



Large Behavioral Models

Foundation models over user event sequences

Rickard Brül Gabrielsson | Unbox AI

Invited talk at JPMorgan AI Research, August 27, 2026



Foundation models.

- Pretrain **one** model on a broad corpus by **self-supervision** — no labels, no human annotation. The data supervises itself.
- **Easily adapted** to many tasks — little or no target data, and no per-task architecture.
- Performance and range of tasks it covers **grows with scale**.

The shift was not a better model per se. It was the end of **one model per task**. (Similar to human intelligence)



Banking, retail, payments, app usage. What a person did, in order, timestamped — rather than what anyone wrote.

Same recipe: one model, pretrained by self-supervision on event sequences, serving many tasks across many catalogs and many domains. We call these **Large Behavioral Models**.

What this talk is

What happened to language is happening to behavior — and it is critical to understand the differences to get it working.



1. **The claim** — what a foundation model for behavior would have to demonstrate.
2. **Why it has not happened yet** — including at the seven institutions that have each built one.
3. **Approach** — enough to make the results legible.
4. **Evidence** — benchmarks, breadth, A/B tests, and what each of them does not establish.
5. **Scaling laws** — what scale actually buys, on four axes and ~ 600 runs.
6. **Conclusions**, and where I think this goes.

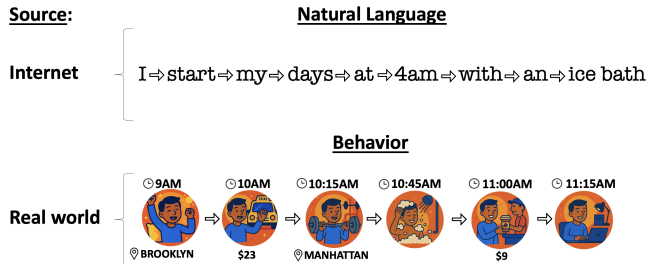


The claim



- P1 Zero-shot / few-shot transfer.** Performs well on a new merchant, catalog and user population with *no* samples from that dataset — where a specialist has nothing to train on at all.
- P2 Breadth from one checkpoint.** The *same weights* competitive with the strongest specialist across industries, subdomains and task families — simultaneously, not one at a time.
- P3 Generality that improves with scale.** Zero-shot transfer improves, and the target data needed for a given accuracy falls, as the pretrained model grows.

None of the three is a marginal accuracy gain over a per-dataset baseline. All are structural, and a specialist cannot exhibit them at any level of tuning.

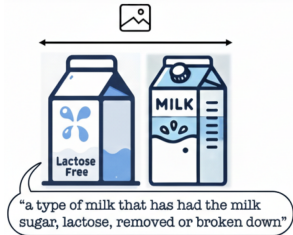


Text and images are the polished **outputs** of human activity.

Behavioral logs are the **trajectory that produced them** — timestamped, priced, located, never written for an audience.

An LLM predicts the next token. An LBM predicts the next *action*, over a candidate set that changes with every deployment — and, within one deployment, over time.

LLMs / Vision Features



Behavior Actions



Regular and lactose-free milk are near-duplicates in text and in pixels.

Behaviorally they are **nearly disjoint**: few users buy both, and lactose-free milk co-occurs with gluten-free bread far more often than with regular milk.

Similarity from descriptions gives answers that sound *reasonable*. Similarity from actions gives answers that are *correct for the user at hand*.



Words are shared across every English corpus. Actions are not.

Every tracking system has its own catalog, its own event taxonomy, and its own notion of what counts as an interaction.

Behavior is far more heterogeneous



Why behavior has not had its
LLM moment yet

Seven institutions in two years. One tested transfer off its corpus.



Model	Corpus	Par.	Data	Ind.	0-shot
HSTU (Meta)	Meta platforms	0.13B	100B ex.	1	No
Netflix FM	Netflix	~1B	100s B int.	1	No
PRAGMA (Revolut)	banking	0.91B	24B ev.	1	No
nuFormer (Nubank)	card txns.	0.30B	500B tok.	1	No
TREASURE (Visa)	card txns.	~0.005B	6B txn.	1	No
TransactionGPT (Visa)	card txns.	0.01B	1B+ txn.	1	No
Stripe PFM	payments	—	10s B txn.	1	No
TradeFM (JPMorgan)	equities	0.49B	10.7B tok.	1	Yes*
BehaviorGPT-v4	multi-industry	12.5B	1.25T act.	3	Yes

Par.: non-embedding parameters — the only comparable count: lookup tables can hold a trillion parameters while computing with very few. TREASURE is the extreme case, 3 layers at $d=256$: ~5M compute-bearing against ~38B in its merchant table. Only TransactionGPT reports it directly; the rest are our counts from published configs, and dashes mean no architecture published.

0-shot: a target *corpus* with no samples from it — several do test new *tasks* on their own. *Next slide.

Kawawa-Beaudan, Borrajo & Veloso, 2026

TradeFM transfers **zero-shot** from US equities to Japanese and Chinese markets — the only entry in that table that tests transfer *off* its training corpus. That is the right experiment.

TradeFM sequences an instrument

quantile-binned price, volume, inter-arrival

One asset, many actors. Identity is the *index* of the series, so dropping it still leaves a complete sequence behind.

We sequence a person

that user's own actions, in order

One actor, many items. The items *are* the sequence, so dropping identity leaves nothing behind.

We both replace IDs with features.



Industrial foundation models

HSTU, Netflix, PRAGMA, nuFormer, TREASURE,

TransactionGPT, Stripe, TradeFM

- Real scale, real deployment
- One company's event stream
- Evaluated by fitting a head on labeled target data — **TradeFM excepted**
- Headline numbers are relative lifts on inaccessible data

⇒ scale helps *within* a platform.

Transferable recommenders

ZESRec, UniSRec, VQ-Rec, Recformer, RecGPT

- Genuinely test zero-shot
- Items carried by text, so they cross platforms
- Pretrained on 10^6 – 10^7 sequences of public catalogs
- Mostly one industry, mostly next-item only

⇒ right regime, but 10^{-6} – 10^{-4} of the scale.



Approach



The tempting move is to treat each item ID as a vocabulary entry. It fails for four reasons.

