

SONA TECHNICAL REPORT

A SINGLE-MODEL GENERATIVE RECOMMENDER FOR YANDEX MUSIC

Sona Team

We introduce SONA, a single-model generative recommender for Yandex Music. In an online A/B test, SONA replaced the entire production cascade — more than 15 candidate generators followed by pre-ranking and ranking models that consume hundreds of features, including signals from large transformer models such as Argus [12] and target-attention scorers — while significantly improving key engagement metrics.

The architecture of SONA unifies candidate generation and ranking around a shared user representation. Its encoder transforms the user’s chronological sequence of logged engagement events into hidden states consumed by both the autoregressive decoder and the Ranking Module. The next-token-prediction and distillation objectives jointly update the encoder, coupling generation and ranking through the same user state. Neither SONA nor its Teacher Ranker uses hand-engineered features; both operate on logged event fields and learned item representations. In the final SONA configuration, the larger teacher supplies ranking targets during training but is absent from serving, leaving the encoder, decoder, and Ranking Module as a single deployed model.

We evaluate SONA in an online A/B experiment using live traffic from My Vibe on smart speakers, one of Yandex Music’s largest recommendation surfaces. On this surface, playback begins without the user specifying an artist, genre, mood, or other preference. Relative to the production control, SONA produced statistically significant uplifts of +4.53% in Active Users, the primary metric, +6.30% in Total Listening Time, and +11.42% in Likes. These effects were incremental to improvements retained from preceding deployments. The Active Users uplift was $2.35\times$ the increment previously delivered by Argus, the strongest model deployed on this surface before SONA. These results show that a single jointly trained model can replace a mature multi-stage recommendation cascade while improving recommendation quality on live traffic.

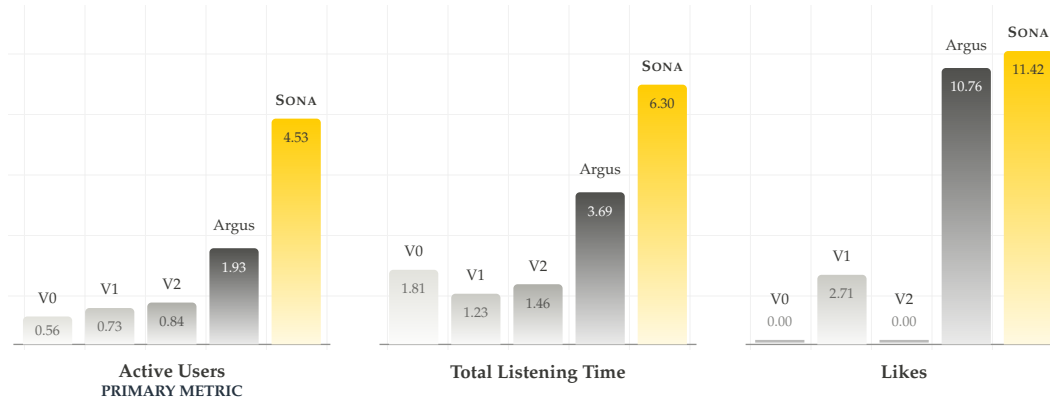


Figure 1 | Successive transformer deployments in My Vibe on smart speakers. V0, V1, V2, and Argus are successive generations of transformer-based recommendation models deployed on the surface before SONA. Bars show incremental A/B uplift in percent; panels use independent scales. Each deployment adds its gain on top of those retained from all previous deployments. SONA more than doubles Argus on the primary Active Users metric.

Contents

1	Introduction	3
2	SONA Overview	4
2.1	Design Philosophy	4
2.2	Overall Framework	5
3	SONA Architecture	5
3.1	Semantic Tokenizer	6
3.2	User History Representation	8
3.3	Encoder	9
3.4	Decoder	10
3.5	Ranking Module	10
4	SONA Training	11
4.1	Dataset Design	11
4.2	Training Framework	11
5	Teacher Ranker	13
5.1	Architecture	13
5.2	Dataset Design	13
5.3	Training Framework	14
6	Training and Inference Infrastructure	14
6.1	Training Optimizations	14
6.2	Online Training Infrastructure	15
6.3	Inference Optimizations	16
7	Evaluation	17
7.1	Evaluation Protocol and Metrics	17
7.2	Candidate Generation	18
7.3	Teacher Ranker Design Ablations	19
7.4	Unified Generation and Ranking	20
7.5	Online A/B Tests	22
8	Conclusion, Limitations, and Future Work	24
A	Contributions and Acknowledgments	27
B	Hyperparameters and Model Cards	27
C	Notation	30

1 Introduction

We present SONA, a generative recommender for Yandex Music that combines candidate generation and ranking in a single model, evaluated in online A/B experiments on live traffic. Its served architecture is a transformer over the user’s chronological event history with two output modules: a *decoder* and a *Ranking Module*. In the final experiment this single model replaces the candidate generation, pre-ranking, and ranking stages of a mature recommendation cascade of separately trained models. The report describes the architecture, training recipe, and infrastructure that allowed this model to surpass the cascade on live traffic.

Yandex Music is a subscription music-streaming service built around personalized listening: alongside search, curated playlists, and editorial content, the home feed, radio-style stations, and the personal My Vibe stream are all driven by recommendations, and recommended playback accounts for a large share of total listening time. The service runs on mobile and desktop applications, the web player, TVs, and smart speakers with a voice assistant. My Vibe on smart speakers is our test platform for SONA. Playback begins without the user specifying an artist, genre, mood, or other preference. This creates a pure-recommendation setting for measuring recommendation quality; the surface is also one of the largest by listening time.

Several properties distinguish the music domain from short-video and e-commerce recommendation and shape the design of SONA. Consumption is largely passive: the stream plays continuously while the user attends to something else, so feedback arrives as plays, skips, and occasional likes rather than deliberate choices. Repeated listens are the norm rather than the exception — returning to a familiar track is often exactly what the user wants, so repetition is signal to be modeled, not redundancy to be filtered out. These two properties combine into the central tension of the domain: familiar tracks reliably sustain time spent, while discovery of new favorites pays off over a longer horizon, so the recommender must continuously balance time spent against novelty.

SONA formulates next-track recommendation as conditional sequence generation. The encoder consumes typed engagement events — including plays, skips, likes, and dislikes — together with their track identities and request context. Following the Semantic ID formulation of Rajput et al. [22] and its production successors [8, 18, 39, 41], the decoder emits each recommendation as a short tuple of discrete Semantic IDs (SIDs) (Section 3.1). Each tuple is expanded into the catalog tracks that share it, and the Ranking Module ranks the resulting candidates (Section 3.4). The two modules share a user encoder that is evaluated once per request, allowing generation and ranking to run in a single served model. Across training and serving, inputs are constructed from logged event fields and learned Semantic IDs; no component uses hand-crafted features.

The Ranking Module’s training supervision comes from a Teacher Ranker (Section 5.1), a sequential transformer over the same event vocabulary as the encoder with a cross-attention candidate scorer. We train it with a two-stage initial recipe: next-item-prediction pre-training learns the sequence representation, and multi-head ranking fine-tuning adapts it to engagement objectives; the trained model is the distillation teacher. Few production reports describe this pre-training — Argus [12] and GPSD [30] among them — and our ablations make it the load-bearing stage: it is what lets the teacher alone, one transformer with no hand-engineered features, replace a ranking model driven by hundreds of features, among them large-scale transformer models such as Argus [12] and target-attention scorers (Section 7.5).

The served model is connected to a continuous online-training loop: an emitted listening event is aggregated into its session over a window of 15 minutes, flows through real-time processing into a queue consumed by the training process, and fresh model weights are delivered to production every 10 minutes. End to end, the latency from event emission to an updated production model is 45 minutes at the median and 60 minutes at the 99th percentile (Section 6.2).

We validate SONA both offline and online. For offline validation, we use *Teacher Recall@k* to measure how closely the Ranking Module preserves the teacher’s ordering over the same generated candidates (Sections 7.1 and 7.4). We then evaluate intermediate and final serving configurations in five online experiments presented in conceptual order. In the final experiment in My Vibe on smart speakers, SONA increases Active Users by +4.53%, Total Listening Time by +6.30%, and Likes by +11.42% (Section 7.5). On the primary Active Users metric, this uplift is $2.35\times$ the increment previously delivered by Argus [12], the strongest model previously deployed there.

2 SONA Overview

2.1 Design Philosophy

A small number of ideas guided us throughout the work on SONA — in choosing the formulation, weighing alternatives, and deciding what to ship. We state each idea and the reasoning behind it below; the rest of the report describes the architecture, training recipe, and infrastructure they shaped.

One model for generation and ranking. Large language models demonstrated that a single model trained end-to-end on a large corpus can absorb functions previously split across specialized components, and single-model generative recommenders carried this recipe into production recommendation [8, 18, 41]. Motivated by these results, we set out to explore the potential of a single-model recommender in the music domain.

One user representation, shared across tasks. We are guided by the idea that a single user representation should serve every recommendation task built on top of it: the encoder cost is paid once per request, and the resulting representation is reused for both generation and ranking. Recent industry deployments show that candidate generation and item-level ranking can share one backbone at production scale [13, 14]. We therefore adopt Gryphon [28], a unified generation-and-ranking design, as the architecture of SONA (Section 3.5).

No hand-engineered features. Across machine-learning domains, models given scale and data have matched pipelines that encode domain structure by hand: image transformers reach the accuracy of architectures built around convolutional inductive bias once pre-training data is plentiful [5], and language-model quality follows smooth scaling curves in parameters and data [11]. Recommendation reproduced the pattern: sequence models over raw engagement events scale the same way [12], and transformer rankers over those sequences reach the quality of feature-rich production rankers [2, 39]. We therefore build SONA and its teacher on logged event fields and learned Semantic IDs alone, and set out to explore how far that reaches in music recommendation.

Dense supervision for ranking without hand-engineered features. Next-token prediction and action prediction supply supervision at very different densities: next-token prediction produces a training target at every position of every sequence, while explicit feedback events are sparse. A strong ranking model without hand-engineered features must recover from data the structure such features would otherwise encode, and this lack of built-in inductive bias calls for training on large-scale datasets. Autoregressive training over engagement sequences is widely adopted as the way to leverage datasets of this scale [7, 39]. We follow this recipe, and we initially train the Teacher Ranker in two stages — next-item-prediction pre-training followed by multi-head ranking fine-tuning [12]; our ablations find the pre-training stage crucial for the teacher’s quality (Table 7.4).

Compress a year of history into a teacher, distill it densely into the student. For encoder-decoder models trained with next-token prediction on chronologically organized data, each training sample carries the user’s full interaction history, so both the dataset’s size on disk and the compute spent encoding it grow far faster than the number of training samples: every sample re-encodes the history from scratch to supervise a handful of new targets. This cost makes such datasets unsuitable for a multi-month span of dates. We therefore train a large-scale ranking model autoregressively on a dataset spanning a year of engagement logs, compressing that history into its weights, and distill its knowledge densely into the Ranking Module, which is trained jointly with the next-token-prediction task on a dataset covering a much shorter range of dates (Sections 4.2 and 5.1).

Aligning generation and ranking through Rollout Distillation. We use beam-search rollouts from the current decoder as the candidate source for Rollout Distillation (Section 4.2). The frozen teacher scores the generated candidates, and the Ranking Module learns to reproduce those scores. Because the rollouts come from the current decoder distribution, the distillation signal follows the candidate distribution the model produces at that point in training. The decoder and Ranking Module also share the user encoder, so this signal shapes the representation that conditions both tasks. In this way, Rollout Distillation aligns generation and ranking inside the joint training objective.

2.2 Overall Framework

Figure 2.1 summarizes SONA’s serving path and training signals. At serving time, the encoder processes the user’s chronological event history once, producing a representation shared by the decoder and Ranking Module. Following the generative-recommendation paradigm, the decoder autoregressively produces a beam of short Semantic ID tuples [8, 18, 22, 39, 41]. A catalog index maps each tuple to one or more tracks, and the Ranking Module uses the shared encoder states to score and order the resulting candidates.

During training, the semantic tokenizer converts each target item into its Semantic ID sequence, providing the next-token-prediction objective for the decoder. The frozen teacher ranker scores decoder-generated candidates and provides distillation targets for the Ranking Module. Both losses back-propagate through the shared encoder, so its representation learns from generation and ranking. In the final SONA configuration, the tokenizer and teacher are used only for training; serving consists of the encoder, decoder, deterministic SID-to-item mapping, and Ranking Module. The teacher was temporarily added to limited-traffic treatments in Online Experiments 1–4 (Section 7.5).

Table 2.1 summarizes the role and lifecycle of each component.

