Clase 220 — Métricas: MAP@k, NDCG, recall@k

Parte: 6 — Sistemas de Recomendación · Fuente: Aggarwal cap. 7 + Järvelin & Kekäläinen, Cumulated Gain-Based Evaluation of IR Techniques (TOIS 2002 — NDCG paper). Duración estimada: 70 min.

Clase 220 — Métricas: MAP@k, NDCG, recall@k

> *Parte: 6 — Sistemas de Recomendación · Fuente: Aggarwal cap. 7 + Järvelin & Kekäläinen, Cumulated Gain-Based Evaluation of IR Techniques (TOIS 2002 — NDCG paper). Duración estimada: 70 min.*

Objetivo

Evaluar recomendadores con las métricas correctas: NO accuracy ni RMSE de rating (irrelevante para top-N). Sí recall@k (¿cuántos relevantes recuperaste en top-k?), precision@k (¿qué % del top-k es relevante?), MAP@k (precision promedio sensible al ranking), NDCG@k (gain descontado por posición). Decidir qué métrica reportar según objetivo de negocio.

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Calcular precision@k, recall@k, F1@k, hit rate desde cero.
- Calcular MAP@k (Mean Average Precision) y entender por qué sensible al ranking dentro del top-k.
- Calcular NDCG@k (Normalized Discounted Cumulative Gain) y entender el descuento logarítmico.
- Diseñar el split correcto: leave-one-out por user vs temporal split vs random.
- Decidir métrica según objetivo: recall (catálogo grande, descubrimiento) vs NDCG (orden importa) vs MAP (búsqueda).

Temas

#	Tema	Por qué importa
1	Por qué NO usar RMSE de rating	Top-N no usa el rating predicho — usa el r
2	Precision@k vs recall@k	Trade-off; recall importa más con catálogo
3	MAP@k: precision promediada por posición	Sensible a poner relevantes ARRIBA del top
4	NDCG@k con descuento log2(i+1)	El relevante en posición 1 vale más que en
5	Leave-one-out vs temporal split	Temporal evita data leakage en RS de tiemp
6	Coverage + diversity (beyond accuracy)	Métricas de "salud" del catálogo.

Definiciones y características

- Recall@k: |relevantes ∩ top-k| / |relevantes totales|. Pregunta: "¿qué % de lo que el user quiere mostré?". Ideal con catálogo grande.
- Precision@k: |relevantes ∩ top-k| / k. Pregunta: "¿qué % del top-k es relevante?". Ideal cuando el usuario consume pocos (top-5).
- F1@k: armónico de precision y recall.
- Hit rate@k: 1 si al menos 1 relevante en top-k, 0 si no. Más simple, popular en deep recsys research.
- AP@k (Average Precision): Σ_i (precision@i × rel(i)) / |relevantes| para i [1, k]. Penaliza relevantes en posiciones bajas.
- MAP@k (Mean AP): promedio de AP@k sobre todos los users.
- DCG@k (Discounted Cumulative Gain): Σ_i rel(i) / log2(i+1). Posiciones bajas valen menos.

- NDCG@k: DCG@k / iDCG@k donde iDCG es el máximo posible. Normaliza a [0, 1].
- Coverage (catalog coverage): |items recomendados al menos 1 vez en N users| / |items totales|.
- Diversity (intra-list): 1 - avg(sim(i, j)) entre pares en la lista. Alto = recomendaciones diversas.

Dataset / recursos

- Dataset: MovieLens 100K.
- Librerías: numpy puro (las métricas son fáciles), opcional recmetrics.

Ejercicios

1. Implementar precision@k, recall@k: dadas listas [1, 0, 0, 1, 0] (relevancia) y k=5, calcular. Verificar contra recmetrics u otra librería.
2. MAP@k: implementar AP@k. Mostrar diferencia con precision@k: AP penaliza tener los relevantes al final.
3. NDCG@k: implementar DCG y iDCG. Mostrar caso donde recall@k es igual pero NDCG distinto porque el orden cambia.
4. Leave-one-out por user: para cada user con ≥5 ratings, dejar el último (cronológico o random) para test, resto para train. Evaluar.
5. Coverage y diversity: para 1000 users, ¿qué % del catálogo fue recomendado? Diversidad media intra-list (cosine entre items recomendados).

