ORGANIZING LLM INFERENCE FOR AGGREGATE PREDICTIONS FROM BEHAVIORAL DATA

Anonymous authors
Paper under double-blind review

ABSTRACT

When using a large language model (LLM) to simulate a user population, how should we combine individual behavioral histories to predict the fraction who take a particular action? We study this question in e-commerce, retrospectively predicting product-level conversion rates from browsing sessions without adapting the model to the task. We vary two design choices, each involving a bias–variance tradeoff. Summarizing a product's sessions before inference is economical but can discard predictive detail, adding bias, whereas judging sessions one at a time and averaging afterward keeps the detail but requires many calls to reduce variance. Evaluating products together in one prompt can refine the model's prior about what is typical in the setting, reducing bias, but also makes each judgment depend on which other products are sampled to share the prompt, adding noise. We evaluate both choices across five LLMs and two datasets. Our main finding is that judging individual sessions outperforms judging summaries in isolation, but once both are compared within a prompt, summaries lead: comparative summaries achieve the highest Spearman correlation with observed rates in 12 of 15 settings, with no alternative significantly better. Comparison thus helps summaries far more than session judgments, overcoming the advantage of behavioral detail. We interpret this disparity through a Bayesian model of the second tradeoff, which suggests that comparison helps more when the inputs in a prompt differ more across products, consistent with what we observe across four kinds of comparative context. Comparative summaries also require a small fraction of the inference cost of the session methods, and with the best model for each outcome they attain 66–89% of the correlation of the best supervised benchmark trained on thousands of labeled products.

1 INTRODUCTION

When a population-level prediction can draw on many individual behavioral observations, how should a large language model (LLM) leverage the evidence? E-commerce offers a natural setting for this question. Firms making product-level pricing, promotion, and inventory decisions often lack timely and relevant outcome data, especially when strategies are new or market conditions change quickly. At the same time, they may observe rich, recent customer behavior, e.g., browsing sequences, search activity, dwell time, and product consideration, that reveals customer intent and frictions. Conventional supervised models typically require substantial task-specific labels to exploit such data. Obtaining those labels through live experimentation can be slow, traffic-intensive, or even infeasible for decisions such as pricing.

Recent work on LLM-based human simulation and market research raises a natural alternative: can pretrained models extract predictive signal from behavioral context without extensive application-specific training (Aher et al., 2023; Brand et al., 2024)? Doing so creates a basic inference-design problem with a natural bias–variance tradeoff. One approach is to prompt an LLM to predict outcomes for individual sessions and then aggregate those judgments. This preserves fine-grained behavioral detail, but can require many model calls per product and incurs estimation variance from finite sampling. Another approach is to aggregate the behavioral evidence first and prompt the model to predict directly from a product-level summary. This greatly reduces inference cost and can incorporate much more historical evidence, but compression may discard predictive information and introduce bias. A second design choice is whether products are evaluated independently or jointly, and it

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

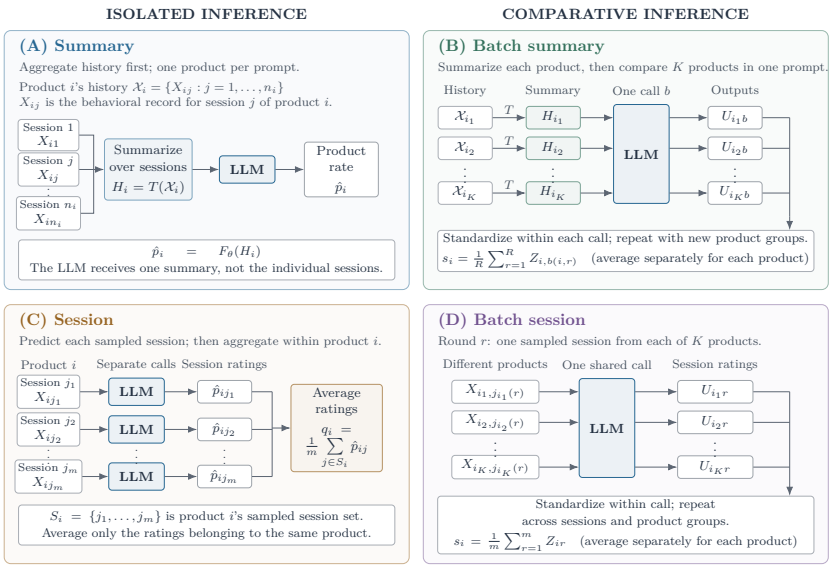


Figure 1: The four main methods, by evidence representation (rows) and prompting context (columns). Summary and Batch summary aggregate sessions before inference; Session and Batch session aggregate session-level judgments afterwards.

carries its own bias–variance tradeoff. Evaluated alone, a product is judged against the model's prior assumptions about what is typical, which need not describe the setting at hand. Placing other products in the same prompt can correct those assumptions, but it also shifts each judgment in ways unrelated to the product. This paper studies how evidence representation, comparative context, and inference budget jointly determine aggregate prediction quality.

We study these choices using two independent e-commerce datasets: Coveo, with approximately 4.93 million anonymized sessions (Tagliabue et al., 2021; Coveo Solutions Inc., 2021), and YooChoose, with approximately 9.25 million sessions (Ben-Shimon et al., 2015). We mask outcome fields and predict product-level add-to-cart and purchase rates on Coveo and purchase rates on YooChoose (where add-to-cart is not available). Inputs contain behavioral signals but exclude product titles, descriptions, images, and semantic embeddings. Prompts contain no labeled examples, and the models receive no task-specific tuning. We develop the procedures using GPT-5.6 on a separate Coveo discovery pool, then evaluate five LLMs on 1,000 held-out Coveo products and replicate the study on YooChoose. The task is retrospective: we predict rates observed under the existing offer. This on-policy prediction is useful in itself when outcomes are sparse, delayed, or unlogged, as for rarely purchased products, and it provides observed targets for studying how to organize inference. Predicting counterfactual outcomes under interventions, our main motivation, involves the same design choices, but testing them there requires experimental data, which can be difficult to obtain.

The design space has two axes, shown in Figure 1. First, aggregation can occur before or after model inference. Summary methods compute product statistics from the complete history of a product's recorded sessions, and request one product-level prediction; session methods obtain a judgment on each of 80 sampled sessions and average these judgments (sample size was set on the discovery pool). Second, prompts can be isolated or comparative. Isolated prompts contain one product, whereas comparative prompts present several products jointly. Comparative prompting is known to improve LLM ranking in information retrieval relative to pointwise scoring (Sun et al., 2023; Qin et al., 2024; Zhuang et al., 2024), while batch prompting can also reduce inference cost (Cheng et al., 2023). Applying comparison to both representations lets us separate the effect of prompting context from the choice of where aggregation occurs.

Because these zero-shot procedures are not calibrated to the task, we evaluate ranking by the Spearman correlation between predicted and observed product-level rates, and leave the accuracy and calibration of predicted levels to future work. Ranking is what many retail decisions require, such as choosing which products to promote or restock. The gains from comparison hold under Pearson correlation (Appendix G).

Across five models and three dataset–outcome pairs (Coveo add-to-cart, Coveo purchase, and YooChoose purchase), we find three consistent patterns.

1. **Session-level inference is stronger in isolation.** When products are evaluated independently, session-level prediction outperforms summary prediction in 14 of 15 settings, with one statistically indistinguishable result and no reversals.
2. **Comparative context disproportionately benefits summaries.** Evaluating 40 product summaries jointly outperforms evaluating them one at a time in all 15 settings. The same intervention has mixed effects on session inference: comparative session prediction improves in 3 settings, all on YooChoose, worsens in 5, all on Coveo, and is indistinguishable in 7. The gain from comparison is larger for summaries than for sessions in 14 of 15 settings.
3. **Comparative summaries provide the strongest accuracy–cost tradeoff.** Comparative summaries outperform isolated session inference in 10 of 15 settings and comparative session inference in 11, with no statistically significant reversal against either. Comparative summaries also need only 125 model calls per model–dataset pair, versus 2,000 for comparative and 80,000 for isolated session inference.

To summarize, our main finding is that comparison helps prediction with product summaries much more than prediction with individual sessions, to the extent that it overcomes the advantage of behavioral detail. To explain this disparity, we propose a Bayesian model in which the other summaries or sessions in a prompt provide evidence about the environment in which a prediction is made (Section 5.2). Conditioning on this evidence can improve interpretation of the focal input and reduce bias, but it also makes the prediction depend on which others are shown, adding noise. The balance depends on the input representation: interpreting a summary statistic may require knowing what is typical across products, which comparison can supply, whereas an individual behavioral trace may have a similar interpretation across environments. We use this distinction to interpret the different gains from comparing summaries and sessions.

Finally, we compare the zero-shot LLM procedures with three supervised benchmarks (ridge, random forest and gradient boosting) trained on thousands of labeled products, and with a single summary statistic selected on the discovery pool and used directly to rank products. While the best supervised benchmark significantly outperforms the best LLM method for every outcome, the best LLM method attains 66% of its Spearman correlation on Coveo add-to-cart, 89% on Coveo purchase, and 85% on YooChoose purchase. The single-feature rule outperforms every LLM method on Coveo add-to-cart, whereas the best comparative summaries exceed it on both purchase outcomes.

The broader lesson, within the procedures and budgets we test, is that preserving more behavioral detail is not sufficient for strong aggregate LLM prediction. Under finite inference budgets, comparative judgment based on summaries can outweigh the benefits of finer-grained inputs. Aggregate prediction with LLMs thus involves two bias–variance tradeoffs: where to aggregate the evidence and what context accompanies each judgment. Evidence representation, prompting context, and inference cost should therefore be studied jointly.

2 Related work

Aggregate prediction from collections. A product's conversion rate is a class proportion, connecting our task to quantification (Forman, 2008). HistNetQ predicts proportions from collections using prevalence labels (Pérez-Mon et al., 2025); distribution regression likewise learns from sampled distributions (Szabó et al., 2016). Wolf et al. (2026) ask whether population-level probabilities elicited from an LLM agree with weighted subgroup estimates, and find that subgroup aggregation can improve agreement with human data. Our question is instead how to organize observed behavioral records for prediction, by summarizing them before inference or aggregating predictions afterward, and how comparative prompting changes which is better.

Comparative prompting and inference efficiency. Our cross-product prompts connect to listwise and pairwise ranking (Sun et al., 2023; Qin et al., 2024) and reference-based pointwise scoring (Long et al., 2025). Batch prompting answers several inputs per call to save cost (Cheng et al., 2023), and BatchPrompt ensembles over input permutations to reduce batching errors (Lin et al., 2024). We build on these techniques rather than claim comparison or batching as contributions, and apply comparison to both summaries and sessions so that its effect is separated from the choice of representation.

Contextual inference and calibration. Our interpretation draws on Bayesian accounts of in-context learning (Xie et al., 2022) and evidence that demonstrations convey input-distribution and formatting information (Min et al., 2022). These motivate, rather than establish, our prior-refinement hypothesis. Contextual calibration estimates label biases from a content-free input (Zhao et al., 2021) and batch calibration from a batch of inputs (Zhou et al., 2024), much as our within-call standardization removes a level shared by a call, though we use it only for ranking.

Behavioral simulation and prediction from records. Predicting shopper responses connects our study to human simulation and market research (Aher et al., 2023; Brand et al., 2024). Shopping benchmarks assess individual behavior (Lu et al., 2026; Wang et al., 2026), while demand simulators aggregate persona predictions using fitted weights (Huang & Wang, 2026), and SimGym simulates A/B tests with traffic-grounded agents (Li et al., 2026). Our summaries are serialized table rows, as in TabLLM (Hegselmann et al., 2023), but each aggregates many sessions of one product, and we compare them with the raw session records. Predicting responses to interventions raises causal challenges (Gui & Toubia, 2025) that do not apply to our retrospective task.

3 Data and evaluation

We evaluate the different inference methods on 1,000 products from each of two public e-commerce datasets. The datasets record different information, and we keep those differences when constructing the inputs.

Coveo. We use the browsing logs released for the Coveo Data Challenge at the SIGIR 2021 eCom workshop (Tagliabue et al., 2021; Coveo Solutions Inc., 2021). We select 1,000 products with at least 100 observed sessions, stratified by number of sessions and price bucket and disjoint from the 280-product discovery pool on which the design was fixed (Section 4). We predict both add-to-cart and purchase.

YooChoose. We use the browsing and purchase logs released for the RecSys Challenge 2015 (Ben-Shimon et al., 2015), which record purchases but no add-to-cart events. The logs record a price only when a product is bought, so we take prices from purchases before August 1, 2014, and behavioral records and target rates from the window afterward; otherwise, prices would reveal which products were purchased. We select 1,000 products with at least 300 sessions in the later window and a priced purchase in the earlier one. We predict the purchase rate.

In both datasets, we mask the focal outcome fields in the prompts. On Coveo, the remaining excluded fields can include time after the focal outcome, whereas YooChoose sessions end at their first purchase; the task is a retrospective reconstruction of observed rates (Appendix F).

Models and coverage. We use GPT-5.6, Qwen3.8, Gemini3.8 Flash, GLM5.3 and Claude Sonnet5, at temperature zero and low reasoning effort. We evaluate each model on the full pool of 1,000 products with every method (Table 7; Appendix F).

Ranking and uncertainty. We measure the association between predicted and observed product ranks by Spearman's rank correlation ρ, with tied values assigned their average rank. Ties are common, since 30% of Coveo purchase rates are exactly zero, but breaking them by sort order instead changes little (Appendix G). We report 95% percentile intervals from 2,000 bootstrap resamples of the product pool. Within a dataset, the same resamples enter every method, model and outcome, so differences are paired. The intervals are not adjusted for multiple comparisons, and they treat the model outputs as fixed and products as independent (Appendix F); intervals for model means refer to these five models.

4 Organizing product-level inference

We define the four methods of Figure 1, labeled A (Summary), B (Batch summary), C (Session) and D (Batch session) in figures and tables.

The prediction target. Let i index products and j index the browsing sessions in which product i was viewed, and write $\mathcal{X}_i = \{X_{ij}\}_{j=1}^{n_i}$ for the product's full history, its n_i recorded sessions. The record X_{ij} describes one such session, for example how long the shopper viewed the product, what else they considered, and whether they arrived by search. Each record has a binary label Y_{ij} indicating

whether the shopper added product i to the cart (or, for the purchase outcome, bought it) in that session; the label is never shown to the models. We treat the sessions that view product i as draws from a product-specific population. The target is the product's conversion rate $p_i = \mathbb{E}_j[Y_{ij}]$, where the expectation is over a session j drawn from that population, and we evaluate predictions against its sample estimate, the observed fraction $\bar{Y}_i = n_i^{-1} \sum_{j=1}^{n_i} Y_{ij}$. We thus evaluate ranking by observed fractions in the recorded window, not by population rates or later outcomes. Every prompt asks for both an add-to-cart (where applicable) and a purchase rate, and we evaluate each against the outcomes the dataset records.

Averaging before the model. For Summary we reduce a product's sessions to a fixed vector of statistics $H_i = T(\mathcal{X}_i)$: means, medians, quartiles and proportions of viewing time and search activity, together with the viewing times, prices and counts of the products considered alongside it. The summary omits the product's traffic counts (its number of sessions and, on Coveo, of search sessions) for two reasons: the session methods do not see them, and in the matched variants that summarize the same 80 sessions the session methods use (Section 5.1), they are fixed by construction, so including them would make the gain from the full history partly reflect traffic volume. Our main findings hold even when we include them (Appendix C). Writing F_θ for the model asked to produce a product rate, the prediction is $\hat{p}_i = F_\theta(H_i)$.

Averaging after the model. For Session we judge each of $m = 80$ sampled sessions of one product in its own call, ask for the probability that the session converts, and average the m probabilities:

$$q_i = \frac{1}{m} \sum_{j \in S_i} \hat{p}_{ij}, \qquad \hat{p}_{ij} = f_\theta(X_{ij}), \tag{1}$$

where f_θ is the model asked to judge a single session and $S_i = \{j_1, \ldots, j_m\}$ is the sampled set.

Putting several products in one prompt. In Batch summary, we present $K = 40$ product summaries in one prompt and ask for all K rates, repeating over $R = 5$ random groupings of the 1,000 evaluation products. In Batch session, we present one sampled session from each of K products and ask for all K probabilities, repeating over $m = 80$ rounds so that every product contributes each of its sampled sessions once.

Standardization within calls. The outputs of one call can share a level and scale specific to that call. We hence standardize within each call before combining calls, which removes such call-level shifts. Let μ_b and σ_b be the mean and standard deviation of the outputs of call b. The score s_i averages

$$Z_{ib} = \frac{U_{ib} - \mu_b}{\sigma_b}, \qquad s_i = \frac{1}{R} \sum_{r=1}^{R} Z_{i,b(i,r)} \tag{2}$$

over the calls containing i, where $b(i, r)$ is the call holding i in grouping r and σ_b is replaced by one when it falls below 10^{-12}; Batch session averages the same way over its m rounds. We leave Session probabilities unstandardized, since each of its calls returns a single probability.