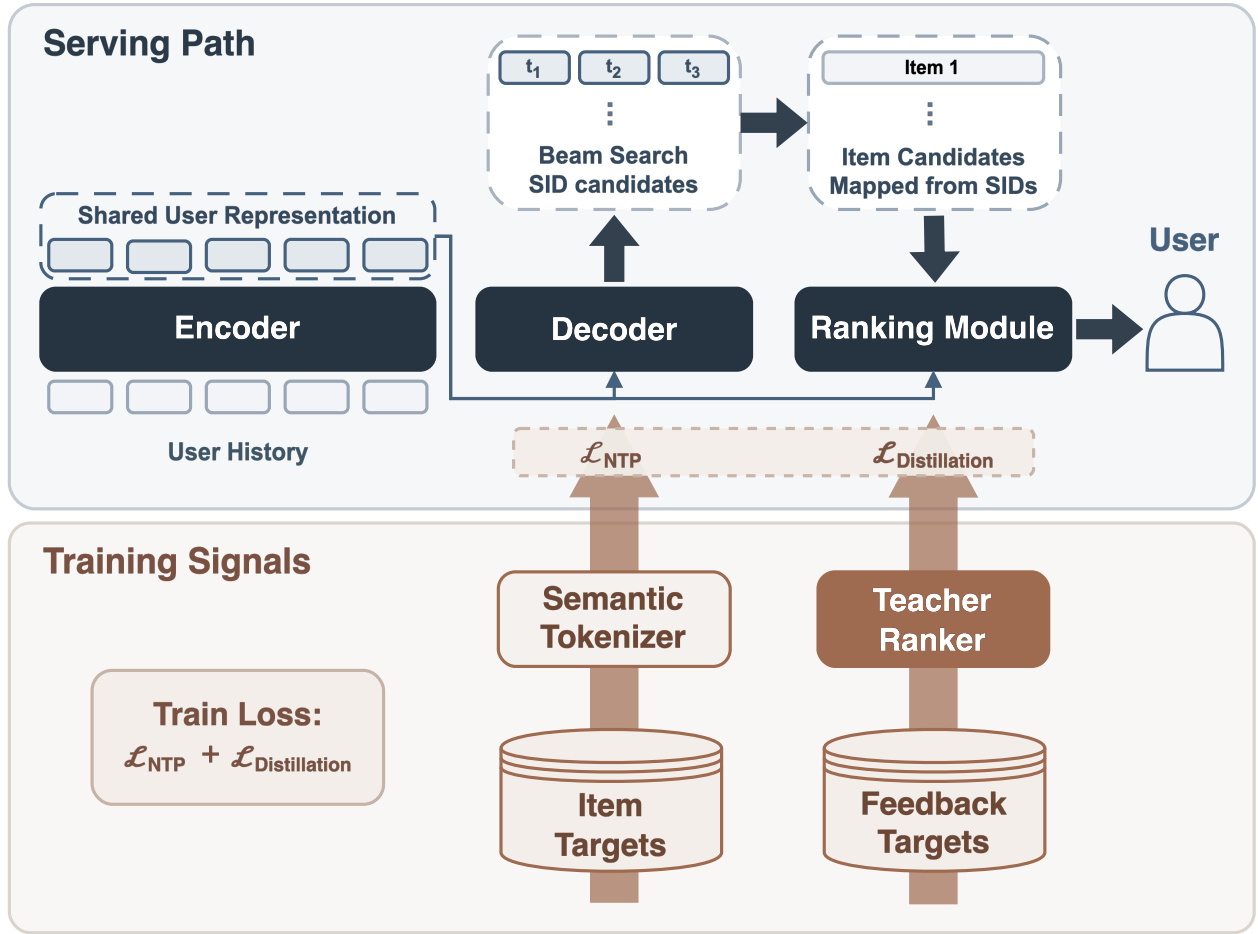


Figure 2.1. SONA architecture and data flow.

3 SONA Architecture

This section details the components of SONA: the semantic tokenizer, user-event representation, shared encoder with History Compression, autoregressive decoder, and Ranking Module.

Component	Consumes	Produces / role	Optimization and lifecycle
Semantic Tokenizer	Dense item representations	A fixed SID tuple for each item, defining decoder targets and the SID-to-item index	Offline; fixed during training and serving
Shared User Encoder	Chronological user events	Hidden states shared by generation and ranking	Training and serving; receives gradients from both next-token and distillation objectives
Semantic ID Decoder	Encoder states and preceding SID tokens	Beam of candidate SID tuples	Next-token training; served
SID-to-item Mapper	Generated SIDs and catalog index	Catalog candidates	Deterministic; no learned parameters
Ranking Module	Encoder states and candidate item / SID features	Multi-feedback scores and final ordering	Teacher-distilled; served
Teacher Ranker	Event histories and candidate items	Item-level distillation targets	Separately trained and frozen

Table 2.1. Final SONA serving components and training-only supervision.

3.1 Semantic Tokenizer

Generative recommenders face large and continuously growing item spaces, which makes the direct generation of atomic item identifiers computationally and architecturally infeasible. Items are therefore tokenized into short coarse-to-fine sequences of discrete codes — *Semantic IDs* — drawn from a small fixed vocabulary [22]. This keeps the output space compact and stable as the catalog grows, and items with similar content or behavioral patterns share leading codes, which transfers knowledge across related tracks and improves generalization to newly added items [18].

The tokenizer realizes this scheme for the music catalog: it assigns every track a fixed tuple of 3 Semantic IDs, drawn from 3 codebooks of 32,000 entries each. These tuples are the decoder’s output vocabulary and define the SID-to-item index used at serving. The tokenizer is built offline in three stages (Figure 3.1): content features from the hidden states of a frozen multimodal LLM, a refinement transformer trained on collaborative pairs, and a residual quantizer.

For each eligible track, the frozen Qwen2.5-Omni [21] encodes the mel-spectrogram of the first 90 seconds of audio together with a textual prompt carrying the title, artists, tags, and additional metadata. Its last-layer states — the track’s *content features* — are refined by a four-layer transformer with four attention heads and model dimension 512, following QARM-style pipelines [15], and mean-pooled into a single 256-dimensional *item embedding* z for the track. Once the transformer is trained and frozen, z is the only representation the quantizer receives.

Collaborative refinement. Content features alone leave substitutability implicit: items that listeners treat as interchangeable are not necessarily neighbors in the content embedding space [9, 32]. The refinement transformer is therefore trained on collaboratively related track pairs, so that tracks with similar listening patterns receive nearby item embeddings — and, after quantization, shared code prefixes. Its loss combines two terms,

$$\mathcal{L} = \mathcal{L}_{\text{InfoNCE}} + \lambda_{\text{align}} \mathcal{L}_{\text{align}}.$$

A batch of m collaborative pairs yields $2m$ refined item embeddings, written z_a for track a of the batch. Each serves once as the anchor; its paired track z_{a+} is the positive and the remaining $2m - 2$ embeddings are negatives:

$$\mathcal{L}_{\text{InfoNCE}} = -\frac{1}{2m} \sum_{a=1}^{2m} \log \frac{\exp(\text{sim}(z_a, z_{a+})/\tau)}{\sum_{b \neq a} \exp(\text{sim}(z_a, z_b)/\tau)},$$

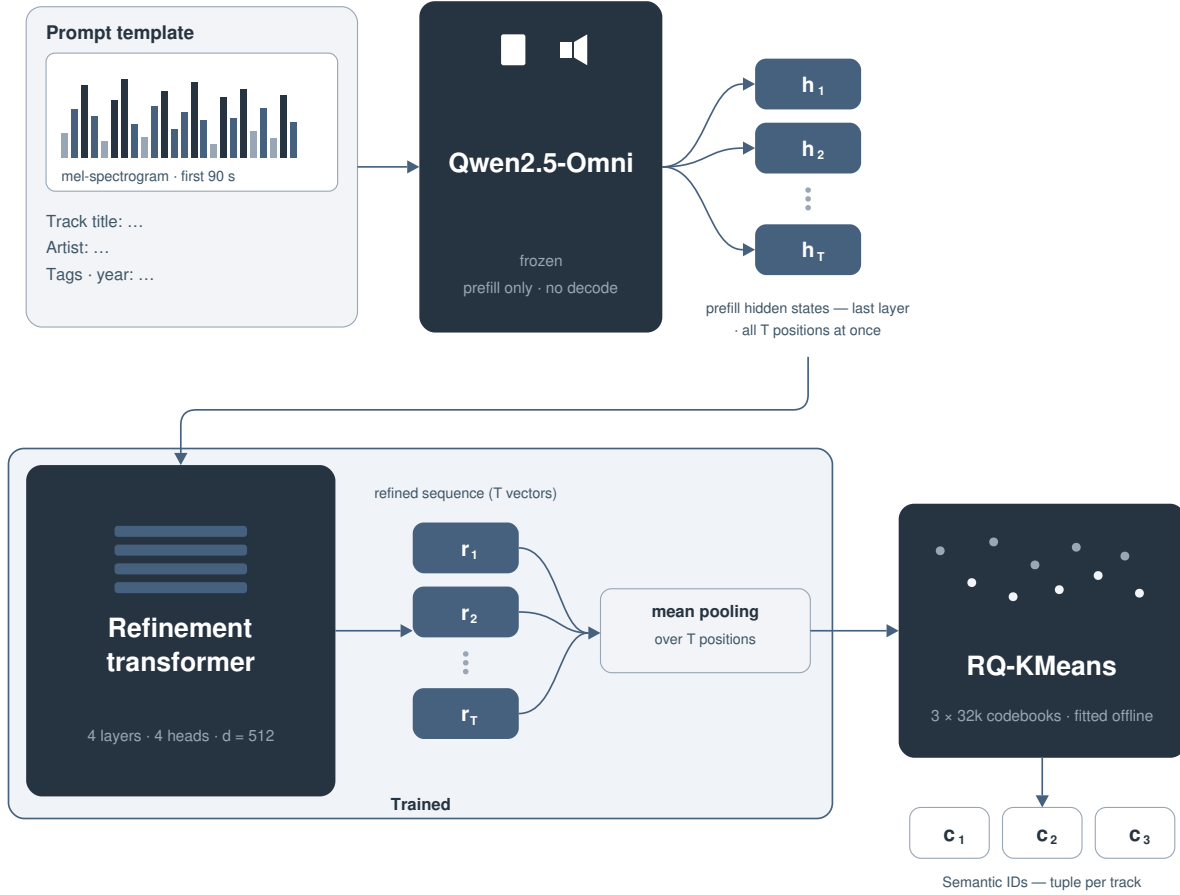


Figure 3.1. Tokenization pipeline. The frozen Qwen2.5-Omni runs in prefill-only mode — no decoding.

with cosine similarity sim and temperature τ . The alignment term keeps each item embedding close to $\bar{h}_a$, the mean of the same track’s content features,

$$\mathcal{L}_{\text{align}} = \frac{1}{2m} \sum_{a=1}^{2m} \|z_a - \bar{h}_a\|^2,$$

with weight $\lambda_{\text{align}} = 0.1$.

Collaborative pairs. Pairs are mined from two sources. The *target-wise* stream collects co-served item sets from the production recommender — groups of items presented together in one recommendation response. The production recommender does not place tracks by the same artist in a single response, so these co-occurrences are cross-artist by construction. The *NPMI* stream scores co-listening of two tracks i and j with normalized pointwise mutual information,

$$\text{NPMI}(i, j) = \frac{1}{-\log p(i, j)} \log \frac{p(i, j)}{p(i) p(j)},$$

where $p(i, j)$ is the probability that i and j co-occur in a user’s listening log over a three-month window and $p(i)$, $p(j)$ are the marginals. Only high-NPMI pairs in which both tracks belong to the same artist are kept, so the two streams are disjoint. Table 3.2 summarizes the sources; the streams are concatenated and shuffled before training.

Pair source	Mining window	Pairs
Target-wise (co-served, cross-artist)	3 weeks	220–240M
NPMI (co-listening, same-artist)	3 months	20M

Table 3.2. Collaborative pair sources for the tokenizer InfoNCE objective.

Residual quantization. The refined item embedding z is mapped to its Semantic ID tuple by residual K-means: 3 levels of 32,000 centroids, initialized with k-means++ and assigned by Euclidean distance. Each level quantizes the residual of the previous one, so the tuple is a coarse-to-fine path through the codebooks. The comparisons that selected this configuration — the codebook size and the item representation supplied to the quantizer — are reported in Section 7.2.

3.2 User History Representation

The encoder input is built from the user’s logged engagement history. For a request, the history is the chronological sequence

$$u = (e_1, \dots, e_T), \quad T \leq T_{\max},$$

where each e_t is one past engagement event. The same selection rule is applied in training and serving. Histories shorter than $T_{\max}$ are padded with a single learned embedding shared across the missing positions.

