ML-Powered Search & Recommendation 101: From Core Concepts to Scalable Systems

High-level overview; individual topics require in-depth exploration

Sergey Polyashov

Agenda

- Why Search & Recommendation Matter
- Success Metrics: Short-Term vs Long-Term Targets
- High-Level Architecture Overview
- Multi-Stage Retrieval & Ranking Funnel
- Candidate Generation: Efficient Filtering
- Ranking: Approaches
- Ranking: Typical High-Level Architecture
- Ranking: GBDT vs Neural Networks
- Design Principles for Large-Scale NN
- Scaling Gaps: LLMs and Recommender Systems
- Trends in Neural Networks for Recommender Systems

Why Search & Recommendation Matter

- Address **Information Overload** and Enhancing **User Experience**
  - Surface relevant items from gigantic catalogs with **Millions to Billions** of objects
- Boost **Engagement** and **Business Metrics**
  - Drive **clicks**, **conversions**, and **purchases**
  - Increase **time spent**, **content consumed**, and **return visits**
  - Optimize for **long-term value (LTV)** with methods like Reinforcement Learning (RL)
  - Support product goals, like **Discovery Scenario** or **Search**
- Real-World Examples:
  - YouTube: 70% of views via recommendations (2018)
  - Amazon: 35% of purchases driven by recommendations (2013)
  - Netflix: 80% of watched content via algorithmic recommendations (2017)

Success Metrics: Short-Term vs Long-Term Targets

Short-Term Targets:

Definition:

- Immediate user actions

Specific:

- Easy to track.
- Fast feedback loop
- Can lead to overfitting to short-term interest

Examples:

- Clicks
- Dwell time
- Add to cart / Purchase
- Engagement in next session

Long-Term Targets:

Definition:

- Satisfaction and retention over time

Specific:

- Better reflects user value
- Encourages exploration and diversity
- Harder to measure and optimize directly

Examples:

- User retention / Return visits
- Subscription continuation
- Diversity of consumed content
- Reduced churn

High-Level Architecture Overview

Search Query → Query Understanding

User → User Profile

Items → Real-Time Data Processor

Actions → Logs

Serving Layer: Index and Ranker

Candidate Generation
KNN:
- User-Item CF
- Matrix Factorization: ALS
- Random Walks
- NN Embeddings

Inverted Index:
- Key/Inv
- Term-Sharded

Heuristics:
- Popularity
- Freshness
- Subscriptions
- Categories
- User History

Feature Generation
Counters:
- User, Query, Item
- User-Item, Query-Item
- Histograms
- Content-based
- Contextual
- Behavioral
- Time Exp Decay

Models based score:
- Linear models
- NN: DSSM, BERT

Models & Business Rules
Multi-Stage Models:
- L1 (top10000). Small. CPU
- L2 (top1000). Mid. CPU
- L3 (top50) Heavy CPU-GPU

Multi-Target:
- Model: p(relevance)
- Model: p(click | shown)
- Model: p(conversion)