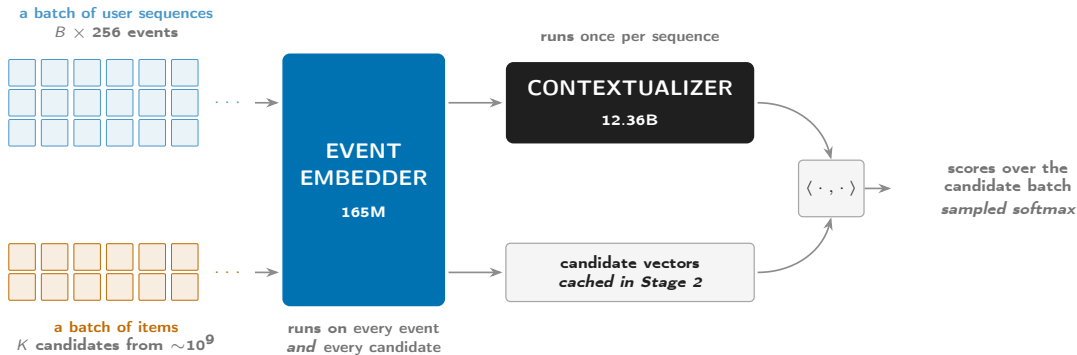
1. **IDs do not cross tracking boundaries.** A SKU means nothing outside the system that minted it, so an ID vocabulary can never be zero-shot.
2. **IDs pool no statistical strength.** Catalogs are brutally heavy-tailed; a tail item's embedding sees a handful of updates.
3. **Features give deeper, more generalizable intelligence.** Understanding features is deeper than understanding IDs.
4. **IDs give no handle for generation.** You cannot ask for an item that does not exist yet.

Instead

Every event is serialized as its **multimodal features** — description text, images, categorical fields, numerical values such as price.

A new catalog is embedded from features with **zero training**.

Two parts, and two different batches



The asymmetry that matters

Both batches pass through the **embedder**; only the sequence batch reaches the **contextualizer**. With 10^9 candidates, a parameter spent describing events is far more expensive than one spent modeling sequences — which is why the scaling laws ahead look nothing like a text LM's.



Stage 1 — learn to describe

Embedder and contextualizer train *jointly*, negatives from within the batch.

The embedder is shaped by the sequential objective on behavior, not by a reconstruction loss. End-to-end beat every frozen pretrained embedder we tried.

Stage 2 — learn to sequence

Embedder **frozen**, candidate vectors precomputed and cached. Only the contextualizer trains, against the full catalog.

Freezing is what makes 10^9 candidates affordable at all.

Pretraining is causal. Bidirectional queries — *which history preceded this outcome* — come from masked fine-tuning. Adaptation is LoRA on the contextualizer plus a trainable final layer: one base model, per-partner adapter swap.



	Layers / width	Parameters
Event embedder	8 / 768	165M
Contextualizer	64 / 4096	12.36B
Total		12.5B

1.25T

user actions

27.5T

tokens

 $\sim 10^9$

catalog entries

Three event families: **transactions** (purchases, returns, receipt lines); **interactions** (clicks, likes, searches); **contextual attributes** (market, store format, age band), which enter as CTRL-style *domains* rather than as extra sequence positions.

The scale comparison is against **BehaviorGPT-v3**, the previous generation: a **0.5B** model on a **similar mixture**, so the data is held fixed across a **25×** span in parameters — two successive generations rather than two ablations.



	Passes per request	Full-catalog ranking?
Semantic-ID generative rec.	$m \times b$ (40 at $m=4, b=10$)	Beam-limited
Prompted LLM ranker	T autoregressive tokens	No — in-context shortlist
LBM	1 + matmul vs. cache	Yes

Why this is load-bearing, not an engineering footnote

Both established routes to generality give up single-step retrieval. In payments and finance, high-frequency low-margin applications cannot absorb either an LLM's token budget or a beam search per request.

Generality that cannot be served at the latency of the decision is not generality you can deploy.



Three commitments

1. **Full-catalog ranking.** No sampled negatives at evaluation.
2. **Temporal splits.** Random splits leak the future.
3. **Disjoint users.** Splits are by user, so every user scored at test is one no model trained on.

Effective Recall@10

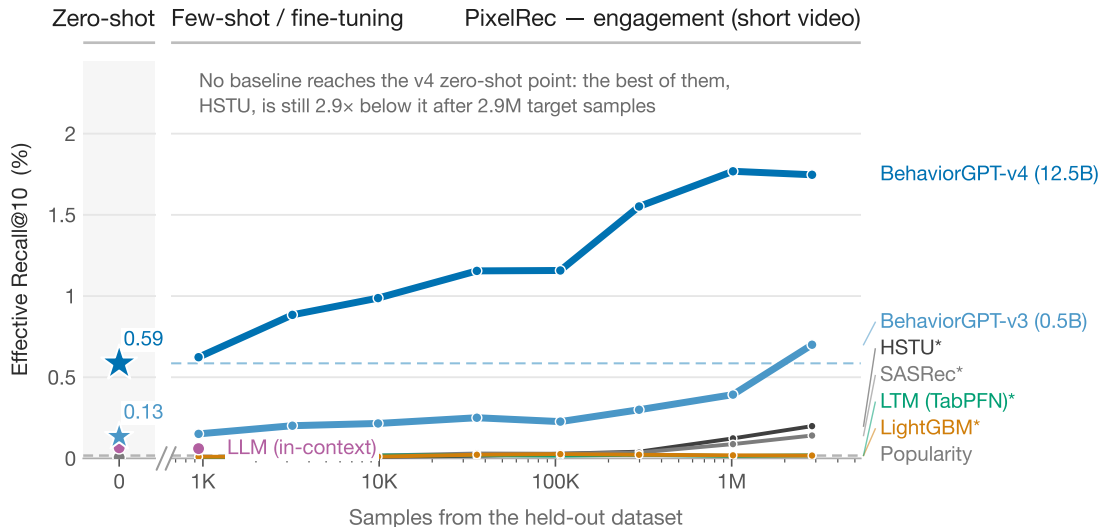
Being able to understand unseen items is often critical.

Preprocessing is identical across methods. We also report $\text{Recall}@_{\{10,50\}}$, $\text{NDCG}@_{\{10,50\}}$ and MRR. The harness will be public.



