

Balancing Trial and Reorder: A Hybrid Sequential Transformer-GBDT Ranker for On-Demand Delivery

Marcel Kurovski
marcel.kurovski@wolt.com
Wolt (DoorDash, Inc.)
Munich, Germany

Attila Nagy
attila.nagy@wolt.com
Wolt (DoorDash, Inc.)
Munich, Germany

Steffen Klempau
steffen.klempau@wolt.com
Wolt (DoorDash, Inc.)
Berlin, Germany

Aleksandr Fedintsev
aleksandr.fedintsev@wolt.com
Wolt (DoorDash, Inc.)
Helsinki, Finland

Abstract

On a delivery platform, personalized store ranking greatly influences what users find and order. Unlike digital-only domains, candidate stores are local and bound by real-time availability and delivery operations. One central modeling tension is between surfacing new stores for trial and preserving ranking quality for sessions with reorder intent.

We present *Universal Venue Ranker* (UVR), a production system deployed at Wolt that pairs a bidirectional transformer encoder for sequential user modeling with a GBDT ranker integrating contextual, user, and store features. Trained across all stores and domains of a country while enforcing local delivery constraints at inference, UVR replaces four previously separate ranking models—three for restaurants, one for retail—with a single unified system. Label smoothing and trial-biased sample weighting steer the model toward new stores, lifting offline trial MRR by +12% to +30% over production while regressing reorder MRR in five of six countries. These regressions leave Global CVR—our core online metric, which blends trial and reorder sessions—statistically unchanged.

We validate UVR in three consecutive A/B tests, the first two across Wolt’s largest operating markets and the third spanning all operating countries and both domains. UVR V1 delivers +5.5% Merchant Trial Rate and +0.16% Global CVR over the previous production ranker; V2 adds a further +0.45% Merchant Trial Rate on top; and V3, our cross-domain unification of the restaurant and retail rankers, adds a further +1.31% Retail Merchant Trial Rate, together accounting for substantial incremental gross order value and a materially simplified serving stack.

CCS Concepts

• Information systems → Recommender systems.

Keywords

Recommender Systems, Sequential Recommendation, Session-aware Recommendation, Learning to Rank, Transformers, Gradient-Boosted Decision Trees, Cross-domain Recommendation, On-Demand Delivery

1 Introduction

Wolt, a DoorDash company, is a local commerce platform operating across 30+ countries and more than 1,000 cities in Europe and Asia for on-demand delivery across restaurants, grocery, and retail. Together with DoorDash, the group serves over 56 million monthly active users across 40+ countries.¹ Users access this selection through multiple surfaces, from multi-list (carousel) surfaces

to single ranked lists [23], making personalized store ranking the central task that determines which stores users see, click, and purchase from. Throughout this paper we use *store* as the umbrella term covering restaurants, grocery, and retail stores.

Recommender systems (RS) on delivery platforms face challenges absent in digital-only domains such as streaming platforms or social networks. Candidate stores are local: eligibility depends on delivery location, distance constraints, real-time courier availability, merchant operating hours and workload, rendering the stores candidate set dynamic and sparse at every request. As a multi-sided marketplace, rankings must balance consumer relevance, merchant exposure, and platform economics (objectives often in tension [15, 35]) while generalizing across heterogeneous verticals (restaurants, grocery, and retail), each with distinct consumption patterns and business constraints. A further tension is between *reorder* and *trial* purchases—the delivery-domain manifestation of the explore-exploit trade-off [3]: users return to trusted stores out of habit or convenience, yet also expect to find stores they have not tried before, and the ranker must actively balance both. We operationalize this trade-off as a constrained multi-objective problem, implemented via empirically-tuned per-session weighting (Section 4.4.2).

User behavior is sequential in nature, and purchase histories encode strong predictive signals for future decisions [17, 27, 28].

We present *Universal Venue Ranker* (UVR), a production system that pairs a bidirectional transformer encoder for sequential user modeling with a gradient-boosted decision-tree (GBDT) ranker integrating contextual and content features. Trained per country across all stores and domains, UVR replaces the previous production ranking stack with a single unified model. We evaluate UVR on offline benchmarks and in large-scale A/B tests, observing improvements in conversion rate and trial rate, which translate into increased gross order value. We present this as a real-world industry case study of session-aware, cross-domain personalization deployed at production scale across 30+ countries.

2 Related Work

Transformer-based Models for Sequential User Modeling. Causal-attention (SASRec [11]) and bidirectional-attention (BERT4Rec [27]) transformer architectures superseded earlier Markov-chain and RNN-based sequential recommenders, with subsequent work advancing training objectives and sampling strategies [20, 21]. They have been adopted across multiple industries; Tran et al. provide a comprehensive survey of transformers in recommendation [28]. Transformers have further proven effective in combination with *Semantic IDs* for retrieval and search [9, 18, 24].

¹As of December 2025

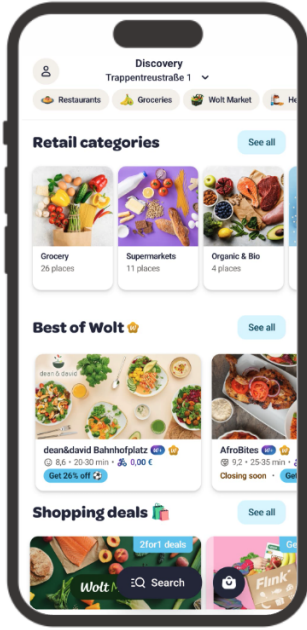


Figure 1: Wolt’s Discovery Page

Recommender Systems in the Delivery Domain. Food and delivery recommendation has been studied through benchmark datasets [2, 30], food-specific methods [12], and production architectures [4, 6, 8, 25, 34]. Relative to general e-commerce, delivery platforms introduce compounding constraints such as local candidate sets, real-time availability, and multi-stakeholder objectives, which remain underexplored in the RS literature. MTGR [8] exemplifies a recent industrial trend toward target-aware, generative multi-candidate architectures that score all candidates within a single model pass. UVR’s classifier shares this *multi-candidate, single-inference* property—one forward pass per request scores the entire candidate slate (Section 4.4.2)—but via a simpler, non-target-aware bidirectional encoder rather than a generative target-aware one.

Reorder vs. Trial Behavior. The tension between reorder and trial behavior has been studied through surrogate metrics for long-term engagement [31] and frameworks for balancing trial and reorder behavior [1, 3, 26, 28], primarily in music and video streaming. In on-demand delivery, users similarly oscillate between reorder from trusted stores and trial of new ones [12], and ranking must actively facilitate both.

Cross-domain Recommender Systems. Cross-domain recommender systems (CDRS) transfer knowledge across domains to mitigate cold-start and improve recommendation quality [5, 16]. Our multi-domain setting calls for this directly: a unified model lets behavior learned in one vertical inform ranking in another, and naturally handles user cold-start across domains.

UVR integrates these four threads into a single production system.

3 Problem Formulation

We frame personalized store ranking as a sequential recommendation problem with explicit time conditioning. Let $\mathcal{U}$ denote the set

Table 1: Notation overview.

Symbol	Meaning
$\mathcal{U}, \mathcal{I}$	user / store sets
u, i	a user / a store
l, t	delivery location, absolute timestamp
S_u	lifetime purchase sequence of user u
$S_u^{<t}$	subsequence of S_u with $t_k^u < t$
$\mathcal{I}_u^{<t}$	set of distinct stores in $S_u^{<t}$
$\mathcal{I}_{u,l}^t$	stores available to u at (l, t)
$r_{u,i}^t$	purchase indicator, $\in \{0, 1\}$
$\text{type}(u, i, t)$	purchase type $\in \{\text{CS}, \text{NEW}, \text{REC}\}$
$h_i(u, l, t)$	transformer (classifier) score for store i
$g(i u, l, t)$	GBDT (ranker) score for store i
$\mathbf{x}_{u,i,l}^t$	ranker feature vector

of users and $\mathcal{I}$ the set of stores. For each user $u \in \mathcal{U}$, let

$$S_u = (i_1, i_2, \dots, i_{n_u}), \quad i_k \in \mathcal{I},$$

denote their lifetime purchase sequence ordered by ascending timestamp, with t_k^u the absolute timestamp of the k -th purchase.