Event embedders. Each event e_t is embedded by three embedders, one per feature group: an *item embedder* over item identity, a *context embedder* over the request surface, and a *feedback embedder* over the observed response. Table 3.3 lists the source signals each embedder reads and the transformations that produce its inputs. Every input is a logged attribute; no hand-engineered features are used.

Embedder	Source signals	Construction
Item	track ID, artist ID, track duration	hash track ID three times and artist ID twice into one shared embedding table; bucket and embed track duration
Context	smart-speaker flag, organic-feed flag	embed both request-surface flags
Feedback	like flag, played time	embed the like flag; divide played time by track duration into the completion ratio, then bucket and embed it

Table 3.3. Historical-event features and their transformations before the three input embedders.

Continuous features. Two continuous quantities reach the embedders, and both are bucketed and embedded rather than used as raw scalars: track duration, and the completion ratio

$$\rho_t = \frac{p_t}{\max(d_t, \epsilon)},$$

where p_t is played time and d_t track duration.

Event token. The item embedder concatenates its five hashed identity features with the track-duration embedding and projects the result to $\mathbb{R}^d$; the context and feedback embedders embed each of their attributes directly at width d and sum them. The event token is the sum of the three embedder outputs:

$$x_t = \text{ItemEmb}(e_t) + \text{ContextEmb}(e_t) + \text{FeedbackEmb}(e_t).$$

Encoder input. A learned CLS token is prepended as a global summary position, giving

$$X = [x_{\text{CLS}}, x_1, \dots, x_T].$$

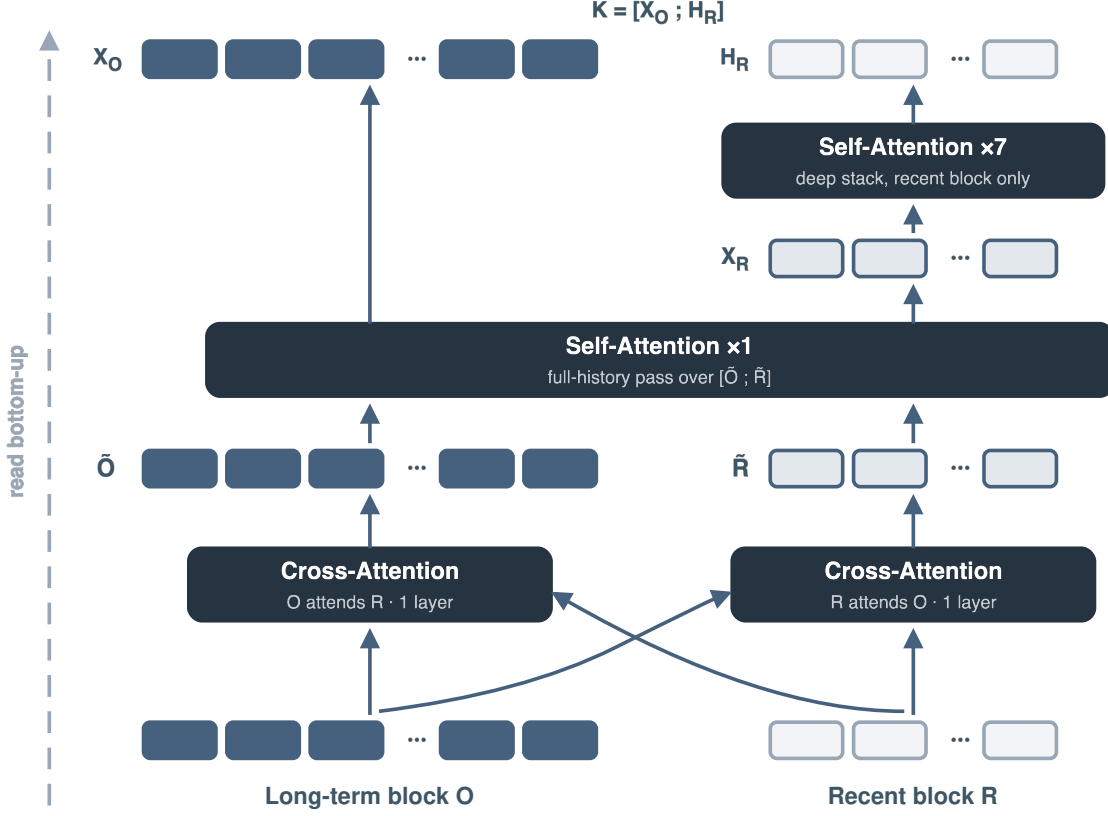


Figure 3.2. History Compression, read bottom-up. Dark: the long-term block O (n_o events); light: the recent block R (n_r events).

3.3 Encoder

For a full encoder, let H denote the hidden-state matrix computed from the event sequence X of Section 3.2,

$$H = \text{Enc}_\theta(X), \quad H \in \mathbb{R}^{(T+1) \times d}.$$

The decoder (Section 3.4) and Ranking Module (Section 3.5) consume a shared encoder memory K , computed once per request. The symbols distinguish the encoder’s direct output from the memory exposed to downstream modules: $K = H$ for a full encoder, whereas History Compression does not construct a full-depth H over all positions and instead forms $K = [X_O; H_R]$. Thus K is the common downstream interface in both cases.

History Compression. Attention cost grows quadratically with history length, and keeping long histories affordable under serving-latency budgets is a recurring constraint for industrial user models [34, 43]. SONA therefore spends its depth unevenly. The history of up to N events is split into a *recent* block R of the n_r most recent events and a *long-term* block O of the preceding $n_o = N - n_r$ events, and the encoder applies four stages (Figure 3.2). History Compression is used only in SONA; the Teacher Ranker (Section 5.1) encodes its history in full.

1. **Cross-attention.** Two single-layer cross-attention blocks let each block read the other — O attends to R and R attends to O — giving enriched blocks $\tilde{O}$ and $\tilde{R}$.
2. **Full-history pass.** One self-attention layer runs over the concatenation $[\tilde{O}; \tilde{R}]$, producing block representations X_O and X_R . This is the only stage that mixes information across the whole history.
3. **Deep stack, recent block only.** A stack of 7 self-attention layers, which carries most of the encoder’s capacity, refines X_R over the n_r recent events only, yielding H_R .
4. **Concatenated read-out.** Modules that cross-attend to the encoder read

$$K = [X_O; H_R],$$

the shared encoder memory: all N positions are visible downstream, while only the recent n_r carry the deep encoding.

Cost. A full L -layer encoder over N events spends $\Theta(L N^2 d)$ in attention; ours pays the deep cost only over the recent block, reducing the attention cost to $\Theta((L n_r^2 + N^2 + 2 n_o n_r) d)$. The saving comes from where capacity is spent, not from discarding history. We adopt $N = 8192$ and $n_r = 2048$; this preserves most of the downstream performance of a full transformer over the same history at about half its inference cost (Section 7.4).

Block design. All transformer blocks use RMSNorm [40] in a pre-normalization layout [35], SwiGLU feed-forward blocks [23], and RoPE positional embeddings [26]; the complete model cards are given in Tables B.4 and B.5.

3.4 Decoder

The tokenizer (Section 3.1) maps each catalog track v to an L -level Semantic ID tuple

$$s(v) = (s_1(v), \dots, s_L(v)), \quad L = 3.$$

The decoder is a shallow autoregressive transformer over the codebook vocabulary (Table B.5). It starts from BOS and generates the tuple one level at a time, conditioned on the encoded history:

$$p_\theta(s_1, \dots, s_L \mid u) = \prod_{\ell=1}^L p_\theta(s_\ell \mid s_{<\ell}, K).$$

From tuples to tracks. At serving time, constrained beam search masks invalid prefixes with the tokenizer trie, so each completed tuple maps to at least one catalog track. The codebook is not collision-free: a typical SID tuple is shared by a small number of catalog tracks, as is intrinsic to the Semantic ID formulation of Rajput et al. [22]. Each completed tuple is therefore expanded into the full collision class of tracks that quantize to it, and the Ranking Module orders the expanded candidates (Section 3.5).

3.5 Ranking Module

The Ranking Module scores each candidate that beam search proposes against the same encoder memory K the decoder attends to, following the unified generation-and-ranking design of Gryphon [28]: Semantic ID generation proposes candidates with high recall, and the Ranking Module orders them (Figure 3.3). The history is encoded once, so scoring adds no second encoder pass.

Candidate representation. Each candidate is represented by three kinds of features: its item ID, its Semantic ID codes $c_1, \dots, c_L$ taken independently, and their prefix n -gram features [45]. Following the unified-embedding scheme of Coleman et al. [3], every feature is mapped by several independent hash functions into the embedding table shared with the encoder (Section 3.2). The looked-up rows are concatenated and projected into a single dense vector e_i .

Scoring. The Ranking Module largely follows the decoder architecture, with the candidate embedding e_i as its input. A stack of pre-norm cross-attention blocks refines this embedding against the encoded history [36],

$$\tilde{e}_i = \text{CrossAttnBlocks}(e_i, K).$$

A feed-forward scoring layer maps the result to n head scores,

$$\mathbf{s}_i = \text{Head}(\tilde{e}_i) \in \mathbb{R}^n,$$

one per Teacher Ranker head, trained by distillation from the teacher jointly with the decoder’s next-token objective (Section 4.2).

Combined score. A fixed weighted combiner collapses the n head scores into a single scalar, and the expanded candidates are ordered by it to produce the final recommendation list.

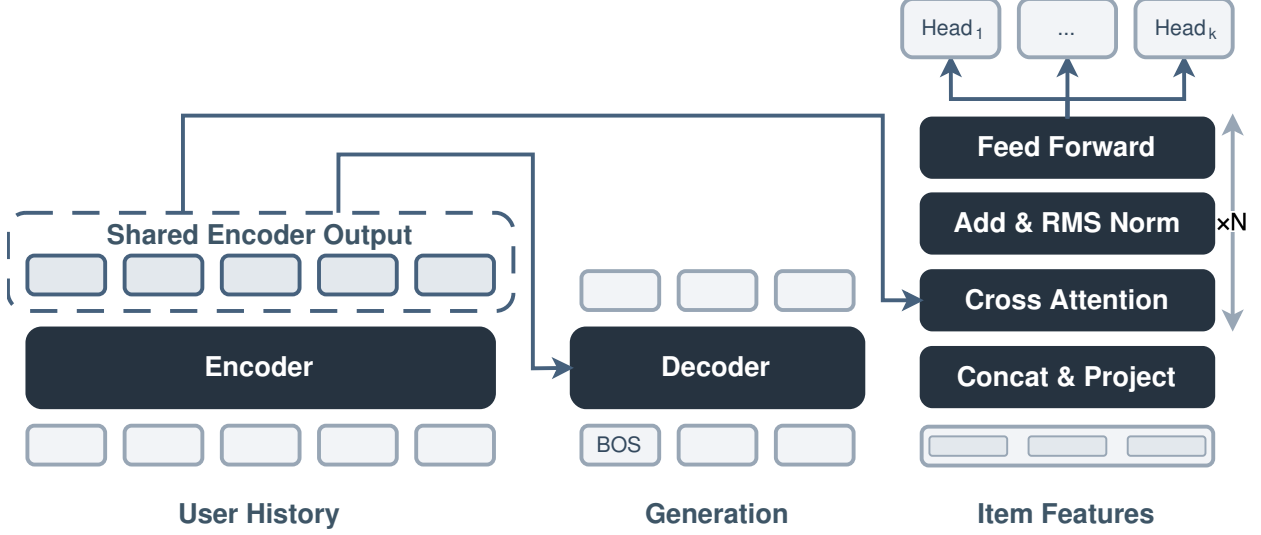


Figure 3.3. Unified generation-and-ranking architecture: the decoder and the Ranking Module attend to the same encoded user history.

4 SONA Training

4.1 Dataset Design

The dataset is request-level: its unit is a single served recommendation request. A sample carries the user’s chronological event history up to the request and the *impressions* — the items served in response, stamped with the request timestamp and the feedback they subsequently received. History events carry exactly the logged attributes of Table 3.3. Samples with an empty history are dropped, and the sample stream inherits the traffic gates of the online loop (Section 6.2).

Supervision. All positive items of a request supervise the model against the shared user history: the tokenizer maps each impression to its Semantic ID tuple, and the positives form the packed target set $C^+(u)$ of the next-token objective — several targets per request, one encoder pass (Section 4.2). The full impression list is the impression candidate source of the distillation objective, and beam-search rollouts supply its second source. Impressions absent from the tokenizer’s catalog snapshot are masked out of the loss.