Re-Ranking rules:
- Diversity: MMR, DPP
- Exploration: Thompson Sampling

Cold Start

Training Pipeline: Labeling, Training

Labeling
- Offline: Judges, LLM
- Online Target: CTR, DwellTime, Conv, RL

Training
- Full Retraining
- Active Learning
- Online Learning

Metrics and Experimentations

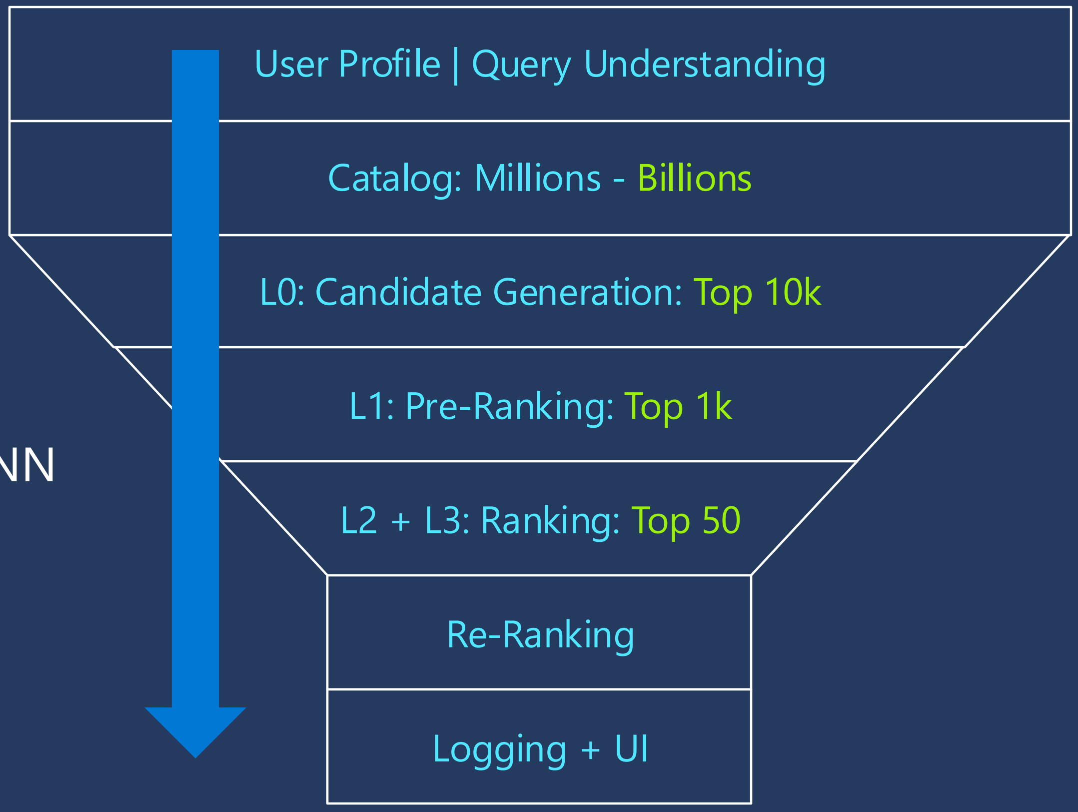
Offline (Judges, Crowd, LLM)
- P/R@K, AUC ROC
- NDCG with Secondary Aspects
- Defect Rate, Leakage
- Coverage

Online
- CTR, Conv Rate, Time Spent, Bounce Rate
- DAU, Retention, Revenue
- Latency, QPS

Multi-Stage Retrieval & Ranking Funnel

Serving Pipeline Stages:

- User Profile | Query Understanding
- L0: Candidate Generation and Filtration
    - High Recall, Diversity, Freshness, Popularity.
- L1: Pre-Ranking:
    - 10k -> 1k best items. (Light Model)
    - High Recall, Diversity. Light Model: GBDT, Fast DNN
- L2 + L3: Ranking:
    - 1000 -> 50 best items (Heavy Models)
    - High NDCG, Diversity. Heavy Models.
- Re-Ranking:
    - Business Rules.



Candidate Generation: Efficient Filtering

Goal: Select a smaller subset (thousands) of relevant items from the large corpus

Optimization Goal: High Recall, High Diversity, Computationally efficient.

Approaches:

- **ANN:** Find closest vectors by embeddings
  - Libraries: Faiss (Meta), ScaNN (Google). Algos: HNSW (fast), IVF+PQ (less memory)
  - Sources of embeddings:
    - **Collaborative Filtering:** User-Item ALS (fast). Issues: Cold start problem
    - **Content-based:** Two-Tower Models (DSSM, BERT-based)
- **Random Walk:** Discover related items. Useful for: Cold start problems and diversity. A bit outdated.
- **Inverted Index:** Primarily in search-based candidate selection
  - Maps each term (word/key) to a list of documents where it appears
  - Each entry may store position, frequency, and/or relevance score
  - Scales to Trillions of documents using:
    - Term sharding for distributed lookup
    - Sorting by relevance (BM25 or better). Top-K trimming to reduce result size efficiently
- **Heuristics:** Simple, rule-based selection
  - Popularity, Recency, Subscriptions, User History, Categories

Ranking: Approaches

Goal: Order the candidates based on relevance and auxiliary objectives.