Judgments, calls, and the discovery pool. A product's prediction rests on one judgment for Summary, five for Batch summary, and 80 for both session methods. The design choices (which statistics the summary uses, how many sessions it is computed from, how many products share a prompt, and how the request is worded) were fixed on a separate discovery pool of 280 Coveo products with GPT-5.6, before any evaluation product was used; we report ablations on that pool in Appendix E.

5 Results

Figure 2 shows the four main methods for all five models; we use the labels A, B, C and D of Figure 1 when naming a paired comparison. All paired differences across settings, with confidence intervals, are in Appendix I.

5.1 Comparing the designs

Session judgments outrank summaries. Session averages judgments of 80 sampled sessions per product, whereas Summary is computed from the product's full history. To compare the two

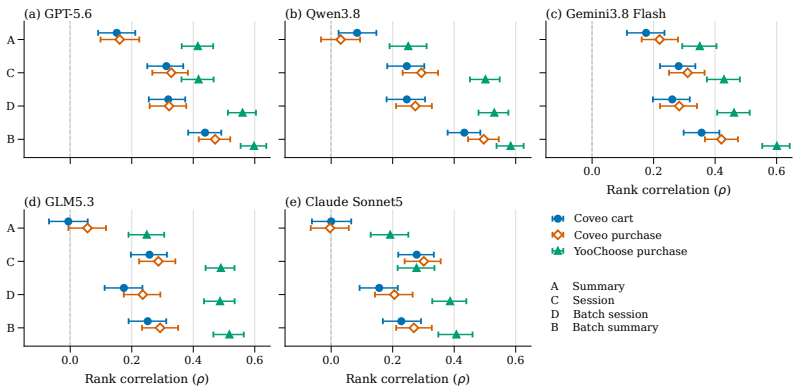


Figure 2: Rank correlations of the four main methods for five LLMs, with 95% product-bootstrap intervals. Each model uses its common support in a 1,000-product pool (Table 7); comparisons in the text use paired intervals.

representations on the same evidence, we recompute the summary statistics from exactly those 80 sessions (Summary-80, A80; Appendix B). Session outperforms it on both datasets: the mean Session−Summary-80 difference is 0.190 $[0.1467, 0.2288]$ for Coveo cart, 0.221 $[0.1784, 0.2610]$ for Coveo purchase and 0.130 $[0.0864, 0.1710]$ for YooChoose. The model-specific intervals lie above zero in 12 of the 15 settings, and none lies below zero. Adding the rest of each product's history to Summary-80, which gives Summary, does not close the gap: Session−Summary is 0.194, 0.211 and 0.132. However, the session advantage requires many judgments. When we recompute Session from $X \leq 80$ of its stored judgments per product (Figure 4 in Appendix M), accuracy rises steadily and reaches Summary-80 within 1 to 20 sessions on Coveo and, on YooChoose, within 2 to 56 sessions for four models; GPT's curve ends just below it, at 0.417 against 0.421.

Comparison greatly improves summaries. Batching and standardizing product-summary predictions improves ranking for every model and outcome on both datasets. The mean Batch summary−Summary gain is 0.260 $[0.2104, 0.3082]$ for Coveo cart, 0.297 $[0.2520, 0.3387]$ for Coveo purchase, and 0.250 $[0.2139, 0.2843]$ for YooChoose purchase, and all 15 model-specific paired intervals are above zero. The gain does not depend on standardization or averaging: a single comparative call per product, neither standardized nor averaged, obtains exactly one judgment per product, as Summary does, and already improves on Summary by 0.149, 0.179 and 0.127, above zero in all 15 settings. Standardization and averaging then add gains that largely overlap: each adds 0.06 to 0.09 on its own, and together they add 0.11 to 0.12 (Appendix K). The overall gain from batching is largest where isolated summaries work poorly: on Coveo, Sonnet rises from 0.000 and −0.004, no better than chance, to 0.228 and 0.269 on identical product statistics.

Comparison helps sessions less consistently. Batch session applies the same design to single sessions, one from each of 40 products per call, and its effect depends on the dataset. The mean Batch session−Session difference is −0.044 and −0.040 on the two Coveo outcomes and 0.062 on YooChoose. On Coveo, no model gains and five of the ten intervals lie below zero; on YooChoose, GPT, Gemini and Sonnet gain, GPT by 0.143 and Sonnet by 0.109, and Qwen and GLM do not. Compared directly, the gain from comparison is larger for summaries than for sessions in 14 of the 15 settings, by 0.30, 0.34 and 0.19 on average, and indistinguishable for GPT on YooChoose (Appendix L).

Batching a product's own sessions into one call lowers accuracy. Session is expensive, at 80 calls per product. The obvious way to save calls is to place a product's 80 sessions in one call and ask for 80 probabilities (Same-product batch), which keeps the requested judgments and uses one-eightieth of the calls. However, it ranks worse. On the same sampled sessions and under the same system message, it trails Session by 0.144, 0.161 and 0.104 on the three outcomes, with the paired interval above zero in 14 of the 15 settings; the exception is GPT on YooChoose (Table 21). Asking instead for a single product rate from the same call (Session bag) does no better (Appendix B).

Comparative summaries lead. Among the four main methods, Batch summary has the highest rank correlation in 12 of the 15 settings; in the other three, Session is higher, but not by a margin our intervals distinguish from zero. Batch summary leads Session by 0.066, 0.086 and 0.118 on the

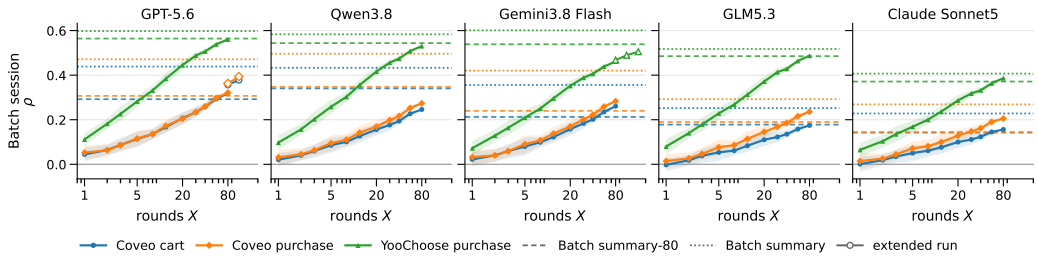


Figure 3: Batch session using X of its 80 rounds. Each point averages random draws of X stored judgments (band ± 1 sd). Dashed: Batch summary-80; dotted: Batch summary. Hollow markers: extended runs (Appendix N).

three outcomes, with the paired interval above zero in 10 of the 15 settings. Even one standardized grouping of Batch summary, which requests a single judgment per product, is ahead of Session's 80 judgments per product on average: the differences are 0.040, 0.055 and 0.056, the last two with intervals above zero (Appendix K).

Batch summary also leads Batch session, by 0.110, 0.126 and 0.056, but the lead comes from the full history: Batch summary-80, which applies the comparative procedure to summaries of the same 80 sampled sessions, differs from Batch session by 0.002, -0.019 and 0.015 on average, with significant model-level differences in both directions, and trails Session on Coveo by 0.042 and 0.059 while leading on YooChoose by 0.078 (Appendix I). Figure 3 recomputes Batch session from X of its 80 rounds. Accuracy rises with X, but Batch session needs 30 to 80 rounds to reach Batch summary-80, for the models that reach it at all, and no model reaches Batch summary within 80 rounds; doubling the sample to 160 sessions, for GPT on Coveo and Gemini on YooChoose, does not close that gap either (Appendix N).

Inference cost. Per model and dataset, Session makes 80,000 calls, Batch session 2,000 for the same judgments, and Batch summary 125. Session costs 1.6 to 3.2 times Batch session, and Batch summary-80, whose five-model mean is within 0.02 of Batch session's on the same 80 sessions, costs at most 12 to 18% as much for every model: requesting fewer judgments saves far more than batching them (Appendix P).

Supervised and single-feature references. We train ridge regression, a random forest, and gradient boosting on thousands of products outside the evaluation and discovery sets, using the traffic counts withheld from the LLM prompts. All of them lead every model's Batch summary on Coveo. The same holds for the tree ensembles on YooChoose (Appendix O). Nevertheless, the best Batch summary reaches 66% of the forest's correlation on Coveo add-to-cart, 89% on Coveo purchase, and 85% on YooChoose. The forest matches these levels with 15–25, 200–400 and 100–200 labeled products (Table 26). A single-feature rule fixed on the discovery pool, the average price of the most-considered competing product, beats every LLM method on Coveo add-to-cart but trails the best comparative summaries on both purchase outcomes (Table 25). The LLM methods thus fare best on purchase, the outcome that matters more in practice.

5.2 Representation and In-Context Inference: A Bayesian Perspective

Comparison improves on Summary in every setting, has mixed effects relative to Session, and lowers accuracy when all sessions in a call belong to one product. To explain these observations, we next propose a simple model of how the other summaries or sessions in a call change a prediction.

We call the summaries or sessions provided in one LLM call its *entries*. Let x_i denote the covariates of entry i, the input to one prediction. We read a zero-shot prediction from x_i alone as the model's estimate of $\mathbb{E}_\pi[p \mid x_i]$, the expected rate given the covariates under the model's prior π over environments. An *environment* θ describes a store's products, including each product's own characteristics, its shoppers, and how their browsing relates to carts and purchases; the online shop behind the Coveo data is one example. The prediction from x_i alone averages over environments, $\mathbb{E}_\pi[p \mid x_i] = \int \mathbb{E}[p \mid x_i, \theta]\, \pi(\theta \mid x_i)\, d\theta$. Let C denote a context supplied in the same prompt, for example summaries of other products, sessions of other products, or other sessions of the same product. Because θ includes each product's characteristics, C is independent of p given θ and x_i, so

with C added, the prediction becomes

$$\mathbb{E}_\pi[p \mid x_i, C] = \int \mathbb{E}[p \mid x_i, \theta]\, \pi(\theta \mid x_i, C)\, d\theta. \tag{3}$$

The context therefore changes the prediction only by updating the prior over θ, which we call prior refinement.

Refinement decomposition. We write $P_\theta(\cdot \mid x_i)$ for the distribution of the context C that our design assigns to entry i when its companions are drawn at random in environment θ. Then, for the environment θ that our data come from, we decompose the change from adding the context C as

$$\mathbb{E}_\pi[p \mid x_i, C] - \mathbb{E}_\pi[p \mid x_i] = m_\theta(x_i) + \eta_\theta(x_i, C), \tag{4}$$

where $m_\theta(x_i) = \mathbb{E}_{C \sim P_\theta(\cdot \mid x_i)}\big[\mathbb{E}_\pi[p \mid x_i, C] - \mathbb{E}_\pi[p \mid x_i]\big]$ is the average change over these draws and $\eta_\theta(x_i, C)$, which has mean zero under $P_\theta(\cdot \mid x_i)$, is the deviation for the companions actually drawn. We call m_θ the mean shift and η_θ the context noise. The mean shift need not be zero: the prediction without context, $\mathbb{E}_\pi[p \mid x_i]$, averages over the environments the model considers plausible, whereas our contexts come from the single environment behind our data.[1] The shift helps when it moves the prediction toward $\mathbb{E}[p \mid x_i, \theta]$, offsetting its bias. For an ideal Bayesian predictor, one that computes these expectations exactly, conditioning on C cannot increase the expected squared error on average over environments drawn from its prior. Because we score ranking, only differences in the shift across products matter.

Properties of the decomposition. For an ideal Bayesian predictor, the decomposition has two properties. *First*, the mean shift is nonzero only if (a) the prior given x_i, $\pi(\theta \mid x_i)$, does not already concentrate on a single environment and (b) the context is informative about the environment given x_i. To first order, the shift is the product of two factors: how strongly the prediction $\mathbb{E}[p \mid x_i, \theta]$ depends on θ, and how far conditioning on the context moves the posterior over the aspects of θ that matter for our evaluation. The first factor depends on the entry's representation. The interpretation of a summary statistic, such as a product's average viewing time, may strongly depend on the environment, whereas a repeated view or a search in a single session signals customer intent more or less similarly across environments. The second factor depends on the nature of the context. Because we evaluate rankings, what matters is how much the context informs how products compare. Variation across products within a call, which we call *task variation*, informs this most directly (Appendix R). Summaries of different products in one call have high task variation. Single sessions of different products likely have little, because sessions vary mostly from shopper to shopper rather than from product to product, and one product's sessions have none.

Second, the context noise comes only from which companions are drawn, so averaging over the n independent contexts in which a product appears divides its variance by n ($n = 5$ for Batch summary, 80 for Batch session and 1 for Same-product batch), while the mean shift m_θ does not change. Comparison therefore trades a possible reduction in bias, through m_θ, against added variance, through η_θ: a second bias–variance tradeoff, alongside the choice of where to aggregate. In this view, standardization and averaging reduce the context noise. Part of η_θ is shared by all entries in a call: a level set by the particular companions. Standardizing within the call (equation (2)) removes this shared level when the call includes several products.

The perspective in action. The results of Section 5.1 are consistent with the decomposition perspective in the following respects.

(1) **Gains follow task variation.** A helpful mean shift, and with it the gain from comparison, should rise with task variation, which we proxy by how informative a single entry is about its product's rate. We measure this informativeness as the rank correlation between the observed rates and a supervised score computed from one entry per product, cross-fitted by product, and order the three kinds of cross-product context by it (Table 31 in Appendix Q). For every outcome, the gain over the isolated counterpart rises with it, from -0.04 to $+0.06$ for single sessions of different products through $+0.15$ to $+0.21$ for summaries of 80 sessions to $+0.25$ to $+0.30$ for full-history summaries, whereas Same-product batch, which places one product's sessions

[1] By the law of iterated expectations, the expectation of the change over the prior is zero: $\mathbb{E}_{\theta \sim \pi(\cdot \mid x_i)} \mathbb{E}_{C \sim P_\theta(\cdot \mid x_i)}\big[\mathbb{E}_\pi[p \mid x_i, C] - \mathbb{E}_\pi[p \mid x_i]\big] = 0$. In other words, $m_\theta(x_i)$ averages to zero over θ drawn from $\pi(\theta \mid x_i)$, but not in general for the environment our data come from.

in one call, trails Session by 0.10 to 0.16. The same ordering fits the dataset difference in Section 5.1, where comparing single sessions helps on YooChoose but not on Coveo: on Coveo, only 13% (cart) and 10% (purchase) of the variance of a cross-fitted session score lies between products, against 41% on YooChoose (Appendix Q), so a call of 40 YooChoose sessions is closer to a comparison of products.

(2) **Precise summaries matter more after comparison.** If comparison refines the prior, more precise summaries should help more after that refinement, since precision matters more once the covariates are better interpreted. Computing the comparative summaries from full histories rather than from the 80 sampled sessions raises the rank correlation by 0.108, 0.145 and 0.041 (Batch summary−Batch summary-80), above zero in 14 of the 15 settings, while the same change leaves the isolated summaries essentially unchanged (Summary−Summary-80 is -0.004, 0.010 and -0.002).

(3) **Same-product context adds little and shares its noise.** A call of one product's sessions has no variation across products, so we expect it to provide little refinement that helps ranking (Appendix R). It does add noise: all 80 judgments share one context ($n = 1$), so the noise does not average out, and standardization cannot remove it without also removing the product's level needed for ranking. We therefore expect a net loss, which is what we observe (Section 5.1). On Coveo the shared noise is directly visible: when a product's 80 sessions are judged in one call, their predictions become much more correlated with each other, and the common part is unrelated to the product's observed rate (Appendix J).

(4) **Standardization and averaging do not drive the comparisons.** Standardization and averaging bring largely overlapping gains to the comparative summaries (Section 5.1), consistent with both reducing common context noise.[2] Standardization changes Batch session, which averages 80 calls, by less than 0.01 on average, with intervals containing zero (Appendix Q). Neither device changes the sign of any comparison: a single unstandardized comparative call already improves summaries in all 15 settings, and without standardization, Batch session still trails Session on Coveo by 0.038 and 0.035 and leads on YooChoose by 0.058.

Model-specific bias. LLMs may not be ideal Bayesian predictors and may respond to context in ways the decomposition does not predict. A context that says little about how products compare offers a check: for an ideal predictor it should produce little mean shift, so any loss it causes should shrink as more contexts are averaged. On Coveo, single sessions of different products are such a context (item (1)). There, Batch session stays within 0.035 of Session at every number of rounds $X \leq 80$ for GPT, Gemini and Qwen, but trails it increasingly for Sonnet and GLM, by 0.122 and 0.083 on cart and 0.096 and 0.051 on purchase at 80 (although 80 rounds cannot rule out noise that averages away more slowly; Appendix Q).

6 Conclusion

Our results point to two bias–variance tradeoffs in zero-shot aggregate prediction. The first is where to aggregate: a summary lets one judgment draw on a product's full history but discards detail, whereas judging sessions one at a time keeps the detail but needs many noisy judgments. The second is what context accompanies each judgment: other entries in the call can refine the model's prior about the environment, reducing bias, but add noise that depends on which entries are present. In our interpretation, the balance depends on how much entries vary across products: comparing summaries helps in every setting, whereas comparing single sessions, which vary mostly across shoppers, helps only on YooChoose, where they vary more across products. In practice, this favors comparing entries that differ mainly across the units being predicted, here products: comparative full-history summaries rank best, with an order of magnitude fewer judgments per product than the session methods.