Time ordering. The dataset is organized chronologically. For initial training, target requests are split into day-level tables and the trainer consumes them from oldest to newest. Once the model enters online training (Section 6.2), it consumes the request stream directly in event order, with no splitting into timestamp chunks. In both stages training sweeps time forward, and every sample conditions only on events that precede its request.

4.2 Training Framework

SONA is trained with a joint generation-and-ranking objective (Figure 4.1). The decoder is supervised by next-token prediction (NTP) on the *positive* impressions of each request only — the items the user engaged with, mapped to their Semantic ID tuples (Section 4.1). The encoder runs once per request, and every positive is decoded as an independent teacher-forced sequence that cross-attends to the shared representation. The Ranking Module is supervised by distillation from the frozen Teacher Ranker (Section 5.1): the teacher exposes n engagement scores per candidate, and the Ranking Module regresses a matching head onto each with an element-wise mean absolute error. Both objectives back-propagate into the shared encoder.

Distillation sources. The distillation objective draws (user, candidate) pairs from two complementary sources: current-decoder rollouts and logged impressions. For each user history, beam search generates B SID tuples using the current decoder. The generated SIDs are resolved into catalog items; successfully resolved items form the rollout candidate set $\mathcal{B}$, while unresolved entries are excluded from Ranking Module

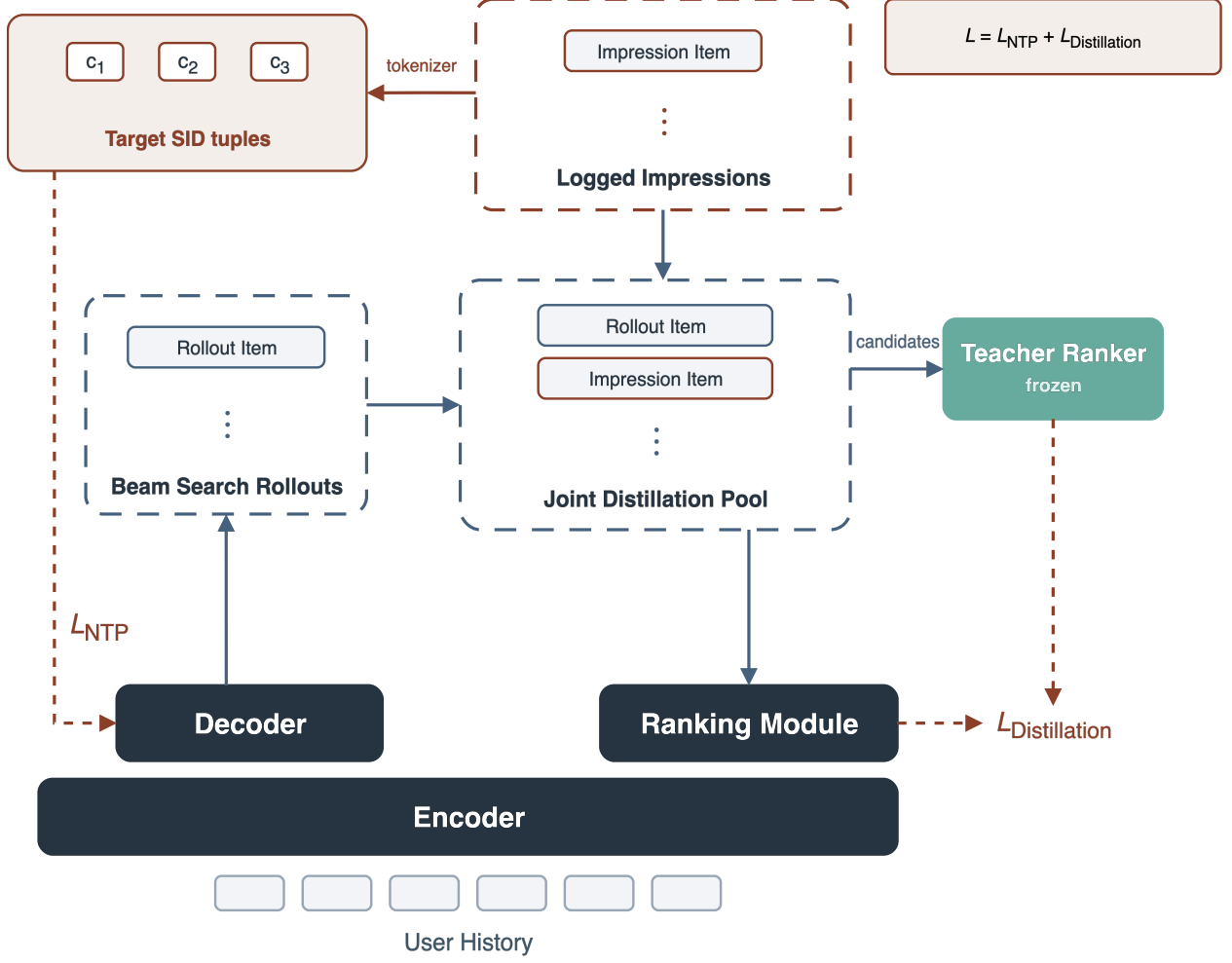


Figure 4.1. Joint training of the decoder ($\mathcal{L}_{\text{NTP}}$) and the Ranking Module ($\mathcal{L}_{\text{Distillation}}$, regression toward the frozen Teacher Ranker) on the shared encoder.

scoring and loss. The rollout set is complemented with the request’s logged impressions $\mathcal{I}$, which broaden teacher supervision to items exposed in production traffic. Writing r_c^h for the teacher’s head- h score of candidate c and $\hat{r}_c^h$ for the Ranking Module’s, each source contributes a per-head mean absolute error,

$$\mathcal{L}_{\text{rollout}} = \sum_{h=1}^n \frac{1}{|\mathcal{B}|} \sum_{c \in \mathcal{B}} |\hat{r}_c^h - r_c^h|, \quad \mathcal{L}_{\text{impression}} = \sum_{h=1}^n \frac{1}{|\mathcal{I}|} \sum_{c \in \mathcal{I}} |\hat{r}_c^h - r_c^h|.$$

The joint loss sums the generative and the two distillation terms,

$$\mathcal{L} = \mathcal{L}_{\text{NTP}} + \mathcal{L}_{\text{rollout}} + \mathcal{L}_{\text{impression}}.$$

Combined ranking score. At serving, the n distilled head scores are collapsed into one scalar by the fixed weighted combiner of Section 3.5. This score reorders the beam candidates; it enters no training loss.

Training stages. *Initial training* starts from random initialization and consumes the dataset chronologically in day-level chunks — days in order, requests shuffled within a day — under a linear warmup-and-decay learning-rate schedule. *Online training* then consumes requests as they arrive (Section 6.2), at a constant learning rate, the terminal rate of the decay.

Optimization. Training uses AdamW with weight decay and gradient clipping; optimizer and schedule hyperparameters are listed in Section B. FSDP shards the trainable parameters, while the frozen teacher stays replicated in bfloat16 on every rank, adding no all-gather cost.

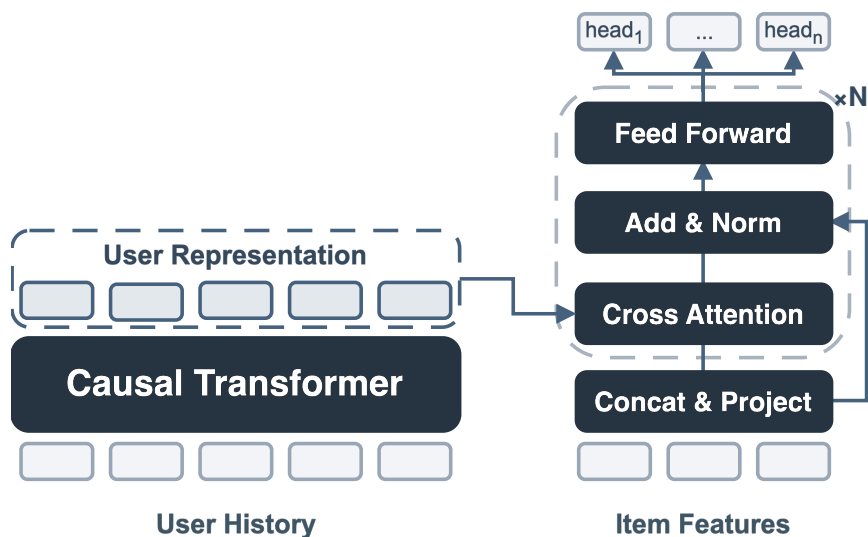


Figure 5.1. Teacher Ranker architecture: a history encoder over the typed event sequence (left) and a candidate scorer that cross-attends to its hidden states (right).

5 Teacher Ranker

5.1 Architecture

The teacher ranker is a large ranker without hand-engineered features, used as the distillation teacher of the Ranking Module (Section 4.2). Freed from the latency and throughput budgets of the final model, it is larger and attends the full user history. It surpasses the production ranker in online experiments (Section 7.5). Figure 5.1 shows the architecture.

The teacher combines a deep user-history encoder over the typed event sequence with a candidate scorer. As in SONA’s Ranking Module (Sections 3.3 and 3.5), the scorer largely follows the decoder architecture: it takes a candidate embedding as input and refines it against the encoder’s hidden states through multi-layer cross-attention [27, 39]. Two differences separate it from the served pair. First, its encoder is *causal*: every position carries the user state at that point of the timeline, which unlocks autoregressive training and timeline-packed supervision (Section 5.3). Second, it does not use History Compression — the deep stack attends the full user history at every layer.

Heads. A set of engagement heads reads from the scorer’s output, each an MLP that narrows to one scalar. One head supplies the primary ranking signal, trained to order events by a graded engagement target; the others are auxiliary feedback heads, restored to calibrated logits by a learned per-head scale and bias (Section 5.3). The complete model card is given in Table B.2.

5.2 Dataset Design

The Teacher Ranker follows a two-stage initial recipe — pre-training with next-item prediction (NIP), then multi-head ranking fine-tuning — followed by continuous refresh (Section 5.3). Both initial stages and the refresh runs draw on the same dataset shape. The dataset is timeline-level: its unit is a span of one user’s chronological event timeline, carrying the same logged attributes as Table 3.3. Pre-training samples consist of the timeline alone, and every event in it is a next-item target. Fine-tuning samples additionally carry the impressions served along the timeline, with their graded engagement feedback. Refresh samples keep the fine-tuning format and are built continuously from newly logged sessions, restricted to the surface the model serves.

Autoregressive consumption. Each sample is consumed with all of its targets at once. The causal encoder provides, at every position, the user state at that point of the timeline, and the cross-attention scorer reads each target through a custom mask that restricts it to the history preceding its anchor (Section 5.3). One

encoder pass therefore supervises every target the timeline carries, where the request-level dataset of Section 4.1 runs the encoder once per request.

Data span. Packing a timeline’s worth of targets into one encoder pass sets the affordable window: the teacher is trained on a *year* of target events, where the joint run of Section 4.1 consumes several weeks. Compressing that year into the teacher’s weights, and distilling it densely into the Ranking Module, is the design idea of Section 2.1.

5.3 Training Framework

The Teacher Ranker is trained autoregressively over the user’s chronological engagement-event sequence, using the recipe of Section 5.2: NIP pre-training learns the user encoder, multi-head ranking fine-tuning specializes it to the signal the Ranking Module consumes, and continuous refresh keeps the fine-tuned teacher current. The contribution of the pre-training stage is ablated in Table 7.4.

Pre-training. The history encoder is pre-trained on next-item prediction, following the autoregressive recipe of Argus [12]: every position of the timeline predicts the next engagement from its causal user state. The objective is an InfoNCE contrastive loss over in-batch negatives mixed with a uniformly sampled tail-negative pool, with LogQ correction [38] for the popularity bias of in-batch sampling. The stage keeps the encoder parameters; the prediction head is discarded.