For any reference time t , we further introduce the time-truncated purchase subsequence $S_u^{<t}$ (ordered) and the corresponding store set $\mathcal{I}_u^{<t}$ (unordered):

$$S_u^{<t} = (i_k \in S_u : t_k^u < t), \\ \mathcal{I}_u^{<t} = \{i \in \mathcal{I} : i \in S_u^{<t}\}.$$

At request time, given a user u at delivery location l and time t , the ranker receives a candidate set $\mathcal{I}_{u,l}^t \subseteq \mathcal{I}$ determined by vicinity and operational constraints.² The task is to impose a ranking on $\mathcal{I}_{u,l}^t$ via a scoring function

$$g(i | u, l, t) \in \mathbb{R}, \quad i \in \mathcal{I}_{u,l}^t,$$

such that descending order by g minimizes the rank of the store the user purchases from.

We denote by $r_{u,i}^t \in \{0, 1\}$ the indicator that user u purchases from store i at time t ³. We assume single-purchase sessions:

$$\sum_{i \in \mathcal{I}_{u,l}^t} r_{u,i}^t = 1. \quad (1)$$

Sessions with multiple purchases are reduced to the store with the smallest production rank that received a purchase; sessions without any purchase are discarded.

We further partition purchases into three types according to a user’s prior history:

$$\text{type}(u, i, t) = \begin{cases} \text{CS} & \text{if } r_{u,i}^t = 1 \text{ and } \mathcal{I}_u^{<t} = \emptyset, \\ \text{NEW} & \text{if } r_{u,i}^t = 1 \text{ and } i \notin \mathcal{I}_u^{<t}, \\ \text{REC} & \text{if } r_{u,i}^t = 1 \text{ and } i \in \mathcal{I}_u^{<t}. \end{cases} \quad (2)$$

Cold-start (cs) purchases are from users with no prior purchase history; NEW and REC distinguish a trial order (first order from a store) from a recurring order (reorder) among returning users.

²The retrieval of $\mathcal{I}_{u,l}^t$ from $\mathcal{I}$ is out of scope for this work; in our setting it is driven by factors such as the size of the delivery area, store availability, and courier availability.

³We use the purchase signal as our primary target; clicks enter Section 4.2 as an auxiliary indicator $c_{u,i,l}^t$. The framework extends naturally to further interaction types (impressions, dwell time, ...).

4 Methodology

As depicted in Figure 2, UVR is a two-stage system:

- (1) a bidirectional transformer *classifier* h that maps a user’s purchase sequence and request context to a vector of per-store scores (Section 4.1);
- (2) a GBDT *ranker* g that consumes h as a single scalar feature alongside content, contextual, and user-store interaction features (Section 4.2).

Concretely, the final score is

$$g(i|u, l, t) = \text{GBDT}\left(h_i(u, l, t), \mathbf{x}_{u,i,l}^t\right), \quad (3)$$

where $h_i(u, l, t) \in \mathbb{R}$ is the raw transformer logit for store i and user u at time t , and $\mathbf{x}_{u,i,l}^t$ collects content and contextual features. We pass the raw logit rather than a softmax probability because the GBDT ranker is invariant to monotonic feature transformations.

UVR’s contribution is twofold: (i) it couples a sequential transformer to a tree-based ranker inside a single deployed stack, a stacking pattern established in production search [33] and instantiated here under hard local-availability constraints; and (ii) purchase-type-aware weighting makes the trial/reorder trade-off directly tunable (Section 4.4.2). The two stages are fit in sequence on disjoint data slices. Both are realized over a unified store vocabulary $\mathcal{I} = \mathcal{I}^{\text{rest}} \cup \mathcal{I}^{\text{retail}}$, covering restaurants and retail. The following sections describe this cross-domain UVR architecture; the experimental study (Section 6) reports on the restaurant-only configurations (V1, V2) followed by the cross-domain variant (V3), whose online A/B result is reported in Section 6.3.

4.1 Classifier: User Sequence Transformer for Next-Purchase Prediction

The classifier h is an encoder-only transformer [29] trained as a multi-class store classifier over $\mathcal{I}$, adopting the bidirectional-attention masked-prediction backbone of BERT4Rec [27] but narrowing its random-position cloze objective to mask only the most recent purchase in S_u .

Partial Target Masking. Three pieces of information are available to the model at inference time and are therefore supplied unmasked: (i) the request timestamp t , from which we derive hour-of-day, day-of-week and calendar week; (ii) the H3⁴ cell ID of the user’s location l (see Section 4.1.1); and (iii) the requested domain (restaurant or retail), which is known because most user surfaces are partitioned by domain. Two pieces of information are unknown and are masked: (a) the target store ID i^* , which is the prediction target, and (b) the product line of i^* , which is masked because the model is queried per domain rather than per product line; supplying the exact product line at the target position would constitute label leakage.

4.1.1 Embedding Layer. Each step of the input sequence is described by six categorical features and one continuous feature, summarized in Table 2. The categoricals are: target store ID $\in \mathcal{I} \cup \{\text{MASK}\}$; product line $\in \text{PL} \cup \{\text{MASK}^{\text{rest}}, \text{MASK}^{\text{retail}}\}$, where the

⁴H3 is an open-source hierarchical hexagonal geospatial indexing system maintained by Uber that partitions the surface of the Earth into nested hexagonal cells. We use resolution 8, whose cells have an average area of approximately 0.74 km², comparable to a small neighborhood or a few city blocks. See <https://h3geo.org/>.

Table 2: Features Representing Purchases in User Sequences

Feature	Source	Masked at target?
store ID	past or target purchase	yes
product line	past or target purchase	yes (per domain)
H3 hexagon ID	delivery location l	no
hour-of-day	timestamp t	no
day-of-week	timestamp t	no
week-of-year	timestamp t	no
$\tilde{\Delta}_k^u$	$t - t_k^u$	no (= $\log \beta$ at target)

two domain-specific mask tokens are appended beyond the real product-line vocabulary; week-of-year; day-of-week; hour-of-day; and the H3 cell ID of the user location. We deliberately do not feed raw latitude and longitude into the model; discretizing location to an H3 cell ID aligns with the granularity of our delivery operations and shares parameters across all requests originating from the same cell, which may also benefit cold-start users.

The continuous feature is the time gap $\Delta_k^u = t - t_k^u$ from each historical position to the target. We transform Δ_k^u by a log-affine map for numerical stability:

$$\tilde{\Delta}_k^u = \log(\alpha \Delta_k^u + \beta), \quad \alpha, \beta > 0. \quad (4)$$

At the target step (where $k = T$ and $\Delta_T^u = 0$), $\tilde{\Delta}_T^u = \log \beta$ acts as a constant offset.