Optimization Goal:

- Short-Term (Widely Used in Industry): NDCG, MRR, P@K based on Offline Judgements and Historical User Feedback
- Long-Term (Under Research/Adoption): Reinforcement Learning, Policy Learning, Sequence Models.

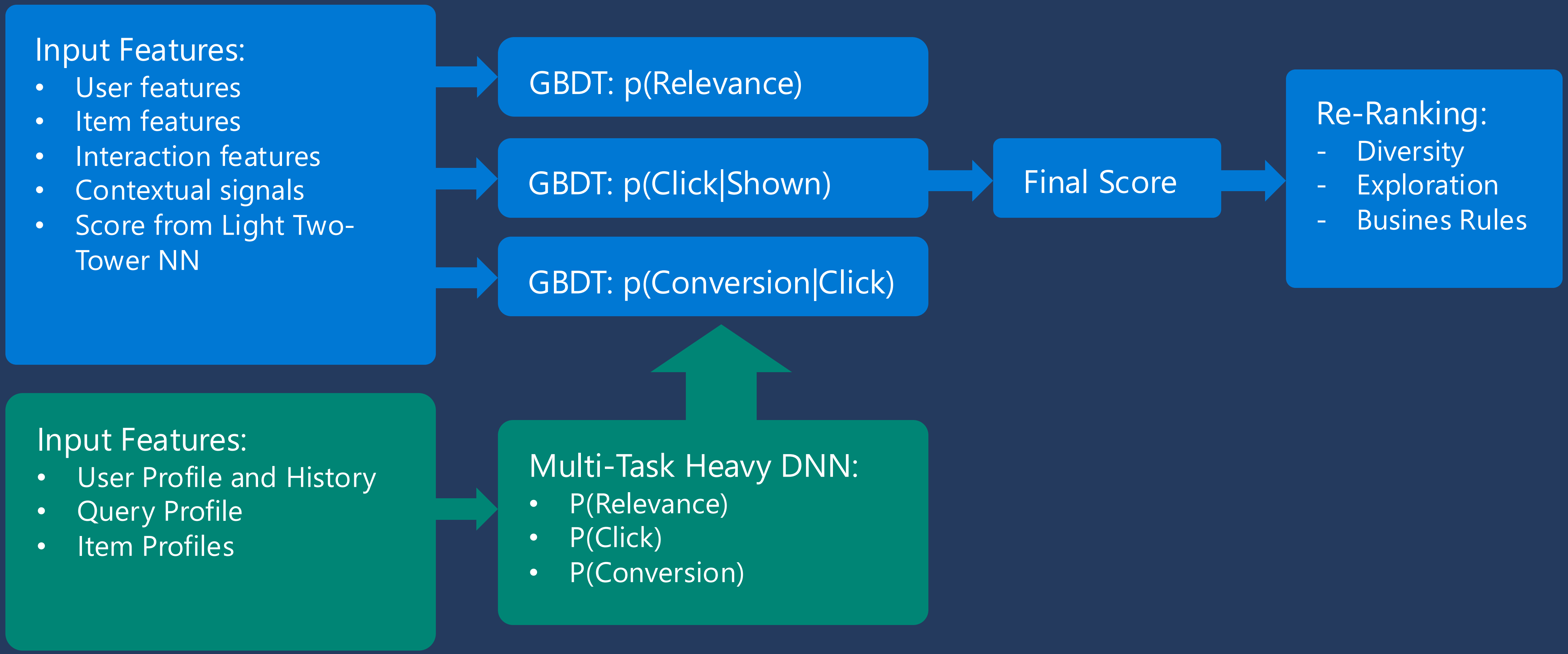
Approaches:

- **GBDT** (e.g. XGBoost, LightGBM, CatBoost): interpretable, fast. Widely used in production as final model.
- **DNN**: DSSM, BERT, Transformer-based models. Often used for ANN selection and as features for GBDT

Challenges:

- **Bias:** Label bias & position bias in logs (implicit feedback)
- **Cold start:** sparse user/item history
- Trade-offs: **Diversity** vs Relevance
- Trade-offs: **Reinforcement Learning (RL):** Modeling the long-term impact of recommendations
- **Exploration** vs Exploitation: Balancing relevant items (Exploitation) with Discovery (Exploration)
- **Interpretability:** Understanding why a recommendation was made. Challenging for complex NN compared to GBDT
- **Privacy (Federated Learning):** Exploring ways to train models without centralizing sensitive user data.
- **Latency**: heavy models need optimization or approximation (e.g., distillation, caching)

Ranking: Typical High-Level Architecture



Ranking: GBDT vs Neural Networks

Gradient Boosted Decision Trees (GBDT)

Widely used in production. Stable.

Pros:
- Works well on structured data
- High interpretability (feature importance)
- Fast training, easy A/B testing and retraining
- Strong baseline, often winning on small data
- Pair-wise, list-wise loss for NDCG
- Low latency

Cons:
- Limited in modeling complex interactions
- Hard to handle sequences, multimodality
- Scales poorly on large datasets

Deep Neural Networks:

Active research and real-world adoption

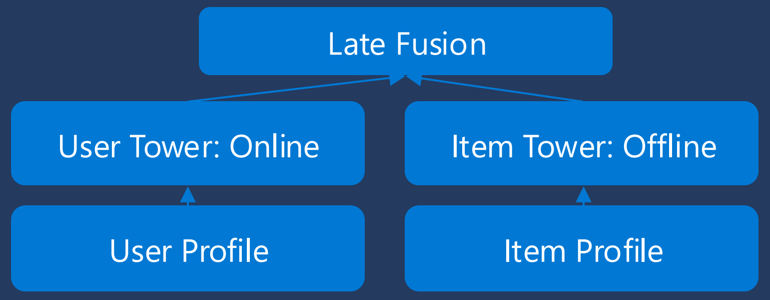
Pros:
- Explicit and implicit feature interaction modeling
- Support for high cardinality features (embeddings)
- Reuse of Embeddings across models
- End-to-end training on sequences
- Multitasking and Transfer learning
- Better scalability on data and parameters

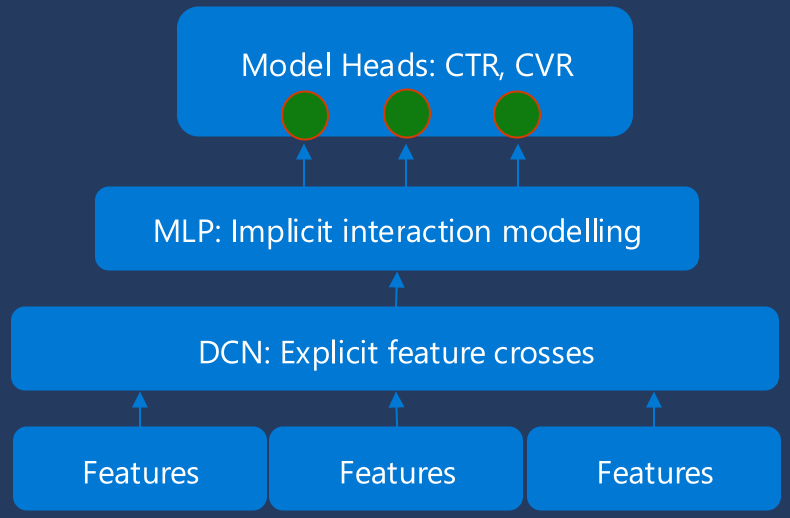
Cons:
- Computationally expensive (latency, inference, training)
- Hard to debug and interpret
- Hallucinations, biases and fairness issues.
- Difficult to fine-tune incrementally
- Sensitive to input noise or prompt changes

Hybrid architectures are common in large-scale pipelines.