Several caveats apply (Appendix F). The prior refinement is our interpretation, checked only through its observable consequences, and the ordering by task variation rests on four kinds of context per outcome (Appendix Q). We fixed the design choices on a discovery pool with GPT-5.6 alone and carried them unchanged to the other four models. Both datasets are e-commerce logs; records in other domains may vary differently across items and individuals, changing how much comparison helps. Finally, whether the ordering of methods carries over to responses to a changed offer, such as a new price, requires experimental data and is the most useful next step.

[2] Standardization also divides by the call's spread, so scores depend partly on the other products in the call.

ETHICS STATEMENT

We use anonymized behavioral data released for the Coveo and YooChoose research challenges. We do not attempt to identify customers or infer protected characteristics. We evaluate product rankings by predicted conversion rate, and we distinguish model predictions from observed customer responses. The records we sent to the model providers were these anonymized public data, and they contain no customer identifiers.

REPRODUCIBILITY STATEMENT

Both datasets are public (Section 3). We reproduce every system message and a rendered user message for the four methods and Session bag on both datasets in Appendix D, generated from the code that issued the calls rather than transcribed; the matched-evidence variants reuse the summary prompts. We give the pools and seeds, feature definitions, model endpoints and execution dates, decoding and retry settings, the coverage gate, the rank convention and the paired bootstrap in Appendix F, the supervised pipeline in Appendix O, and inference usage in Appendix P. Every number in the results tables and figures is produced by a script from the saved predictions, including the doubled-sample runs; Figures 3 and 4 re-aggregate them without new inference and, at full depth, reproduce the reported arms. Two limits apply. The models are served through provider endpoints rather than fixed snapshots, and their outputs vary between calls even at temperature zero, so a rerun will differ in individual judgments. Coveo's pool and session-sample seeds came from a process-dependent Python string hash, so the sample cannot be redrawn; we instead record the identifiers of the sampled sessions, from which a script rebuilds it. The anonymized supplementary material contains the code, call designs, product pools and sampled-session identifiers, every saved model prediction, and the outputs the paper was built from. It does not include the session samples or feature tables; its scripts rebuild them from the original logs, available from the dataset providers, and regenerate the reported tables, figures, and numbers. We will release these materials publicly upon publication.

STATEMENT ON THE USE OF GENERATIVE AI

We used generative AI tools (Claude and ChatGPT), separately from the models evaluated in the study, to execute the methodologies and perform analyses that help interpret results. We did not use them to generate data or to formulate or prove mathematical claims. We also used them to produce initial drafts of sections of the appendices, to edit for readability, and to identify and format references. We reviewed all AI-assisted work: every reported number is regenerated by script from the saved predictions, and citations were checked against the original publications. We take responsibility for the final content of this work.

REFERENCES

Gati V. Aher, Rosa I. Arriaga, and Adam Tauman Kalai. Using large language models to simulate multiple humans and replicate human subject studies. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 337–371, 2023. URL https://proceedings.mlr.press/v202/aher23a.html.

David Ben-Shimon, Alexander Tsikinovsky, Michael Friedmann, Bracha Shapira, Lior Rokach, and Johannes Hoerle. RecSys challenge 2015 and the YOOCHOOSE dataset. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pp. 357–358, 2015. doi: 10.1145/2792838.2798723. URL https://dbs-home-page.s3.amazonaws.com/dbs/papers/recsyschallengeyc2p.pdf.

James Brand, Ayelet Israeli, and Donald Ngwe. Using GPT for market research. In *Proceedings of the 25th ACM Conference on Economics and Computation*, 2024. doi: 10.1145/3670865.3673479.

Zhoujun Cheng, Jungo Kasai, and Tao Yu. Batch prompting: Efficient inference with large language model APIs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 792–810, 2023. doi: 10.18653/v1/2023.emnlp-industry.74. URL https://aclanthology.org/2023.emnlp-industry.74/.

Coveo Solutions Inc. SIGIR e-commerce data challenge: Dataset and documentation, 2021. URL https://github.com/coveooss/SIGIR-ecom-data-challenge.

George Forman. Quantifying counts and costs via classification. *Data Mining and Knowledge Discovery*, 17:164–206, 2008. doi: 10.1007/s10618-008-0097-y. URL https://link.springer.com/article/10.1007/s10618-008-0097-y.

George Gui and Olivier Toubia. The challenge of using LLMs to simulate human behavior: A causal inference perspective, 2025. URL https://arxiv.org/abs/2312.15524v3. Version 3, November 23, 2025; first circulated in 2023.

Stefan Hegselmann, Alejandro Buendia, Hunter Lang, Monica Agrawal, Xiaoyi Jiang, and David Sontag. TabLLM: Few-shot classification of tabular data with large language models. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pp. 5549–5581, 2023. URL https://proceedings.mlr.press/v206/hegselmann23a.html.

Chengpiao Huang and Kaizheng Wang. LLM-powered virtual population for demand simulation and pricing, 2026. URL https://arxiv.org/abs/2606.16183v1. Version 1, June 15, 2026.

Han Li, Vibhor Malik, Zahra Zanjani Foumani, Alberto Castelo, Shuang Xie, Ailin Fan, Keat Yang Koay, Yuanzheng Zhu, Meysam Feghhi, Ronie Uliana, Zhaoyu Zhang, Angelo Ocana Martins, Mingyu Zhao, Francis Pelland, Jonathan Faerman, Nikolas LeBlanc, Aaron Glazer, Andrew McNamara, Zhong Wu, and Lingyun Wang. SimGym: A framework for A/B test simulation in e-commerce with traffic-grounded VLM agents, 2026. URL https://arxiv.org/abs/2605.19219v1. Version 1, May 19, 2026.

Jianzhe Lin, Maurice Diesendruck, Liang Du, and Robin Abraham. BatchPrompt: Accomplish more with less. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/5d8c01de2dc698c54201c1c7d0b86974-Abstract-Conference.html.

Kehan Long, Shasha Li, Chen Xu, Jintao Tang, and Ting Wang. Precise zero-shot pointwise ranking with LLMs through post-aggregated global context information. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2025. URL https://arxiv.org/abs/2506.10859.

Yuxuan Lu, Jing Huang, Yan Han, Bingsheng Yao, Sisong Bei, Yaochen Xie, Yisi Sang, Qi He, and Dakuo Wang. Can LLM agents simulate multi-turn human behavior? evidence from real online customer behavior data. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 43961–43977, 2026. doi: 10.18653/v1/2026.acl-long.2034. URL https://aclanthology.org/2026.acl-long.2034/.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11048–11064. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.emnlp-main.759. URL https://aclanthology.org/2022.emnlp-main.759/.

Olaya Pérez-Mon, Alejandro Moreo, Juan José del Coz, and Pablo González. Quantification using permutation-invariant networks based on histograms. *Neural Computing and Applications*, 37:3505–3520, 2025. doi: 10.1007/s00521-024-10721-1. URL https://link.springer.com/article/10.1007/s00521-024-10721-1. Published online December 14, 2024; February 2025 issue.

Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, and Michael Bendersky. Large language models are effective text rankers with pairwise ranking prompting. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 1504–1518, 2024. URL https://aclanthology.org/2024.findings-naacl.97/.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. Is ChatGPT good at search? investigating large language models as re-ranking agents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 14918–14937, 2023. URL https://aclanthology.org/2023.emnlp-main.923/.

Zoltán Szabó, Bharath K. Sriperumbudur, Barnabás Póczos, and Arthur Gretton. Learning theory for distribution regression. *Journal of Machine Learning Research*, 17(152):1–40, 2016. URL https://jmlr.org/papers/v17/14-510.html.

Jacopo Tagliabue, Ciro Greco, Jean-Francis Roy, Bingqing Yu, Patrick John Chia, Federico Bianchi, and Giovanni Cassani. SIGIR 2021 e-commerce workshop data challenge. *arXiv preprint arXiv:2104.09423*, 2021. URL https://arxiv.org/abs/2104.09423.

Ziyi Wang, Yuxuan Lu, Wenbo Li, Amirali Amini, Bo Sun, Yakov Bart, Weimin Lyu, Jiri Gesi, Tian Wang, Jing Huang, Yu Su, Upol Ehsan, Malihe Alikhani, Toby Jia-Jun Li, Lydia Chilton, and Dakuo Wang. OPeRA: A dataset of observation, persona, rationale, and action for evaluating LLMs on human online shopping behavior simulation. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 43942–43960, 2026. doi: 10.18653/v1/2026.acl-long.2033. URL https://aclanthology.org/2026.acl-long.2033/.

Patrik Wolf, Thomas Kleine Buening, Andreas Krause, and Celestine Mendler-Dünner. Partition, prompt, aggregate: Statistical self-consistency in language models, 2026. URL https://arxiv.org/abs/2607.15277v1. Version 1, July 16, 2026.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit Bayesian inference. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2111.02080.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 12697–12706, 2021. URL https://proceedings.mlr.press/v139/zhao21c.html.

Han Zhou, Xingchen Wan, Lev Proleev, Diana Mincu, Jilin Chen, Katherine Heller, and Subhrajit Roy. Batch calibration: Rethinking calibration for in-context learning and prompt engineering. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2309.17249.

Shengyao Zhuang, Honglei Zhuang, Bevan Koopman, and Guido Zuccon. A setwise approach for effective and highly efficient zero-shot ranking with large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2024. doi: 10.1145/3626772.3657813. URL https://arxiv.org/abs/2310.09497.

A GUIDE TO THE APPENDICES

In Table 1, we list each appendix with the part of the main text it supports.

Table 1: Appendix Guide.

Appendix	Contents	Supports
Methods and inputs		
B	Diagnostic variants: Session bag and the matched-evidence summaries; method key	Sections 4, 5
C	Every method with the withheld traffic counts restored	Section 4
D	System messages and rendered user messages	Section 4
E	Discovery-pool ablations: summary statistics, wording, batch size, evidence depth, a mechanical summary	Section 4
Data and evaluation		
F	Pools, features, models, execution, coverage, rank convention, bootstrap, session sharing, observation window	Section 3
G	Ordinal tie-breaking, Pearson correlation, unstandardized batch outputs	Sections 1, 3
Results (Section 5.1)		
H	Rank correlation of every method, model and outcome	Figure 2
I	All 15 paired contrasts, by model and outcome, and Batch summary-80 against Session	Section 5.1
J	Batching a product's own sessions into one call	Section 5.1
K	Comparison, standardization and averaging in Batch summary; one grouping against Session	Section 5.1
L	Difference in comparative gains between the representations	Section 5.1
M	Accuracy of Session against the number of judgments per product	Section 5.1
N	Batch session on 160 sessions per product	Figure 3
O	Supervised benchmarks and the single-feature rule: training pool, fitting, paired comparisons, training-size curve	Section 5.1
P	Calls, tokens, charges and latency by model and logged block	Section 5.1
Interpretation (Section 5.2)		
Q	Informativeness of single entries, effect of standardization, session methods at equal depth	Section 5.2
R	Same-product and cross-product context	Section 5.2

B DIAGNOSTIC METHODS AND PREDICTION GRANULARITY

The diagnostic variants change one part of a main method at a time. Summary-80 and Batch summary-80 recompute the product statistics from the same 80 sessions used by Session and Batch session, so that summaries and sessions can be compared on the same evidence. Session bag gives the model the same 80 records as Session in one call and asks for a single product rate, so that the aggregation happens within the call rather than by averaging. Same-product batch (Appendix J) gives the model the same 80 records in one call but asks for a probability for each, which we average. Matching the evidence also removes information: with a fixed sample, `n_sessions` equals 80 for every product and Coveo's `n_search_sessions` is rescaled to the same 80, so the matched summaries carry no traffic volume. This is one of the two reasons the reported summaries withhold the traffic counts (Appendix C). Table 2 gives the stored result identifier for each method name and label.

Averaging session judgments or asking for one product rate. Session and Session bag receive the same sampled records of each product but differ in two respects: what the model is asked to produce, 80 per-session probabilities that we average or a single product rate, and how the records are batched, 80 calls or one. Session performs better, by 0.131 $[0.0876, 0.1704]$ for Coveo cart, 0.167 $[0.1242, 0.2054]$ for Coveo purchase and 0.173 $[0.1365, 0.2084]$ for YooChoose purchase, above zero in 14 of the 15 settings, with GPT on YooChoose the one null and none where Session bag is higher. Same-product batch, which batches the sessions as Session bag does but asks for a probability for each, separates the two respects. Holding the batching fixed in this way, asking for per-session judgments makes no difference on Coveo, at -0.013 $[-0.0475, 0.0216]$ for cart and 0.006 $[-0.0304, 0.0406]$ for purchase, with individual models split in both directions (per-session

Table 2: Method names and source identifiers. C denotes the main session method throughout the paper; the source arm named C is the diagnostic R.

Code	Method	Stored identifier	Role
A	Summary	`A`	Full-history summary; one product judgment
B	Batch summary	`B`	Full-history summaries; comparative product judgments
C	Session	`C_sess`	One session per call; average the 80 probabilities
D	Batch session	`D`	Cross-product sessions; average standardized judgments
A80	Summary-80	`A80`	Isolated summary from the sampled 80 sessions
B80	Batch summary-80	`B80`	Comparative summaries from those 80 sessions
R	Session bag	`C`	Same 80 records; one product judgment
—	Same-product batch	`solo_vs_batched`	Same 80 records in one call; average the 80 probabilities

judgments are worse for Gemini on cart and better for GLM on purchase). On YooChoose it helps, by 0.070 $[0.0382, 0.0997]$, above zero for Gemini, GLM and Qwen, the three models for which Session bag ranks lowest, and indistinguishable from zero for GPT and Sonnet. The remainder of the Session advantage comes from spreading the judgments across calls: 0.144 and 0.161 on Coveo, essentially all of it, and 0.104 on YooChoose (Table 21). Asking for one product rate instead of 80 averaged probabilities therefore lowers accuracy little beyond what batching the records into one call already does, except for three models on YooChoose.

Comparisons with matched evidence. Batch summary-80 improves on Summary-80 in all 15 model–outcome comparisons, with every paired interval above zero. Comparative processing therefore helps summaries even when their evidence is restricted to the sampled sessions. The mean Session−Summary-80 differences are 0.190 $[0.1467, 0.2288]$ for Coveo cart, 0.221 $[0.1784, 0.2610]$ for Coveo purchase, and 0.130 $[0.0864, 0.1710]$ for YooChoose purchase. These comparisons hold sessions fixed, while the records and summaries still differ in fields, representation, and the number of judgments averaged.

C Summaries with the traffic counts

The reported summary block withholds a product's traffic counts: its number of sessions, and on Coveo, its number of search sessions. They are withheld for two reasons. The session methods receive 80 sampled sessions per product and therefore cannot observe these counts, and the matched-evidence variants fix these counts by construction (at 80 sessions for every product), so a full-history increment computed with them present would mix the precision of the behavioral statistics with the appearance of a feature that ranks products on its own. We ran both specifications on every model and dataset, changing nothing else, and report them side by side in Table 3. Session and Batch session receive sessions rather than summary columns and are unaffected.

Restoring the counts lowers the isolated summary on Coveo, from 0.081 to 0.048 for cart and from 0.093 to 0.031 for purchase, even though the counts (session count and search-session count) rank products at 0.088 and 0.222 for cart and 0.174 and 0.300 for purchase on their own. It raises the isolated summary slightly on YooChoose, from 0.291 to 0.309, where session count ranks at only 0.034. No ordering among the methods changes on either dataset. Two YooChoose results are weaker with the counts present and lose their separation from zero: the full-history increment Batch summary−Batch summary-80, at 0.020 against 0.041, and Batch summary−Batch session, at 0.027 against 0.056. Every other contrast keeps its sign and its significance.

D Prompts

Every prompt is a system message that fixes the analyst role and the output schema, followed by a user message that carries the evidence. Both are reproduced below, one method at a time, with the system message verbatim and the user message as it was rendered for the first call of the reported run. Long user messages are elided in the middle; the elision is marked and says what was removed.

Table 3: Summaries without and with the two traffic counts, mean of five models; the counts-free block is the reported specification. The session methods do not read the summary columns and are identical across blocks. A star marks a paired interval excluding zero.

	Coveo cart		Coveo purchase		YooChoose purchase	
	free	counts	free	counts	free	counts
Arm levels						
Summary	0.081	0.048	0.093	0.031	0.291	0.309
Batch summary	0.341	0.338	0.390	0.376	0.541	0.512
Summary-80	0.085	0.064	0.083	0.064	0.293	0.313
Batch summary-80	0.233	0.226	0.245	0.245	0.501	0.493
Paired contrasts						
C−A	0.194*	0.227*	0.211*	0.273*	0.132*	0.114*
C−A80	0.190*	0.212*	0.221*	0.240*	0.130*	0.110*
B−A	0.260*	0.289*	0.297*	0.345*	0.250*	0.203*
B−C	0.066*	0.063*	0.086*	0.072*	0.118*	0.089*
B−D	0.110*	0.107*	0.126*	0.112*	0.056*	0.027
B80−D	0.002	-0.005	-0.019	-0.019	0.015	0.007
B−B80	0.108*	0.112*	0.145*	0.131*	0.041*	0.020

Nothing here is transcribed. The prompts are rendered by the same code that issued the calls, applied to the same pools, session samples, features and product groupings as the reported runs, so the text below matches what was sent. Three details are visible in the renderings. Batch session (D) gives the model one session from each of 40 different products and does not say which session belongs to which product, since each product contributes exactly one. Session (C) and Session bag (R) read the same 80 sessions of one product and differ in how they are batched and in what they ask for: one session per call and a probability for it, against all 80 in one call and a single product rate. Their system messages differ in that one sentence and nowhere else. The summary methods list the same feature block in the same order, the 18 columns of Appendix F, with absent values written `null`.