Each categorical feature has its own learned embedding function. Writing $C = \{\text{store, pl, hex, hour, dow, week}\}$ for the set of categorical fields, we associate one embedding function per field,

$$E_c: \mathcal{X}_c \rightarrow \mathbb{R}^{d_c}, \quad c \in C,$$

where $\mathcal{X}_c$ is the input vocabulary for field c (for example, $\mathcal{X}_{\text{store}} = \mathcal{I} \cup \{\text{MASK}\}$ and $\mathcal{X}_{\text{pl}} = \text{PL} \cup \{\text{MASK}^{\text{rest}}, \text{MASK}^{\text{retail}}\}$). We denote by x_k^c the value of field c at step k . The input representation at step k is obtained by concatenating the six per-field embeddings with the transformed continuous gap and projecting to the model dimension d_{model} :

$$\mathbf{e}_k = W_{\text{in}} \left[(E_c(x_k^c))_{c \in C} \parallel \tilde{\Delta}_k^u \right] + \mathbf{b}_{\text{in}} + \mathbb{1}[k = T] \mathbf{d}_{\text{dom}(i^*)}, \quad (5)$$

where $\parallel$ denotes concatenation, $(E_c(x_k^c))_{c \in C}$ is the concatenation of the per-field embeddings in a fixed canonical order, $W_{\text{in}}, \mathbf{b}_{\text{in}}$ are the parameters of the linear input projection, and the indicator $\mathbb{1}[k = T]$ restricts the domain embedding $\mathbf{d}_{\text{dom}(i^*)} \in \mathbb{R}^{d_{\text{model}}}$ to the target step.

Two-level domain conditioning. We inject the requested domain at the target step at two points of the embedding pipeline. *Pre-projection*, the target’s product line carries either $\text{MASK}^{\text{rest}}$ or $\text{MASK}^{\text{retail}}$, which is processed by the product-line embedding alongside the other five categoricals. *Post-projection*, the additive term $\mathbb{1}[k = T] \mathbf{d}_{\text{dom}(i^*)}$ in Eq. 5 is a dedicated domain embedding, restricted by the indicator to the target step. At $k = T$ we additionally substitute the store and product-line inputs by their mask tokens, $x_T^{\text{store}} = \text{MASK}$ and $x_T^{\text{pl}} = \text{MASK}^{\text{dom}(i^*)}$.

We found that supplying the domain signal only through the pre-projection mask led to severe dilution as the representation propagated through successive encoder layers; the additive post-projection embedding instead acts as a clean, isolated signal in d_{model} -space and sharpens conditioning without disrupting the rest

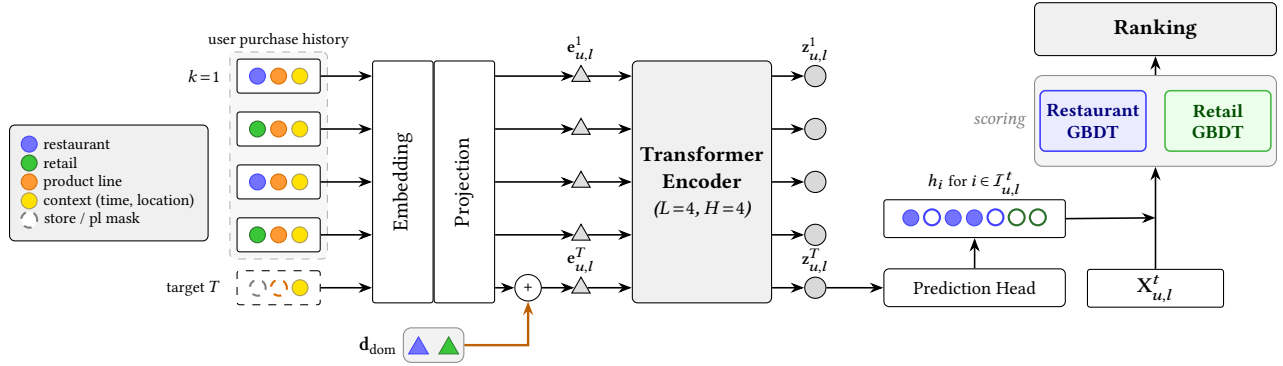


Figure 2: UVR architecture: bidirectional Transformer classifier h feeding domain-specific GBDT rankers g .

of the encoder input. We also experimented with FiLM-style feature-wise affine modulation [19] of the encoder activations but did not observe a benefit over the simpler additive scheme.

4.1.2 Encoder Layer.⁵

We stack $L = 4$ identical transformer encoder layers with pre-LayerNorm [32], multi-head self-attention with $H = 4$ heads, and a feed-forward sub-layer of hidden size $d_{ff} = 160$. The model dimension is $d_{model} = 128$. Sequences shorter than the context length T are left-padded with zero embeddings; padding positions are additionally excluded from attention via a key-padding mask. No causal mask is applied: the target step attends to all historical positions. We denote the encoder output at the target step by $z_{u,l}^T \in \mathbb{R}^{d_{model}}$.

4.1.3 Prediction Layer. A single linear head with weight matrix $W_{out} \in \mathbb{R}^{|I| \times d_{model}}$ and bias $b_{out} \in \mathbb{R}^{|I|}$ projects $z_{u,l}^T$ to a logit vector over the full store vocabulary,

$$h(u, l, t) = W_{out} z_{u,l}^T + b_{out} \in \mathbb{R}^{|I|},$$

and we write $h_i(u, l, t) = [W_{out}]_{i,:} z_{u,l}^T + b_{out,i}$ for the per-store logit.

At training time we minimize a per-sample-weighted, label-smoothed softmax cross-entropy loss whose denominator is restricted to the in-domain vocabulary $I^{\text{dom}(i^*)}$ of the ground-truth store, preventing cross-domain leakage during gradient updates. With label-smoothing parameter $\varepsilon \in [0, 1)$, the smoothed target distribution is

$$\tilde{y}_j = (1 - \varepsilon) \mathbf{1}[j = i^*] + \frac{\varepsilon}{|I^{\text{dom}(i^*)}|}, \quad j \in I^{\text{dom}(i^*)},$$

and the loss is

$$\mathcal{L}(\theta) = - \sum_{(u,l,t,i^*) \in \mathcal{D}} w_{u,i^*}^t \sum_{j \in I^{\text{dom}(i^*)}} \tilde{y}_j \log \frac{\exp h_j(u, l, t)}{\sum_{k \in I^{\text{dom}(i^*)}} \exp h_k(u, l, t)}. \quad (6)$$

The per-sample weights w_{u,i^*}^t are tied to the purchase type defined in Equation 2 and follow the ordering $w^{\text{NEW}} > w^{\text{CS}} > w^{\text{REC}}$, biasing the gradient towards stores the user has not yet purchased from while still rewarding correct predictions of returning purchases. At inference time the classifier is queried once per request with its (u, l, t) context; only the raw logits $h_i(u, l, t)$ for the candidate stores $i \in I_{u,l}^t$ are retained and passed to the ranker.

⁵Reported values are for V1 and V2. V3 widens the architecture to $H = 6$, $d_{model} = 192$, $d_{ff} = 384$ and trains for $E = 12$ epochs (instead of the $E = 15$ used by V1 and V2); the encoder depth $L = 4$ and all other hyperparameters in Section 4.4 are unchanged.

Cold-start handling. For cold-start users the encoder input collapses to a single (partially masked) target step. The cold-start signal is therefore implicit in the input length: with no historical positions to attend to, the model produces a domain-, location- and time-conditioned prior over stores from the target step alone. Note that the contextual features at the target (hex, hour, day-of-week, week) are not themselves sufficient to identify a user as cold-start, since two co-located requests at the same timestamp from different users (one cold-start, one returning) share those features exactly; the discriminating signal is the absence of historical positions, not the contents of the target step.

4.2 Ranker: GBDT for Listwise Store Ranking

The ranker g is a GBDT model implemented with CatBoost [22], taking per-(user, candidate-store) feature vectors $x_{u,i,l}^t$ over the candidate set $I_{u,l}^t$; training and label construction are detailed in Section 4.4.2.

The feature vector $x_{u,i,l}^t$ comprises four groups:

- **Classifier signal** (single scalar): the per-candidate transformer logit $h_i(u, l, t)$.
- **Store content features**: rating, conversion rate, retention rate, and assortment size (e.g. menu size for restaurants, in-stock item count for retail), etc.
- **Contextual features**: delivery distance, delivery price, and location-derived signals, etc.
- **User-store interaction features**: recent click counts, mean basket size, and trial ratio, etc.