Fine-tuning. The pre-trained encoder is warm-started, the candidate scorer is attached with random initialization, and the model is trained as a multi-objective ranker. The primary objective is *pairwise*. Each impression is mapped to a graded engagement scale — like > play > skip > dislike — and the pairwise head is trained to order temporally adjacent impressions with different grades by a logistic loss on its score difference. Over the set $\mathcal{P}$ of ordered adjacent pairs (i, j) in which i carries the higher grade,

$$\mathcal{L}_{\text{pair}} = -\frac{1}{|\mathcal{P}|} \sum_{(i,j) \in \mathcal{P}} \log \sigma(\text{score}_{\text{pair}}(i) - \text{score}_{\text{pair}}(j)). \quad (5.1)$$

Pairs form both within one recommendation request and across the boundary of two adjacent ones, and only impressions served in recommendation requests are eligible: fine-tuning stays on the distribution the ranker is used on. The auxiliary objectives are *pointwise*: each remaining head fits one binary feedback target with binary cross-entropy $\mathcal{L}_{\text{BCE}}^k$ on its calibrated logit. The heads are trained jointly against the sum of the pairwise and pointwise losses,

$$\mathcal{L}_{\text{teacher}} = \mathcal{L}_{\text{pair}} + \sum_k \mathcal{L}_{\text{BCE}}^k.$$

Each impression group attends the history preceding its anchor through the custom cross-attention mask of Section 5.2, implemented as a dedicated Triton kernel [29]. Fine-tuning extends the attended history well beyond the pre-training length, so the scorer sees the long-range context available at scoring time.

Continuous refresh. After fine-tuning, the teacher is rolled forward to the present and then refreshed daily on newly logged sessions. These continuation runs narrow the target pairs to the recommendation surface whose generative model the teacher supervises, specializing its signal to that surface.

Optimization. All stages use Adam in two parameter groups — encoder and heads — with a linear-warmup schedule, following Khrylchenko et al. [12], under bfloat16 autocast. Sequence lengths, negative counts, and learning rates are listed in the model card (Table B.2).

6 Training and Inference Infrastructure

6.1 Training Optimizations

SONA trains generation and ranking jointly and evaluates the frozen teacher on the generated candidates in the same step. We co-locate all three computations on one set of GPU workers.

Keep the data path off the critical path. After network fetch and CPU preprocessing, the input pipeline buffers prepared batches for the GPU to consume. Catalog item hashes are computed by the asynchronous input pipeline while training proceeds. Data preparation, hash computation, and transfer can therefore overlap GPU model execution.

Optimize GPU execution. We use bfloat16 mixed precision to shorten the processing time of each batch. FSDP [44] shards the trainable model state. FlashAttention [4] supplies efficient attention kernels, and `torch.compile` optimizes the dense model graph.

Preserve numerical stability. QK normalization [10] bounds attention logits, and gradient clipping limits outlier updates. We also tune the learning-rate schedule carefully, including warmup before the peak rate and decay during training. These controls are important when combining mixed precision, long histories, and the joint generation–ranking objective.

6.2 Online Training Infrastructure

Figure 6.1 shows the online-training data path. Updates to the served model come from a continuous loop: user actions stream into a real-time event-processing system, are aggregated per user into sessions, joined with the inference-time context, and emitted as training samples; a GPU trainer consumes those samples from a message queue and updates the model through the joint objective of Section 4.2 — next-token prediction and Ranking Module distillation; the updated model is checkpointed and synced back to serving every 10 minutes. The Teacher Ranker runs its own training pipeline (Section 5.3) and is refreshed into the trainer every 24 hours. Measured from when a user event is emitted until the model update trained on it reaches serving, the end-to-end latency is 45 minutes at the median and 60 minutes at the 99th percentile. The session-attribution window of 15 minutes described below and the deployment cadence of 10 minutes account for most of the median budget; event processing, queueing, training, and artifact upload occupy the remainder.

Session aggregation. The event stream is partitioned by user and consumed by a stream processor whose per-user state is backed by a sharded distributed key–value store. The processor groups consecutive events into *sessions* closed by an idle gap of a few minutes and by a minimum attribution window of 15 minutes; the attribution window also serves as a correctness buffer that lets the inference log catch up to the events that triggered it (below). Because the stream processor wakes a worker only when a new message arrives, session timeouts are driven by internal timer messages that the worker enqueues for itself.

Traffic filtering. Emitted sessions pass through several gates: standard fraud and bot-activity filters; a history-length gate that keeps the gradient from being dominated by very short histories; and a model blacklist that drops traffic served by probe or evaluation models, which would otherwise close a feedback loop with the training data.

Concurrent sample streams. Session aggregation is configured per model. Several models are registered at a time, each with its own session definition, traffic gates, and sample format, and each emitting its own stream. Several online updates therefore run side by side over one pass of the user events — one feeding the deployed model, the others feeding candidate variants under evaluation.

User profile storage. The user history that serving and sample construction both read is held in two stores behind one logical profile. A key–value store holds the long history and is rebuilt on a batch cadence, so its tail trails live traffic. A real-time profile, written directly from the event stream, holds the last 72 hours of events, capped at 1000 events per user. A read merges the two by timestamp and deduplicates the overlap, resolving collisions in favor of the batch side. The window on the real-time side is what keeps the split worthwhile: the stream processor holds and updates a bounded tail per user instead of the full 8192-event history, which is what keeps its state size and write load affordable at the request rate of the surface.

History join via the inference service. A training sample is meaningful only if the user history it shows the model matches the history the inference saw when the candidate was produced; otherwise the gradient is computed against a state the model never observed. Two design choices make this tractable. First, the inference service writes an *inference log* entry to a replicated key–value store at the moment of serving, keyed by the request identifier and carrying the history-cutoff timestamp at inference time; the session aggregator

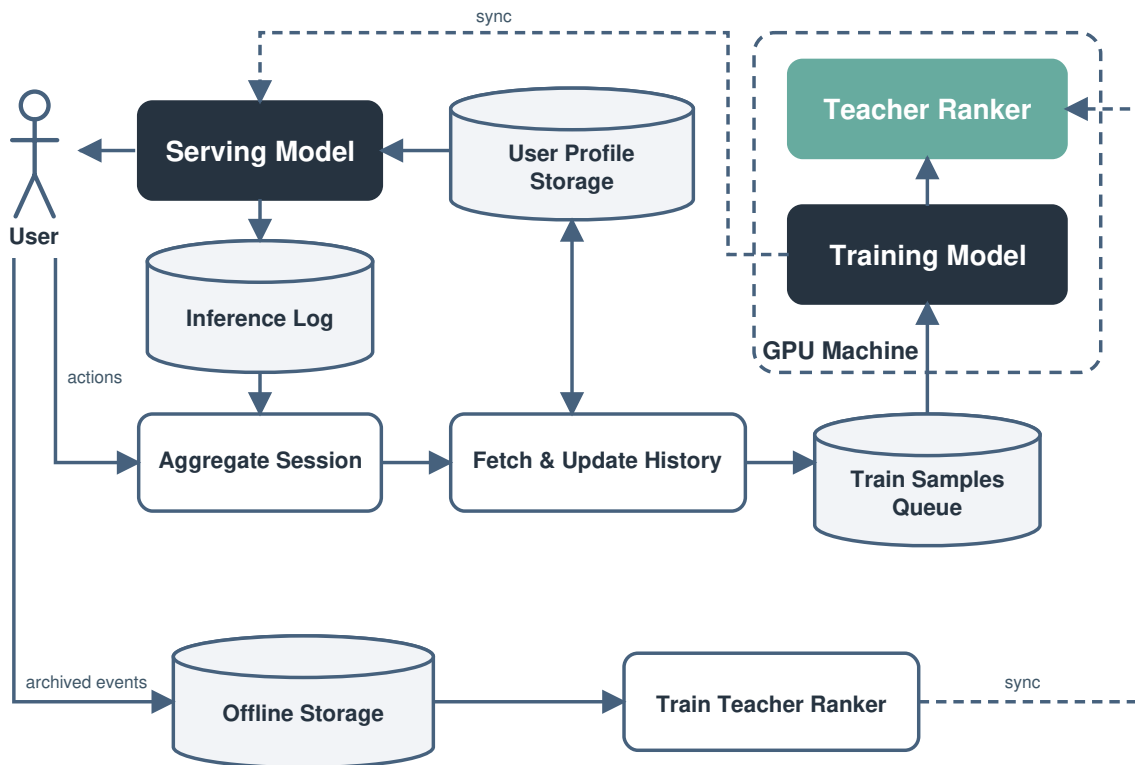


Figure 6.1. Online-training infrastructure for the final SONA configuration. In the diagram, “Teacher Ranker” is the frozen distillation teacher, “Train Teacher Ranker” its training pipeline, “Training Model” / “Serving Model” the trainer and serving copies of SONA, and “Inference Log” the per-request serving record that the session aggregator joins by session identifier.

joins this entry by session identifier and stamps the sample with the cutoff. Second, the user history itself is not propagated through the event stream. The aggregated session is posted back to the inference service, which re-reads the profile from the two stores and rebuilds the sample through the same computation it runs at serving time (Section 6.3). Carrying history inline in every event would inflate the streaming bandwidth by orders of magnitude relative to the rate of training samples, whereas a rebuild from storage stays at the rate of emitted samples.

Update loop. The emitted training samples land in a message queue with at-least-once delivery and single-digit-second read latency. The GPU trainer consumes from the queue and applies the joint objective of Section 4.2 — the next-token-prediction and Ranking Module distillation terms — using the current model as the rollout source for distillation. Updated models are uploaded to a model-artifact store every 10 minutes and pulled by the serving fleet on the same cadence.

6.3 Inference Optimizations

The served SONA model contains the shared encoder, autoregressive decoder, and Ranking Module. In the final production configuration, the frozen teacher remains outside the serving path and is evaluated only by the trainer; temporary teacher-ranked A/B treatments used a separate experimental path (Section 7.5). Generation and item-level ranking are executed by the same GPU worker and share the encoded user state. With the optimizations described below, the served model reaches a model FLOPs utilization (MFU) of 41% during inference.

Sample construction. A separate CPU inference service assembles the model input: it reads the user profile (Section 6.2) and computes the event attributes of Table 3.3 through a feature store, then sends the finished

sample to the GPU fleet. The same feature-store processing is used at every stage — initial training, online training, and inference — so event attributes are computed identically on all three paths.

NVIDIA Triton Inference Server. NVIDIA Triton Inference Server separates request preparation from model execution. An asynchronous CPU batcher groups variable-length user histories, truncates and pads them to fixed input shapes, and dispatches each packed batch to a GPU worker. Fixed shapes allow the worker to capture the complete model path as a CUDA graph during initialization. Steady-state inference copies new inputs into the captured graph and replays it, reducing framework and kernel-launch overhead across the complete request.

Model execution. The history encoder runs once per packed batch. Its hidden states condition every Semantic ID decoding step and are reused by the Ranking Module. Intermediate candidates remain on GPU throughout generation and ranking. The model subgraphs are compiled with `torch.compile`, and the main computation uses `bfloat16`.

Beam-search selection. With a beam width of 1024 and 32,000 logits at each Semantic ID level, top- k selection is a material decoding cost. We apply the radix-based selector of Ye et al. [37] within each parent beam, then merge the retained candidates into the next global beam. Catalog-prefix constraints remove invalid extensions, and the key-value (KV) cache follows the selected parent hypotheses.

Decoder attention. Large-beam decoding produces many independent one-token queries with key-value sequences of at most 3 positions, a geometry poorly matched to the default FlashAttention tile sizes [4]. We therefore use a decoder self-attention kernel with tile sizes tuned for this geometry, substantially reducing computation overhead. The decoder maintains a KV cache across generation steps.

7 Evaluation

We evaluate four questions. First, how do tokenizer choice, model capacity, and data volume affect candidate generation? Second, how do pre-training, history length, and scorer depth affect the Teacher Ranker’s ranking objective? Third, which training and architecture choices let the unified model preserve both target-track retrieval and the teacher’s ordering? Finally, does the selected single-model configuration improve engagement online? Table 7.1 maps each question to the model under test and the reported evidence.

Evaluation question	Model under test	Reported evidence
Candidate generation	NTP-only encoder-decoder	Target-track Recall@ k ; train NTP loss at matched exposure for the capacity sweep
Teacher Ranker design	Teacher Ranker variants under a shared offline protocol	Weighted pair accuracy (WPA) across pre-training, history-length, and scorer-depth ablations
Unified-model transfer	Encoder-decoder and Ranking Module trained jointly with NTP and distillation; teacher used only for distillation targets and reference scores	Target-track Recall@ k , Teacher Recall@ k , and WPA
Online performance	Intermediate treatments and the complete SONA serving configuration	Relative changes in engagement metrics against the control arm from the same experiment

Table 7.1. Scope of the evaluation.

7.1 Evaluation Protocol and Metrics

Target-track Recall. Let n_i be the number of target tracks for request i , and let $m_i(k)$ be the number of those targets found among the first k resolved track predictions. Target-track Recall@ k is the macro-average of $\min\{m_i(k), k\} / \min\{n_i, k\}$ over held-out requests with at least one target. The model-capacity sweep instead reports train NTP loss at matched target exposure; lower train loss is better.

