser
Recall@10	0.1510	0.1080	0.1190	<b>0.1393**</b>	0.1592	0.0759	0.1127	<b>0.1435**</b>	0.1544	0.1089	0.1174	<b>0.1425**</b>
Recall@20	0.2389	0.1737	0.1906	<b>0.2254**</b>	0.2507	0.1266	0.1823	<b>0.2254**</b>	0.2428	0.1743	0.1914	<b>0.2266**</b>
Recall@50	0.4005	0.2970	0.3271	<b>0.3822**</b>	0.4165	0.2309	0.3081	<b>0.3818**</b>	0.4076	0.3084	0.3322	<b>0.3839**</b>
NDCG@10	0.3412	0.2774	0.2844	<b>0.3208**</b>	0.3546	0.2058	0.2575	<b>0.3317**</b>	0.3509	0.2767	0.3832	<b>0.3276**</b>
NDCG@20	0.3368	0.2679	0.2827	<b>0.3186**</b>	0.3509	0.1998	0.2557	<b>0.3253**</b>	0.3447	0.2686	0.2791	<b>0.3229**</b>
NDCG@50	0.3669	0.2840	0.3105	<b>0.3483**</b>	0.3838	0.2174	0.2807	<b>0.3542**</b>	0.3754	0.2905	0.3043	<b>0.3520**</b>
Movielens-10m	BPR				WMF				LightGCN			
	Retrain	SISA	GraphEraser	RecEraser	Retrain	SISA	GraphEraser	RecEraser	Retrain	SISA	GraphEraser	RecEraser
Recall@10	0.1951	0.1620	0.1556	<b>0.1798**</b>	0.2094	0.1332	0.1135	<b>0.1994**</b>	0.1994	0.1221	0.1055	<b>0.1881**</b>
Recall@20	0.3016	0.2507	0.2470	<b>0.2808**</b>	0.3177	0.2142	0.1766	<b>0.3031**</b>	0.3059	0.2033	0.1676	<b>0.2899**</b>
Recall@50	0.4688	0.3906	0.3979	<b>0.4436**</b>	0.4838	0.3515	0.2838	<b>0.4655**</b>	0.4748	0.3314	0.2632	<b>0.4537**</b>
NDCG@10	0.3425	0.3002	0.2688	<b>0.3223**</b>	0.3704	0.2499	0.2209	<b>0.3575**</b>	0.3555	0.2289	0.2139	<b>0.3369**</b>
NDCG@20	0.3534	0.3053	0.2813	<b>0.3319**</b>	0.3784	0.2579	0.2243	<b>0.3642**</b>	0.3635	0.2476	0.2141	<b>0.3352**</b>
NDCG@50	0.3946	0.3366	0.3219	<b>0.3719**</b>	0.4169	0.2918	0.2484	<b>0.4015**</b>	0.4037	0.2808	0.2288	<b>0.3844**</b>

# Evaluación (eficiencia)

## Desempeño en Recomendación Top-N

- RecEraser reduce el tiempo de desentrenamiento en más de 10 veces en MovieLens-10m
- Podemos tolerar más *shards* para datasets más grandes para mejorar eficiencia de *unlearning*.