Summary (A). System message:

```
You are an e-commerce conversion analyst. You will be given aggregate behavioral statistics
↪ for one product. Estimate the fraction of browsing sessions containing this product that
↪ end in an add-to-cart, and the fraction that end in a purchase. Return ONLY a JSON object
↪ of the form {"cart_rate": <float>, "purchase_rate": <float>}. No prose.
```

User message, Coveo, from the reported run:

```
=== PRODUCT ===:
  price_bucket: 9
  avg_focal_dwell: 16.99
  median_focal_dwell: 0
  p25_focal_dwell: 0
  p75_focal_dwell: 16.9
  search_rate: 0
  avg_search_rank: null
  median_search_rank: null
  avg_comp_top1_dwell: 66.99
  avg_comp_top2_dwell: 30.59
  avg_comp_top3_dwell: 22.78
  median_comp_top1_dwell: 30
  avg_comp_top1_price: 8.16
  avg_comp_top2_price: 8.24
  avg_comp_top3_price: 8.16
  median_comp_top1_price: 9
  avg_n_competitors: 2.79
  median_n_competitors: 1

Predict cart_rate and purchase_rate for this product. JSON object only.
```

User message, YooChoose, from the reported run:

```
=== PRODUCT ===:
  price: 229
  avg_focal_dwell: 151.4
  median_focal_dwell: 43.1
  p25_focal_dwell: 0
  p75_focal_dwell: 120.2
  special_offer_rate: 0.6677
  category_click_rate: 0.3323
```

```
  brand_click_rate: 0
  avg_comp_top1_dwell: 383.6
  avg_comp_top2_dwell: 183
  avg_comp_top3_dwell: 146.3
  median_comp_top1_dwell: 179.9
  avg_comp_top1_price: 2976
  avg_comp_top2_price: 891.7
  avg_comp_top3_price: 1250
  median_comp_top1_price: 329
  avg_n_competitors: 4.486
  median_n_competitors: 2

Predict cart_rate and purchase_rate for this product. JSON object only.
```

Batch summary (B). System message:

```
You are an e-commerce conversion analyst. You will be given aggregate behavioral statistics
↪ for several products. For each, estimate the fraction of browsing sessions containing it
↪ that end in an add-to-cart, and the fraction that end in a purchase. Return ONLY a JSON
↪ array, one object per product, of the form {"product": <int>, "cart_rate": <float>, "
↪ purchase_rate": <float>}. No prose.
```

User message, Coveo, from the reported run:

```
=== PRODUCT 0 ===:
  price_bucket: 9
  avg_focal_dwell: 16.99
  median_focal_dwell: 0
  p25_focal_dwell: 0
  p75_focal_dwell: 16.9
  search_rate: 0
  avg_search_rank: null
  median_search_rank: null
  avg_comp_top1_dwell: 66.99
  avg_comp_top2_dwell: 30.59
  avg_comp_top3_dwell: 22.78
  median_comp_top1_dwell: 30
  avg_comp_top1_price: 8.16
  avg_comp_top2_price: 8.24
  avg_comp_top3_price: 8.16
  median_comp_top1_price: 9
  avg_n_competitors: 2.79
  median_n_competitors: 1

=== PRODUCT 1 ===:

[... the remaining 39 product blocks, in the same format ...]


Predict cart_rate and purchase_rate for each product. JSON array only.
```

User message, YooChoose, from the reported run:

```
=== PRODUCT 0 ===:
  price: 229
  avg_focal_dwell: 151.4
  median_focal_dwell: 43.1
  p25_focal_dwell: 0
  p75_focal_dwell: 120.2
  special_offer_rate: 0.6677
  category_click_rate: 0.3323
  brand_click_rate: 0
  avg_comp_top1_dwell: 383.6
  avg_comp_top2_dwell: 183
  avg_comp_top3_dwell: 146.3
  median_comp_top1_dwell: 179.9
  avg_comp_top1_price: 2976
  avg_comp_top2_price: 891.7
  avg_comp_top3_price: 1250
  median_comp_top1_price: 329
  avg_n_competitors: 4.486
  median_n_competitors: 2

=== PRODUCT 1 ===:

[... the remaining 39 product blocks, in the same format ...]


Predict cart_rate and purchase_rate for each product. JSON array only.
```

Session (C). System message:

```
You are an e-commerce conversion analyst. You will be given browsing sessions for a product.
↪ For each session, estimate the probability that the user adds that session's focal product
↪  to their cart and the probability that they purchase it. Return ONLY a JSON array, one
↪ object per session, of the form {"session": <int>, "p_add_cart": <float>, "p_purchase": <
↪ float>}. No prose.
```

User message, Coveo, from the reported run:

```
Session 0: focal_price=9.0, time_spent_viewing=0.0s, number_of_views=1, discovery_order=2,
↪ discovered_via_search=False, session_length=958s, total_searches_in_session=1,
↪ num_competing_products=7, top_competing_products=[(price=8.0, time_spent_viewing=107.4s,
↪ discovery_order=4), (price=8.0, time_spent_viewing=49.0s, discovery_order=5), (price=9.0,
↪ time_spent_viewing=46.9s, discovery_order=1)]

Predict p_add_cart and p_purchase for each session. JSON array only.
```

User message, YooChoose, from the reported run:

```
Session 0: focal_price=229, time_spent_viewing=0.0s, number_of_views=1, discovery_order=2,
↪ click_context=special_offer, session_length=36s, num_competing_products=1,
↪ top_competing_products=[(price=261, time_spent_viewing=36.6s, discovery_order=1)]

Predict p_add_cart and p_purchase for each session. JSON array only.
```

Batch session (D). System message:

```
You are an e-commerce conversion analyst. You will be given browsing sessions drawn from
↪ SEVERAL DIFFERENT products. For each session, estimate the probability that the user adds
↪ that session's focal product to their cart and the probability that they purchase it.
↪ Return ONLY a JSON array, one object per session, of the form {"session": <int>, "
↪ p_add_cart": <float>, "p_purchase": <float>}. No prose.
```

User message, Coveo, from the reported run:

```
Session 0: focal_price=4.0, time_spent_viewing=0.0s, number_of_views=1, discovery_order=1,
↪ discovered_via_search=False, session_length=0s, total_searches_in_session=0,
↪ num_competing_products=0, top_competing_products=[none]
Session 1: focal_price=9.0, time_spent_viewing=0.0s, number_of_views=2, discovery_order=1,
↪ discovered_via_search=False, session_length=365s, total_searches_in_session=0,
↪ num_competing_products=2, top_competing_products=[(price=9.0, time_spent_viewing=0.0s,
↪ discovery_order=2), (price=10.0, time_spent_viewing=0.0s, discovery_order=3)]
Session 2: focal_price=6.0, time_spent_viewing=0.0s, number_of_views=1, discovery_order=5,
↪ discovered_via_search=False, session_length=128s, total_searches_in_session=0,
↪ num_competing_products=5, top_competing_products=[(price=4.0, time_spent_viewing=22.0s,
↪ discovery_order=2), (price=4.0, time_spent_viewing=0.0s, discovery_order=1), (price=7.0,
↪ time_spent_viewing=0.0s, discovery_order=3)]

[... sessions 3 to 38, each the single session of a different product ...]

Session 39: focal_price=8.0, time_spent_viewing=0.0s, number_of_views=1, discovery_order=1,
↪ discovered_via_search=False, session_length=1028s, total_searches_in_session=1,
↪ num_competing_products=1, top_competing_products=[(price=5.0, time_spent_viewing=31.6s,
↪ discovery_order=2)]

Predict p_add_cart and p_purchase for each session. JSON array only.
```

User message, YooChoose, from the reported run:

```
Session 0: focal_price=3978, time_spent_viewing=49.4s, number_of_views=1, discovery_order=2,
↪ click_context=category_1, session_length=119s, num_competing_products=2,
↪ top_competing_products=[(price=3664, time_spent_viewing=70.1s, discovery_order=1), (price=
↪ null, time_spent_viewing=0.0s, discovery_order=3)]
Session 1: focal_price=22515, time_spent_viewing=114.7s, number_of_views=1, discovery_order
↪ =3, click_context=category_7, session_length=3481s, num_competing_products=6,
↪ top_competing_products=[(price=null, time_spent_viewing=2919.6s, discovery_order=6), (
↪ price=25028, time_spent_viewing=309.9s, discovery_order=2), (price=26075,
↪ time_spent_viewing=72.0s, discovery_order=1)]
Session 2: focal_price=2408, time_spent_viewing=35.3s, number_of_views=1, discovery_order=4,
↪ click_context=category_4, session_length=474s, num_competing_products=5,
↪ top_competing_products=[(price=2408, time_spent_viewing=289.3s, discovery_order=3), (price
↪ =null, time_spent_viewing=87.3s, discovery_order=2), (price=6282, time_spent_viewing=41.4s
↪ , discovery_order=1)]

[... sessions 3 to 38, each the single session of a different product ...]

Session 39: focal_price=1046, time_spent_viewing=12.2s, number_of_views=2, discovery_order=3,
↪  click_context=category_3, session_length=2981s, num_competing_products=7,
↪ top_competing_products=[(price=1105, time_spent_viewing=2188.9s, discovery_order=4), (
```

```
↪ price=null, time_spent_viewing=679.1s, discovery_order=2), (price=null, time_spent_viewing
↪ =53.1s, discovery_order=1)]

Predict p_add_cart and p_purchase for each session. JSON array only.
```

Session bag (R). System message:

```
You are an e-commerce conversion analyst. For each product below you are shown a sample of
↪ the browsing sessions in which shoppers viewed it. Using those sessions, estimate for each
↪  product the fraction of its sessions that end in an add-to-cart, and the fraction that
↪ end in a purchase. Return ONLY a JSON array, one object per product, of the form {"product
↪ ": <int>, "cart_rate": <float>, "purchase_rate": <float>}. No prose.
```

User message, Coveo, from the reported run:

```
=== PRODUCT 0 ===  (80 sessions)
Session 0: focal_price=9.0, time_spent_viewing=0.0s, number_of_views=1, discovery_order=2,
↪ discovered_via_search=False, session_length=958s, total_searches_in_session=1,
↪ num_competing_products=7, top_competing_products=[(price=8.0, time_spent_viewing=107.4s,
↪ discovery_order=4), (price=8.0, time_spent_viewing=49.0s, discovery_order=5), (price=9.0,
↪ time_spent_viewing=46.9s, discovery_order=1)]
Session 1: focal_price=9.0, time_spent_viewing=5.8s, number_of_views=3, discovery_order=1,
↪ discovered_via_search=False, session_length=22s, total_searches_in_session=0,
↪ num_competing_products=0, top_competing_products=[none]
Session 2: focal_price=9.0, time_spent_viewing=68.6s, number_of_views=1, discovery_order=1,
↪ discovered_via_search=False, session_length=582s, total_searches_in_session=0,
↪ num_competing_products=1, top_competing_products=[(price=8.0, time_spent_viewing=0.0s,
↪ discovery_order=2)]

[... sessions 3 to 78 of the same product, in the same format ...]

Session 79: focal_price=9.0, time_spent_viewing=0.0s, number_of_views=1, discovery_order=4,
↪ discovered_via_search=False, session_length=269s, total_searches_in_session=0,
↪ num_competing_products=5, top_competing_products=[(price=9.0, time_spent_viewing=0.0s,
↪ discovery_order=1), (price=9.0, time_spent_viewing=0.0s, discovery_order=2), (price=9.0,
↪ time_spent_viewing=0.0s, discovery_order=3)]

Predict cart_rate and purchase_rate for each product. JSON array only.
```

User message, YooChoose, from the reported run:

```
=== PRODUCT 0 ===  (80 sessions)
Session 0: focal_price=229, time_spent_viewing=0.0s, number_of_views=1, discovery_order=2,
↪ click_context=special_offer, session_length=36s, num_competing_products=1,
↪ top_competing_products=[(price=261, time_spent_viewing=36.6s, discovery_order=1)]
Session 1: focal_price=229, time_spent_viewing=81.3s, number_of_views=1, discovery_order=5,
↪ click_context=special_offer, session_length=1124s, num_competing_products=12,
↪ top_competing_products=[(price=261, time_spent_viewing=166.4s, discovery_order=6), (price
↪ =554, time_spent_viewing=146.3s, discovery_order=7), (price=null, time_spent_viewing=135.9
↪ s, discovery_order=11)]
Session 2: focal_price=229, time_spent_viewing=0.0s, number_of_views=1, discovery_order=2,
↪ click_context=special_offer, session_length=391s, num_competing_products=1,
↪ top_competing_products=[(price=null, time_spent_viewing=391.7s, discovery_order=1)]

[... sessions 3 to 78 of the same product, in the same format ...]

Session 79: focal_price=229, time_spent_viewing=0.0s, number_of_views=1, discovery_order=2,
↪ click_context=category_10, session_length=18s, num_competing_products=1,
↪ top_competing_products=[(price=345, time_spent_viewing=18.5s, discovery_order=1)]

Predict cart_rate and purchase_rate for each product. JSON array only.
```

E DISCOVERY-POOL DESIGN ABLATIONS

The design choices behind Figure 1 – which statistics the summary contains, how many sessions it is computed from, how many products share a prompt, and how the request is worded – were fixed on the discovery pool before any interaction with the evaluation pools. Here we vary each choice in turn on that same discovery pool (280 Coveo products) with GPT-5.6, so the evaluation pools remain untouched by anything exploratory.

Every cell is Batch summary and differs from a reference cell in exactly one respect. The reference is the curated block at batch 40 with the reported prompt, re-run through the same program as the ablations rather than taken from the earlier discovery runs, whose records use a different format; all contrasts below are therefore internal to one code path. Every cell uses the reported 18-column summary, with the two traffic counts withheld as in the evaluation runs. Running the same grid with the counts present leaves the depth and wording results unchanged; the only difference is that the smallest batch sizes then fall below the reference, at -0.066 and -0.060 for batch 5, as does batch 280 on cart, at -0.044. Reference correlations are 0.307 for cart and 0.303 for purchase. Intervals are paired product bootstraps with a shared index set.

Summary statistics. The columns are not a selected subset. They are the complete set of non-outcome, non-identifier columns produced by the feature builder: a price bucket, traffic counts, four focal-dwell statistics, three search statistics, and the dwell, price and count statistics of the three most-considered competing products, of which the reported summary renders all but the traffic counts (Appendix C). The construction crosses each behavioral quantity with conventional descriptive statistics – mean, median, quartiles, counts and rates – and keeps every result. Six columns are excluded, five because they are derived from the outcome (`n_add_to_cart`, `n_purchased`, `n_displaced`, `n_competitor_buy`, `competitor_buy_rate`) and would leak the target, and one because it is a category identifier rather than a statistic. No column was added or removed on the basis of predictive performance, and the feature file predates the experiments reported here.

Prompt wording. Two paraphrases carrying identical information and an identical output contract change the correlation by -0.009 and $+0.004$ on cart and -0.003 and $+0.000$ on purchase, every interval containing zero. The reported results are not an artifact of one phrasing. All reported runs use the first wording, which is the one reproduced in Appendix D; the two paraphrases are included with the supplementary code.

Batch size. Holding the number of groupings at five, so each product receives five judgments at every size, correlation is flat in batch size (Table 4). No size from 5 to 280 differs from the reference by a margin the paired intervals distinguish from zero, on either outcome; the largest movement is $+0.024$ at batch 140 on cart. Batch 40 therefore sits inside a flat region rather than at a favorable extreme, and the inference-cost comparison in Section 5 does not depend on the choice. Three features of this curve deserve care. Batch size sets both the number of products compared and the number of points entering the within-call standardization, so at batch 5 the score is divided by the spread of five values. At the other end a single call holds the entire discovery pool, so the five groupings at batch 280 repeat one comparison set instead of varying it, and averaging over comparison contexts disappears. And batch 1 would be Summary exactly, so the curve traces a continuous path between the isolated and comparative methods rather than treating batching as a switch.

Evidence depth. Recomputing the same statistics from the first K sampled sessions shows how accuracy depends on the number of sessions a summary draws on (Table 5). Correlation rises with depth, apart from a small dip between 10 and 20 sessions on both outcomes, at a constant 35 calls per cell, which is the property that makes aggregation attractive: additional evidence enters the prediction without additional inference. Both outcomes are still rising at 240 sessions, the deepest sample we hold, and the cells below 80 all sit below the reference. Products retain every session they have, so the realized depth at a cap of 240 averages 202: 280 products reach 80, 250 reach 120, 203 reach 160 and 155 reach 240.