Evidence

P1: zero-shot transfer, and what it costs a specialist to catch up



*cannot rank items unseen in training · 78,796-item catalogue, 79,399 held-out targets



P1 — zero-shot transfer

Zero-shot, no target data	R@10
BehaviorGPT-v4	0.59%
BehaviorGPT-v3	0.13%
LLM in-context	0.065%
Popularity	0.018%

No baseline reaches the zero-shot point.

After 2.9M target samples HSTU is still 2.9× below a model that saw none.

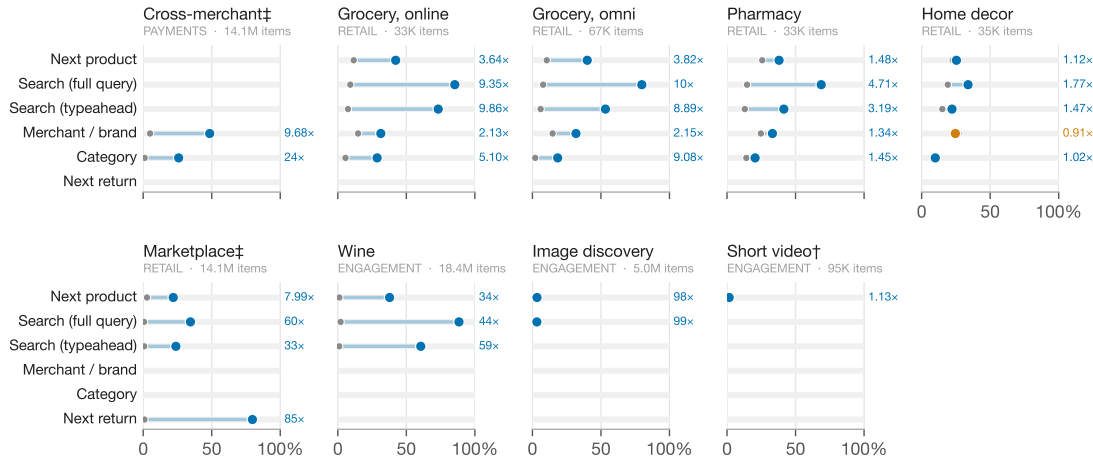
P3 — generality improving with scale

v4 is 4.3× v3 zero-shot on a corpus neither had seen — **same mixture, same protocol**, so the gap is scale. Short video is also an industry absent from pretraining, so this tests whether the multimodal route rather than language carries the transfer.

P2: breadth from one checkpoint



● BehaviorGPT-v4 (one checkpoint) ● Best specialist trained on that corpus



Recall@10 over the full catalogue, hit@1 for the attribute tasks, which hold the merchant on the cross-merchant stream and the product brand elsewhere.

Figure right of each row is $v4 \div \text{specialist}$; a blank row is a task that stream does not carry. [†] zero-shot transfer to an unseen catalogue. [‡] nine panels,

eight streams: Cross-merchant and Marketplace are one 14.1M stream, split by what each task needs — merchant and category from rail-level fields, the rest from the SKU.



What is being compared

- One checkpoint, **no** per-dataset architecture, objective or vocabulary change.
- Against the strongest specialist *for each corpus*, each trained on it — one model against a different opponent everywhere.
- 8 streams, 6 task families, **32 cells**.

Ahead on **31 of 32**. Median **6.6×**, max 99×, min 0.91× — and the one loss is on the slide because it is the kind of cell a favourable aggregate would quietly drop.

Recall@10 over the full catalog for ranking tasks, hit@1 for the attribute tasks. The attribute row is the merchant on the cross-merchant stream and the product brand elsewhere. Partner corpora anonymized by industry and subdomain.

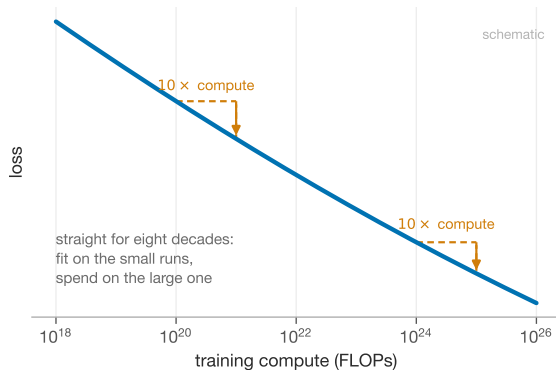


Area	Metric / baseline	Effect
Recommendations	CTR vs. internal recommender	+29%
	Conversion vs. most-bought-products	10×
Engagement	Clicks on featured content vs. control	+11.7%
	Orders after click	+19.5%
Search	Conversion vs. vendors A / B / C	+5.7 / +7.5 / +9.4%
	Conversion vs. vendor D	+16%
	Search-term suggestion interactions	+49.1%
	Time on site (3:17 → 4:03)	+23%
Merchandising	Dynamic categorization vs. market leader	+11%
	In-store sales, dynamic assortment	+2.2%

Read these carefully. Vendors anonymized; deployments are partner-specific and span several years; each baseline is whatever that partner was running, and it differs from row to row. These are **not** controlled comparisons against a named product. They establish that the approach survives contact with production.



Scaling laws



One self-supervised objective, more data, more compute — and loss falls on a straight line for eight decades.

The straightness is the return: fit on cheap runs, commit to an expensive one.

Does behavior get the same deal?



Scaling laws turned LLM development from a sequence of ad hoc runs into an allocation problem.

Given a compute budget: how many parameters, on how much data, at what batch size, with which recipe?

Why this stack needs its own answer

The now-standard behavioral architecture — event embedder → decoder-only contextualizer — has **two knobs text LMs do not have**: how to split parameters between the two parts, and how many negatives to score against once the embedder is frozen.

These models are large enough that a mistake on either is expensive, and there has been **no Chinchilla-style guidance** for them.