Homework verificable

Notebook con:

1. Implementación de 6 métricas: recall@10, precision@10, F1@10, MAP@10, NDCG@10, hit rate@10. Validar contra librería.
2. Aplicar a 4 modelos (kNN, ALS, content, hybrid de Clases 216-219). Tabla comparativa.
3. Análisis: ¿qué métrica decidiría cuál modelo gana? Discutir.
4. Beyond accuracy: coverage, diversity, novelty (1 - popularity rank promedio). ¿Algún modelo gana en accuracy pero pierde en diversity?
5. Temporal split: si tu dataset tiene timestamps, hacer split por fecha (no por ratio). Comparar con random split. Discutir por qué temporal es más realista.

Criterio de aceptación: las 6 métricas coinciden con recmetrics ±1e-6; la elección de "mejor modelo" depende de la métrica; el estudiante justifica una arquitectura con métricas de negocio.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
MAP@k diferente a la librería	Diferencias de convención: ¿dividís por mi
NDCG@k > 1	Bug: estás computando DCG sin normalizar p
Recall=0.0 cuando esperaba algo	Probablemente recomendaste items que ya es
RMSE alto pero recall@10 alto	Esperado en implicit. RMSE no importa para
Random split en dataset temporal → métrica	Data leakage: train tiene futuro vs test.
Coverage 100% → "óptimo"	Coverage 100% probablemente significa reco

Diversity alta pero NDCG bajo	El modelo está recomendando cosas random.

Preguntas frecuentes

¿Qué métrica reportar?

Casi siempre: NDCG@k + recall@k + coverage. NDCG mide calidad del ranking, recall mide cobertura del relevante, coverage mide salud del catálogo. Si tu UI muestra top-5: k=5. Top-20: k=20.

¿MAP o NDCG?

NDCG es más expresivo (acepta relevancia graduada: rating 5 > rating 3). MAP solo binario. En implicit feedback (binary): equivalentes en práctica. Default: NDCG.

¿RMSE alguna vez sirve?

Sí, en sistemas de rating prediction donde el rating predicho se muestra al usuario (ej. Netflix mostraba "92% match"). Si solo mostrás top-N: NO usar RMSE.

¿Leave-one-out o random split?

LOO por user es estándar en RS papers — emula la situación "predecir el próximo item". Random split puede sesgar (data leakage si hay temporalidad). Para producción real: temporal split (train con datos hasta T-7d, test con T-7d a T-0d).

¿Cómo manejo cold-start en evaluación?

(1) Reportar métricas por segmento (cold vs warm users; new vs popular items). (2) En "all", incluir cold-start (es honesto, aunque baje promedio). (3) NO excluir cold-start users — el sistema real los tiene.

¿Online metrics vs offline?

Las offline (NDCG, recall) son proxies. Online (CTR, conversion, watch time, retention) son la verdad. La correlación offline-online suele ser positiva pero no perfecta. En producción: A/B test (Clase 204).

Referencias

- Aggarwal Recommender Systems: The Textbook (Springer, 2016), cap. 7 — Evaluating Recommender Systems.
- Järvelin, K. & Kekäläinen, J. Cumulated Gain-Based Evaluation of IR Techniques (ACM TOIS 2002) — paper original de NDCG.
- recmetrics — libreria con todas implementadas.
- Beyond accuracy: evaluating recommender systems by coverage and serendipity (Ge et al., RecSys 2010).

Siguiente clase

Clase 221 — Cold-start problem

Apéndice: notebook (primer bloque)

Implementación de las 6 métricas + comparativa entre 3 recomendadores sintéticos.

```
import numpy as np

def precision_at_k(rel, k):
    """rel: array binario de relevancia en orden de ranking. precision sobre top-k."""
    return rel[:k].sum() / k

def recall_at_k(rel, k, n_relevants):
    return rel[:k].sum() / max(n_relevants, 1)

def ap_at_k(rel, k):
    """Average Precision: precision en posiciones donde hay un relevante, promediada."""
    rel_k = rel[:k]
    if rel_k.sum() == 0: return 0.0
    precisions = [(rel_k[:i+1].sum() / (i+1)) * rel_k[i] for i in range(k)]
    return sum(precisions) / min(k, rel.sum())

def dcg_at_k(rel, k):
    rel_k = rel[:k]
    return float(np.sum(rel_k / np.log2(np.arange(2, k + 2))))

def ndcg_at_k(rel, k):
    ideal = np.sort(rel)[::-1]
    idcg = dcg_at_k(ideal, k)
    return dcg_at_k(rel, k) / idcg if idcg > 0 else 0.0
```

Archivos complementarios

- notebook.ipynb