Designing a Personalized Content Recommendation System

Problem, Data, Cleaning, Models, Inference, and Deployment

Chandan J

July 23, 2025

Agenda

What Are We Building?

Goal

Develop an algorithm that **personalizes** content (media, articles, products, etc.) for each user to maximize engagement, satisfaction, or business KPIs.

Key Questions

- What **content types**? (videos, news, songs, courses, products)
- Which **signals**? (clicks, watch time, ratings, purchases, dwell time)
- What metric optimizes success? (CTR, NDCG@10, retention, revenue)
- Real-time vs. batch; on-device vs. cloud; latency constraints?

Example Use Cases

Domain	Personalization Task
News App	Rank daily articles per user based on reading history and topics of interest.
OTT/Streaming	Recommend next movies/episodes; continue watching; cold-start for new users.
E-Learning	Suggest courses/modules matching skills and completed lessons.
E-Commerce	"Customers like you also bought"; re-rank search results for conversion.
Social Media Feed	Order posts/stories balancing relevance, freshness, and diversity.

Data Sources

User Signals

- Explicit: ratings, likes/dislikes, thumbs up
- Implicit: clicks, watch time, scroll depth, add-to-cart.
- Context: time, device, location, session info.

Item Metadata

- Text (title, description, tags, categories).
- Audio/Video features (embeddings).
- Creator info, publish time, popularity.

Public Benchmark Datasets

Dataset	Domain	Users/Items	Signals
MovieLens (100K/1M/20M)	Movies	943/6k ...	Ratings (1–5)
Amazon Reviews (2018)	E-commerce	Millions	Ratings, reviews, timestamps
GoodBooks-10k	Books	53k/10k	Ratings
Netflix Prize	Movies	480k/17k	Ratings
Last.fm 1K	Music	1k/65k	Play counts
Yelp Open Dataset	Local biz	1.6M/200k	Ratings, reviews
RecSys Challenge sets	Varies yearly	Varies	Clicks, orders, add-to-cart

Building the Interaction Log

1. Define a unified schema: `user_id`, `item_id`, `timestamp`, `event_type`, `value`.
2. Convert raw events to implicit scores (e.g., view $\rightarrow$ 1, complete $\rightarrow$ 3).
3. Handle missing/erroneous IDs, timestamps, duplicates.
4. Filter bots/outliers (excessive clicks in short time).

Cleaning & Splitting

Temporal split: train on past, validate/test on future to avoid leakage.

Minimum interaction thresholds (e.g., users with $\geq$5 actions).

Negative sampling for implicit data (items user didn't interact with).

Normalize continuous features (popularity, recency).

Text cleanup: lowercase, stopwords, n-grams, embeddings.

Baseline Methods

Non-personalized: top popular, trending, newest.

Content-based: TF-IDF / embedding similarity of item metadata to user profile.

Neighborhood CF: User-based or item-based kNN using cosine/pearson similarity.

Matrix Factorization Family

ALS / SGD MF: Learn latent user/item vectors minimizing MSE.

BPR-MF: Pairwise ranking loss for implicit feedback.

SVD++: Incorporates implicit signals (clicks) + explicit ratings.

Neural Recommenders

Two-Tower / NCF

- Separate user and item encoders.
- Dot product / MLP for matching.
- Good for ANN retrieval (FAISS, ScaNN).

Sequence Models

- GRU4Rec, SASRec, Transformer4Rec.
- Predict next-item from session history.
- Handle context and order of interactions.

Advanced/Hybrid Approaches

Graph-based: GCNs/LightGCN on user–item bipartite graphs.

Context-aware: Wide & Deep, DeepFM, xDeepFM.

Knowledge Graph Recsys: leverage entity relations.

Hybrid: Combine collaborative + content signals.

Re-ranking: Diversity, novelty, fairness constraints.

Typical Training Loop (Ranking Model)

```
for epoch in range(E):
    model.train()
    for users, pos_items, neg_items in loader:
        pos_scores = model(users, pos_items)
        neg_scores = model(users, neg_items)
        loss = bpr_loss(pos_scores, neg_scores)  # or CE, MSE, etc.
        loss.backward()
        optimizer.step(); optimizer.zero_grad()

    val_ndcg = evaluate(model, val_data, k=10)
    early_stopping(val_ndcg)
    save_checkpoint(...)
```

Serving / Inference Pipeline

Two-Stage Architecture

1. **Candidate Generation** (fast, approximate)
   - ANN search on item embeddings
   - Retrieve top 200–1000 candidates
2. **Ranking** (slower, accurate)
   - Rich features + deep model
   - Output final top-k list

Online Considerations

- Latency budgets (e.g., < 100 ms)
- Caching popular results
- Real-time feature updates (streaming)

Offline Metrics

Ranking: HitRate@k, NDCG@k, MRR, MAP.

Classification/AUC: ROC-AUC, PR-AUC for click prediction.

Rating Prediction: RMSE, MAE.

Beyond-accuracy: Diversity, novelty, serendipity, coverage.

Online Testing

A/B testing on production traffic: CTR, retention, revenue uplift.

Interleaving tests for fine-grained pairwise comparison.

Guardrail metrics: latency, complaint rate, content policy violations.

Production Stack

Feature store (Feast), model registry (MLflow), experiment tracker (W&B).

Batch (Spark) + stream (Kafka/Flink) pipelines.

Model versioning, canary releases.

Monitoring & Ethics

Drift detection: user taste shifts, new items.

Bias/fairness: exposure imbalance, filter bubbles.

Privacy: GDPR/CCPA; minimize PII, anonymize logs.

Feedback loops: integrate user feedback/corrections.

Takeaways

Start with clear objectives and measurable metrics.

Build a robust data pipeline: clean, temporal splits, negative samples.

Compare baselines (popularity, CF) before complex neural models.

Two-stage serving (retrieve & rank) is practical at scale.

Continuous monitoring, ethical checks, and iteration are essential.

Questions?