$C = 6ND$ — **six FLOPs per parameter, per token.**

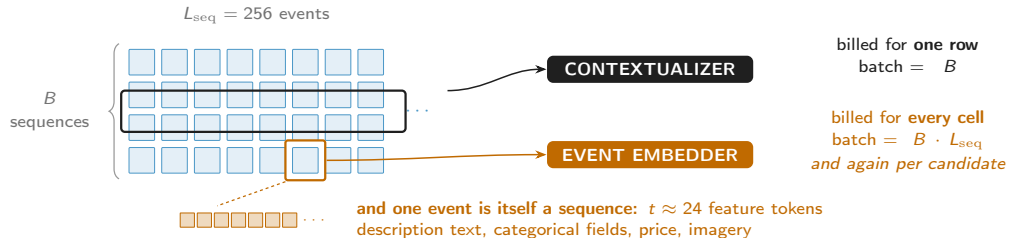
$$F_{\text{step}} = 6 \underbrace{B L_{\text{seq}}}_{\text{tokens in the step}} \underbrace{p}_{\text{parameters}}$$

Every parameter is visited once per token, so **parameter count and cost are the same quantity** up to a constant. That is why a text LM can be planned with two numbers.

Why an LBM needs a third

Our two parts do not see the same batch, and the part that dominates the bill sees one that is not measured in tokens at all.

The embedder's batch is not the model's batch (even without the candidate pool)





$$F_{\text{step}} = 6BL_{\text{seq}} \left(\underbrace{t p_e}_{\text{embedder}} + \underbrace{p}_{\text{contextualizer}} + \underbrace{3BL_{\text{seq}}h}_{\text{in-batch scoring}} \right) \quad N_{\text{eff}} = p + t p_e$$

Four axes, jointly varied

1. Embedder share $s = p_e/N$
2. Batch size B
3. Model/data allocation, N vs. D
4. Negative pool K after embedder freeze

~ 600 runs, $C \in [10^{15}, 10^{19}]$ FLOPs, 408 validation evaluations.

Real retail corpus, $\sim 10^8$ catalog. AdamW, bf16, FSDP,

$L_{\text{seq}}=256$.

The term that changes everything

$t=24$ is the embedder's context, the feature tokens one event expands into, so its parameters enter the FLOP count with a **multiplier of 24**.

Compute is not a function of parameter count alone — and that single fact drives the first result.



Full-vocabulary loss

Cross-entropy over the deployed catalog. Calibrated — but the *same ranking* sits at losses several nats apart depending on what it was scored against.

Ranking — Recall@{10, 50}, NDCG, MRR

What deployment is actually judged on. Blind to calibration, sensitive to the very top of the list.

Coverage

How much of the catalog is ever ranked highly. A diversity check: a model can win Recall@10 while collapsing onto the head.

Entropy

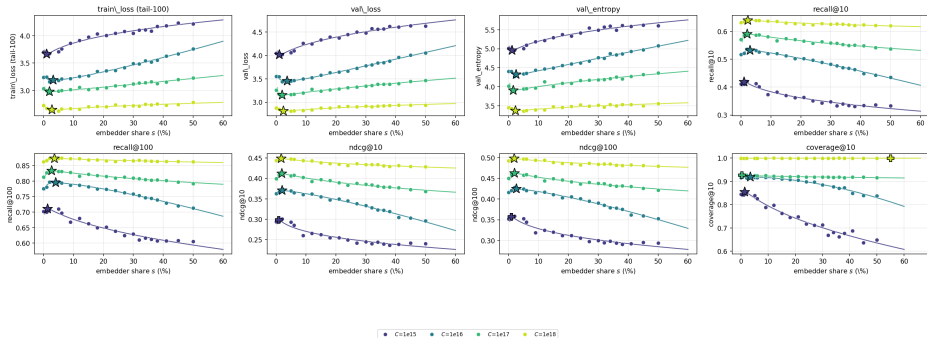
Diagnostic on the sequence model itself, not a deployment target.

The scaling laws do depend on the metric to some extent

Axis 1 — how much of the model should describe events?



Phase 1W embedder-share sweep, per metric (single LR per cell = train-loss-best; smooth fits = signed two-term starvation model $\text{sign} \cdot y(s) = E + as^\alpha + bs^{-\beta}$; * = analytic s^* inside swept range, + = s^* extrapolated)

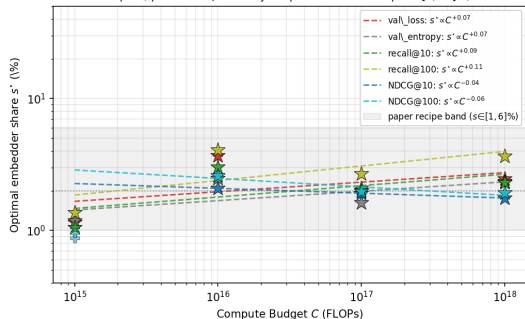


$$\text{sign} \cdot y(s) = \underbrace{as^\alpha}_{\text{contextualizer starved}} + \underbrace{bs^{-\beta}}_{\text{embedder starved}} + E, \quad s^* = \left(\frac{b\beta}{a\alpha} \right)^{1/(\alpha+\beta)}$$

$s^* \approx 2\%$, and it does not move



How s^* scales with compute, per metric (* = analytic optimum inside swept $s \in [1, 50]\%$, faded + = boundary)



Across **four decades of compute** the optimal embedder share sits in a narrow band around $s^* \approx 2\%$ (val-loss range 1.1–3.7%).

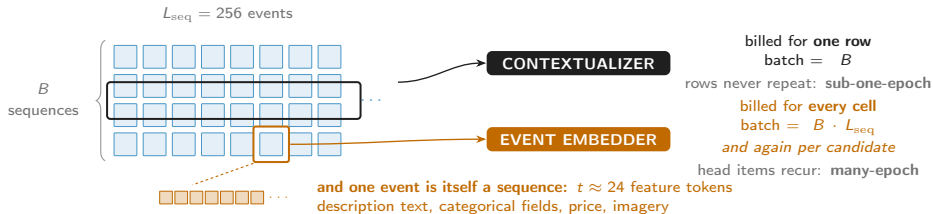
$s^* \propto C^{+0.07}$ — indistinguishable from zero.

An independent depth sweep re-reads it at 1.80–1.82%.

Flat within 2–6% ($\Delta \leq 0.06$ nat). Going 6% $\rightarrow$ 27% costs ~ 0.40 nat at 10^{15} .

The opposite of multimodal encoders — LLaVA and Flamingo put tens of percent into the encoder.