Design Principles for Large-Scale NN

- Late Fusion & Bi-Encoder:
  - Separate User|Query Tower (online) and Item Tower (offline, precomputed)
  - Can preserve 80%+ of profit with 100x speedup
- Contrastive Learning:
  - Loss: InfoNCE or NT-Xent. Trains on positive vs hard negative pairs
  - Enables dot-product compatible embeddings
  - Reported uplift: up to +100% profit in retrieval
- Embedding Compression: Hashing, Quantization, Distillation
- Remove Bias:
  - Feedback Loop, Popularity bias, Position bias
  - Strategy: add context tower during training, drop at inference
- Hard Negatives Mining: avoid trivial negatives
- Multi-Signal Learning:
  - Multi-Modal: text, image, tabular
  - Multi-Domain: search queries, watch history, cart events
- Sequential Modeling:
  - Transformer Encoder: feed recent events first





Scaling Gaps: LLMs and Recommender Systems

NLP, Computer Vision (LLMs)

Scales well

- Long input sequences (text, pixels)
- Dense labels & strong supervision
- Pretraining tasks like next-token prediction
- Deep transformer architectures
- Latency-tolerant (seconds ok)
- Scale improves quality (scaling laws)

Recommender Systems

Doesn't Scale Easily

- Massive embedding tables (billions of user/item IDs)
- Tiny MLPs or towers (milliseconds constraints)
- Short behavioural sequences (3–30 user actions)
- Sparse, implicit feedback (clicks, skips)
- No universal self-supervised task
- Hard latency constraint (<50ms)
- No clear scaling law (limited by bias, noise)

Recommender models hit unique scaling limits:
latency, implicit feedback, massive embeddings, **domain-specific bias**
not easily solved by just making models deeper or wider.

Trends in Neural Networks for Recommender Systems

- **Neural Ranking:** Shift from GBDT to DNN: YouTubeDNN, Wide&Deep, DIN, DLRM
- **Multi-Stage Pipelines:** Bi-encoders for fast recall + DNN for final ranking.
- **LLMs:** interpret embeddings and generate answers based on vectors alone
  - Demystifying Embedding Spaces using LLMs (Google, 2024)
- **Model Architecture Trends:**
  - HyperFormer, HiFormer transformer innovations (DeepMind, 2023)
- **Scaling Recommender Systems:** Scaling laws have been shown to apply to embeddings, sequences
  - Understanding Scaling Laws for Recommendation Models (Meta, 2020)
  - Actions Speak Louder than Words: Trillion-Parameter Transducers (Meta, 2024)
  - Wukong: Scaling Law for Large-Scale Recommendation (Meta, 2024)
- **Sequence Modeling:** Moving beyond Next-Item Prediction toward richer modelling: multimodal, lifelong, time-aware
  - PinnerFormer (Pinterest, 2022)
  - Incorporating Time in Sequential Models (Amazon, 2023)
- **Graph NNs**: Use user–item graphs to improve recommendations, especially in the long tail.
  - **Inductive**: aggregates neighbor features, leverages content, generalizes to unseen nodes.. PinSage (Pinterest)
  - **Transductive**: learns embeddings from the full graph, suited for fixed node sets. TwHIN (TikTok)
- **Reinforcement Learning**: RL is used for retention, LTV, long-term goals. Exploration mitigates feedback loops & bias
  - UNEX-RL (Kuaishou, 2024)
  - Long-Term Value of Exploration (DeepMind, 2024)
  - Navigating the Feedback Loop (Netflix, 2023)

Thank you