An exhaustive mechanical summary. Replacing the curated block with a mechanical one – count, mean, standard deviation, minimum, median and maximum of every numeric field in the session record, with no selection of any kind – changes the correlation by $+0.023$ $[-0.0823, +0.1332]$ on cart and $+0.007$ $[-0.0987, +0.1157]$ on purchase. The mechanical block carries 144 columns against the reported block's 18 and involves no domain judgment at all, and we detect no difference

from the curated block. The intervals are wide, about ± 0.1, so this does not establish that the two are equivalent; it shows only that the result does not depend on the curated statistics by a margin this pool can detect.

Table 4: Batch size, discovery pool, GPT-5.6. Every size uses five groupings, so each product receives five judgments; at 280, one call holds the whole pool. A star marks a paired interval excluding zero.

Products per prompt	Calls	Cart	Purchase
5	280	0.280	0.276
10	140	0.297	0.283
20	70	0.288	0.280
40 (reference)	35	0.307	0.303
80	20	0.324	0.305
140	10	0.331	0.294
280	5	0.317	0.311

Table 5: Evidence depth, discovery pool, GPT-5.6: statistics recomputed from the first K sampled sessions, 35 calls per cell. Products with fewer sessions keep all of them, so the realized mean depth is shown. A star marks a paired interval excluding zero.

Sessions (cap / realized mean)	Products reaching cap	Cart	Purchase
5 / 5	280	0.057*	0.023*
10 / 10	280	0.150*	0.127*
20 / 20	280	0.132*	0.119*
40 / 40	280	0.203*	0.165*
80 / 80 (reference)	280	0.307	0.303
120 / 119	250	0.371	0.330
160 / 151	203	0.453*	0.379
240 / 202	155	0.502*	0.410*

F PROTOCOL AND REPRODUCIBILITY

Here we record what a reader would need in order to rerun the evaluation: how the pools were drawn, what the models were shown, how the runs were executed, and how the reported statistics are computed.

Product pools and sampling. The Coveo pool contains 1,000 distinct products, selected with at least 100 sessions and stratified by depth decile and price-bucket tercile across 30 cells. It is disjoint from the discovery pools used to develop the procedures, and its behavioral inputs and outcome rates describe the same observation period. We sample 80 sessions per product by reservoir sampling during a pass through the trace. Each record includes up to three co-considered products chosen by dwell time. A session containing two evaluated products can contribute to both histories.

For YooChoose, the 1,000-product pool requires at least 300 evaluation-window sessions and a positive-price purchase before August 1, 2014. Historical purchases supply prices; evaluation-window sessions supply behavior and target rates. The summary and session methods use these same information sources. The selection excludes products with no priced sales history. Competitor price is missing when the competitor has no priced purchase in the earlier window, even though focal price is present for every selected product.

The realized pools and prompt designs define the evaluations. Coveo's pool artifact stores seed 2327118 and its session-sample artifact stores 1422810. Their original seed expression uses a process-dependent Python string hash, so rerunning that expression can change the selected products or sessions. Reproduction requires the stored artifacts. The YooChoose construction uses the explicit seed 20260911. Coveo discovery results inform the procedures subsequently evaluated on separate products and on YooChoose.

Behavioral features. Coveo's 20 columns comprise price bucket and traffic; four focal-dwell statistics; search count, search share, and two search-rank statistics; four competitor-dwell statistics; four competitor-price statistics; and two competitor-count statistics. YooChoose has 19 columns: price and traffic; the same four focal-dwell statistics; special-offer rate, category-click rate, and

brand-click rate; and the same competitor-dwell, competitor-price, and competitor-count groups. The two blocks reflect the fields available in each dataset. The reported summary prompts render all of these except the traffic counts, 18 columns on each dataset; we report the arms with them restored in Appendix C, and the supervised benchmark uses the full block.

Session inputs retain individual records rather than these marginal summaries. The Coveo renderer additionally exposes view counts and consideration order. Coveo summary features use four significant digits and render missing values as `null`. Outcome fields are masked in the prompted records. Recomputing A80 and B80 updates the summary features from the fixed 80-session sample, so comparisons with the full history change the behavioral statistics alone, the traffic counts being withheld from every summary prompt.

Models and execution. The prompts are reproduced in Appendix D. We use the endpoints in Table 6, with temperature zero and low reasoning effort. Qwen uses an eightfold token-budget scale. The execution records span September 9–20, 2026 for Coveo and September 11–20 for YooChoose. The endpoints are provider slugs rather than immutable model snapshots; we preserve endpoint identifiers and execution dates as the version record.

Table 6: Model endpoints used across the two datasets.

Model	Endpoint
GPT-5.6	`openai/gpt-5.6`
Qwen3.8	`qwen/qwen3.8-2.4t-a95b`
Gemini3.8 Flash	`google/gemini-3.8-flash`
GLM5.3	`z-ai/glm-5.3`
Claude Sonnet5	`anthropic/claude-sonnet-5`

Each call is attempted up to six times, doubling the token limit from a per-cell base up to 32,000 tokens. An empty parse triggers a retry; a call exhausting retries contributes no checkpoint entry. Because the sampling temperature is zero, a reply that the parser rejects for its shape rather than its content tends to recur on retry, so such a call is usually lost rather than recovered. Two response parsers in the execution code were strict in this way: the product-level parser required a JSON array and rejected an otherwise complete single-object reply, and the session parser discarded a response whose enclosing array was truncated instead of keeping the complete records inside it. We repaired both parsers to accept any complete record, to log rejected replies, and to report a call that yields nothing, then re-ran the affected calls: 38 single-product session calls and 22 cross-product session calls across three models and both datasets. Five product-level predictions rejected only for their shape were recovered from the logged replies rather than re-requested, since the model had already answered them at temperature zero. Every method now covers the full 1,000-product pool for every model on both datasets (Table 7). Comparative methods use 40 products per call. Five summary groupings produce 125 planned calls; 80 cross-product session rounds produce 2,000. Isolated summaries and session bags each require 1,000 planned calls, and the per-session arm 80,000. The coverage gate is 97%; in cross-product session inference, calls with fewer than 80% of their 40 session predictions parsed are dropped.

Common support. We use one model-specific intersection of products across all seven methods for every arm level and paired contrast. We report those counts in Table 7. Differences between a model's methods therefore use the same products. After the refills described above, every model's support is the full 1,000-product pool on both datasets.

Aggregation and rank convention. For B, B80, and D, we standardize within each call using the population standard deviation, replacing a divisor at or below 10^{-12} with one, and average by product across calls. A, A80, and R each produce one product prediction. C averages the raw probabilities returned by its 80 single-session calls; with one output per call there is nothing within a call to standardize against.

For a prediction vector s and observed rates $\bar{Y}$, we report Spearman's correlation $\rho = \mathrm{Corr}(R(s), R(\bar{Y}))$, where R assigns tied values the average of the ranks they span (`scipy.stats.rankdata` with `method="average"`). The same convention is used for every method, contrast, supervised benchmark and the single-feature reference. In Appendix G, we report the effect of breaking ties by sort order instead.

Table 7: Prediction coverage: product calls for A, A80 and R (expected 1,000), 40-product calls for B and B80 (125), and parsed session predictions for C and D (80,000). n is the common support for all seven arms; Coveo coverage is the same for both outcomes.

Model	n	A	C	D	B	A80	B80	R
Coveo								
GPT-5.6	1000	1,000	80,000	80,000	125	1,000	125	1,000
Qwen3.8	1000	1,000	80,000	80,000	125	1,000	125	1,000
Gemini3.8	1000	1,000	80,000	80,000	125	1,000	125	1,000
GLM5.3	1000	1,000	80,000	80,000	125	1,000	125	1,000
Sonnet5	1000	1,000	80,000	80,000	125	1,000	125	1,000
YooChoose								
GPT-5.6	1000	1,000	80,000	80,000	125	1,000	125	1,000
Qwen3.8	1000	1,000	80,000	80,000	125	1,000	125	1,000
Gemini3.8	1000	1,000	80,000	80,000	125	1,000	125	1,000
GLM5.3	1000	1,000	80,000	80,000	125	1,000	125	1,000
Sonnet5	1000	1,000	80,000	79,999	125	1,000	125	1,000

Paired bootstrap. Within each dataset, we use 2,000 product resamples from NumPy's random generator with seed seven. The index set is drawn once, before any model, method, or outcome is processed, and resamples product positions in the shared pool ordering, so every level, contrast, and difference of differences in the paper uses the same resampled products. A model's common-support mask is applied to its prediction vector before resampling; a resampled position outside that model's support enters as a missing value and is dropped pairwise inside the correlation. Replicates are therefore aligned by product identity across models; with every support at the full pool, no product is dropped. We compute paired differences within each replicate and report the 2.5th and 97.5th percentiles. For model means, we average the model-specific values within each replicate before taking percentiles. The intervals condition on saved predictions.

Session sharing between products. Resampling products treats them as independent, which shared shopping sessions could violate. The stored session samples record behavior without session identifiers, so we measure sharing in the released logs directly. Sharing is common at the population level: 11.5% of Coveo sessions touching a pooled product touch more than one, giving 27,327 of the 499,500 product pairs at least one shared session; on YooChoose the figures are 24.4% and 108,753 pairs.

For the session methods, that statistic overstates the dependence that reaches the estimates. Each pooled product appears in hundreds of logged sessions, a median of 279 on Coveo and 643 on YooChoose, but contributes only 80 to the experiment, and a sharing pair shares a median of one logged session. Scaling each pair's shared count by the probability that a shared session enters both 80-session samples gives the expected overlap within the evaluated evidence: a mean of 0.131 shared sessions per sharing pair on Coveo and 0.039 on YooChoose, with 518 and 750 of the 499,500 pairs, about one in a thousand, expected to share even one sampled session. Averaged over all pairs this is roughly 0.01% of a product's 80 sessions per pair; summed over its partners, a product's 80 sampled sessions are expected to overlap other products' samples about 7 times on Coveo and 8 on YooChoose, spread thinly across many pairs. The argument does not extend to the full histories behind the summary methods or to the observed rates, which draw on all logged sessions and share them more often; the product bootstrap does not account for that dependence, and we report it without a cluster correction.

Further limitations. The summary is one particular compression, though on the discovery pool we detect no difference from an exhaustive block of 144 mechanical summary columns, within wide intervals (Appendix E). We measure ranking alone and leave calibration and cardinal accuracy aside. Our intervals condition on the saved predictions and treat products as independent, although full histories share sessions across products, as discussed above, and they do not include variation from drawing new sessions or new comparative groupings.

Observation window of the session records. We remove the focal cart and purchase fields from every prompted record. On Coveo, the remaining fields describe completed sessions: session length runs from the first to the last logged event, including cart and purchase events, and a product's viewing

time runs from a detail view to the next event of any kind, so both can include time after the focal outcome. Session length appears only in the session records, while viewing-time statistics enter both the records and the summaries, so the methods differ in some fields under this regime; we do not measure how that affects their ordering. On YooChoose, where purchases are logged separately from clicks, our preprocessing truncates each session at the time of its first purchase, so its records contain no activity after a purchase: session length runs to the last click before that time, and that last click contributes no viewing time. On both datasets the task is a retrospective reconstruction of observed rates, not a forecast made before the outcome. YooChoose prices come from purchases before the August 2014 split for both focal and competing products.

G METRIC SENSITIVITY

We recompute the results under three alternatives to the reported estimator, holding the predictions, the common supports, and the bootstrap index set fixed. The alternatives are ordinal ranks that break ties by sort order rather than averaging them, Pearson correlation, and Spearman correlation applied to batch outputs that have not been standardized within the call. We report the two headline contrasts under each in Table 8.

Rank convention. We rebuilt both full reports with ties broken by sort order (`argsort` applied twice) and compared every arm level and all 15 paired contrasts for every model and outcome. The largest shift is 0.022 in an arm level and 0.030 in a paired contrast. Of the 225 paired intervals, 2 change significance, one in each direction, and no significant result reverses sign: Session bag−Summary-80 for Gemini on Coveo purchase is significant under midranks but not under ordinal ranks, and Session−Summary-80 for Sonnet on YooChoose, 0.055, is significant under ordinal ranks but not under midranks. For the two headline contrasts in Table 8 the largest shift is 0.010, with no change in significance.

Pearson correlation. The observed rates are strongly right-skewed, with skewness $+2.16$ and $+2.46$ on the two Coveo outcomes and $+1.85$ on YooChoose, and 30.2% of Coveo purchase rates are exactly zero. Pearson correlation therefore weights a small number of high-conversion products heavily, and it measures a linear relationship the methods are not designed to deliver: the batch methods standardize within each call, which removes level information by construction. We report it as a sensitivity analysis rather than as a second primary metric.

Both findings replicate. All 15 Batch summary−Summary intervals remain above zero, and Batch session−Session remains weak, with two intervals above zero under Pearson against three under the reported estimator.

Unstandardized batch outputs. Averaging raw batch outputs instead of within-call standardized scores preserves all 30 conclusions: every Batch summary−Summary interval is still above zero, and Batch session−Session is still above zero in three settings. Standardization sharpens the comparative gain rather than creating it; we separate the two in Appendix K.

H DETAILED ARM LEVELS

Every table uses the method names and source mapping in Table 2. We display point estimates to three decimal places and interval endpoints to four; figures use the full-precision machine-readable values. Coveo has two outcomes and YooChoose one. Each model uses its common support across all seven methods. The mean of five is an equal-weight average of model-specific correlations, not an ensemble of predictions or a correlation on pooled data.

Table 8: The two headline contrasts under four estimators: the reported Spearman correlation (midranks) on standardized outputs, ordinal ranks with ties broken by sort order, Pearson correlation, and Spearman correlation on unstandardized outputs. A star marks a paired 95% interval excluding zero.

Model	Spearman	Ordinal ranks	Pearson	Unstandardized
Batch summary−Summary, Coveo cart				
GPT-5.6	0.287*	0.289*	0.234*	0.236*
Sonnet5	0.228*	0.228*	0.173*	0.211*
Gemini3.8	0.181*	0.186*	0.152*	0.141*
GLM5.3	0.258*	0.255*	0.160*	0.208*
Qwen3.8	0.349*	0.354*	0.318*	0.313*
Batch summary−Summary, Coveo purchase				
GPT-5.6	0.310*	0.305*	0.235*	0.278*
Sonnet5	0.273*	0.264*	0.214*	0.258*
Gemini3.8	0.201*	0.198*	0.152*	0.165*
GLM5.3	0.236*	0.232*	0.190*	0.190*
Qwen3.8	0.465*	0.459*	0.405*	0.414*
Batch summary−Summary, YooChoose purchase				
GPT-5.6	0.183*	0.186*	0.201*	0.179*
Sonnet5	0.215*	0.219*	0.175*	0.192*
Gemini3.8	0.251*	0.261*	0.271*	0.092*
GLM5.3	0.269*	0.268*	0.248*	0.201*
Qwen3.8	0.333*	0.339*	0.270*	0.320*
Batch session−Session, Coveo cart				
GPT-5.6	0.006	0.006	0.018	0.001
Sonnet5	-0.122*	-0.122*	-0.144*	-0.119*
Gemini3.8	-0.021	-0.020	-0.027*	-0.020
GLM5.3	-0.083*	-0.083*	-0.092*	-0.066*
Qwen3.8	0.000	0.000	-0.013	0.017
Batch session−Session, Coveo purchase				
GPT-5.6	-0.007	-0.006	-0.008	-0.023
Sonnet5	-0.096*	-0.095*	-0.083*	-0.087*
Gemini3.8	-0.027*	-0.026*	-0.039*	-0.024*
GLM5.3	-0.051*	-0.050*	-0.053*	-0.032
Qwen3.8	-0.020	-0.020	-0.033*	-0.008
Batch session−Session, YooChoose purchase				
GPT-5.6	0.143*	0.143*	0.119*	0.124*
Sonnet5	0.109*	0.109*	0.091*	0.113*
Gemini3.8	0.033*	0.032*	0.025	0.034*
GLM5.3	-0.003	-0.003	-0.005	-0.005
Qwen3.8	0.029	0.029	0.031	0.023

Table 9: Coveo cart: rank correlations with 95% product-bootstrap intervals, on each model's common support (Table 7); the mean of five has its own interval. R is the session-bag diagnostic.

Arm	GPT-5.6	Qwen3.8	Gemini3.8	GLM5.3	Sonnet5	Mean
A	0.151 [0.0908,0.2115]	0.084 [0.0241,0.1470]	0.175 [0.1133,0.2353]	−0.006 [-0.0690,0.0568]	0.000 [-0.0625,0.0653]	0.081 [0.0410,0.1213]
C	0.313 [0.2505,0.3675]	0.246 [0.1824,0.3029]	0.281 [0.2208,0.3357]	0.258 [0.1971,0.3143]	0.278 [0.2182,0.3338]	0.275 [0.2161,0.3275]
D	0.319 [0.2556,0.3739]	0.246 [0.1802,0.3054]	0.261 [0.1978,0.3176]	0.175 [0.1118,0.2347]	0.156 [0.0924,0.2166]	0.231 [0.1703,0.2866]
B	0.438 [0.3835,0.4912]	0.433 [0.3787,0.4851]	0.356 [0.2977,0.4138]	0.252 [0.1898,0.3123]	0.228 [0.1681,0.2921]	0.341 [0.2881,0.3959]
A80	0.171 [0.1086,0.2300]	0.032 [-0.0346,0.0913]	0.132 [0.0680,0.1919]	0.065 [0.0022,0.1246]	0.027 [-0.0326,0.0895]	0.085 [0.0424,0.1252]
B80	0.292 [0.2321,0.3486]	0.341 [0.2808,0.3946]	0.212 [0.1477,0.2742]	0.178 [0.1133,0.2400]	0.143 [0.0785,0.2067]	0.233 [0.1722,0.2903]
R	0.235 [0.1741,0.2934]	0.174 [0.1129,0.2339]	0.225 [0.1624,0.2833]	0.029 [-0.0348,0.0930]	0.059 [-0.0049,0.1203]	0.145 [0.1054,0.1803]

Table 10: Coveo purchase: rank correlations with 95% product-bootstrap intervals, on each model's common support (Table 7); the mean of five has its own interval. R is the session-bag diagnostic.