Why so small? Return to the two batches



The $24\times$ is what makes comparable compute look like 2% of parameters

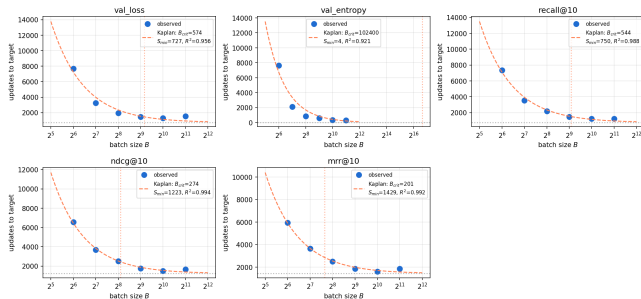
A parameter describing events is **billed** $24\times$, and its effective data is **exhausted first** (Less capacity needed for embedder). Equal cost on the two halves falls at $s = 1/(1+t) = \mathbf{4\%}$, inside the flat band — and at $s^* \approx \mathbf{2\%}$ the embedder still takes a third of the FLOPs. So read s^* as a **compute split**, not a parameter split.

Swept in Stage 1, negatives in-batch — so both effects come from the sequence grid alone, with no candidate pool.

Axis 2 — critical batch size depends on the metric



Phase 2 v3: per-metric Kaplan/McCandlish fits $S(B) = S_{\min}(1 + B_{\text{crit}}/B)$



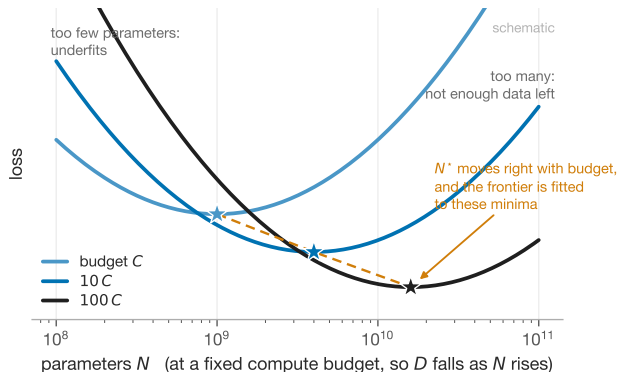
$$S_m(B) = S_m^{\min} \left(1 + \frac{B_{\text{crit}}^{(m)}}{B} \right)$$

Metric	B_{crit}
val_loss	574
recall@10	544
NDCG@10	274
MRR@10	201

A $\sim 3\times$ **spread** depending on what you optimize: the knee for a top-heavy ranking metric arrives far earlier than for loss.

Practical: $B \approx 2048$, $\text{LR} \approx 2 \times 10^{-3} \sqrt{B/512}$, cosine, 5% warm-up.

What “compute-optimal” means



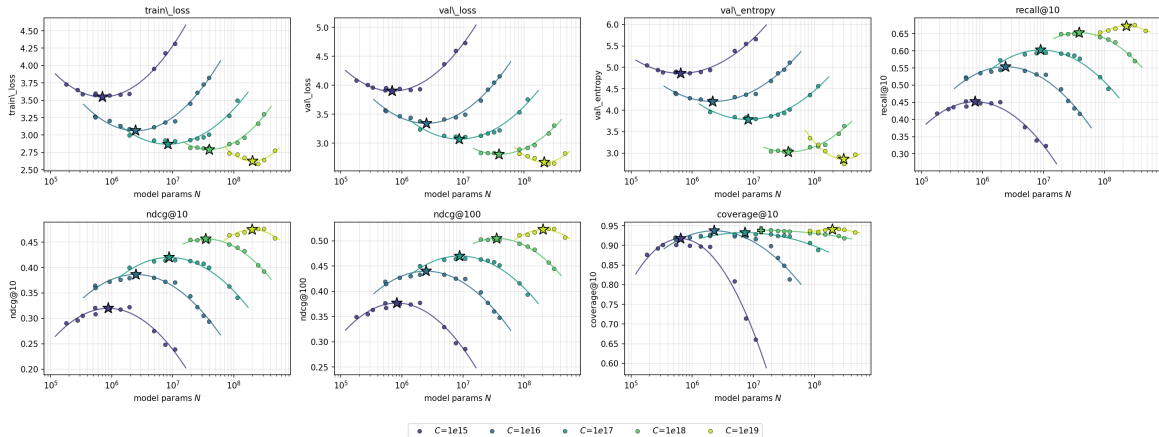
Hold compute fixed. Parameters and data now trade against each other: every parameter added is a token you cannot afford.

Sweep N at that budget and loss traces a **U** — underfitting on one side, too little data on the other. Its minimum is N^* , and D^* follows.

Repeat at several budgets and the minima trace the frontier $N^*(C)$.

For text: $N^* \propto C^{0.5}$, $D/N \approx 20$. The next slides ask what it is here.

Phase 3 iso-FLOP curves, per metric (metric value vs N along each compute budget; * = per-budget optimum)

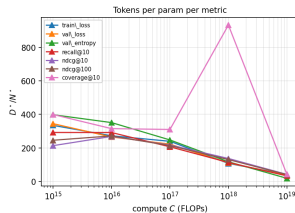
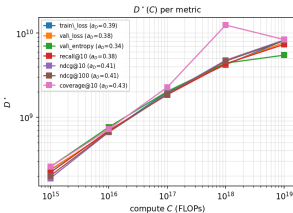
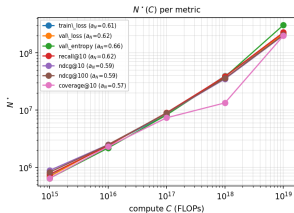


Iso-FLOP curves per metric, parabolic fits in $\log N$, analytic minimum per budget.

The frontier, and where it differs from text



Phase 3: how the Chinchilla frontier moves when we change the optimisation target



$$N^*(C) = 3.35 \times 10^{-4} C^{0.617 \pm 0.025}$$

C	10 ¹⁵	10 ¹⁶	10 ¹⁷	10 ¹⁸	10 ¹⁹
D/N	344	265	222	110	36

Parameter-heavier than Chinchilla (0.617 vs. ≈ 0.5), yet far more data-hungry at small budgets: D/N meets the text rule of thumb (≈ 20) only at the top of our range. Importing Chinchilla at 10^{15} – 10^{17} overbuilds the model by an order of magnitude.