Teacher Recall. Teacher Recall@ k evaluates ranking fidelity conditional on candidate generation: for a held-out request r , the decoder produces a candidate pool C_r , and the Ranking Module and teacher rank the same pool. If $\mathcal{R}$ is the set of evaluated requests and their scalar scores are s_{RM} and s_{teacher} , respectively, then

$$\text{TeacherRecall@}k = \frac{1}{|\mathcal{R}|} \sum_{r \in \mathcal{R}} \frac{1}{k} \left| \text{Top}_k(s_{\text{RM}}^{(r)}; C_r) \cap \text{Top}_k(s_{\text{teacher}}^{(r)}; C_r) \right|,$$

where the intersection is computed within the common candidate pool. When a configuration has no Ranking Module, decoder likelihood supplies its ordering.

Weighted pair accuracy. Weighted pair accuracy measures engagement-based ordering on held-out pairs of temporally adjacent impressions with different target weights. If t_i is the predefined target weight of impression i and $\mathcal{P}$ contains pairs with $t_i > t_j$, then

$$\text{WeightedPairAccuracy} = \frac{\sum_{(i,j) \in \mathcal{P}} (t_i - t_j) \mathbf{1}\{\text{score}_{\text{pair}}(i) > \text{score}_{\text{pair}}(j)\}}{\sum_{(i,j) \in \mathcal{P}} (t_i - t_j)}. \quad (7.1)$$

Pairs farther apart in target weight therefore contribute more. All three offline metrics are higher-is-better.

7.2 Candidate Generation

These experiments evaluate an NTP-only encoder–decoder, without the Ranking Module or teacher. The tokenizer and data-volume ablations use Target-track Recall@ k on held-out requests; the capacity sweep compares train NTP loss at matched target exposure. We vary the tokenizer, model size, and training-data volume.

Semantic tokenizer. Each tokenizer maps a dense track representation to a three-code Semantic ID tuple using residual K-means. We vary codebook size and item representation independently, regenerating the IDs and retraining the retrieval model for each setting. CLMR, a contrastively trained music-audio encoder [24], provides the audio-only baseline. Holding CLMR embeddings fixed, we compare three-level quantizers with 8,192 or 32,000 entries per codebook. Holding the quantizer configuration at three 32,000-entry codebooks, we compare raw CLMR, CLMR with an InfoNCE-trained residual projection on collaborative pairs and the Qwen2.5-Omni audio-and-metadata representation with transformer-based collaborative refinement from Section 3.1.

		Item representation	Recall@10	Recall@100	Recall@1000
Codebook size	Recall@1000	CLMR audio embedding	0.1848	0.4712	0.8111
$3 \times 8,192$	0.8036	CLMR + collaborative			
$3 \times 32,000$	0.8111	projection	0.2005	0.4873	0.8139
(a) Codebook size.		Qwen2.5-Omni + collaborative			
		refinement	0.2171	0.5362	0.8524
		(b) Item representation.			

Table 7.2. Semantic-tokenizer ablations. Each subtable varies one component and holds the other fixed. All variants use the same held-out set.

We select the $3 \times 32\text{k}$ codebook and refined Qwen2.5-Omni representation, which provides the strongest retrieval result in this ablation.

Model size. The model-size sweep varies encoder–decoder capacity while fixing the layer allocation, data window, objective, and 8192-event histories. We report train NTP loss against cumulative packed-target exposure. The Small, Medium, and $2 \times$ Medium configurations contain 20M, 130M, and 260M transformer-core parameters, respectively. Their total learnable counts, including embeddings and output heads, are 264M, 485M, and 659M.

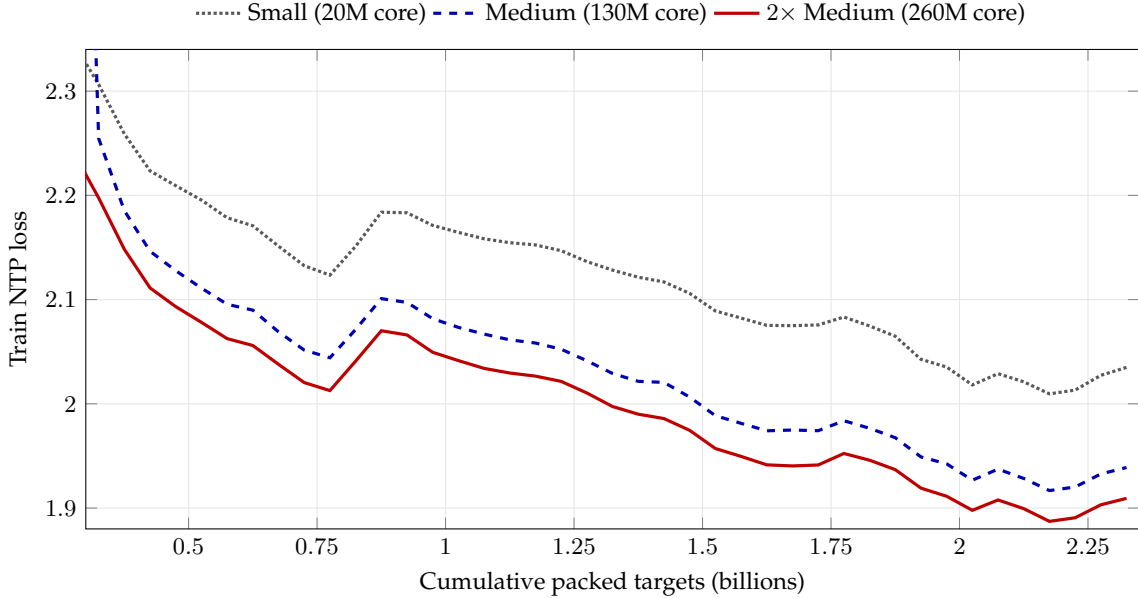


Figure 7.1. Train NTP loss of the Small, Medium, and 2× Medium configurations at matched cumulative packed-target exposure.

Increasing capacity consistently lowers train NTP loss over the measured exposure range.

Training-data volume. We next fix the Medium backbone, 2k-event histories, tokenizer, held-out set, and optimization recipe. Each run makes one chronological pass over a longer log window.

Training window	Train loss	Recall@10	Recall@100	Recall@1000
1 week	2.2735	0.1798	0.4979	0.8333
2 weeks	2.1061	0.2126	0.5576	0.8640
4 weeks	2.0800	0.2388	0.5993	0.8849
8 weeks	2.0427	0.2519	0.6178	0.8947

Table 7.3. Final encoder–decoder backbone train loss and recall as the unique chronological training-data window grows. Each run makes one pass over its window.

The point estimates improve monotonically as the one-pass training window grows, including from 4 weeks to 8 weeks. Together with the separated capacity curves, the sweep shows no observed plateau within the measured ranges.

7.3 Teacher Ranker Design Ablations

These experiments compare Teacher Ranker variants using the held-out engagement-pair protocol and WPA defined above. They isolate choices within the teacher — initial pre-training, attended history length, and candidate-scorer depth.

Pre-training. We compare the two-stage initial recipe with direct fine-tuning from random initialization. Both variants use an 8k-event history and a six-layer candidate scorer; the only change is whether next-item-prediction pre-training precedes ranking fine-tuning.

Configuration	Weighted pair accuracy
Teacher, no pre-training	0.6153
Teacher, two-stage initial recipe (selected)	0.6215

Table 7.4. Teacher Ranker pre-training ablation at fixed 8k-event history and six-layer candidate scorer. The first row removes only the next-item-prediction pre-training stage of Section 5.3.

The gain establishes NTP pre-training as part of the selected Teacher Ranker recipe.

History length and scorer depth. We next vary both axes while holding the remaining teacher configuration fixed.

History	Scorer depth	Weighted pair accuracy
2k	1	0.6080
	2	0.6105
	4	0.6124
	6	0.6134
	8	0.6138
8k	1	0.6159
	2	0.6189
	4	0.6207
	6 (selected)	0.6215
	8	0.6219

Table 7.5. Teacher Ranker WPA point estimates as candidate-scorer depth and attended history length vary on the same held-out engagement-label set.

Longer attended histories provide consistent gains across scorer depths. We use the six-layer, 8k-event configuration in subsequent distillation and limited-traffic experiments.

7.4 Unified Generation and Ranking

With the teacher configuration fixed, these offline experiments evaluate the encoder, decoder, and Ranking Module trained jointly with NTP and distillation. The teacher supplies distillation targets and evaluation reference scores but is not part of the evaluated model. We report the metrics defined in Section 7.1 while varying the distillation loss, candidate source, training rollout beam size, Ranking Module depth, and history encoding.

Experimental sequence. All variants use the components of Section 3 and a 2-week training window. We report encoder, decoder, and Ranking Module layer counts separately in the E , D , and R columns of Table 7.7. A dash in the R column denotes decoder-likelihood ordering without a Ranking Module. Table 7.6 varies one factor at a time around a shared reference: MAE distillation, rollouts plus logged impressions, a training rollout beam of 32, 2k-event histories, and a model with $E = 7$, $D = 2$, and $R = 1$. The daggered entry in each block is therefore the same reference run. Every variant is evaluated with the fixed serving beam $B_{\text{eval}} = 1024$. We then increase R to 4 and use that architecture for the history ablations.

Configuration	Target-track Recall@1000	Teacher Recall@10	Teacher Recall@100	WPA
<i>Distillation loss</i>				
MAE [†]	0.8615	0.5654	0.7344	0.5892
MSE	0.8671	0.5748	0.7329	0.5915
Huber	0.8679	0.5568	0.7220	0.5894
KL	0.8662	0.5452	0.7151	0.5860
<i>Distillation candidate source</i>				
Rollouts + impressions [†]	0.8615	0.5654	0.7344	0.5892
Rollouts only	0.8610	0.5618	0.7288	0.5730
Impressions only	0.8663	0.2983	0.5281	0.5872
<i>Training rollout beam size</i>				
32 [†]	0.8615	0.5654	0.7344	0.5892
64	0.8565	0.5661	0.7359	0.5901
128	0.8616	0.5762	0.7427	0.5905

Table 7.6. One-factor unified-model ablations around the shared reference: MAE, rollouts plus impressions, training rollout beam 32, 2k-event history, $E = 7$, $D = 2$, and $R = 1$. The daggered entries are the same reference run.

Distillation loss. We compare MAE, MSE, Huber, and pairwise KL on a held-out split (Table 7.6). No loss dominates across retrieval, ranking fidelity, and engagement-based ordering, so we retain MAE as the reference for the subsequent ablations.

Distillation candidate distribution. Rollouts come from the current decoder; impressions come from logged requests. Impressions alone have the highest retrieval point estimate but much lower Teacher Recall than either configuration containing rollouts. Combining both sources keeps Teacher Recall close to rollouts only and has the higher WPA point estimate, so we use both sources.

Training rollout beam size. The training beam B_{train} determines how many decoder candidates the Ranking Module sees per training prompt; it is distinct from the fixed evaluation beam B_{eval} . Increasing B_{train} from 32 to 128 gives the highest Teacher Recall point estimates but quadruples the number of rollout candidates scored during training. We retain $B_{\text{train}} = 32$ as the lower-cost reference (Table 7.6).

Configuration		Metrics						
History encoding	History Length	E	D	R	Target-track Recall@1000	Teacher Recall@10	Teacher Recall@100	WPA
Full attention	2k	7	2	–	0.8656	0.0381	0.1936	0.5478
Full attention	2k	7	2	1	0.8615	0.5654	0.7344	0.5892
Full attention	2k	7	2	4	0.8656	0.6005	0.7501	0.5937
Full attention	8k	7	2	4	0.8722	0.6586	0.7997	0.6029
History Compression	8k	7	2	4	0.8709	0.6474	0.7893	0.6033

Table 7.7. Ranking Module depth followed by history-length and history-encoding ablations. The E , D , and R columns report layer counts separately; $R = -$ denotes decoder-likelihood ordering without a Ranking Module. The depth study increases R from 1 to 4, and the history study fixes the latter depth.

Ranking Module depth and history encoding. Table 7.7 first scales the Ranking Module with 2k-event histories. Moving from decoder-likelihood ordering ($R = -$) to one scorer layer substantially raises Teacher Recall and WPA; increasing R to 4 raises the Teacher Recall point estimate further.

With $R = 4$ fixed, extending full attention to 8k events gives the highest Target-track Recall and Teacher Recall point estimates. History Compression (Section 3.3 and Figure 3.2) has close values and the highest WPA point estimate while applying the deep encoder only to recent events. We select this configuration for the final model.

7.5 Online A/B Tests

Unless marked [†], reported treatment–control deltas are significant at $p < 0.01$. Five controlled A/B tests evaluate serving configurations on My Vibe traffic from smart speakers. Experiments 1–3 examine teacher-assisted configurations, Experiment 4 evaluates the distilled serving model, and Experiment 5 evaluates the complete SONA system.