Two domain-specific rankers are trained because the content and contextual feature sets differ structurally between restaurant and retail candidates (e.g. menu size is restaurant-only; in-stock item count is retail-only).

4.3 Coupling and Cross-Domain Transfer

Cross-domain transfer enters through (a) the chronologically interleaved sequence $S_u^{<t}$, (b) the per-position product-line embedding, and (c) the two-level domain conditioning at the target step (Section 4.1.1). A user's restaurant history thus informs their retail next-store prediction and vice versa, and the per-domain rankers inherit this transfer through the shared classifier logit $h_i(u, l, t)$.

Table 3: Comparison of UVR versions. V1, V2, and V3 are evaluated in Section 6.

Version	Domains	Ranker Loss	Notable additions
V1	restaurants	PAIRLOGIT (country-specific max_pairs)	—
V2	restaurants	YETIRANK-PAIRWISE:NDCG	label smoothing; trial weighting
V3	restaurants + retail	YETIRANK-PAIRWISE:NDCG	cross-domain coupling; product-line embedding

4.4 Model Training and Validation

We train one country-specific UVR model for each of the 30 countries Wolt operates in, and re-train it daily. This is a deliberate trade-off between training complexity, resource utilization, model degradation, and store cold-start. We do not observe a clear performance trend over the first ~28 days post-training; the daily cadence is instead driven by store cold-start: the store, H3 cell, and product-line embedding tables (Section 4.1.1) are populated only for values observed during training, making the classifier *transductive* with respect to these three vocabularies but *inductive* with respect to time (hour-of-day, day-of-week, week-of-year embeddings are allocated exhaustively at construction). Without daily retraining the number of unseen stores accumulates rapidly and degrades predictions. A complementary mitigation through supplementary content-based store features is left for future work, e.g. by adopting DenseRec [13].

We train and validate UVR in two stages: first the classifier (Section 4.4.1), then the ranker (Section 4.4.2), since the trained classifier is needed to score the candidate slates that constitute the ranker’s training, validation, and test data. The two stages operate on disjoint data slices: the classifier is trained on prefixes $S_u^{<(t-2d)}$, i.e. the last purchase of any sequence is guaranteed to happen before the ranker’s validation and test sessions. This prevents both the ranker’s validation day (which drives early stopping) and its held-out test day from leaking into the classifier’s training set.

4.4.1 Classifier.

Data split. We adopt a user-based split of user purchase sequences into a 90% training set $\mathcal{D}_c^{\text{train}}$ and a 10% validation set $\mathcal{D}_c^{\text{val}}$. Any sequence in $\mathcal{D}_c^{\text{val}}$ that references a store ID, an H3 cell ID, or a product line absent from $\mathcal{D}_c^{\text{train}}$ is moved back to the training set; given the transductive nature of the classifier with respect to these three feature fields (Section 4.4), this guarantees that every entity observed at validation time also occurs in the training corpus and therefore has an associated learned embedding. Sequences are capped to the most recent M purchases per user, since we observe diminishing returns from longer histories, consistent with the findings of Pancha et al. [17]. See appendix Section B for a detailed analysis.

Validation decomposition. We further partition $\mathcal{D}_c^{\text{val}}$ by the type of the target purchase, defined in Equation 2. Writing i_T^u for the most recent (target) purchase of user u at time t_T^u ,

$$\mathcal{D}_c^{\text{val}\tau} = \{S_u \in \mathcal{D}_c^{\text{val}} : \text{type}(u, i_T^u, t_T^u) = \tau\}, \quad \tau \in \{\text{CS}, \text{NEW}, \text{REC}\}. \quad (7)$$

This decomposition lets us monitor cold-start, trial, and reorder performance independently during training.

Optimization. We train the transformer for $E = 15$ epochs with a training batch size of 256 and a validation batch size of 1024. We use AdamW[14] with learning rate $\eta = 10^{-3}$ and weight decay $\omega = 0.15$, combined with a multi-step learning-rate schedule that decays η by a factor of $\gamma = 0.1$ at epochs $\{4, 6, 7\}$. Training runs in mixed-precision bfloat16 autocast on a single A10G GPU, with the loss in Equation 6 computed at label-smoothing $\varepsilon = 0.2$ and per-sample weights ordered $w^{\text{NEW}} > w^{\text{CS}} > w^{\text{REC}}$.

Checkpoint selection. After each epoch we evaluate the model on the three validation subsets in Equation 7. For each store i , let $\mathcal{N}_i \subset \mathcal{I}$ be the set of same-country stores within Haversine distance D of i , i itself included. For each S_u we score the classifier on $\mathcal{N}_{i_T^u}$, compute the rank of i_T^u , and aggregate to mean reciprocal rank (MRR) across $\mathcal{D}_c^{\text{val}\tau}$. On the NEW slice we further exclude $\mathcal{I}_u^{<t}$ from the scored set, so the target competes only against stores the user has not yet purchased from. We retain the checkpoint with the highest MRR on $\mathcal{D}_c^{\text{valNEW}}$, optimizing explicitly for trial recommendations; the REC and CS MRRs serve as guardrails.

4.4.2 Ranker.

Data. The ranker operates over user sessions rather than purchase sequences. Following the assumptions in Section 3, we focus on single-purchase sessions (Equation 1); sessions without any purchase are dropped as low-intent noise. A session in which several stores received a purchase is normalized to a single-purchase session by keeping $r_{u,i}^t = 1$ for the purchased store with the smallest production rank and setting $r_{u,j}^t \leftarrow 0$ for every other purchased store $j \neq i$ in the same session, so the single-purchase constraint of Equation 1 holds for every session in the resulting dataset. We define the ranker dataset as a set of session tuples

$$\mathcal{D}_r = \{(u, l, t, i^+, \mathcal{I}_{u,l}^t, \mathbf{X}_{u,l}^t, \rho^{\text{prod}}, \mathbf{o}_{u,l}^t, \mathbf{c}_{u,l}^t)\}, \quad (8)$$

where the session is uniquely identified by the (u, l, t) triple, $i^+ \in \mathcal{I}_{u,l}^t$ is the purchased store (the unique i for which $r_{u,i}^t = 1$ after the normalization above), $\mathbf{X}_{u,l}^t = (\mathbf{x}_{u,i,l}^t)_{i \in \mathcal{I}_{u,l}^t}$ collects the per-candidate feature vectors, $\rho^{\text{prod}} : \mathcal{I}_{u,l}^t \rightarrow \{1, \dots, |\mathcal{I}_{u,l}^t|\}$ is the production ranking served at request time, and $\mathbf{o}_{u,l}^t, \mathbf{c}_{u,l}^t \in \{0, 1\}^{|\mathcal{I}_{u,l}^t|}$ are per-candidate impression and click indicators ($o_{u,i,l}^t = 1$ iff i was shown to the user during the session, $c_{u,i,l}^t = 1$ iff i was clicked). At this stage $\mathbf{x}_{u,i,l}^t$ contains only content, contextual, and user-store interaction features; the classifier signal $h_i(u, l, t)$ is appended later (“Coupling with the classifier” below).