Arm	GPT-5.6	Qwen3.8	Gemini3.8	GLM5.3	Sonnet5	Mean
A	0.161 [0.0984,0.2245]	0.031 [-0.0331,0.0940]	0.220 [0.1612,0.2793]	0.056 [-0.0059,0.1164]	−0.004 [-0.0664,0.0572]	0.093 [0.0569,0.1298]
C	0.328 [0.2669,0.3825]	0.293 [0.2325,0.3477]	0.311 [0.2503,0.3657]	0.287 [0.2240,0.3418]	0.301 [0.2389,0.3561]	0.304 [0.2442,0.3552]
D	0.322 [0.2583,0.3773]	0.274 [0.2112,0.3274]	0.283 [0.2210,0.3411]	0.236 [0.1742,0.2930]	0.205 [0.1424,0.2649]	0.264 [0.2042,0.3181]
B	0.471 [0.4179,0.5202]	0.496 [0.4452,0.5450]	0.420 [0.3668,0.4742]	0.292 [0.2334,0.3510]	0.269 [0.2114,0.3272]	0.390 [0.3420,0.4382]
A80	0.168 [0.1036,0.2292]	0.008 [-0.0543,0.0676]	0.114 [0.0511,0.1738]	0.083 [0.0166,0.1440]	0.042 [-0.0223,0.1056]	0.083 [0.0422,0.1222]
B80	0.307 [0.2426,0.3626]	0.347 [0.2875,0.4032]	0.240 [0.1747,0.3004]	0.190 [0.1263,0.2502]	0.142 [0.0792,0.2032]	0.245 [0.1856,0.2992]
R	0.225 [0.1643,0.2812]	0.185 [0.1247,0.2496]	0.189 [0.1289,0.2496]	0.009 [-0.0543,0.0722]	0.077 [0.0153,0.1378]	0.137 [0.0984,0.1746]

Table 11: YooChoose purchase: rank correlations with 95% product-bootstrap intervals, on each model's common support (Table 7); the mean of five has its own interval. R is the session-bag diagnostic.

Arm	GPT-5.6	Qwen3.8	Gemini3.8	GLM5.3	Sonnet5	Mean
A	0.415 [0.3625,0.4649]	0.251 [0.1899,0.3103]	0.350 [0.2931,0.4044]	0.249 [0.1895,0.3057]	0.192 [0.1284,0.2508]	0.291 [0.2524,0.3276]
C	0.417 [0.3617,0.4663]	0.502 [0.4513,0.5480]	0.429 [0.3732,0.4806]	0.490 [0.4401,0.5341]	0.278 [0.2162,0.3359]	0.423 [0.3730,0.4684]
D	0.560 [0.5126,0.6039]	0.530 [0.4792,0.5762]	0.461 [0.4069,0.5124]	0.487 [0.4349,0.5345]	0.387 [0.3287,0.4395]	0.485 [0.4343,0.5314]
B	0.597 [0.5541,0.6375]	0.584 [0.5378,0.6258]	0.601 [0.5533,0.6426]	0.517 [0.4654,0.5642]	0.407 [0.3485,0.4597]	0.541 [0.4983,0.5796]
A80	0.421 [0.3670,0.4700]	0.255 [0.1934,0.3138]	0.388 [0.3307,0.4426]	0.180 [0.1194,0.2373]	0.222 [0.1615,0.2844]	0.293 [0.2528,0.3324]
B80	0.564 [0.5160,0.6066]	0.544 [0.4970,0.5867]	0.539 [0.4855,0.5843]	0.485 [0.4330,0.5321]	0.371 [0.3138,0.4237]	0.501 [0.4557,0.5418]
R	0.407 [0.3545,0.4581]	0.154 [0.0912,0.2137]	0.291 [0.2327,0.3468]	0.194 [0.1327,0.2517]	0.202 [0.1431,0.2614]	0.250 [0.2121,0.2851]

I DETAILED PAIRED CONTRASTS

The following tables report the 15 pairwise contrasts for every model and outcome. A star marks a reported paired 95% interval excluding zero, based on its unrounded endpoints. For a model mean, the interval uses the distribution of the mean paired contrast across shared bootstrap draws; it cannot be constructed by averaging or subtracting interval endpoints. These are individual comparisons without a multiplicity adjustment.

The tables do not include Batch summary-80 against Session, which we compute from the same predictions and bootstrap draws. The model-mean difference is -0.042 $[-0.0763, -0.0059]$ for Coveo cart, -0.059 $[-0.0921, -0.0246]$ for Coveo purchase and 0.078 $[0.0471, 0.1090]$ for YooChoose. On both Coveo outcomes, Session is higher for Sonnet, Gemini and GLM and Batch summary-80 is higher for Qwen, with intervals excluding zero, and GPT is indistinguishable; on YooChoose Batch summary-80 is higher for four models and GLM is indistinguishable. Against Batch session, Batch summary-80 is ahead with an interval excluding zero in 3 of the 15 settings and behind in 4.

Table 12: Coveo cart: paired differences in rank correlation, group 1 of 3, with 95% product-bootstrap intervals. A star marks an unrounded interval excluding zero; no multiplicity adjustment.

Model	$B-A$	$D-C$	$B-C$	$D-A$	$C-A$
GPT-5.6	$+\mathbf{0.287}^*$ [0.2239,0.3481]	$+0.006$ [-0.0182,0.0293]	$+\mathbf{0.126}^*$ [0.0786,0.1744]	$+\mathbf{0.167}^*$ [0.0988,0.2359]	$+\mathbf{0.161}^*$ [0.0922,0.2308]
Qwen3.8	$+\mathbf{0.349}^*$ [0.2758,0.4227]	$+0.000$ [-0.0273,0.0265]	$+\mathbf{0.187}^*$ [0.1357,0.2431]	$+\mathbf{0.162}^*$ [0.0781,0.2344]	$+\mathbf{0.162}^*$ [0.0772,0.2360]
Gemini3.8	$+\mathbf{0.181}^*$ [0.1141,0.2427]	-0.021 [-0.0441,0.0023]	$+\mathbf{0.075}^*$ [0.0212,0.1300]	$+\mathbf{0.085}^*$ [0.0130,0.1546]	$+\mathbf{0.106}^*$ [0.0321,0.1739]
GLM5.3	$+\mathbf{0.258}^*$ [0.1799,0.3373]	$-\mathbf{0.083}^*$ [-0.1212,-0.0434]	-0.006 [-0.0734,0.0618]	$+\mathbf{0.181}^*$ [0.0983,0.2571]	$+\mathbf{0.264}^*$ [0.1818,0.3418]
Sonnet5	$+\mathbf{0.228}^*$ [0.1525,0.2984]	$-\mathbf{0.122}^*$ [-0.1664,-0.0768]	-0.050 [-0.1130,0.0117]	$+\mathbf{0.156}^*$ [0.0788,0.2328]	$+\mathbf{0.278}^*$ [0.1973,0.3527]
Mean of five	$+\mathbf{0.260}^*$ [0.2104,0.3082]	$-\mathbf{0.044}^*$ [-0.0675,-0.0202]	$+\mathbf{0.066}^*$ [0.0174,0.1184]	$+\mathbf{0.150}^*$ [0.0918,0.2025]	$+\mathbf{0.194}^*$ [0.1341,0.2471]

Table 13: Coveo cart: paired differences in rank correlation, group 2 of 3, with 95% product-bootstrap intervals. A star marks an unrounded interval excluding zero; no multiplicity adjustment.

Model	$C-A80$	$R-A80$	$C-R$	$A-A80$	$B-B80$
GPT-5.6	$+\mathbf{0.142}^*$ [0.0848,0.1957]	$+0.065$ [-0.0075,0.1295]	$+\mathbf{0.077}^*$ [0.0239,0.1289]	-0.019 [-0.0920,0.0530]	$+\mathbf{0.146}^*$ [0.1001,0.1943]
Qwen3.8	$+\mathbf{0.214}^*$ [0.1477,0.2831]	$+\mathbf{0.142}^*$ [0.0645,0.2210]	$+\mathbf{0.072}^*$ [0.0016,0.1387]	$+0.052$ [-0.0241,0.1322]	$+\mathbf{0.092}^*$ [0.0420,0.1440]
Gemini3.8	$+\mathbf{0.149}^*$ [0.0907,0.2060]	$+\mathbf{0.093}^*$ [0.0213,0.1632]	$+\mathbf{0.057}^*$ [0.0002,0.1131]	$+0.044$ [-0.0331,0.1195]	$+\mathbf{0.144}^*$ [0.0908,0.1995]
GLM5.3	$+\mathbf{0.192}^*$ [0.1179,0.2656]	-0.036 [-0.1224,0.0568]	$+\mathbf{0.228}^*$ [0.1512,0.3067]	-0.071 [-0.1601,0.0222]	$+\mathbf{0.074}^*$ [0.0245,0.1248]
Sonnet5	$+\mathbf{0.251}^*$ [0.1919,0.3123]	$+0.033$ [-0.0433,0.1122]	$+\mathbf{0.219}^*$ [0.1495,0.2870]	-0.027 [-0.0915,0.0488]	$+\mathbf{0.085}^*$ [0.0398,0.1329]
Mean of five	$+\mathbf{0.190}^*$ [0.1467,0.2288]	$+\mathbf{0.059}^*$ [0.0185,0.0991]	$+\mathbf{0.131}^*$ [0.0876,0.1704]	-0.004 [-0.0470,0.0408]	$+\mathbf{0.108}^*$ [0.0639,0.1548]

J BATCHING A PRODUCT'S OWN SESSIONS

Session uses 80 calls per product. The obvious way to save calls is to put all 80 of its sampled sessions into one call and ask for 80 probabilities, which keeps the evidence and the requested judgments and uses one call instead of 80. We ran that variant on every model and dataset, on

Table 14: Coveo cart: paired differences in rank correlation, group 3 of 3, with 95% product-bootstrap intervals. A star marks an unrounded interval excluding zero; no multiplicity adjustment.

Model	$B80-D$	$B-D$	$B-R$	$R-A$	$B80-A80$
GPT-5.6	-0.027 [-0.0644,0.0129]	$\mathbf{+0.120}^*$ [0.0697,0.1693]	$\mathbf{+0.203}^*$ [0.1388,0.2660]	$\mathbf{+0.084}^*$ [0.0066,0.1550]	$\mathbf{+0.121}^*$ [0.0649,0.1732]
Qwen3.8	$\mathbf{+0.095}^*$ [0.0560,0.1371]	$\mathbf{+0.187}^*$ [0.1347,0.2449]	$\mathbf{+0.259}^*$ [0.1848,0.3333]	$\mathbf{+0.090}^*$ [0.0032,0.1690]	$\mathbf{+0.309}^*$ [0.2440,0.3765]
Gemini3.8	$\mathbf{-0.048}^*$ [-0.0825,-0.0139]	$\mathbf{+0.095}^*$ [0.0414,0.1532]	$\mathbf{+0.131}^*$ [0.0633,0.2041]	$+0.049$ [-0.0320,0.1251]	$\mathbf{+0.080}^*$ [0.0190,0.1400]
GLM5.3	$+0.004$ [-0.0414,0.0496]	$\mathbf{+0.077}^*$ [0.0160,0.1403]	$\mathbf{+0.223}^*$ [0.1395,0.3083]	$+0.036$ [-0.0470,0.1212]	$\mathbf{+0.113}^*$ [0.0370,0.1908]
Sonnet5	-0.013 [-0.0643,0.0387]	$\mathbf{+0.072}^*$ [0.0089,0.1331]	$\mathbf{+0.169}^*$ [0.0870,0.2489]	$+0.059$ [-0.0312,0.1486]	$\mathbf{+0.116}^*$ [0.0612,0.1731]
Mean of five	$+0.002$ [-0.0320,0.0366]	$\mathbf{+0.110}^*$ [0.0594,0.1626]	$\mathbf{+0.197}^*$ [0.1434,0.2517]	$\mathbf{+0.064}^*$ [0.0164,0.1079]	$\mathbf{+0.148}^*$ [0.1065,0.1879]

Table 15: Coveo purchase: paired differences in rank correlation, group 1 of 3, with 95% product-bootstrap intervals. A star marks an unrounded interval excluding zero; no multiplicity adjustment.

Model	$B-A$	$D-C$	$B-C$	$D-A$	$C-A$
GPT-5.6	$\mathbf{+0.310}^*$ [0.2475,0.3697]	-0.007 [-0.0334,0.0190]	$\mathbf{+0.143}^*$ [0.0980,0.1880]	$\mathbf{+0.161}^*$ [0.0908,0.2260]	$\mathbf{+0.168}^*$ [0.0958,0.2345]
Qwen3.8	$\mathbf{+0.465}^*$ [0.3902,0.5357]	-0.020 [-0.0475,0.0091]	$\mathbf{+0.203}^*$ [0.1525,0.2559]	$\mathbf{+0.243}^*$ [0.1637,0.3172]	$\mathbf{+0.263}^*$ [0.1850,0.3395]
Gemini3.8	$\mathbf{+0.201}^*$ [0.1345,0.2620]	$\mathbf{-0.027}^*$ [-0.0509,-0.0034]	$\mathbf{+0.110}^*$ [0.0615,0.1608]	$+0.064$ [-0.0099,0.1323]	$\mathbf{+0.091}^*$ [0.0176,0.1582]
GLM5.3	$\mathbf{+0.236}^*$ [0.1594,0.3119]	$\mathbf{-0.051}^*$ [-0.0917,-0.0091]	$+0.005$ [-0.0609,0.0737]	$\mathbf{+0.180}^*$ [0.0995,0.2607]	$\mathbf{+0.231}^*$ [0.1476,0.3100]
Sonnet5	$\mathbf{+0.273}^*$ [0.2012,0.3417]	$\mathbf{-0.096}^*$ [-0.1420,-0.0470]	-0.032 [-0.0977,0.0316]	$\mathbf{+0.209}^*$ [0.1326,0.2896]	$\mathbf{+0.305}^*$ [0.2251,0.3825]
Mean of five	$\mathbf{+0.297}^*$ [0.2520,0.3387]	$\mathbf{-0.040}^*$ [-0.0648,-0.0152]	$\mathbf{+0.086}^*$ [0.0391,0.1314]	$\mathbf{+0.171}^*$ [0.1151,0.2261]	$\mathbf{+0.211}^*$ [0.1548,0.2640]

Table 16: Coveo purchase: paired differences in rank correlation, group 2 of 3, with 95% product-bootstrap intervals. A star marks an unrounded interval excluding zero; no multiplicity adjustment.

Model	$C-A80$	$R-A80$	$C-R$	$A-A80$	$B-B80$
GPT-5.6	$\mathbf{+0.160}^*$ [0.1033,0.2155]	$+0.057$ [-0.0148,0.1266]	$\mathbf{+0.104}^*$ [0.0487,0.1564]	-0.007 [-0.0759,0.0660]	$\mathbf{+0.164}^*$ [0.1205,0.2077]
Qwen3.8	$\mathbf{+0.285}^*$ [0.2115,0.3535]	$\mathbf{+0.177}^*$ [0.0966,0.2563]	$\mathbf{+0.108}^*$ [0.0432,0.1729]	$+0.023$ [-0.0589,0.0983]	$\mathbf{+0.149}^*$ [0.1054,0.1959]
Gemini3.8	$\mathbf{+0.196}^*$ [0.1327,0.2563]	$\mathbf{+0.074}^*$ [0.0011,0.1458]	$\mathbf{+0.122}^*$ [0.0641,0.1739]	$\mathbf{+0.105}^*$ [0.0292,0.1826]	$\mathbf{+0.181}^*$ [0.1303,0.2346]
GLM5.3	$\mathbf{+0.204}^*$ [0.1297,0.2767]	-0.074 [-0.1581,0.0197]	$\mathbf{+0.278}^*$ [0.1951,0.3534]	-0.027 [-0.1150,0.0601]	$\mathbf{+0.102}^*$ [0.0535,0.1526]
Sonnet5	$\mathbf{+0.259}^*$ [0.2011,0.3201]	$+0.036$ [-0.0414,0.1098]	$\mathbf{+0.224}^*$ [0.1544,0.2925]	-0.046 [-0.1209,0.0303]	$\mathbf{+0.127}^*$ [0.0775,0.1722]
Mean of five	$\mathbf{+0.221}^*$ [0.1784,0.2610]	$\mathbf{+0.054}^*$ [0.0168,0.0943]	$\mathbf{+0.167}^*$ [0.1242,0.2054]	$+0.010$ [-0.0336,0.0531]	$\mathbf{+0.145}^*$ [0.1003,0.1882]

Table 17: Coveo purchase: paired differences in rank correlation, group 3 of 3, with 95% product-bootstrap intervals. A star marks an unrounded interval excluding zero; no multiplicity adjustment.