Target metric	a_N	a_D
val_loss	0.617	0.383
val_ppl	0.617	0.383
recall@10	0.616	0.384
NDCG@10	0.586	0.414
coverage@10	0.574	0.426

± 0.025 to ± 0.079 .

Cross-budget exponents are **stable** — a_N within $[0.57, 0.66]$ for every metric.



Stage 1 — embedder training

Negatives are whatever else is *in the batch*. One more candidate means one more embedder pass, so the pool is capped by batch size at $\sim 5\text{--}10\text{K}$ — not a free choice.

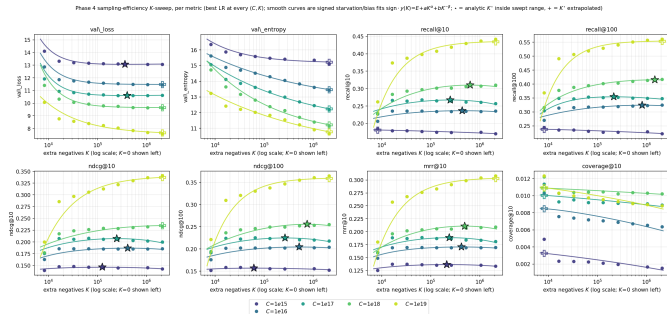
Stage 2 — embedder frozen

Every catalog vector is computed once and **cached**. A negative now costs one dot product against that cache, not an embedder pass — so we sample K from the **full catalog** — $\sim 10^8$ on the study corpus, and tractably 13.6M of it.

Which is what turns K into a scaling axis

Sampling from the deployed catalog rather than from the batch is what makes training match serving. It also makes the size of the negative pool a knob somebody has to set — and nothing in the FLOP budget says what it should be.

Axis 4 — negatives, and a constraint that is not FLOPs



$$y(K) = \underbrace{aK^\alpha}_{\text{starvation}} + \underbrace{bK^{-\beta}}_{\text{sampling bias}} + E$$

Starvation: per-step cost is linear in K ,

so at fixed C more negatives buys fewer steps. Bias: the sampled-softmax partition estimator has variance $\propto 1/K$.

Useful band $K \in [2.5, 9] \times 10^5$.

Loss is punished by sampling bias far harder than ranking is — another place the two disagree.

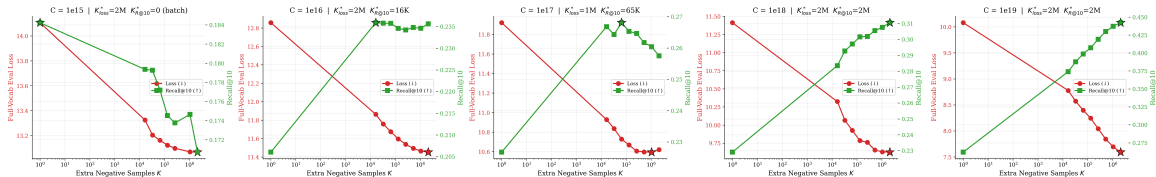
At $C = 10^{19}$

Every metric is **still improving** at the largest K we can train. The binding constraint becomes **memory**, not FLOPs.

The evaluation metric is part of the scaling law

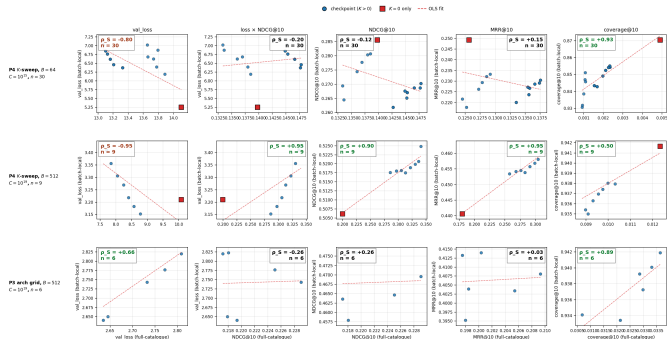


Loss-Ranking Divergence: K_{loss}^* vs K_{ranking}^*



Sometimes the metrics contradict each other outright

Batch-local vs. full-catalog: the agreement is not stable



Same checkpoints, scored both ways. Agreement swings from -0.95 to $+0.95$ depending on the metric, the batch size, the budget, and how much genuine quality spread the cells have — **loss does not even keep its sign**. Ranking metrics track the full catalog when cells differ in real quality, and stop tracking it when they are near-tied: **fine for triage, never for the final call**.



Scored against a candidate set $\mathcal{B}$ drawn from q , the model minimizes

$$\mathcal{L}(h, \mathcal{B}) = -\log \frac{\exp\langle h, e_{\theta}(y^+) \rangle}{\sum_{x \in \mathcal{B}} \exp\langle h, e_{\theta}(x) \rangle} \implies f_{\theta}^*(y | h) \propto \frac{p(y | h)}{\mathbf{q}(y)}$$

Unbiased only with a $-\log q(x)$ correction on every logit. Without it, the optimum is the true conditional *divided by the sampling distribution*.

Stage 1 — negatives from the batch

Every candidate is also somebody's target, so $q \propto p$ and the two cancel: the minimizer is **uniform on the catalog**. Popularity is **unidentifiable** — only within-batch rank order is learned.

Stage 2 — K uniform extras

$q \rightarrow$ uniform as K grows, so $p/q \rightarrow p$: the minimizer approaches the **true conditional**, and K sets how nearly.

Not a measurement artifact, then. The $1/q$ correction the in-batch objective never had to learn is what the full-catalog loss charges for.