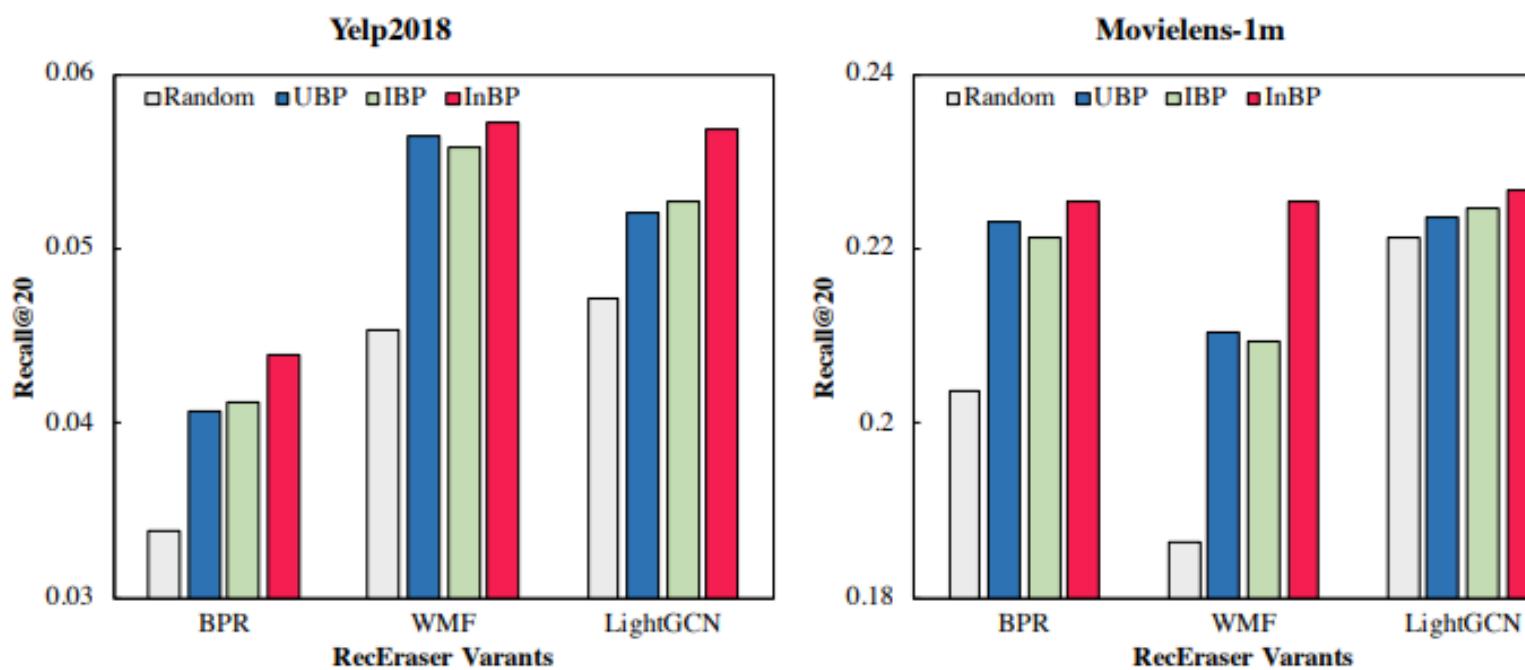
**Table 4: Comparison results of running time (minute [m]). For RecEraser, only the corresponding submodel and the aggregation part need to be retrained when receiving an unlearning request of data.**

		Yelp2018			MovieLens-1m			MovieLens-10m		
		BPR	WMF	LightGCN	BPR	WMF	LightGCN	BPR	WMF	LightGCN
<b>Retrain</b>		400m	75m	1090m	330m	13m	545m	3250m	292m	25330m
<b>RecEraser</b>	<b>Shard Training</b>	39m	2.5m	52m	12m	0.9m	17m	240m	14m	610m
	<b>Aggregation Training</b>	13m	1.3m	34m	10m	0.2m	14m	80m	4m	330m
	<b>Total</b>	52m	3.8m	86m	22m	1.1m	31m	320m	16m	940m

# Estudio de Ablación

## Particionado

- InBP (partición basada en interacciones) supera a UBP (usuarios), IBP (ítems) y partición aleatoria.