Time-based split. We use the most recent 30 days of sessions and adopt a time-based train/validation/test split, disjoint from the classifier’s user-based split: the first 28 days form $\mathcal{D}_r^{\text{train}}$, the second-to-last day forms $\mathcal{D}_r^{\text{val}}$ (used for early stopping), and the last day is held out as $\mathcal{D}_r^{\text{test}}$. The two-day cutoff applied to the classifier’s training horizon (Section 4.4) is what enforces the disjointness: where a training prefix $S_u^{<t'}$ allowed to terminate on the ranker’s validation or test day, the corresponding target purchase would be the same event that the ranker uses as its label on $\mathcal{D}_r^{\text{val}}$ or $\mathcal{D}_r^{\text{test}}$, and the classifier logit $h_{i^+}(u, l, t)$ subsequently fed to the ranker as

a feature would carry direct knowledge of the held-out target. By training the classifier only on prefixes $S_u^{<(t-2d)}$ we ensure no such overlap can occur and that $h_i(u, l, t)$, evaluated on $\mathcal{D}_r^{\text{val}}$ and $\mathcal{D}_r^{\text{test}}$, is a true out-of-sample prediction.

Training-set restriction. On $\mathcal{D}_r^{\text{train}}$ we restrict each session to its impressed stores $\{i \in \mathcal{I}_{u,l}^t : o_{u,i,l}^t = 1\}$. These are harder negatives than stores the user never saw: the production system already considered them attractive enough to display, so the ranker must learn to discriminate purchases from *competitive* alternatives rather than from random stores in the catalogue. On $\mathcal{D}_r^{\text{val}}$ and $\mathcal{D}_r^{\text{test}}$ we keep the entire candidate set $\mathcal{I}_{u,l}^t$, including unseen stores, because this matches inference: at request time the ranker must order the full slate without knowing which subset will be impressed to the user. The training data is therefore drawn from historical user session logs joined with impression events, while validation and test data couple the same sessions with the full backend slate.

Coupling with the classifier. For each session $(u, l, t, \dots) \in \mathcal{D}_r$, we reconstruct the user’s purchase sequence as it stood at request time, $S_u^{<t}$, and append the partially masked target step. The target step carries the request’s H3 cell l , calendar features derived from t , and the domain-specific product-line mask MASK^{dom} (with $\text{dom} \in \{\text{rest}, \text{retail}\}$ chosen according to which per-domain ranker is being trained). A single forward pass of the trained classifier h on this input yields the unified logit vector, from which we gather the per-candidate logits $\{h_i(u, l, t)\}_{i \in \mathcal{I}_{u,l}^t}$ and append $h_i(u, l, t)$ to $\mathbf{x}_{u,i,l}^t$ as one additional ranker feature. Note that h is invoked once per session, not once per (session, candidate) pair: the candidate logits are read out from the same forward pass via a single $\mathcal{O}(|\mathcal{I}_{u,l}^t|)$ gather, so the offline join scales linearly in the number of sessions rather than in the number of (session, candidate) pairs.

Loss and labels. Both rankers use the session identifier as the ranking group. In V1 we used a pairwise BPR-style logistic loss (PAIRLOGIT) with the number of negatives `max_pairs` tuned per country, and negatives drawn from impressed-but-not-purchased stores. With V2’s trial-oriented upgrade we switched to the list-aware pairwise objective YETIRANKPAIRWISE with `mode=NDCG` [7], which let us additionally incorporate clicks as graded relevance. Concretely, YetiRank builds a position-dependent weight for each pair by repeatedly perturbing the current model scores with random noise and re-ranking; pairs of candidates that land adjacent near the top of these perturbed rankings accumulate a larger weight than pairs deep in the list, concentrating learning signal on the comparisons most likely to affect the top of the served ranking [7]. Setting `mode=NDCG` selects NDCG as the target metric this weighting is calibrated toward. We assign a per-row label as a function of the purchase indicator $r_{u,i}^t$, the click indicator $c_{u,i,l}^t$, and the user-store relationship $\text{rel}(u, i, t) \in \{\text{NEW}, \text{REC}\}$, defined as `REC` if $i \in \mathcal{I}_u^{<t}$ and `NEW` otherwise—the binary projection of Equation 2 extended to

click rows, with `cs` folded into `NEW`:

$$y_{u,i,l}^t = \begin{cases} \lambda^{\text{P,NEW}} & \text{if } r_{u,i}^t = 1, \text{ rel} = \text{NEW}, \\ \lambda^{\text{P,REC}} & \text{if } r_{u,i}^t = 1, \text{ rel} = \text{REC}, \\ \lambda^{\text{C,NEW}} & \text{if } r_{u,i}^t = 0, c_{u,i,l}^t = 1, \text{ rel} = \text{NEW}, \\ \lambda^{\text{C,REC}} & \text{if } r_{u,i}^t = 0, c_{u,i,l}^t = 1, \text{ rel} = \text{REC}, \\ 0 & \text{otherwise.} \end{cases} \quad (9)$$

The case predicates partition $\{0, 1\}^2$ along the purchase axis first, so a row that records both a click and a purchase is treated as a purchase rather than a click. The constants are calibrated so that purchase signal dominates click signal and click signal in turn dominates the absence of interaction: $\lambda^{\text{P,NEW}} = \lambda^{\text{P,REC}} > \lambda^{\text{C,NEW}} > \lambda^{\text{C,REC}} > 0$. The two purchase-row labels are equal because the trial-vs-reorder distinction is already captured at the session level via group weights, applied to the listwise NDCG-style loss; the row label only needs to express the relative strength of the implicit-feedback signal. We additionally use per-session weights ordered $w^{\text{NEW}} > w^{\text{REC}}$, biasing the ranker toward trial sessions while preserving reorder performance. Unlike the classifier, the ranker collapses `cs` and non-`cs` new sessions into a single w^{NEW} ; a separate `cs` weight could yield finer control and is left to future work.

We frame this as a constrained multi-objective problem: maximize trial ranking quality subject to a bounded reorder regression. The per-session weight ratio ($w^{\text{NEW}}/w^{\text{REC}}$) is this constraint’s empirically tuned Lagrange multiplier; Appendix A’s grid sweep traces the resulting Pareto frontier as the multiplier varies. Tuning a single scalarization weight empirically, rather than deriving it analytically, mirrors established practice in multi-objective learning-to-rank and explore/exploit systems [3, 26].

Per-domain rankers. We train one ranker for the restaurant domain (introduced in V1) and, since V3, an additional ranker for the retail domain. Both share the same loss and label scheme and most optimization settings. Both are trained for $T_r = 500$ boosting iterations with tree depth 8, learning rate 0.05, and L2 leaf regularization 3.0, with early stopping on $\mathcal{D}_r^{\text{val}}$ after 50 rounds without improvement.

4.5 Testing and Benchmarking

After training has concluded, we evaluate the per-domain ranker on the held-out last day of sessions $\mathcal{D}_r^{\text{test}}$. We compute MRR per purchase type from Equation 2—separately for `cs`, `NEW`, and `REC` targets—and across four scoring functions: a benchmark b , the production ranking ρ^{prod} , the classifier h , and the trained ranker g , yielding twelve MRR values per evaluation run and model domain. The benchmark b combines a binary reorder indicator with a store-relative popularity score; aside from the reorder indicator (which is itself user-specific) it uses no further personalized signal, providing a strong yet *weakly personalized* reference. On the `NEW` slice we consistently observe

$$\text{MRR}_b < \text{MRR}_{\rho^{\text{prod}}} < \text{MRR}_g \quad \text{and} \quad \text{MRR}_h < \text{MRR}_g, \quad (10)$$

in every country and for both versions, while the position of h relative to ρ^{prod} is country-dependent (Table 4). On `REC` the ordering depends on the operating point: at trial-weighted settings g can fall below both h and ρ^{prod} (Section 6.2, Appendix A). We therefore

apply the safeguard on the trial side: a new ranker is blocked from roll-out if its MRR on the **NEW** segment fails to exceed MRR_b at 95% confidence (the lower bound of a two-sigma interval around MRR_g must exceed MRR_b). Performance on **REC** is monitored as a guardrail and surfaces a warning, but does not block deployment.

