Clase 219 — Recomendadores híbridos

Parte: 6 — Sistemas de Recomendación · Fuente: Burke, Hybrid Recommender Systems: Survey and Experiments (UMUAI 2002) + Kula, Metadata Embeddings for User and Item Cold-start Recommendations (LightFM, 2015). Duración estimada: 75 min.

Clase 219 — Recomendadores híbridos

> *Parte: 6 — Sistemas de Recomendación · Fuente: Burke, Hybrid Recommender Systems: Survey and Experiments (UMUAI 2002) + Kula, Metadata Embeddings for User and Item Cold-start Recommendations (LightFM, 2015). Duración estimada: 75 min.*

Objetivo

Combinar CF (filtrado colaborativo, Clase 216-217) + content-based (Clase 218) para conseguir lo mejor de ambos: serendipia + cold-start + explicabilidad. Aplicar los 7 patrones de hybrid de Burke (2002) y usar LightFM (que aprende un modelo único con CF + features).

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Diferenciar los 7 patrones de hybrid: weighted, switching, mixed, feature combination, cascade, feature augmentation, meta-level.
- Implementar un hybrid weighted: score = α × score_cf + (1-α) × score_content.
- Implementar un hybrid switching: usar content para users con <N interactions, CF para el resto.
- Usar LightFM como hybrid built-in: el modelo aprende embeddings que combinan CF + content features.
- Tunear α con validation y entender por qué α óptimo varía por user/item segment.

Temas

#	Tema	Por qué importa
1	7 patrones de Burke (2002)	Vocabulario para discutir arquitecturas.
2	Weighted hybrid: α × CF + (1-α) × CB	El más simple y muy efectivo.
3	Switching: cold-start triage	"Si user tiene <5 ratings, usá content".
4	LightFM: hybrid aprendido	Embeddings que combinan ambas señales.
5	Tuning α por segmento	Cold-start: peso a content; long-tail: pes
6	Two-tower modelo (concept)	El sucesor moderno de LightFM.

Definiciones y características

- Weighted: combinar scores por suma ponderada. score = α × CF + (1-α) × CB. α se tunea con validación.
- Switching: elegir uno u otro según contexto. Ej: cold-start (user nuevo) → content; usuario maduro → CF.
- Mixed: presentar resultados de ambos en la misma página (ej: carousel "porque te gustó X" + carousel "popular ahora").
- Feature combination: agregar señales CF (ratings) como features a un modelo content-based, o viceversa.
- Cascade: primero filtra con un modelo (e.g. CF top-100), después re-rankea con otro (e.g.

content-based fine-grained).
- Feature augmentation: usar el output de un modelo (e.g. cluster CF del user) como feature de otro.
- Meta-level: el modelo aprende cuándo usar cada uno (gating network, mixture of experts).
- LightFM: librería que aprende un único embedding por user/item combinando CF + content features. Funciona en pure CF, pure content, o hybrid.
- Two-tower architecture: red neuronal con dos encoders (user tower + item tower), ambos producen embeddings que se combinan via dot product. SOTA moderno para retrieval.

Dataset / recursos

- Dataset: MovieLens 100K con u.item que tiene géneros (19 binarios).
- Librerías: lightfm>=1.17, scipy.sparse, pandas.

Ejercicios

1. Weighted hybrid manual: tomar scores de Clase 217 (ALS) y Clase 218 (content-based). Combinar score = α × cf + (1-α) × cb para α {0, 0.25, 0.5, 0.75, 1}. Reportar NDCG@10 para cada α.
2. Switching por user: si interactions(u) < 5: usar content; si no: usar CF. Comparar contra weighted para users en distintos segmentos (new/mature).
3. LightFM hybrid: entrenar LightFM(loss='warp') con item_features (géneros) y user_features (demographics). Comparar NDCG vs pure CF (sin features).
4. Cold-start eval: held-out incluye items nuevos (no en train) y users nuevos (sin ratings). Comparar pure CF (~0%), pure content (~OK), LightFM hybrid (~mejor).
5. Cascade: top-100 con CF, re-rankear top-10 con content (boost a items con descripción similar al historial del user).