Active Users is the primary metric; we also report Total Listening Time, Likes, “Repeat” Commands, and Deeply Engaged Users. Every table entry is a relative percentage change from the production control arm in the same experiment.

Experiment 1: Likes score. An NTP-only encoder–decoder generates candidates, which the teacher orders using its pairwise score plus a weighted likes score. We compare the production control with three teacher-assisted treatments: pairwise scoring alone (weight 0), and likes-score weights α and 2α . Each group contains 5% of users. The teacher attends to 8k events and the encoder–decoder to 2k; its serving cost limits traffic and statistical power (Table 7.8).

Metric	Pairwise only (0)	+ α Likes score	+ 2α Likes score
Total Listening Time	+2.05%	+2.45%	+2.28%
Likes	−1.03% [†]	+6.56%	+11.93%
“Repeat” Commands	+7.43%	+12.89%	+12.57%
Active Users	+1.51%	+1.83%	+1.53%
Deeply Engaged Users	+2.61%	+2.87%	+2.80%

Table 7.8. Experiment 1: control-relative changes for Likes-score weights in the teacher’s ranking score, on the NTP-only encoder–decoder stack.

The results support retaining a non-zero likes score in the teacher.

Experiment 2: Semantic ID features in the Teacher Ranker. Both treatments use the same NTP-only encoder–decoder for candidate generation and the teacher for ordering. The ablation changes only the teacher’s candidate features: one teacher is trained without Semantic ID (SID) features, while the other includes them. Control and both treatments each contain 4% of users (Table 7.9).

Metric	Teacher without SID features	Teacher with SID features
Total Listening Time	+2.60%	+3.18%
Likes	+11.13%	+19.86%
“Repeat” Commands	+15.88%	+20.52%
Active Users	+1.98%	+2.23%
Deeply Engaged Users	+2.95%	+3.80%

Table 7.9. Experiment 2: control-relative changes when the teacher’s SID candidate features are the only difference between treatments; both use the same NTP-only encoder–decoder.

We therefore retain SID features in the teacher.

Experiment 3: Teacher Ranker and candidate generation. We test the teacher with two candidate-generation paths. One treatment replaces only the production ranker; the other also replaces production candidate generation with the encoder–decoder. Both use only the teacher’s pairwise score, matching the scoring objective across treatments. Serving cost limits each split to 3% of users (Table 7.10).

Metric	Teacher Ranker	Enc-dec + Teacher Ranker
Total Listening Time	+1.64%	+3.63%
Likes	+1.48% [†]	-0.54% [†]
“Repeat” Commands	+4.68% [†]	+5.02% [†]
Active Users	+1.22%	+2.72%
Deeply Engaged Users	+1.83%	+4.06%

Table 7.10. Experiment 3: the Teacher Ranker replacing the production ranker, and the NTP-only encoder-decoder with Teacher Ranker ordering replacing the full stack.

The Teacher Ranker can successfully replace the production ranker, and the further replacement of candidate generation by the encoder-decoder shows promising results toward replacing the full stack.

Experiment 4: distillation. We compare the distilled model with the same candidate generator re-ranked by the teacher. Both treatments use the encoder-decoder trained with the joint objective of Section 4.2 and 2k-event histories. One orders candidates with the distilled Ranking Module; the other uses the Teacher Ranker, which attends to 8k events. Both include the likes score. Control and both treatments each contain 8% of users (Table 7.11).

Metric	Distilled model	Distilled + Teacher Ranker
Total Listening Time	+1.62%	+4.32%
Likes	+7.12%	+11.81%
“Repeat” Commands	+15.25%	+14.93%
Active Users	+1.41%	+2.81%
Deeply Engaged Users	+2.78%	+5.11%

Table 7.11. Experiment 4: the distilled single model, alone and re-ranked by the Teacher Ranker, replacing the full stack.

The distilled-model result shows that the production cascade can be replaced without serving the teacher.

Experiment 5: SONA. The fifth experiment studies the further improvement of the recipe by extending the user history to 8192 events with History Compression. It has two splits — control and SONA, the full recipe described in this report — and ran for 7 days on 15% of randomly selected users in each split. Hard business rules that restrict production output are preserved (Table 7.12).

Metric	Relative change vs control
Total Listening Time	+6.30%
Likes	+11.42%
“Repeat” Commands	+17.99%
Active Users	+4.53%
Deeply Engaged Users	+7.37%

Table 7.12. Experiment 5: the single SONA model replaces the whole recommendation stack.

A key difference from Experiment 4 is the available user context: Experiment 4 uses 2k-event histories, whereas Experiment 5 extends the history to 8192 events with History Compression. The offline history ablation (Table 7.7) shows material gains from the longer context, so this change is an important part of the complete SONA configuration. Because the two online experiments were conducted separately, their difference does not isolate the causal contribution of history length alone.

On the primary Active Users metric, SONA’s +4.53% control-relative uplift is 2.35 $\times$ as large as the +1.93% uplift previously recorded for Argus [12], the strongest model previously deployed on this surface.

8 Conclusion, Limitations, and Future Work

This technical report demonstrates that a mature industrial recommendation stack — candidate generation, pre-ranking, and ranking — can be replaced by a single model trained and served end to end. SONA generates candidates and ranks them with one transformer over the user’s raw event history, supervised by next-token prediction and by distillation from a Teacher Ranker without hand-engineered features. In its final online A/B test, it improves all five reported engagement metrics relative to the production control (Section 7.5). The remainder of this section lists the limitations of the present system and the directions we are pursuing next.

Toward full deployment. Despite the major improvements in core platform metrics, we have not yet deployed SONA on full traffic. SONA introduces significant changes to the existing stack, and we want to validate it in a long-term experiment spanning a multi-month window. We also observed that SONA’s catalog coverage is lower than that of the production stack, and we are going to investigate this effect.

Other surfaces. This report validates SONA on one of our recommendation surfaces. Full deployment requires validating the approach on the other surfaces, and we leave this investigation for future work.

Model scaling. Across the measured capacity and data ranges, the reported point estimates continue to improve at the largest tested settings (Section 7.2). We will continue enlarging the model’s capacity, including sparse mixture-of-experts backbones, the adoption of sparse and linear attention [16], and a persistent cross-request KV cache.

Test-time scaling. The potential of test-time scaling is still underexplored. Beam search provides such an axis — widening the beam trades inference compute for candidate quality — but how to leverage test-time scaling properly remains an open question.

Content exploration. We did not notice any degradation in content exploration, which could be explained by SONA being trained on production logs. We leave further investigation of fresh-content behavior for future work.

Reinforcement learning. Reinforcement-learning post-training has proved effective for LLMs [1, 6, 17, 19, 25], and industrial generative recommenders increasingly use RL to align generation with engagement and business objectives [8, 18, 20, 31, 33, 42, 46]. Recommendation is a natural fit: logged user feedback supplies a verifiable reward, and that reward is tightly coupled with the business objectives the platform optimizes. We adopt a training stage only once a controlled comparison demonstrates its contribution, and for RL we have not yet built that evidence; the recipe reported here is therefore fully supervised. Promising directions for SONA include long-term reward modeling and RLVR-style post-training on user feedback, and we leave their controlled evaluation to future work.

References

- [1] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.
- [2] Fedor Borisjuk, Lars Hertel, Ganesh Parameswaran, Gaurav Srivastava, Sudarshan Srinivasa Ramanujam, Borja Ocejo, Peng Du, Andrei Akterskii, Neil Daftary, Shao Tang, Daqi Sun, Qiang Charles Xiao, Deepesh Nathani, Mohit Kothari, Yun Dai, Guoyao Li, and Aman Gupta. From features to transformers: Redefining ranking for scalable impact. *arXiv preprint arXiv:2502.03417*, 2025. URL <https://arxiv.org/abs/2502.03417>.
- [3] Benjamin Coleman, Wang-Cheng Kang, Matthew Fahrbach, Ruoxi Wang, Lichan Hong, Ed H. Chi, and Derek Zhiyuan Cheng. Unified embedding: Battle-tested feature representations for web-scale ml systems. In *Advances in Neural Information Processing Systems*, 2023. URL <https://arxiv.org/abs/2305.12102>.
- [4] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *arXiv preprint arXiv:2205.14135*, 2022. URL <https://arxiv.org/abs/2205.14135>.

- [5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021. URL <https://arxiv.org/abs/2010.11929>.
- [6] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- [7] Ruidong Han, Bin Yin, Shangyu Chen, He Jiang, Fei Jiang, Xiang Li, Chi Ma, Mincong Huang, Xiaoguang Li, Chunzhen Jing, Yueming Han, Menglei Zhou, Lei Yu, Chuan Liu, and Wei Lin. Mtgr: Industrial-scale generative recommendation framework in meituan. *arXiv preprint arXiv:2505.18654*, 2025. URL <https://arxiv.org/abs/2505.18654>.
- [8] Xuegang Hao, Ming Zhang, Alex Li, et al. Oxygenrec: An instruction-following generative framework for e-commerce recommendation. *arXiv preprint arXiv:2512.22386*, 2025. URL <https://arxiv.org/abs/2512.22386>.
- [9] Ruining He, Lukasz Heldt, Lichan Hong, Raghunandan Keshavan, Shifan Mao, Nikhil Mehta, Zhengyang Su, Alicia Tsai, Yueqi Wang, Shao-Chuan Wang, Xinyang Yi, Lexi Baugher, Baykal Cakici, Ed H. Chi, Cristos Goodrow, Ningren Han, He Ma, Romer Rosales, Abby Van Soest, Su-Lin Wu, Weilong Yang, and Yilin Zheng. Plum: Adapting pre-trained language models for industrial-scale generative recommendations. *arXiv preprint arXiv:2510.07784*, 2025. URL <https://arxiv.org/abs/2510.07784>.
- [10] Alex Henry, Prudhvi Raj Dachapally, Shubham Pawar, and Yuxuan Chen. Query-key normalization for transformers. *Findings of EMNLP*, 2020. URL <https://arxiv.org/abs/2010.04245>.
- [11] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020. URL <https://arxiv.org/abs/2001.08361>.
- [12] Kirill Khrylchenko, Artem Matveev, Sergei Makeev, and Vladimir Baikalov. Scaling recommender transformers to one billion parameters. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2026. URL <https://arxiv.org/abs/2507.15994>.
- [13] Hanyu Li, Yi-Ping Hsu, Aditya Mantha, Prabhat Agarwal, Laksh Bhasin, Jialu Wang, Hongtao Lin, Bella Huang, Yaxin Li, Xinyi Li, et al. Unipinrec: Unifying generative retrieval and ranking at pinterest scale. *arXiv preprint arXiv:2606.00422*, 2026.
- [14] Yu Liu and Jiangxia Cao. Harmonizing generative retrieval and ranking in chain-of-recommendation. *arXiv preprint arXiv:2604.25787*, 2026. URL <https://arxiv.org/abs/2604.25787>.
- [15] Xinchun Luo, Jiangxia Cao, et al. Qarm: Quantitative alignment multi-modal recommendation at kuaishou. *arXiv preprint arXiv:2411.11739*, 2024.
- [16] Artem Matveev, Vladislav Tytskiy, Sergei Makeev, and Sergei Liamshev. Gated bidirectional linear attention for generative retrieval. *arXiv preprint arXiv:2606.07317*, 2026. URL <https://arxiv.org/abs/2606.07317>.
- [17] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.
- [18] OneRec Team. Onerec technical report. *arXiv preprint arXiv:2506.13695*, 2025. URL <https://arxiv.org/abs/2506.13695>.
- [19] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 2022. URL <https://arxiv.org/abs/2203.02155>.
- [20] Junyan Qiu et al. Ega-v1: Unifying online advertising with end-to-end learning. *arXiv preprint arXiv:2505.19755*, 2025.
- [21] Qwen Team. Qwen2.5-omni technical report. *arXiv preprint arXiv:2503.20215*, 2025. URL <https://arxiv.org/abs/2503.20215>.
- [22] Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Tran, Jonah Samost, et al. Recommender systems with generative