5 Training and Serving Infrastructure

We launch parallel daily training jobs for each country on our internal Kubernetes cluster, orchestrated via Flyte⁶. User interaction sequences for inference are generated during training and written to an online feature store to keep training and serving features synchronized. Since purchases are relatively sparse signals, daily updates were sufficient in our setting.

Additional feature groups—including content, contextual, and user-store interaction features—are computed through separate batch pipelines and published to the online feature store at cadences ranging from a few hours to daily, depending on upstream data availability.

At inference time, the ranking service retrieves all required features from the online feature store, computes the transformer-based classifier logits on CPUs, and applies the downstream GBDT ranker to produce the final store ranking. The end-to-end serving pipeline has a p99 latency of approximately 60 ms, including feature retrieval, model inference, and serialization.

6 Experiment Results

We report offline and online results for V1 and V2, the two production rollouts of UVR; their architectural and loss-function differences are summarized in Table 3. V1 and V2 are restaurant-only; V3, our cross-domain unification, is evaluated separately below (Section 6.3). The offline benchmarks in this section therefore focus on the restaurant domain, which dominates Wolt’s order volume. Substantial offline uplifts gave us the conviction to carry both versions forward to A/B tests: V1 was rolled out in 2025 while V2 was deployed to production and replaced V1 in early 2026. V3 then replaced V2 a few months later. Throughout the section, we focus on a subset of our largest operating countries by purchase volume, which together account for more than 50% of Wolt’s worldwide order volume and provide a representative cross-section while keeping experimental complexity manageable.

6.1 Baselines

We compare UVR against the production stack it was designed to subsume, which consists of four ranking models split across two domains. In the restaurant domain we operate three models: Neural Collaborative Filtering [10] with BPR-loss as a first-pass ranker (FPR), a DNN-based context- and content-aware second-pass ranker (SPR), and a GBDT-based ranker for cold-start users (CSR). In the retail domain we operate a single GBDT-based ranker (RR). The offline experiments below report uplifts of UVR over this stack, and the online A/B tests (Section 6.3) use FPR+SPR/CSR as the V1 control arm.

Table 4: Relative MRR uplift of V1/V2 over production, by country and purchase type (NEW**/**REC**); classifier h and ranker g reported separately.**

Country	Model	UVR V1		UVR V2	
		NEW	REC	NEW	REC
ISR	Classifier	−0.5%	−3.6%	+2.2%	−4.2%
	Ranker	+19.7%	+5.0%	+30.3%	−12.5%
GRC	Classifier	+4.7%	−8.7%	+5.0%	−8.7%
	Ranker	+17.6%	−5.0%	+22.3%	−7.8%
DEU	Classifier	+4.8%	−17.0%	+6.4%	−16.6%
	Ranker	+19.8%	−11.4%	+29.0%	−27.3%
FIN	Classifier	−4.8%	+6.2%	−3.5%	+7.7%
	Ranker	+7.5%	+11.2%	+12.5%	+6.7%
HUN	Classifier	−0.6%	+0.3%	−1.0%	+1.3%
	Ranker	+7.7%	+9.4%	+14.3%	−1.4%
DNK	Classifier	−3.1%	+4.0%	−1.7%	+5.7%
	Ranker	+9.0%	+7.6%	+14.5%	−5.8%

6.2 Offline Testing

We report the relative MRR improvement of the classifier h and ranker g over production, broken down by purchase type; absolute volumes are omitted due to confidentiality. Headline numbers are anchored at a Sunday peak-volume test day, with results on other days directionally consistent. Table 4 reports per-country uplifts for V1 and V2 on **NEW** and **RECURRING** purchases; we omit **CS** as its session volume in this slice is negligible. A single-country study with multiple training seeds shows run-to-run standard deviation of ~10-30% relative to the mean uplifts; individual cells are therefore directional.

V1 (PairLogit) over the production ranker. The ranker g improves over the classifier h in all 12 of 12 (country, purchase-type) cells, confirming that content and contextual features carry information complementary to the sequence signal alone. The improvement holds uniformly across both **NEW** and **REC** purchases: every country gains on **NEW** (range +7.5% to +19.8%), and the ranker also recovers ground on **REC** relative to the classifier in every country.

V2 (YetiRankPairwise:NDCG) over V1. The picture splits along the trial/reorder axis. On **NEW**, V2 amplifies the V1 gains uniformly across all six countries (e.g. ISR +19.7% → +30.3%, DEU +19.8% → +29.0%), in line with the design intent of the listwise NDCG-aware loss. On **RECURRING** purchases, however, the classifier alone outperforms the V2 ranker in five of six countries (ISR, DEU, FIN, HUN, DNK; only GRC remains in the ranker’s favor), suggesting that the trial-oriented loss penalizes reorder relevance more aggressively than V1’s PairLogit. Section 6.3 tests whether this trade-off surfaces in production; Appendix A dissects the ranker-level levers driving it via a controlled ablation with the classifier held fixed.

Operating point selection. From the sweep of Appendix A we discard every setting with a negative **NEW** MRR uplift, then take the setting with the largest **NEW** uplift whose **REC** regression stays within −15%. This makes the constraint of Section 4.4.2 explicit. DEU is the one country where no setting satisfied the bound.

⁶<https://flyte.org/>

Table 5: Online A/B outcomes per UVR version (V1 vs. FPR+SPR/CSR; V2 vs. V1; V3 vs. V2+RR). Merchant Trial Rate gains are significant at $\alpha = 0.05$; Global CVR is reported with its p -value.

Version	Merchant Trial Rate	Global CVR
V1	+5.5%	+0.16% ($p = 0.017$)
V2	+0.45%	-0.05% ($p = 0.398$)
V3	+1.31% [†]	+0.04% ($p = 0.457$)

[†]Retail domain.

6.3 Online A/B Testing

The offline uplifts in Section 6.2, combined with our internal experience with the predecessor FPR and SPR rankers (where offline MRR gains had consistently translated into online wins), motivated three consecutive A/B tests. All three tests used a 50:50 user-level random split over all registered users, cold-start and returning alike. V1 and V2 ran for two weeks across the ten largest operating countries; V3 ran for three weeks across all operating countries, reflecting the larger sample needed for retail-domain significance. The target metrics were the daily global conversion rate (Global CVR) and the daily Merchant Trial Rate, which captures first-time ordering from a store. Table 5 summarizes the per-version outcomes.

Three observations stand out. First, the V1 Merchant Trial Rate gain of +5.5% confirms that UVR effectively pushes trial in production; combined with the simultaneous +0.16% Global CVR uplift, this shows that the trial push does not come at the expense of conversions. Second, V2’s +0.45% Merchant Trial Rate gain on top of an already strong V1 baseline confirms that the listwise loss, along with label smoothing and an additional trial feature, continues to deliver incremental trial value. Third, V3’s +1.31% Merchant Trial Rate gain is measured on the retail domain—V1 and V2’s figures above are restaurant-domain—confirming that unifying the restaurant and retail rankers lifts retail discovery with no significant change in restaurant-domain trial rate or blended Global CVR. Ads revenue regressed significantly under both V1 and V3 (−0.43% for V3), which we read as confirmation of effectiveness: personalized ranking outgrew the Ads stack, which still relied on a variant of the previous production restaurant ranker. RR was retired upon V3’s rollout, completing the consolidation of four legacy rankers into one.

6.4 Discussion and Limitations

The online results confirm that the trial side of the trade-off can be steered effectively, and that steering it is compatible with a conversion gain (V1: +5.5% Merchant Trial Rate with +0.16% Global CVR). The partial offline reorder regressions of Section 6.2 left the blended metric statistically unchanged ($|\Delta| < 0.1\%$, $p > 0.39$). Beyond ranking quality, consolidating FPR, SPR, CSR, and RR into a single cross-domain model (V3) substantially reduces operational complexity, retiring the last of four legacy rankers. The control arm reflects the original rank, which includes ad placements and deduplication beyond the ranker, so a direct model-to-model comparison would likely narrow the observed gap.