Model	$B80-D$	$B-D$	$B-R$	$R-A$	$B80-A80$
GPT-5.6	-0.015 [-0.0538,0.0242]	$\mathbf{+0.150}^*$ [0.1034,0.2014]	$\mathbf{+0.246}^*$ [0.1842,0.3070]	$+0.064$ [-0.0120,0.1369]	$\mathbf{+0.139}^*$ [0.0867,0.1920]
Qwen3.8	$\mathbf{+0.074}^*$ [0.0284,0.1175]	$\mathbf{+0.223}^*$ [0.1684,0.2779]	$\mathbf{+0.311}^*$ [0.2401,0.3762]	$\mathbf{+0.154}^*$ [0.0715,0.2403]	$\mathbf{+0.339}^*$ [0.2668,0.4103]
Gemini3.8	$\mathbf{-0.044}^*$ [-0.0773,-0.0086]	$\mathbf{+0.137}^*$ [0.0867,0.1895]	$\mathbf{+0.232}^*$ [0.1663,0.2959]	-0.031 [-0.1078,0.0441]	$\mathbf{+0.125}^*$ [0.0645,0.1857]
GLM5.3	$\mathbf{-0.046}^*$ [-0.0899,-0.0022]	$+0.056$ [-0.0039,0.1129]	$\mathbf{+0.283}^*$ [0.2043,0.3639]	-0.047 [-0.1316,0.0379]	$\mathbf{+0.107}^*$ [0.0326,0.1848]
Sonnet5	$\mathbf{-0.063}^*$ [-0.1106,-0.0154]	$\mathbf{+0.064}^*$ [0.0022,0.1245]	$\mathbf{+0.192}^*$ [0.1093,0.2672]	$+0.081$ [-0.0063,0.1657]	$\mathbf{+0.101}^*$ [0.0445,0.1578]
Mean of five	-0.019 [-0.0522,0.0156]	$\mathbf{+0.126}^*$ [0.0787,0.1754]	$\mathbf{+0.253}^*$ [0.2023,0.3007]	$+0.044$ [-0.0006,0.0907]	$\mathbf{+0.162}^*$ [0.1204,0.2024]

Table 18: YooChoose purchase: paired differences in rank correlation, group 1 of 3, with 95% product-bootstrap intervals. A star marks an unrounded interval excluding zero; no multiplicity adjustment.

Model	$B-A$	$D-C$	$B-C$	$D-A$	$C-A$
GPT-5.6	$\mathbf{+0.183}^*$ [0.1418,0.2202]	$\mathbf{+0.143}^*$ [0.1104,0.1760]	$\mathbf{+0.180}^*$ [0.1269,0.2366]	$\mathbf{+0.145}^*$ [0.0898,0.2021]	$+0.002$ [-0.0621,0.0644]
Qwen3.8	$\mathbf{+0.333}^*$ [0.2751,0.3910]	$+0.029$ [-0.0028,0.0601]	$\mathbf{+0.082}^*$ [0.0375,0.1291]	$\mathbf{+0.280}^*$ [0.2049,0.3534]	$\mathbf{+0.251}^*$ [0.1810,0.3197]
Gemini3.8	$\mathbf{+0.251}^*$ [0.1969,0.3062]	$\mathbf{+0.033}^*$ [0.0041,0.0600]	$\mathbf{+0.172}^*$ [0.1283,0.2150]	$\mathbf{+0.111}^*$ [0.0471,0.1805]	$\mathbf{+0.079}^*$ [0.0112,0.1456]
GLM5.3	$\mathbf{+0.269}^*$ [0.2007,0.3365]	-0.003 [-0.0444,0.0401]	$+0.028$ [-0.0112,0.0655]	$\mathbf{+0.238}^*$ [0.1621,0.3086]	$\mathbf{+0.241}^*$ [0.1759,0.3023]
Sonnet5	$\mathbf{+0.215}^*$ [0.1522,0.2840]	$\mathbf{+0.109}^*$ [0.0665,0.1537]	$\mathbf{+0.130}^*$ [0.0862,0.1746]	$\mathbf{+0.195}^*$ [0.1305,0.2603]	$\mathbf{+0.086}^*$ [0.0209,0.1540]
Mean of five	$\mathbf{+0.250}^*$ [0.2139,0.2843]	$\mathbf{+0.062}^*$ [0.0338,0.0900]	$\mathbf{+0.118}^*$ [0.0833,0.1550]	$\mathbf{+0.194}^*$ [0.1426,0.2436]	$\mathbf{+0.132}^*$ [0.0845,0.1782]

Table 19: YooChoose purchase: paired differences in rank correlation, group 2 of 3, with 95% product-bootstrap intervals. A star marks an unrounded interval excluding zero; no multiplicity adjustment.

Model	$C-A80$	$R-A80$	$C-R$	$A-A80$	$B-B80$
GPT-5.6	-0.004 [-0.0638,0.0524]	-0.014 [-0.0766,0.0488]	$+0.010$ [-0.0394,0.0573]	-0.006 [-0.0523,0.0388]	$\mathbf{+0.033}^*$ [0.0085,0.0577]
Qwen3.8	$\mathbf{+0.247}^*$ [0.1837,0.3103]	$\mathbf{-0.101}^*$ [-0.1798,-0.0211]	$\mathbf{+0.348}^*$ [0.2809,0.4152]	-0.004 [-0.0733,0.0629]	$\mathbf{+0.040}^*$ [0.0150,0.0655]
Gemini3.8	$+0.041$ [-0.0239,0.1015]	$\mathbf{-0.097}^*$ [-0.1719,-0.0240]	$\mathbf{+0.138}^*$ [0.0845,0.1900]	-0.037 [-0.1002,0.0200]	$\mathbf{+0.062}^*$ [0.0304,0.0942]
GLM5.3	$\mathbf{+0.309}^*$ [0.2478,0.3705]	$+0.014$ [-0.0644,0.0938]	$\mathbf{+0.296}^*$ [0.2244,0.3627]	$+0.068$ [-0.0087,0.1442]	$+0.032$ [-0.0022,0.0672]
Sonnet5	$+0.055$ [-0.0073,0.1188]	-0.020 [-0.0918,0.0504]	$\mathbf{+0.075}^*$ [0.0064,0.1395]	-0.031 [-0.0988,0.0336]	$\mathbf{+0.036}^*$ [0.0016,0.0701]
Mean of five	$\mathbf{+0.130}^*$ [0.0864,0.1710]	$\mathbf{-0.044}^*$ [-0.0846,-0.0019]	$\mathbf{+0.173}^*$ [0.1365,0.2084]	-0.002 [-0.0353,0.0293]	$\mathbf{+0.041}^*$ [0.0181,0.0634]

Table 20: YooChoose purchase: paired differences in rank correlation, group 3 of 3, with 95% product-bootstrap intervals. A star marks an unrounded interval excluding zero; no multiplicity adjustment.

Model	$B80-D$	$B-D$	$B-R$	$R-A$	$B80-A80$
GPT-5.6	$+0.004$ [-0.0341,0.0437]	$+0.037$ [-0.0093,0.0846]	$\mathbf{+0.191}^*$ [0.1330,0.2511]	-0.008 [-0.0694,0.0532]	$\mathbf{+0.143}^*$ [0.1046,0.1796]
Qwen3.8	$+0.013$ [-0.0294,0.0575]	$\mathbf{+0.053}^*$ [0.0024,0.1057]	$\mathbf{+0.430}^*$ [0.3614,0.5005]	$\mathbf{-0.097}^*$ [-0.1825,-0.0119]	$\mathbf{+0.289}^*$ [0.2314,0.3469]
Gemini3.8	$\mathbf{+0.077}^*$ [0.0405,0.1128]	$\mathbf{+0.139}^*$ [0.0966,0.1822]	$\mathbf{+0.310}^*$ [0.2491,0.3679]	-0.059 [-0.1316,0.0111]	$\mathbf{+0.151}^*$ [0.0975,0.2018]
GLM5.3	-0.002 [-0.0420,0.0403]	$+0.030$ [-0.0182,0.0760]	$\mathbf{+0.323}^*$ [0.2513,0.3967]	-0.055 [-0.1460,0.0323]	$\mathbf{+0.305}^*$ [0.2405,0.3688]
Sonnet5	-0.015 [-0.0590,0.0284]	$+0.020$ [-0.0350,0.0732]	$\mathbf{+0.205}^*$ [0.1326,0.2723]	$+0.010$ [-0.0610,0.0848]	$\mathbf{+0.149}^*$ [0.0902,0.2087]
Mean of five	$+0.015$ [-0.0189,0.0508]	$\mathbf{+0.056}^*$ [0.0141,0.0984]	$\mathbf{+0.292}^*$ [0.2496,0.3337]	-0.042 [-0.0871,0.0033]	$\mathbf{+0.207}^*$ [0.1728,0.2388]

the same sampled sessions and under the same system message as Session (Appendix D), and it ranks worse (Table 21): by 0.144 $[0.1050, 0.1795]$ on Coveo cart, 0.161 $[0.1246, 0.1965]$ on Coveo purchase and 0.104 $[0.0697, 0.1375]$ on YooChoose, with the paired interval above zero in 14 of the 15 settings, with GPT on YooChoose the one null.

The pattern is consistent with the shared context noise of Section 5.2. When a product's 80 judgments share a call, any shift specific to that call is common to all of them and survives the average, whereas across 80 separate calls such shifts average out. On the logit scale, $\log\big(p/(1-p)\big)$ with probabilities clipped to $[10^{-3}, 1-10^{-3}]$, the intraclass correlation of a product's per-session predictions is 0.011 to 0.021 across the five models on Coveo when the sessions are judged separately, and 0.035 to 0.114 when they share a call, every increase distinguishable from zero; 11 to 30% of the batched arm's departure from the per-session one is a single shift common to the whole call, and that shift is unrelated to the product's observed rate, at rank correlations between -0.093 and 0.008. On YooChoose the per-session intraclass correlation is 0.029 to 0.086 when the sessions are judged separately, and sharing a call raises it by 0.011 to 0.035 for three models and not at all for Sonnet and GLM, so the shared shift does not account for the whole loss there. The transform matters only for the scale of these numbers: on raw probabilities the same quantities are 0.012 to 0.014 per-session against 0.022 to 0.064 in a shared call on Coveo cart, positive for every model. The session-level predictions of the two arms still agree closely, at correlations of 0.73 to 0.93, so batching lowers product-level accuracy while leaving the session-level predictions largely unchanged. With one call per product, we cannot tell whether this shared shift is random, so that a rerun of the same call would change it, or a systematic response to the particular set of sessions in the call.

Table 21: Session (one session per call) against Same-product batch (all 80 in one call) on the same sessions: rank correlations and their paired difference. A star marks an interval excluding zero.

	Coveo cart			Coveo purchase			YooChoose purchase		
	one	all 80	diff	one	all 80	diff	one	all 80	diff
GPT-5.6	0.313	0.198	+0.114*	0.328	0.193	+0.135*	0.417	0.400	+0.017
Sonnet5	0.278	0.106	+0.172*	0.301	0.101	+0.200*	0.278	0.144	+0.134*
Gemini3.8	0.281	0.134	+0.147*	0.311	0.148	+0.163*	0.429	0.372	+0.057*
GLM5.3	0.258	0.095	+0.163*	0.287	0.131	+0.156*	0.490	0.362	+0.128*
Qwen3.8	0.246	0.122	+0.124*	0.294	0.142	+0.152*	0.502	0.319	+0.182*
Mean of five	—	—	+0.144*	—	—	+0.161*	—	—	+0.104*

K STANDARDIZATION AND AVERAGING IN THE COMPARATIVE SUMMARIES

Batch summary differs from Summary in three ways at once: it compares 40 products in a call, it standardizes the outputs within that call, and it averages five groupings. Removing the last two in

turn separates them (Table 22). A single comparative call with no standardization already improves on Summary in all 15 settings, by 0.149, 0.179 and 0.127 on the three outcomes, so the comparison itself accounts for more than half of the gain. Standardizing that single call adds 0.086, 0.087 and 0.061; averaging five unstandardized groupings instead adds 0.073, 0.082 and 0.070; doing both adds only a little more than either. The overlap is consistent with both devices removing the same call-level variation, one by correcting each call and the other by averaging across calls, although standardization also depends on real differences among the products sharing a call. Consistent with that, two groupings of the standardized arm agree at rank correlations of 0.72 to 0.89 on Coveo, against 0.28 to 0.62 unstandardized, which leaves averaging little to remove. Giving Summary five judgments per product without comparison, by averaging five identical calls, adds 0.012, 0.013 and 0.063 (last row of Table 22).

A single standardized grouping of Batch summary requests one judgment per product, as Summary does. It is ahead of Session, which averages 80 judgments per product, by 0.040 $[-0.0071, 0.0908]$ on Coveo cart, 0.055 $[0.0090, 0.0995]$ on Coveo purchase and 0.056 $[0.0219, 0.0923]$ on YooChoose, with model-level intervals above zero in 2, 3 and 3 settings and below zero in one, GLM on YooChoose.

Table 22: Gain of Batch summary over Summary with and without within-call standardization, at one and five groupings: mean of five models, with the number of settings whose interval lies above zero. The fourth row is the reported arm; the last row averages five identical Summary calls, without comparison.

Variant	Coveo cart	Coveo purchase	YooChoose purchase
Batch summary, one grouping, raw	+0.149 (5/5)	+0.179 (5/5)	+0.127 (5/5)
Batch summary, one grouping, standardized	+0.235 (5/5)	+0.266 (5/5)	+0.188 (5/5)
Batch summary, five groupings, raw	+0.222 (5/5)	+0.261 (5/5)	+0.197 (5/5)
Batch summary, five groupings, standardized	+0.260 (5/5)	+0.297 (5/5)	+0.250 (5/5)
Summary, five identical calls averaged	+0.012 (1/5)	+0.013 (1/5)	+0.063 (5/5)

L Comparative gains across representations

Comparison helps summaries more than sessions in 14 of the 15 settings (Section 5.1). In Table 23, we estimate that difference directly rather than inferring it from the relative size of two gains. For each model and outcome we form

$$\left(\widehat{\rho}_B - \widehat{\rho}_A\right) - \left(\widehat{\rho}_D - \widehat{\rho}_C\right)$$

within each bootstrap replicate and report the percentile interval of that quantity. Because both gains are computed on the same replicate and the same common support, this is a paired estimate rather than a comparison of two separate intervals.

Fourteen of the 15 intervals are above zero and one contains zero, GPT on YooChoose; none is below. This estimates the difference between two complete procedures, each of which changes comparison context and standardization together, and for summary batching also repeats the product judgment. It does not isolate the contribution of any single prompting element.

M Session depth curves

In Figure 4, we recompute Session from $X \leq 80$ of its stored judgments per product, averaging over random draws, so no re-prompting is involved. Accuracy rises steadily with X and reaches Summary-80, which needs a single judgment, for every model on Coveo and four on YooChoose; GPT's curve on YooChoose ends just below it. Summary and Summary-80 are close on average, differing by -0.004, 0.010 and -0.002 over the five models. Figure 3 shows the same analysis for Batch session.

Table 23: Paired difference of differences $(\rho_B - \rho_A) - (\rho_D - \rho_C)$; positive values mean that comparison helps summaries more than sessions. A star marks an interval excluding zero.

Model	Estimate	95% interval
Coveo cart		
GPT-5.6	0.281*	[0.2117, 0.3475]
Sonnet5	0.350*	[0.2669, 0.4321]
Gemini3.8	0.201*	[0.1311, 0.2670]
GLM5.3	0.341*	[0.2561, 0.4244]
Qwen3.8	0.349*	[0.2697, 0.4246]
Coveo purchase		
GPT-5.6	0.317*	[0.2454, 0.3825]
Sonnet5	0.369*	[0.2870, 0.4472]
Gemini3.8	0.228*	[0.1583, 0.2933]
GLM5.3	0.287*	[0.2075, 0.3695]
Qwen3.8	0.485*	[0.4006, 0.5649]
YooChoose purchase		
GPT-5.6	0.040	[-0.0089, 0.0864]
Sonnet5	0.106*	[0.0221, 0.1916]
Gemini3.8	0.218*	[0.1603, 0.2803]
GLM5.3	0.271*	[0.1912, 0.3455]
Qwen3.8	0.304*	[0.2373, 0.3651]

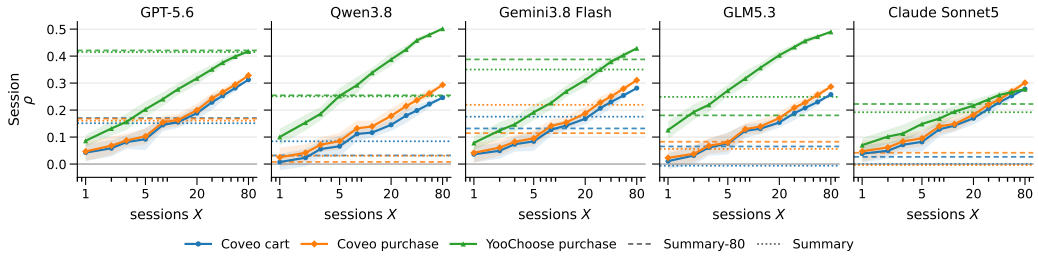


Figure 4: Session using X of each product's 80 sessions. Each point averages random draws of X stored judgments (band ± 1 sd). Dashed: Summary-80; dotted: Summary.