Axis	Question	Swept	Result, and how much the metric moves it
Architecture	How large should the embedder be?	$s = 0\text{--}50\%$	$s^* \approx 2\%$, flat over four decades — and across metrics
Batch size	Where is the data-efficiency knee?	$B = 64\text{--}2048$	$B_{\text{crit}} = 201\text{--}574$ — a $3\times$ spread across metrics
Allocation	How does C split into (N, D) ?	$C = 10^{15}\text{--}10^{19}$	$N^* \propto C^{0.617}$, D/N from 340 to 36 — transfers, winners within 15%
Negatives	How many, once the embedder is frozen?	$K = 0\text{--}2\text{M}$	$K \sim 10^5\text{--}10^6$, memory-bound early — and the exponent itself differs , so the law does not transfer
Metric regime	& Do metrics and candidate pools rank cells the same?	cross-metric, batch-local full-catalog	vs. Agree within a regime; batch-local loss is not full-catalog loss

So pick the target metric *before* you fit, not after

Barely moves the embedder share or the allocation. Moves the batch knee $3\times$. On negatives it changes the exponent — there the law itself is metric-specific.

LR $\approx 2 \times 10^{-3} \sqrt{B/512}$, cosine, 5% warm-up, selected on *training* loss. Do not import Chinchilla below 10^{18} .



- **Your problems are behavioral.** Fraud, disputes, approval, credit — sequences of one customer's actions, not documents.
- **Your data is not language.** A catalog of merchants and instruments, not a fixed vocabulary, so events have to enter through features.
- **Which means an embedder**, and the moment you have one, the text-LM accounting stops applying: parameters cost $24\times$ on one side of the model and not the other.

And your metric is rarely perplexity

A language lab targets perplexity, which is what the loss already measures. You target fraud precision at a fixed alert budget, approval rate at fixed loss, dispute recall at fixed review capacity — all **top-of-list** metrics, which is exactly where we measured the recipe diverging: earlier batch-size knees, smaller models, more data, fewer negatives, and at high compute the wrong sign on the loss.



Conclusions



- P1 Zero-shot / few-shot transfer holds.** On a public benchmark whose catalog and users the model had never seen, no baseline reaches our zero-shot point; after **2.9M** target samples HSTU is still $2.9\times$ behind a model that saw none.
- P2 Breadth from one checkpoint holds.** **31 of 32** cells against the strongest specialist trained on each corpus, median **$6.6\times$** — merchant and category at rail level among them.
- P3 Generality improves with scale**, on the one genuinely out-of-distribution test we have: v3 to v4 is $4.3\times$ zero-shot, same mixture and same protocol. **Two sizes is a direction, not a curve.**



- The LLM moment for behavior is here.
- The siloed prediction failed, so the likely equilibrium is **a small number of general behavioral models** rather than one per company per function — which is what happened in language after GPT-3.
- In deployment I expect **composition, not replacement**: the LBM for the behavior, an LLM for the interface over it.

Rickard Brül Gabrielsson | Unbox AI | rickard@unboxai.com | www.unboxai.com



Thank you

`rickard@unboxai.com | www.unboxai.com`



What the evidence covers

- **Merchant** and **category** prediction on a cross-merchant transaction stream: 9.7× and 24× the best specialist, 14.1M catalog.
- Those are the fields a rail carries, so this is **rail-level** evidence rather than an analogy to it.
- Return prediction at 85× — structurally a dispute problem.
- Fraud indicators present as auxiliary labels.

What it does not cover

- ! **Retail merchants only.** No utilities, travel, subscriptions, P2P or cash-adjacent spend — so no test of *non-retail* merchant categories.
- ! **No issuer-side view.** No interchange, no authorization or decline stream, no cross-issuer coverage of one cardholder.
- ! No credit, risk, AML or treasury evaluation.

The honest form of the claim

Merchant and category on a cross-merchant stream is genuinely payments-shaped evidence, and it is strong. It is not evidence that the same holds once the merchant mix leaves retail, which is the part your data would settle.

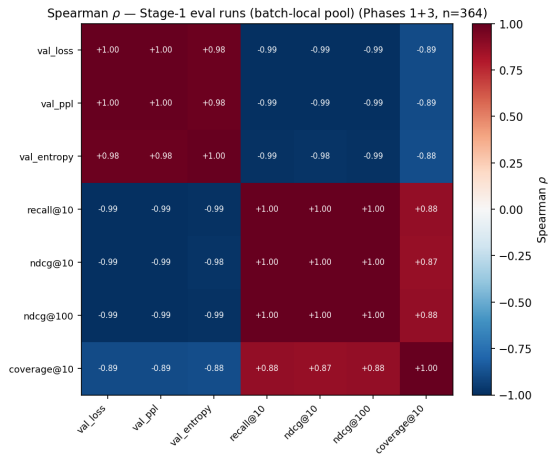


On the model claims

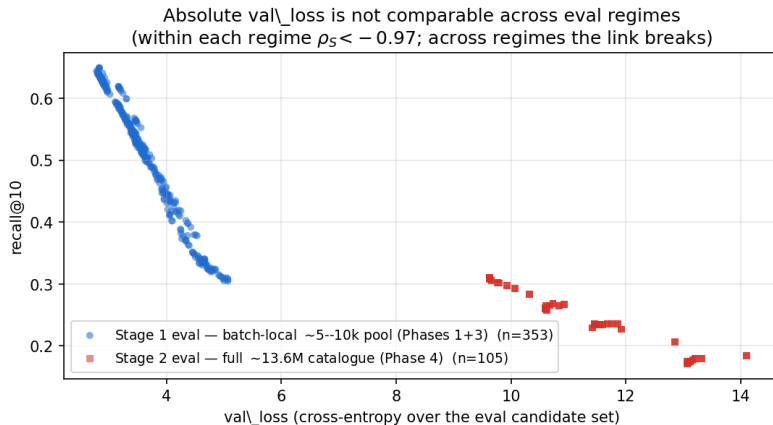
- P3 rests on **two model sizes** — a direction, not a fitted trend.
- Public-benchmark transfer is reported on **one dataset**; a second is evaluated but its v4 run is not final.
- Pretraining data is proprietary: you can evaluate the checkpoint, not reproduce it.
- Public benchmarks are proxies for deployment.
- We do not record corpus membership per evaluation.
- A/B baselines differ by partner and by year.

On the scaling laws

- Fitted three orders of magnitude below the production model.
- One corpus, one architecture family.
- Stage-1 evaluation is batch-local, so absolute losses are not comparable across budgets.
- Training sequence length fixed at 256 — context-length scaling untouched.
- 13 of 25 cells in the K sweep are boundary extrapolations.



Stage 1 ($n=285$) and Stage 2 ($n=123$). Within a stage, loss and ranking agree at $|\rho_S| \geq 0.94$; coverage decouples and fades further as budget grows.