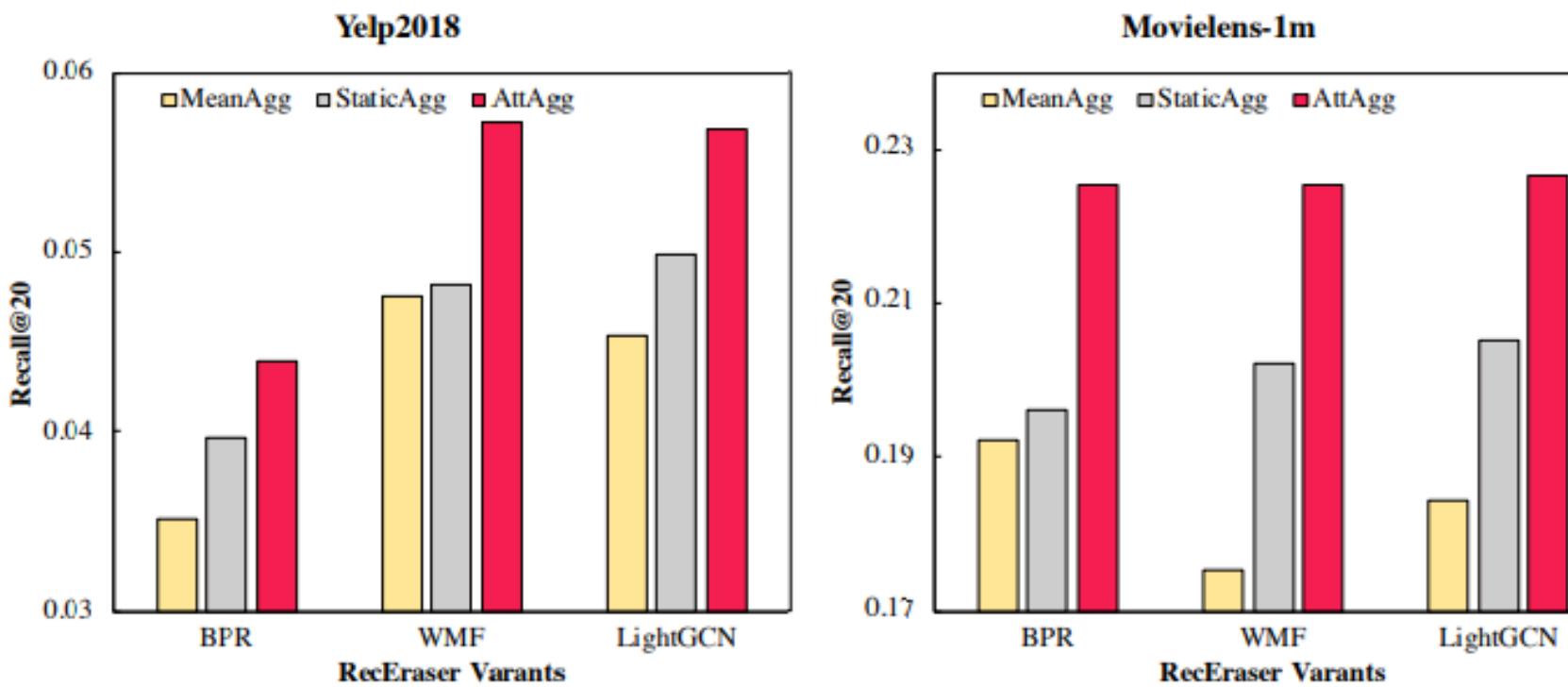


**Figure 2: Performance comparison of different data partition methods on Yelp2018 and MovieLens-1m datasets.**

# Estudio de Ablación

## Agregación

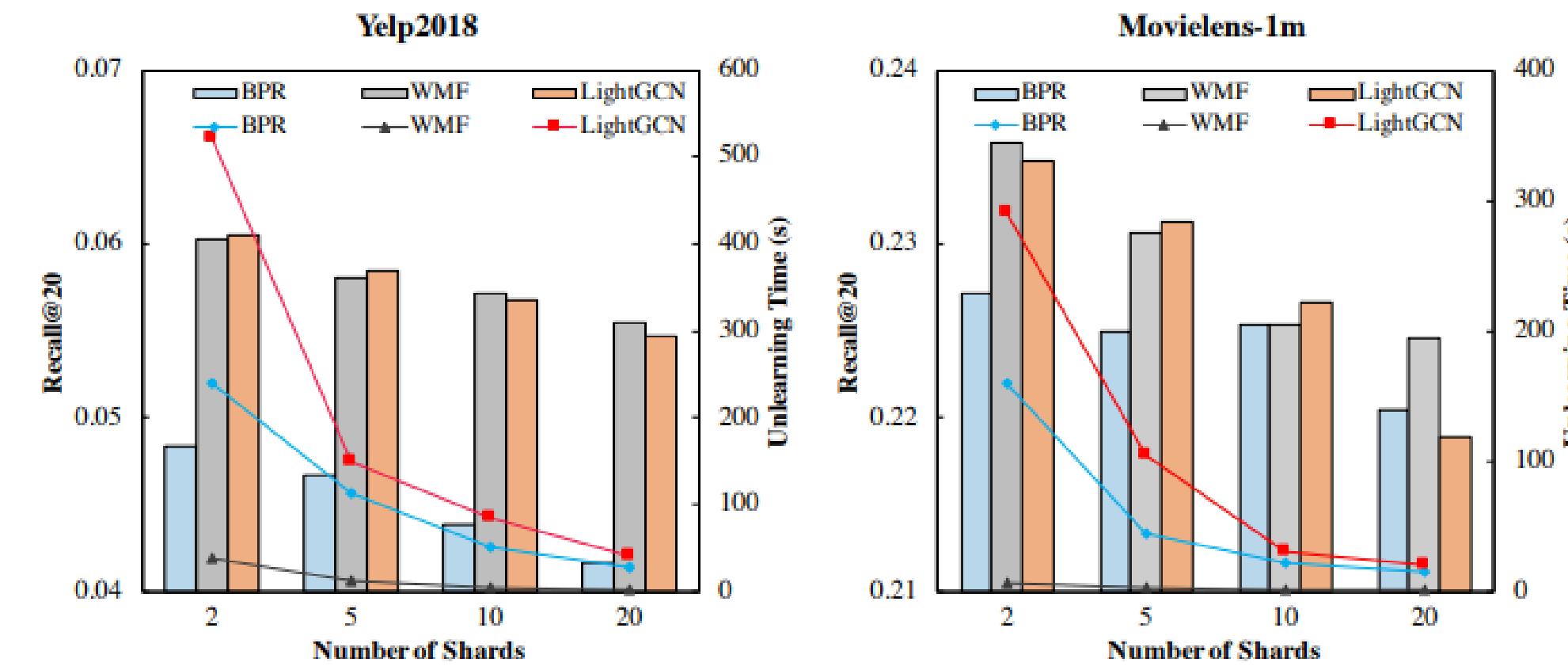
- La agregación adaptiva con *attention* (AttAgg) supera a MeanAgg y StaticAgg.



**Figure 3: Performance comparison of different data aggregation methods on Yelp2018 and MovieLens-1m datasets.**

# Impacto de Número de Shards

- Balance de desempeño y eficiencia.
- Más particiones → mayor eficiencia de desentrenamiento.
- Demasiadas particiones significa pérdida de información colaborativa y precisión.



# Resumen de Resultados

- RecEraser logra un equilibrio entre **eficiencia** y **precisión**.
- Supera a métodos anteriores en **desempeño**.
- **InBP + *attention based aggregation*** es la combinación más efectiva.
- **Escalable** y aplicable con **distintos** modelos base y datasets.

# Referencias

- [1] Chong Chen, Fei Sun, Min Zhang, and Bolin Ding. 2022. Recommendation Unlearning. In Proceedings of the ACM Web Conference 2022 (WWW '22). Association for Computing Machinery, New York, NY, USA, 2768–2777. <https://doi.org/10.1145/3485447.3511997>
- [2] Yang Zhang, Fuli Feng, Chenxu Wang, Xiangnan He, Meng Wang, Yan Li, and Yongdong Zhang. 2020. How to Retrain Recommender System? A Sequential Meta-Learning Method. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (Virtual Event, China). Association for Computing Machinery, New York, NY, USA, 1479–1488.
- [3] Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. 2021. Graph Unlearning. arXiv preprint arXiv:2103.14991 (2021)
- [4] Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hen- grui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. 2021. Machine unlearning. In 2021 IEEE Symposium on Security and Privacy (SP). IEEE, 141–159.