The V3 result validates the business case for cross-domain unification but does not by itself isolate the contribution of cross-domain

transfer, i.e., whether retail benefits from shared restaurant purchase signal, or simply from finally having a transformer+GBDT stack rather than a single GBDT (RR). Two ablations would sharpen the attribution: a domain-siloed transformer+GBDT retail model, separating transfer from the mere presence of a sequence stage, and an otherwise-identical ranker trained without the classifier logit, isolating the transformer’s marginal contribution. The closest evidence we report is the classifier-versus-ranker comparison of Table 4 alongside the two purely tree-based production baselines (CSR, RR); feature sets, losses, and training data differ, so neither is controlled. Under a fixed engineering timeline we prioritized the online tests; each ablation is a single training run and we will report both.

Three limitations are worth noting. First, UVR runs on purchase histories with up to one day of delay and does not exploit real-time interaction signals; the architecture could readily accommodate these as first-class sequence events. Second, the transductive prediction head restricts the classifier to stores seen during training, leaving new stores reliant on the GBDT’s content features alone. Third, restricting the sequence to purchases leaves other useful signals (clicks, browsing, dwell time) untapped, which could benefit users with sparse histories.

7 Conclusion

We presented UVR, a production transformer–GBDT system that consolidates four specialized rankers—three for restaurants, one for retail—into a single unified model. Three consecutive A/B tests validated the approach: V1 delivered +5.5% Merchant Trial Rate with a simultaneous +0.16% Global CVR uplift; V2 added a further +0.45% Merchant Trial Rate; and V3, our cross-domain unification, delivered a further +1.31% Retail Merchant Trial Rate while retiring the last domain-specific legacy ranker. The hybrid architecture makes the trial/reorder balance an explicit operating point: the per-session weight ratio $w^{\text{NEW}}/w^{\text{REC}}$ and the sampled-negative count max_pairs scalarize a constrained multi-objective problem, and 150 ranker retrains trace the frontier they span (Appendix A). At the ratio we deploy, this buys +12% to +30% offline trial MRR over production and costs reorder MRR in five of six countries, a trade that did not surface in Global CVR. Future work focuses on target-aware conditioning of the transformer on the candidate stores, inductive content store representations for cold-start mitigation, and real-time interaction signals for short-term intent.

Acknowledgments

The authors would like to thank Adam Smaili, Daniel Lazar, Paavo Camps, Tanja Menković, Ibrahim Omer Celik, Dzung Nguyen, Sergio Gonzalez Sanz as well as the Machine Learning Platform, Analytics and Product Management teams at Wolt who contributed or supported us throughout this project.

References

- [1] Ashton Anderson, Lucas Maystre, Ian Anderson, Rishabh Mehrotra, and Mounia Lalmas. 2020. Algorithmic Effects on the Diversity of Consumption on Spotify. In *Proceedings of The Web Conference 2020* (Taipei, Taiwan) (WWW '20). Association for Computing Machinery, New York, NY, USA, 2155–2165. doi:10.1145/3366423.3380281
- [2] Zhenisbek Assylbekov et al. 2023. Delivery Hero Recommendation Dataset (DHRD): A Novel Dataset for Benchmarking Recommendation Algorithms. In *Proceedings of the 17th ACM Conference on Recommender Systems (RecSys '23)*. ACM.
- [3] Minmin Chen, Yuyan Wang, Can Xu, Ya Le, Mohit Sharma, Lee Richardson, Su-Lin Wu, and Ed Chi. 2021. Values of user exploration in recommender systems. In *Proceedings of the 15th ACM Conference on recommender systems*. 85–95.
- [4] Alex Egg. 2021. Online learning for recommendations at grubhub. In *Proceedings of the 15th ACM Conference on Recommender Systems*. 569–571.
- [5] Ali Mamdouh Elkahky, Yang Song, and Xiaodong He. 2015. A Multi-View Deep Learning Approach for Cross Domain User Modeling in Recommendation Systems (WWW '15). International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 278–288. doi:10.1145/2736277.2741667
- [6] Zhichao Feng, Junjie Xie, Kaiyuan Li, Yu Qin, Pengfei Wang, Qianzhong Li, Bin Yin, Xiang Li, Wei Lin, and Shangguang Wang. 2024. Context-based fast recommendation strategy for long user behavior sequence in Meituan Waimai. In *Companion Proceedings of the ACM Web Conference 2024*. 355–363.
- [7] Andrey Gulin, Igor Kuralenok, and Dmitry Pavlov. 2011. Winning The Transfer Learning Track of Yahoo!'s Learning To Rank Challenge with YetiRank. In *Proceedings of the Yahoo! Learning to Rank Challenge (Proceedings of Machine Learning Research, Vol. 14)*. PMLR, 63–76. <https://proceedings.mlr.press/v14/gulin11a.html>
- [8] Ruidong Han, Bin Yin, Shangyu Chen, He Jiang, Fei Jiang, Xiang Li, Chi Ma, Mincong Huang, Xiaoguang Li, Chunzhen Jing, et al. 2025. Mtrg: Industrial-scale generative recommendation framework in meituan. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*. 5731–5738.
- [9] Ruining He, Lukasz Heldt, Lichan Hong, Raghunandan Keshavan, Shifan Mao, Nikhil Mehta, Zhengyang Su, Alicia Tsai, Yueqi Wang, Shao-Chuan Wang, et al. 2026. Plum: Adapting pre-trained language models for industrial-scale generative recommendations. In *Proceedings of the ACM Web Conference 2026*. 8093–8104.
- [10] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web (Perth, Australia) (WWW '17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 173–182.
- [11] Wang-Cheng Kang and Julian McAuley. 2018. Self-Attentive Sequential Recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*. 197–206. doi:10.1109/ICDM.2018.00035
- [12] Jiayu Li, Aixin Sun, Weizhi Ma, Peijie Sun, and Min Zhang. 2024. Right Tool, Right Job: Recommendation for Repeat and Exploration Consumption in Food Delivery. In *Proceedings of the 18th ACM Conference on Recommender Systems (RecSys '24)*. Association for Computing Machinery, New York, NY, USA, 643–653. doi:10.1145/3640457.3688119
- [13] Jan Malte Lichtenberg, Antonio De Candia, and Matteo Ruffini. 2025. DenseRec: Revisiting Dense Content Embeddings for Sequential Transformer-based Recommendation. In *EARL Workshop at the 19th ACM Conference on Recommender Systems (RecSys '25)*. https://earl-workshop.github.io/pdf/recsys2025-workshops_paper_167.pdf
- [14] Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *7th International Conference on Learning Representations (ICLR)*. New Orleans, LA, USA. <https://openreview.net/forum?id=Bkg6RiCqY7>
- [15] Rishabh Mehrotra and Benjamin Carterette. 2019. Recommendations in a marketplace. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 580–581.
- [16] Zahra Nazari, Christophe Charbuillet, Johan Pages, Martin Laurent, Denis Charrier, Briana Vecchione, and Ben Carterette. 2020. Recommending podcasts for cold-start users based on music listening and taste. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1041–1050.
- [17] Nikil Pancha, Andrew Zhai, Jure Leskovec, and Charles Rosenberg. 2022. PinnerFormer: Sequence Modeling for User Representation at Pinterest. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Washington DC, USA) (KDD '22). Association for Computing Machinery, New York, NY, USA, 3702–3712. doi:10.1145/3534678.3539156
- [18] Gustavo Penha, Edoardo D'Amico, Marco De Nadai, Enrico Palumbo, Alessandro Tamborrino, Ali Vardasbi, Max Lefarov, Shawn Lin, Timothy Heath, Francesco Fabbri, et al. 2025. Semantic ids for joint generative search and recommendation. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems*. 1296–1301.
- [19] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. 2018. FiLM: Visual Reasoning with a General Conditioning Layer. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI '18)*. AAAI Press, 3942–3951. doi:10.1609/aaai.v32i1.11671
- [20] Aleksandr Petrov and Craig Macdonald. 2022. A Systematic Review and Replicability Study of BERT4Rec for Sequential Recommendation. In *Proceedings of the 16th ACM Conference on Recommender Systems* (Seattle, WA, USA) (RecSys '22). Association for Computing Machinery, New York, NY, USA, 436–447. doi:10.1145/3523227.3548487
- [21] Aleksandr Vladimirovich Petrov and Craig Macdonald. 2023. gsarec: Reducing overconfidence in sequential recommendation trained with negative sampling. In *Proceedings of the 17th ACM Conference on Recommender Systems*. 116–128.
- [22] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. 2018. CatBoost: unbiased boosting with categorical features. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems (Montréal, Canada) (NIPS'18)*. Curran Associates Inc., Red Hook, NY, USA, 6639–6649.
- [23] Behnam Rahdari, Branislav Kveton, and Peter Brusilovsky. 2022. The Magic of Carousels: Single vs. Multi-List Recommender Systems. In *Proceedings of the 33rd ACM Conference on Hypertext and Social Media (Barcelona, Spain) (HT '22)*. Association for Computing Machinery, New York, NY, USA. doi:10.1145/3511095.3531278
- [24] Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Tran, Jonah Samost, et al. 2023. Recommender systems with generative retrieval. *Advances in Neural Information Processing Systems* 36 (2023), 10299–10315.
- [25] Meituan Research. 2025. User Long-Term Multi-Interest Retrieval Model for Recommendation. In *Proceedings of the 19th ACM Conference on Recommender Systems (RecSys '25)*. ACM.
- [26] Yi Su, Xiangyu Wang, Elaine Ya Le, Liang Liu, Yuening Li, Haokai Lu, Benjamin Lipshitz, Sriraj Badam, Lukasz Heldt, Shuchao Bi, et al. 2024. Long-term value of exploration: Measurements, findings and algorithms. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. 636–644.
- [27] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (Beijing, China) (CIKM '19)*. Association for Computing Machinery, New York, NY, USA, 1441–1450. doi:10.1145/3357384.3357895
- [28] Viet-Anh Tran, Guillaume Salha-Galvan, Bruno Sguerra, and Romain Hennequin. 2024. Transformers Meet ACT-R: Repeat-Aware and Sequential Listening Session Recommendation. In *Proceedings of the 18th ACM Conference on Recommender Systems (Bari, Italy) (RecSys '24)*. Association for Computing Machinery, New York, NY, USA, 486–496. doi:10.1145/3640457.3688139
- [29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
- [30] Hai Wang, Lei Zhao, Kaile Yan, Xiaoyi Wang, Yile Liang, and Jinghua Hao. 2025. Meituan: Operational-Level On-Demand Food Delivery Data for the 2025 INFORMS TSL Data-Driven Research Challenge. Available at SSRN 5696423 (2025).
- [31] Yuyan Wang, Mohit Sharma, Can Xu, Sriraj Badam, Qian Sun, Lee Richardson, Lisa Chung, Ed H. Chi, and Minmin Chen. 2022. Surrogate for Long-Term User Experience in Recommender Systems. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Washington DC, USA) (KDD '22). Association for Computing Machinery, New York, NY, USA, 4100–4109. doi:10.1145/3534678.3539073
- [32] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. 2020. On Layer Normalization in the Transformer Architecture. In *Proceedings of the 37th International Conference on Machine Learning (ICML) (Proceedings of Machine Learning Research, Vol. 119)*. PMLR, 10524–10533.
- [33] Dawei Yin, Yuening Hu, Jiliang Tang, Tim Daly, Mianwei Zhou, Hua Ouyang, Jianhui Chen, Changsung Kang, Hongbo Deng, Chikashi Nobata, Jean-Marc Langlois, and Yi Chang. 2016. Ranking Relevance in Yahoo Search. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, 323–332. doi:10.1145/2939672.2939677
- [34] Yu Zhang et al. 2021. A Dual Augmented Two-tower Model for Online Large-scale Recommendation. In *Proceedings of the 3rd Workshop on Deep Learning Practice for High-Dimensional Sparse Data (DLP-KDD '21)*.
- [35] Yong Zheng. 2019. Multi-stakeholder recommendations: case studies, methods and challenges. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 578–579.