Homework verificable

Notebook con:

1. 4 modelos sobre MovieLens 100K:
- Pure CF (implicit ALS).
- Pure content-based (sentence-transformers).
- Weighted hybrid (α tunado).
- LightFM hybrid (con item features).
1. Tabla comparativa: NDCG@10, recall@10, coverage, diversity, tiempo train, tiempo predict.
2. Análisis por segmento: cold-start users (≤3 ratings), warm users (≥20). ¿Cuál gana en cada?
3. Curva α vs NDCG@10 para weighted hybrid.
4. Documentación: arquitectura recomendada para 3 casos: e-commerce, streaming, news.

Criterio de aceptación: weighted hybrid o LightFM gana sobre pure CF en al menos uno de los segmentos; el α óptimo está justificado con números; las recomendaciones por arquitectura son sensatas.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
Weighted hybrid no mejora sobre CF	α no tuneado, o scores en escalas distinta
LightFM peor que pure CF	Item features con poca info (e.g. 19 géner

Switching tiene "jump" entre user con 4 vs	Discontinuidad. Fix: weighted con α(n) = s
Cold-start eval reporta ~0 para CF	Esperado. Fix: ese es el punto del ejercic
LightFM WARP loss muy lento	WARP es O(n_items × negatives) por ejemplo
Mismas recomendaciones para todos los user	Probablemente α=0 o α=1 colapsa todo. Fix:

Preguntas frecuentes

¿Qué hybrid pattern elegir?

Empezá con weighted o switching — son los más simples e interpretables. LightFM si tenés features ricos. Two-tower deep solo si tenés equipo ML + escala que lo justifique.

¿LightFM sigue siendo relevante en 2026?

Sí para datasets pequeños/medianos con features. Para grandes escalas: two-tower (TF Recommenders, recsys-pytorch). Pero LightFM es el sweet spot "complejidad-calidad" para casos chicos-medianos.

¿Cómo tuneo α?

(1) Grid search con validation NDCG@10. (2) Bayesian opt si tenés más hiperparámetros. (3) Por segmento (cold-start users → α bajo, mature → α alto). (4) Online via contextual bandit (avanzado).

¿Mixed (carousels) cuenta como hybrid?

Sí — es el patrón más usado en producción real. Spotify/Netflix muestran múltiples carousels con distintas estrategias ("porque viste X", "popular en tu país", "nuevos lanzamientos"). Cada carousel es un recomendador distinto; el "hybrid" es la página final.

¿Cuándo NO usar hybrid?

Si: (1) tu dataset es 100% interactions sin features útiles → pure CF. (2) Tus items no tienen historial pero tenés metadata rica → pure content. (3) Sos un startup con 1 ML eng → empezá simple, sumá hybrid cuando duela.

¿Two-tower vs LightFM?

Two-tower (TF Recommenders, PyTorch) escala mejor + permite features más complejos (text embeddings, image embeddings). LightFM es matriz factorization + lineal sobre features. Para 2026 SOTA: two-tower. Para empezar: LightFM.

Referencias

- Burke, R. Hybrid Recommender Systems: Survey and Experiments (UMUAI 2002) — los 7 patrones canónicos.
- Kula, M. Metadata Embeddings for User and Item Cold-start Recommendations (DLRS 2015) — LightFM paper.
- LightFM docs.
- TensorFlow Recommenders — two-tower moderno.
- Aggarwal cap. 6 — Ensemble-Based and Hybrid Recommender Systems.

Siguiente clase

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

Apéndice: notebook (primer bloque)

3 estrategias hybrid sobre un dataset sintético + cold-start eval. Requiere: pip install lightfm scipy scikit-learn.

```
import numpy as np
from scipy.sparse import csr_matrix
from sklearn.metrics.pairwise import cosine_similarity

rng = np.random.default_rng(42)
n_users, n_items, n_features = 500, 200, 10

# Item features (géneros multi-hot) y user preferences latentes
item_features = rng.binomial(1, 0.25, (n_items, n_features)).astype(float)
user_prefs = rng.normal(0, 1, (n_users, n_features))

# Generar ratings sintéticos: user gusta items que matchean sus preferencias
scores_true = user_prefs @ item_features.T
noise = rng.normal(0, 0.5, scores_true.shape)
interact_prob = 1 / (1 + np.exp(-(scores_true + noise)))
R = (rng.random(scores_true.shape) < 0.1 * interact_prob).astype(float)
R_sparse = csr_matrix(R)
print(f'interactions: {R_sparse.nnz:,}')
```

Archivos complementarios

- notebook.ipynb