- retrieval. *Advances in Neural Information Processing Systems*, 2023. URL <https://arxiv.org/abs/2305.05065>.
- [23] Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.
 - [24] Janne Spijkervet and John Ashley Burgoyne. Contrastive learning of musical representations. *arXiv preprint arXiv:2103.09410*, 2021. URL <https://arxiv.org/abs/2103.09410>.
 - [25] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. Learning to summarize from human feedback. *Advances in Neural Information Processing Systems*, 2020. URL <https://arxiv.org/abs/2009.01325>.
 - [26] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021.
 - [27] Yijia Sun, Shanshan Huang, Zhiyuan Guan, Qiang Luo, Ruiming Tang, Kun Gai, and Guorui Zhou. Grank: Towards target-aware and streamlined industrial retrieval with a generate-rank framework. In *Proceedings of the ACM Web Conference 2026 (WWW '26)*, 2026. URL <https://arxiv.org/abs/2510.15299>.
 - [28] Daria Tikhonovich, Oleg Sorokin, Vladislav Dodonov, Mariia Ulianova, and Ilya Murzin. Gryphon: A unified architecture for semantic-id generation and item-level scoring in industrial recommendations. *arXiv preprint arXiv:2606.08604*, 2026.
 - [29] Philippe Tillet, H. T. Kung, and David Cox. Triton: An intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages (MAPL)*, 2019. URL <https://doi.org/10.1145/3315508.3329973>.
 - [30] Chunqi Wang, Bingchao Wu, Zheng Chen, Lei Shen, Bing Wang, and Xiaoyi Zeng. Scaling transformers for discriminative recommendation via generative pretraining. *arXiv preprint arXiv:2506.03699*, 2025. URL <https://arxiv.org/abs/2506.03699>.
 - [31] Shen Wang et al. Onelive: Dynamically unified generative framework for live-streaming recommendation. *arXiv preprint arXiv:2602.08612*, 2026.
 - [32] Wenjie Wang, Honghui Bao, Xinyu Lin, Jizhi Zhang, Yongqi Li, Fuli Feng, See-Kiong Ng, and Tat-Seng Chua. Learnable item tokenization for generative recommendation. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, 2024. URL <https://arxiv.org/abs/2405.07314>.
 - [33] Zhipeng Wei et al. Oneloc: Geo-aware generative recommender systems for local life service. *arXiv preprint arXiv:2508.14646*, 2025.
 - [34] Lee Xiong, Zhirong Chen, Rahul Mayuranath, Shangran Qiu, Arda Ozdemir, et al. Llatte: Scaling laws for multi-stage sequence modeling in large-scale ads recommendation. *arXiv preprint arXiv:2601.20083*, 2026. URL <https://arxiv.org/abs/2601.20083>.
 - [35] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer architecture. In *Proceedings of the 37th International Conference on Machine Learning*, 2020. URL <https://proceedings.mlr.press/v119/xiong20b.html>.
 - [36] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International conference on machine learning*, pages 10524–10533. PMLR, 2020.
 - [37] Zihao Ye, Lequn Chen, Ruihang Lai, Wuwei Lin, Yineng Zhang, Stephanie Wang, Tianqi Chen, Baris Kasikci, Vinod Grover, Arvind Krishnamurthy, and Luis Ceze. Flashinfer: Efficient and customizable attention engine for LLM inference serving. In *Proceedings of Machine Learning and Systems*, 2025. URL https://proceedings.mlsys.org/paper_files/paper/2025/file/dbf02b21d77409a2db30e56866a8ab3a-Paper-Conference.pdf.
 - [38] Xinyang Yi, Ji Yang, Lichan Hong, Derek Zhiyuan Cheng, Lukasz Heldt, Aditee Kumthekar, Zhe Zhao, Li Wei, and Ed Chi. Sampling-bias-corrected neural modeling for large corpus item recommendations. In *RecSys*, 2019.
 - [39] Jiaqi Zhai et al. Actions speak louder than words: Trillion-parameter sequential transducers for generative recommendations. *arXiv preprint arXiv:2402.17152*, 2024.

- [40] Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems*, 2019. URL <https://arxiv.org/abs/1910.07467>.
- [41] Jun Zhang, Yi Li, Yue Liu, et al. Gpr: Towards a generative pre-trained one-model paradigm for large-scale advertising recommendation. *arXiv preprint arXiv:2511.10138*, 2025. URL <https://arxiv.org/abs/2511.10138>.
- [42] Kun Zhang et al. Onemall: One architecture, more scenarios — end-to-end generative recommender family at kuaishou e-commerce. *arXiv preprint arXiv:2601.21770*, 2026.
- [43] Wei Zhang, Dai Li, Chen Liang, Fang Zhou, Zhongke Zhang, Xuewei Wang, Ru Li, Yi Zhou, Yaning Huang, Dong Liang, et al. Scaling user modeling: Large-scale online user representations for ads personalization in meta. In *Companion Proceedings of the ACM Web Conference (WWW '24 Companion)*, 2024. URL <https://arxiv.org/abs/2311.09544>.
- [44] Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, et al. Pytorch fsdp: Experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*, 2023.
- [45] Carolina Zheng, Minhui Huang, Dmitrii Pedchenko, Kaushik Rangadurai, Siyu Wang, Fan Xia, Gaby Nahum, Jie Lei, Yang Yang, Tao Liu, et al. Enhancing embedding representation stability in recommendation systems with semantic id. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems*, pages 954–957, 2025.
- [46] Zuowu Zheng et al. Ega-v2: An end-to-end generative framework for industrial advertising. *arXiv preprint arXiv:2505.17549*, 2025.

A Contributions and Acknowledgments

Contributors are listed alphabetically by first name. * marks contributors who have since departed from the team.

Contributors. Alexandr Udeneev, Aleksei Krasilnikov*, Alexey Nadtochiy, Andrey Semenov, Andrey Tsykunov, Anna Krivonos, Anna Lipkina, Artem Matveev*, Daniil Burlakov, Daniil Leschev*, Daria Tikhonovich, Denis Burshtein, Ekaterina Dmitrieva, Eugene Krofto, Grigorii Khlystov*, Ilya Murzin, Kirill Golovko, Ksenia Sycheva, Leonid Dmitriev, Mariia Rozaeva, Mariia Ulianova, Mikhail Sandul, Nikolai Savushkin, Oleg Sorokin, Roman Odobesku, Semyon Panenko, Sergei Liamaev*, Sergei Makeev*, Vadim Shilov, Veronika Ivanova, Viktor Yanush, Vladimir Baikarov*, Vladislav Dodonov, Vladislav Tytskiy*.

Acknowledgments. We would like to extend special thanks to Alexander Ploshkin, Alexey Gusakov, Alexey Pismenny, Nikita Stepanov, Petr Ermakov, Petr Garmider, and Sergey Kastruyulin for their substantial support throughout this work.

B Hyperparameters and Model Cards

B.1 Semantic Tokenizer

Parameter	Value
Audio foundation model	Qwen2.5-Omni [21]
Audio input length	90 s mel-spectrogram
Embedding refresh threshold	500 lifetime plays
Qwen output sequence length	up to 640 hidden vectors
Refinement transformer	4 layers, 4 heads, model dim 512
Item embedding dimension	256
Refinement contrastive temperature	0.1
Refinement alignment weight	0.1
Semantic ID codebooks	3 levels, 32,000 codes per level

Table B.1. Tokenizer model card.

B.2 Teacher Ranker

Parameter	Value
Encoder layers	10 (causal)
Candidate-scorer layers	6
Hidden dimension	1024
Attention heads	16 (head dim 64)
Hash embedding tables	$2^{19} \times 256$ (item / artist), $2^{19} \times 512$ (semantic prefix)
Per-level SID embeddings	32001×128
Maximum positions	8193
Dropout	0.1 (encoder); 0 (candidate scorer)
Total parameters	0.6B (embedding-table dominated)
Pre-training history length	2k events
Fine-tuning history length	8k events
NIP in-batch negatives	$2^{13} = 8192$
LR schedule	$10^{-5} \rightarrow 10^{-4}$ over 3000 warmup steps, then constant (both groups)
Gradient clipping	1.0

Table B.2. Teacher Ranker card. Initial training has two stages (Section 5.3); only the encoder is reused between pre-training and multi-head ranking fine-tuning. The resulting model is then refreshed continuously.

B.3 Ranking Module

Parameter	Value
Blocks	4 cross-attention layers
Hidden dimension	1024
Attention heads	4 (head dim 256)
Feed-forward dimension	4096
Scoring outputs	2 teacher-head scores
Rollouts per prompt	32
Loss weights	NTP : rollout : impression = 1 : 1 : 1

Table B.3. Ranking Module card. Distilled from the Teacher Ranker jointly with encoder–decoder backbone training (Section 4.2).

B.4 Encoder–Decoder Backbone

Parameter	Value
Layer allocation	7 recent-block layers; 1 full-history layer; 1 bridge per direction
Hidden dimension	1024
Attention heads	16 (head dim 64)
Feed-forward dimension	2816
History lengths	8192 events total; 2048 recent (+1 CLS)
Hash-bucket embedding table	$2^{19} \times 256$ (shared)

Table B.4. Encoder model card.

Parameter	Value
Layers	2 (causal self-attention + cross-attention over encoder states)
Semantic ID vocabulary	3 codebooks $\times$ 32,000 (+ BOS row)
Output heads	$3 \times \text{Linear}(1024 \rightarrow 32000)$, untied
Serving decoding	constrained beam search, beam 1024

Table B.5. Decoder model card.

Parameter	Value
Optimizer	AdamW (fused), weight decay 0.1, gradient clipping 1.0
LR schedule (initial training)	linear, $10^{-5} \rightarrow 3 \times 10^{-4} \rightarrow 7 \times 10^{-5}$, 3000-step warmup
Effective batch size	16,384 samples

Table B.6. Encoder–decoder backbone optimization.

B.5 Online Training and Model Refresh

Configuration	Value
Online update mode	continuous optimization over the event-ordered request stream
LR schedule	constant 7×10^{-5} , the terminal rate of the initial-training schedule
Training objective	next-token prediction and Ranking Module distillation
Session-attribution window	15 minutes
Serving-model sync cadence	every 10 minutes
Teacher snapshot refresh cadence	every 24 hours

Table B.7. Online-training and model-refresh configuration (Section 6.2).

C Notation

Symbol	Definition
<i>User history and encoder</i>	
$u = (e_1, \dots, e_T)$	chronological user engagement history; e_t is one logged event
$T, T_{\max}$	history length and its cap
X	embedded event sequence, the encoder input (Section 3.2)
$H = \text{Enc}_\theta(X)$	direct hidden-state output of a full encoder, $H \in \mathbb{R}^{(T+1) \times d}$
d	hidden dimension
θ	learnable model parameters
<i>Semantic tokenizer</i>	
v	catalog track
$s(v) = (s_1, \dots, s_L)$	Semantic ID tuple of track v ; s_ℓ is the level- ℓ code, L the number of levels
$z_a, \bar{h}_a$	item embedding of track a and the mean of its content features
m	collaborative pairs per training batch
sim, τ	cosine similarity and InfoNCE temperature
λ_{align}	weight of the alignment loss term
$\text{NPMI}(i, j)$	normalized pointwise mutual information of co-listening tracks i and j ; $p(i, j), p(i)$ are co-occurrence and marginal probabilities
<i>History Compression</i>	
N	maximum attended history length
O, R	long-term and recent blocks of $n_o = N - n_r$ and n_r events
$\hat{O}, \hat{R}$	blocks enriched by bidirectional cross-attention
X_O, X_R	block representations after the full-history pass
H_R	deep-stack encoding of the recent block X_R
K	shared encoder memory exposed to the decoder and Ranking Module; $K = H$ for a full encoder and $K = [X_O; H_R]$ with History Compression
<i>Training</i>	
$C^+(u)$	packed positive target set of the next-token objective
B	rollout beam size
$\mathcal{B}, \mathcal{I}$	rollout candidate set and logged impressions of a request, the two distillation sources
$r_c^h, \hat{r}_c^h$	teacher and Ranking Module head- h scores of candidate c ; n is the number of distilled heads
$\mathcal{L}_{\text{NTP}}, \mathcal{L}_{\text{rollout}}, \mathcal{L}_{\text{impression}}$	next-token loss and the two distillation losses of the joint objective (Section 4.2)
<i>Teacher Ranker</i>	
$\mathcal{P}$	ordered pairs of temporally adjacent impressions with different engagement grades
$\text{score}_{\text{pair}}(i)$	pairwise-head score of impression i
$\mathcal{L}_{\text{pair}}, \mathcal{L}_{\text{BCE}}^k, \mathcal{L}_{\text{teacher}}$	pairwise loss, per-head pointwise losses, and their sum (Section 5.3)
<i>Evaluation</i>	
k	ranking cutoff
$\text{Top}_k(\cdot)$	top- k item set under a scoring function
$r, \mathcal{R}$	evaluated request/history and the set of such requests
C_r	decoder-generated candidate pool for request r
$s_{\text{RM}}^{(r)}, s_{\text{teacher}}^{(r)}$	Ranking Module and teacher scores over C_r
t_i	predefined target weight of impression i 's engagement label

Table C.1. Notation used throughout the report.