A Trial vs. Reorder Trade-off

To isolate the ranker’s contribution to the trial vs. reorder trade-off, we hold the classifier fixed and re-train only the ranker across 3 per-session weighting regimes $w^{\text{NEW}}/w^{\text{REC}} \in \{1.5, 2.0, 3.0\}$ ($w^{\text{REC}}=1$; the three clusters in Figure 3) and 50 `max_pairs` negative-sampling settings per regime (dot color). All 150 runs share the same country, day, and random seed.

Axes report relative MRR uplift over the production baseline on `NEW` (x) and `REC` (y) purchases. The clusters trace a Pareto-style frontier: $w^{\text{NEW}}/w^{\text{REC}}=1.5$ yields strong reorder gains with a small trial regression, 2.0 balances both at modest positive uplifts, and 3.0 delivers a strong trial uplift at the cost of reorder. Within each cluster, increasing `max_pairs` (low $\rightarrow$ high) consistently shifts the model results toward stronger reorder and weaker trial. Together these two ranker-only levers enabled us to tune the trade-off between trial and reorder.

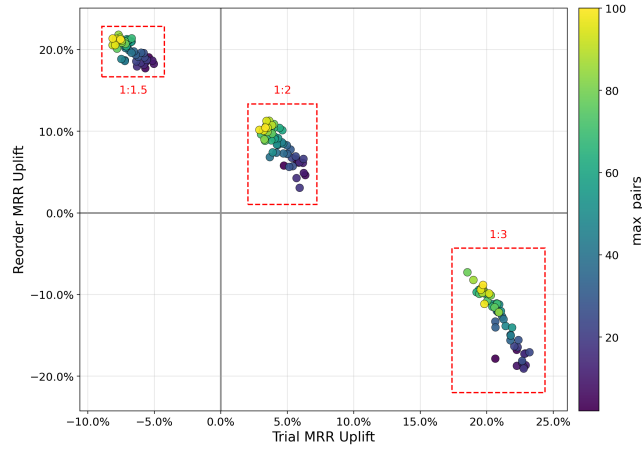


Figure 3: Ranker ablation: relative MRR uplift over production on `NEW` (x) vs. `REC` (y) with the classifier held fixed.

B MRR vs. Maximum Sequence Length

Figure 4 shows that reorder MRR rises sharply up to ~ 30 purchases and then plateaus, while trial and cold-start MRR are largely insensitive to sequence length. This motivates the moderate sequence cap M used in training (Section 4.4.1).

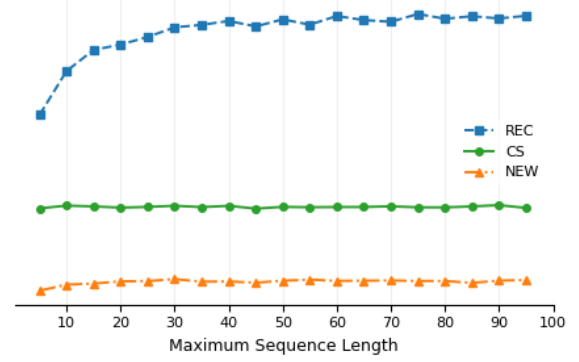


Figure 4: Classifier MRR on the `REC`, `CS`, and `NEW` validation slices vs. the maximum input sequence length.