The same recall@10 sits at absolute losses ~ 7 nats apart. Almost all of the gap is the candidate-pool denominator: the partition function is computed over a $\sim 1500\times$ larger set.

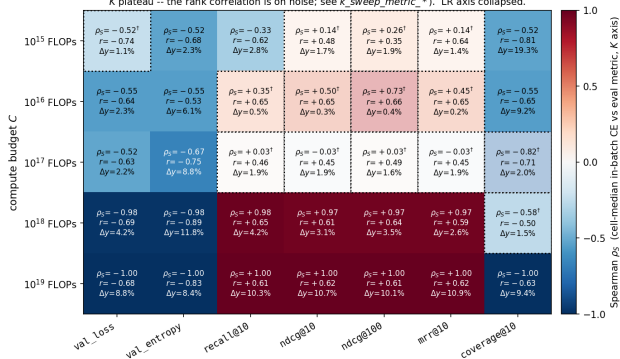
Backup — training loss stops being a surrogate



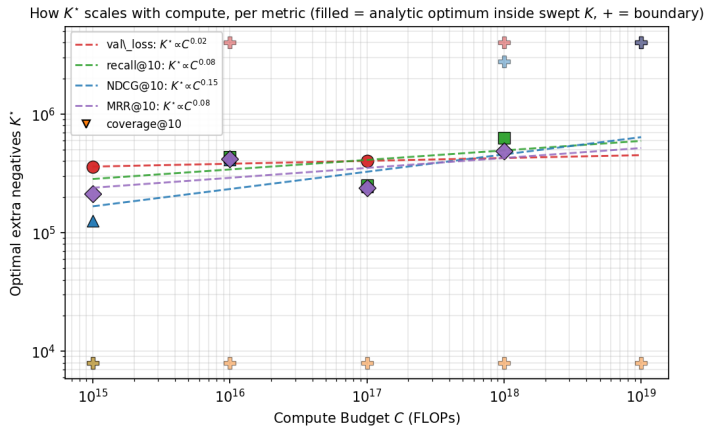
Stage-2 sampling-efficiency surrogacy: at each compute budget, how well does the cheap K -comparable in-batch training loss (fixed in-batch pool; not the K -dependent combined objective) track the K choices on the deployed full-vocab metric ($|V_2| \approx 13.6M$)?

Cell colour = Spearman ρ_S (rank); in-cell values give ρ_S , Pearson r (magnitude-aware sanity check -- low $|r|$ with high $|\rho_S|$ flags non-linear / non-monotone K -sweeps), and $\Delta y = \text{IQR} / |\text{median}|$ of the eval metric across the K axis ($K \in \{0, \dots, 2.1M\}$, $n=8-9$).

Cells with $\Delta y < 2\%$ are faded and dotted-outlined with † (eval essentially tied across the K plateau -- the rank correlation is on noise; see $k_sweep_metric_*$). LR axis collapsed.



In-batch training loss anti-tracks every deployed metric at high compute. Select checkpoints on it and you select the wrong ones.



Weak positive slope, $K^* \propto C^{0.08-0.15}$, fitted on interior cells only — 13 of 25 cells are boundary extrapolations, which is why I would not quote this exponent with confidence.

Grid-winner allocations (val_loss)

C	N^*	D^*	R@10	NDCG
10^{15}	532K	84M	0.453	0.302
10^{16}	1.95M	251M	0.544	0.380
10^{17}	10.9M	537M	0.594	0.414
10^{18}	19.4M	2.63B	0.649	0.455
10^{19}	251M	3.65B	0.673	0.476

Iso-FLOP reference cells, $s = 6\%$

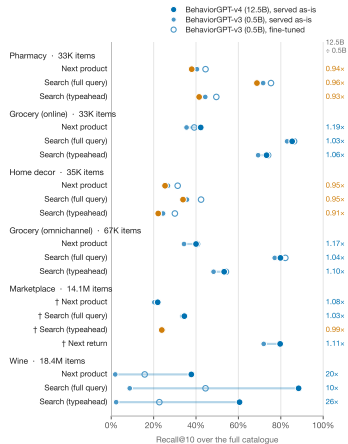
C	h_{ctx}	L_{ctx}	B	N
10^{15}	128	8	64	1.8M
10^{16}	232	8	128	5.5M
10^{17}	408	8	256	16.9M
10^{18}	452	24	256	60.7M

Recall@10 improves 49% relative and NDCG@10 58% relative across the four decades, so the gains are in the ranking metrics and not only in the loss.

Three-term fit to training loss (48 points, RMSE 0.084 nats):

$$L(N, D) = 2.570 + 2.46 \times 10^3 N^{-0.696} + 8.08 \times 10^2 D^{-0.384}.$$

Backup — scale comparison across corpora



† the 0.5B run covers only part of this cell's targets: 74% of next-product, 29% of returns. Each model is therefore scored on the targets it ranked rather than on a common set.



v4 ahead of v3 on 12 of 19 cells, both served as-is.

v3 wins seven cells, concentrated in pharmacy and home decor: next product, full-query search and typeahead search in each.

Those are corpora where the older checkpoint already has strong coverage, which is where the extra scale should buy least.

Both checkpoints train on the **same mixture** and are scored under the same protocol, so neither the data nor the metric explains the gap.

They are still successive *generations*, so size is not the only thing that changed — and 12 of 19 across two points is a direction, not a curve.



- **Domain conditioning.** Market, store format and age band as CTRL-style domains, costing no sequence positions, moved accuracy $0.35 \rightarrow 0.386$ ($\approx 10\%$) — concentrated on users with fewer than 50 events.
- **Separate softmax per distribution.** Mixing distributions in one softmax hurts cross-domain alignment. At 24K candidates, separating them beat *doubling* the candidate count twice over.
- **Test-time compute in behavior space.** Session-level next-purchase accuracy $13\% \rightarrow 25\%$ under additional inference-time search.
- **Method transfer to workforce data.** A 5M-parameter model on 43M workforce events reached 91% accuracy and $F_1 = 0.93$ on attrition — the method at small scale, not this checkpoint.
- **Embedders.** End-to-end training on behavior beat every frozen pretrained embedder we tried, including much larger ones.