N DOUBLING THE SESSION SAMPLE

Batch session samples 80 sessions per product and requests one judgment on each, while Batch summary is computed from the product's full history. Section 5.1 attributes part of the difference between them to that gap in the amount of evidence each method uses. We test the attribution directly with Batch session-d, which repeats Batch session on 160 distinct sampled sessions per product instead of 80. Every other feature is retained: one session from each of 40 products per prompt, within-prompt standardization, and averaging by product.

The deeper arm doubles the inference cost, so we ran it for one model per dataset, choosing, on each dataset, the model for which a shortfall is most informative. On Coveo that is GPT-5.6, which has the highest Batch session rank correlation of the five models on both outcomes, so a shortfall there cannot be attributed to a model that ranks sessions poorly. On YooChoose most models are close to Batch summary, so a reduction would not be measurable; Gemini3.8 Flash is the exception, with the one sizeable gap. The Coveo comparison is restricted to the 743 products with at least 160 recorded sessions so that every arm in Table 24 is scored on one set of products; Batch summary is recomputed on the same restriction and is therefore not identical to the value in Figure 2. All 1,000 YooChoose products qualify. Calls lost in the Coveo run leave every product with at least 147 of its 160 judgments; the table uses all of them, while the extended curve in Figure 3 stops at 112 rounds, the deepest grid point every product supports.

Doubling the sample raises the session method by 0.084 $[0.0271, 0.1418]$ on Coveo cart, 0.094 $[0.0392, 0.1502]$ on Coveo purchase, and 0.044 $[0.0216, 0.0663]$ on YooChoose, every interval above zero. The number of sessions the method draws is therefore part of what separates it from the summary methods. It is not the whole of it: the Batch summary advantage remains above

zero at doubled depth in all three comparisons. Doubling the sample therefore does not close the gap, which is what we can show; it does not establish that no larger sample would, and the deeper arm is still improving at 160 sessions.

Table 24: Doubled session sample: GPT-5.6 on the 743 Coveo products with at least 160 sessions, and Gemini3.8 Flash on all 1,000 YooChoose products. Each arm's rank correlation and paired difference from Batch summary on the same products; a star marks an interval excluding zero.

		Coveo cart		Coveo purchase	
Arm	Sessions / judgments	ρ	B−arm	ρ	B−arm
Batch summary	full history / 5	0.510	—	0.521	—
Batch session	80 / 80	0.324	0.186*	0.330	0.192*
Batch session-d	160 / 160	0.408	0.103*	0.424	0.098*
		YooChoose purchase			
Batch summary	full history / 5	0.601	—		
Batch session	80 / 80	0.461	0.139*		
Batch session-d	160 / 160	0.505	0.096*		

The deeper arm also costs twice what Batch session costs, at 4,000 planned calls and 160,000 requested judgments per model and dataset, against 125 calls and 5,000 judgments for Batch summary. The additional evidence that closes part of the gap requires additional inference, whereas the summaries incorporate more sessions at no additional inference cost (Section 5.1).

O HELD-OUT SUPERVISED BENCHMARKS

The benchmarks in Section 5 are fit on products no LLM ever saw and scored on the same 1,000 evaluation products. Here we describe the training pool and the fitting, and give the comparisons model by model.

Training pool. The benchmarks are trained on products that no LLM ever saw, so the 1,000 evaluation products and their labels play no part in any fit. On Coveo we take every SKU in the feature file with at least 100 recorded sessions – the eligibility bar the evaluation pool itself used – and remove every product ever sent to a model in this project, including the discovery and evaluation pools. That leaves 8,410 training products. On YooChoose the evaluation pool required at least 300 sessions in the evaluation window and a price from the earlier window, and only 732 such products sit outside the YooChoose pools sent to a model, so we lower the bar to 100 sessions and train on 2,530 products; the 732-product fit at the pool's own bar is reported below. Features for the YooChoose training products are built by the same trace pass and the same block function as the evaluation pool, on each product's full evaluation-window history.

Fitting. Ridge selects its penalty from 30 logarithmically spaced values between 10^{-3} and 10^{4} using `RidgeCV`. The random forest uses 400 trees, a minimum leaf size of five, and seed zero. Gradient boosting is `HistGradientBoostingRegressor` at its defaults with seed zero. Missing features are replaced by zero, and columns are centered and scaled using the training rows only. The predictors use the full feature block, 20 columns for Coveo and 19 for YooChoose, which includes the two traffic counts the summary prompts withhold. Intervals resample the saved predictions on the shared product bootstrap rather than refitting inside each replicate, so every benchmark interval is paired with the LLM intervals it is compared against.

Paired comparisons with the LLM methods. The random forest and the gradient-boosted tree each lead every model's Batch summary in all 15 model–outcome settings, with the paired interval above zero in each. The narrowest gaps are Qwen on Coveo purchase, 0.064 $[0.0252, 0.1033]$, and Gemini on YooChoose, 0.107 $[0.0728, 0.1422]$; the widest is Sonnet on Coveo cart, 0.432. Ridge leads in 12 of the 15, the exceptions being GPT, Gemini and Qwen on YooChoose, where the intervals contain zero.

Where zero-shot comes closest. The two Coveo outcomes share products, features and prompts and differ only in the label, which makes them a clean comparison. Cart is the easier target for the supervised references, with the forest at 0.660 against 0.560 on purchase, yet the best Batch

Table 25: Supervised benchmarks trained on products no LLM saw (8,410 Coveo, 2,530 YooChoose) and scored on the 1,000 evaluation products. Features are the full product block, including the traffic counts withheld from the summary prompts. The single-feature rule ranks products by the negative average price of the most-considered competing product, fixed on the discovery pool. The last two rows repeat Batch summary.

Method	Coveo cart	Coveo purchase	YooChoose purchase
Ridge	0.629 [0.5864,0.6675]	0.529 [0.4795,0.5752]	0.612 [0.5683,0.6503]
Random forest	0.660 [0.6187,0.6963]	0.560 [0.5133,0.6042]	0.708 [0.6697,0.7394]
Gradient boosting	0.659 [0.6183,0.6988]	0.560 [0.5123,0.6031]	0.702 [0.6648,0.7342]
Single-feature rule	0.519 [0.4688,0.5640]	0.371 [0.3124,0.4230]	0.360 [0.3052,0.4129]
Best Batch summary	0.438	0.496	0.601
Mean Batch summary	0.341	0.390	0.541

summary reaches 66% of the forest on cart and 89% on purchase; the LLM methods do not share that ordering, with the best Batch summary at 0.438 on cart and 0.496 on purchase. Purchase is the rarer and noisier label: a mean rate of 0.0073 against 0.0400 for cart, no purchase at all for 30% of products, and about 35% of the variance across products attributable to binomial sampling error, against 9% for cart.

A single-feature reference. To measure what the LLM methods add over the strongest single behavioral signal, we rank products by one summary column, chosen without evaluation labels. On the Coveo discovery pool we take the counts-free column with the largest mean absolute Spearman correlation across the two outcomes and fix its sign there; this is the average price of the most-considered competing product, entered negatively (0.402, narrowly ahead of its median and of the other competitor-price averages). Applied unchanged, it reaches 0.519 $[0.4688, 0.5640]$ on Coveo cart, 0.371 $[0.3124, 0.4230]$ on Coveo purchase and 0.360 $[0.3052, 0.4129]$ on YooChoose (Table 25). On Coveo cart it leads every model's Batch summary, by 0.081 to 0.291, with every paired interval excluding zero. On Coveo purchase the model-mean difference is 0.019 and indistinguishable from zero: GPT and Qwen lead the rule, Sonnet trails it, and the rest cannot be separated. On YooChoose Batch summary leads the rule by 0.181 on average, above zero for four models and indistinguishable for Sonnet. The LLM methods thus fare best on purchase, which neither this rule nor a small labeled sample captures.

The matched YooChoose pool. Restricting YooChoose training to the 732 products that clear the evaluation pool's own 300-session bar changes little: the forest reaches 0.712 against 0.708 on the larger pool, gradient boosting 0.680 against 0.702, and ridge 0.600 against 0.612. Training on products that look exactly like the test products, rather than on more of them, is not what produces the gap.

Training-size curve. In Table 26, we refit the forest on random subsets of the training pool and score every fit on the same 1,000 evaluation products, so the curve and the LLM levels are on one scale. Each value averages six draws. The forest is still improving at the full pool in all three outcomes, so the label counts quoted in Section 5 hold at the tested scale and are not a supervised limit. Each evaluation pool contains 1,000 products, and the millions of sessions in each dataset are inputs to those product rows rather than additional labeled observations, so no benchmark at a larger labeled sample is available here and we claim none.

P INFERENCE USAGE

Tables 27 and 28 contain the totals across the six arms the logs identify; Session bag was run before the reported summary runs and its usage is not separable, so it is excluded. Tables 29 and 30 give the recorded blocks. Charges use list prices at execution time; output tokens include reasoning tokens, which are billed as output. The latency entries describe the observed run under its concurrency

Table 26: Random forest by number of labeled training products drawn from the held-out pool, scored on the 1,000 evaluation products; mean and standard deviation over six draws. The full-pool row is a single fit and can differ from Table 25 in the third decimal. The last row repeats the best Batch summary, which uses no labels.

Training products	Coveo cart	Coveo purchase	YooChoose purchase
15	0.348 (0.156)	0.279 (0.170)	—
25	0.461 (0.121)	0.276 (0.143)	0.400 (0.107)
50	0.554 (0.017)	0.389 (0.122)	0.474 (0.026)
100	0.565 (0.018)	0.457 (0.038)	0.535 (0.030)
200	0.600 (0.015)	0.464 (0.030)	0.628 (0.015)
400	0.620 (0.011)	0.506 (0.026)	0.650 (0.014)
800	0.634 (0.007)	0.527 (0.009)	0.679 (0.004)
1,600	0.648 (0.003)	0.542 (0.007)	0.694 (0.001)
3,200	0.654 (0.004)	0.552 (0.009)	—
Full pool	0.660 (8,410)	0.562 (8,410)	0.709 (2,530)
Best Batch summary	0.438	0.496	0.601

and retry settings. They are not intrinsic model-speed estimates or sequential end-to-end pipeline runtimes.

The two full-history summary methods share a usage record for A+B, and the two matched summary methods share A80+B80. We preserve these blocks rather than allocate their costs to methods without supporting logs. Calls include retries and smoke tests where present. Offline feature computation, session sampling, and evaluation are not included.

Basis of the main-text cost comparison. At the same 80,000 judgments, Session costs $103.88 against $38.96 for Batch session for GPT on Coveo, $36.99 against $17.97 for Gemini, $38.02 against $21.96 for GLM, $77.54 against $47.80 for Sonnet, and $234.93 against $74.47 for Qwen. The difference is instruction overhead: for GPT on Coveo, Session uses 17.3M input tokens against 8.8M. On exactly the same 80 sessions, the two matched-evidence summary methods together used 1.35M input tokens and 0.20M output tokens for GPT on Coveo, against 8.79M and 1.78M for Batch session, and cost $5.09 against $38.96. Because that block is pooled, the combined figure bounds either summary method from above. Reasoning tokens account for essentially none of the output for Sonnet (at most 0.1%) and 70–90% for Qwen everywhere, which produces the fourfold spread across models under Batch session and the sixfold spread under Session. Across the six arms the logs identify, the recorded charges are $751.27 on Coveo and $761.51 on YooChoose.

Table 27: Coveo: total inference usage for A, B, A80, B80, C and D, including retries. Tokens are in millions, and output includes reasoning; prices are list prices at run time. Seconds per call are specific to this run; offline data preparation is excluded.

Model	Calls	Input M	Output M	Reason M	Reason %	Out/call	s/call	USD
GPT-5.6	84,266	28.774	7.955	3.875	48.7	94	3.7	153.01
Qwen3.8	84,271	32.857	45.164	38.792	85.9	536	8.2	336.70
Gemini3.8	84,303	33.111	9.253	3.695	39.9	110	2.0	59.53
GLM5.3	84,264	29.644	5.449	1.136	20.9	65	1.4	65.48
Sonnet5	84,259	39.400	5.775	0.001	0.0	69	2.3	136.55
Total cost								751.27

Q Informativeness of context, standardization, and session methods at equal depth

The analyses behind Section 5.2 use only the saved inputs and predictions; no model was called again. In Table 31, we set the gain from each kind of comparative context against the informativeness of a single entry in the call.

Informativeness of a single entry. For sessions, we fit a gradient-boosted classifier to predict a session's outcome from the fields of its record as rendered in the prompts (Appendix D), with five folds split by product so that no product's own sessions contribute to its score, and take the predicted

Table 28: YooChoose: total inference usage for A, B, A80, B80, C and D, including retries. Tokens are in millions, and output includes reasoning; prices are list prices at run time. Seconds per call are specific to this run; offline data preparation is excluded.

Model	Calls	Input M	Output M	Reason M	Reason %	Out/call	s/call	USD
GPT-5.6	84,259	27.618	8.294	4.191	50.5	98	3.8	154.76
Qwen3.8	84,273	33.069	47.559	40.952	86.1	564	8.7	351.49
Gemini3.8	84,332	32.724	8.411	2.857	34.0	100	2.0	56.08
GLM5.3	84,319	29.294	5.660	1.436	25.4	67	1.5	65.91
Sonnet5	84,258	37.717	5.783	0.006	0.1	69	2.1	133.26
Total cost								761.51

Table 29: Coveo: inference usage by recorded arm block; the pooled A+B and A80+B80 logs cannot be split by arm. Tokens are in millions, output includes reasoning, and counts include retries.

Block	Calls	Input M	Output M	Reason M	Reason %	Out/call	s/call	USD
GPT-5.6								
A+B	1,133	1.360	0.197	0.077	39.1	174	3.9	5.09
A80+B80	1,133	1.345	0.200	0.076	38.1	176	3.8	5.09
C	80,000	17.280	5.776	3.622	62.7	72	3.5	103.88
D	2,000	8.788	1.782	0.099	5.6	891	9.9	38.96
Qwen3.8								
A+B	1,133	1.531	1.952	1.695	86.8	1,723	24.8	14.78
A80+B80	1,131	1.508	1.585	1.358	85.7	1,401	19.4	12.52
C	80,002	20.382	32.361	29.248	90.4	404	6.5	234.93
D	2,005	9.435	9.267	6.491	70.1	4,622	59.6	74.47
Gemini3.8								
A+B	1,131	1.649	0.289	0.099	34.4	255	2.9	2.32
A80+B80	1,130	1.648	0.272	0.088	32.2	240	3.0	2.26
C	80,037	19.047	6.054	3.508	57.9	76	1.9	36.99
D	2,005	10.766	2.638	0.000	0.0	1,316	7.9	17.97
GLM5.3								
A+B	1,130	1.368	0.195	0.049	25.3	173	3.7	2.77
A80+B80	1,129	1.357	0.188	0.045	23.7	166	4.4	2.73
C	80,001	17.988	2.918	0.714	24.5	36	1.1	38.02
D	2,004	8.931	2.148	0.328	15.3	1,072	11.0	21.96
Sonnet5								
A+B	1,129	1.899	0.182	0.000	0.0	162	3.0	5.62
A80+B80	1,129	1.884	0.183	0.000	0.1	162	2.8	5.59
C	80,000	23.757	3.002	0.000	0.0	38	2.1	77.54
D	2,001	11.859	2.408	0.001	0.0	1,203	9.8	47.80

log-odds as the session score. We report the share of the score's variance that lies between products (the intraclass correlation over each product's 80 sessions), the rank correlation between the product's observed rate and the score of one randomly drawn session per product (averaged over 50 draws), and the same for the average over all 80 sessions. For summaries, we fit a gradient-boosted regressor to the 18 summary columns given to the models, trained on the ranks of the observed rates with five-fold cross-fitting repeated five times; one entry is the product's summary. These scores are fit with labels and measure what the inputs contain, not what the LLMs extract. The LLMs' own session ratings show the same contrast between datasets: judged separately, 1.1 to 2.1% of their variance lies between products on Coveo and 2.9 to 8.6% on YooChoose (Appendix J).

Field by field, the behavioral fields of a Coveo session vary mostly within products: 1.2% of the variance of viewing time and 0.5 to 1.7% of the search fields lie between products, and at most 19% for any competitor field. On YooChoose the exact price is constant across a product's sessions and the click context nearly so (79% between products), and these two fields account for most of the difference in Table 32.

Effect of standardization. In Table 33, we compare each comparative arm with and without within-call standardization, on the same common supports and bootstrap draws as the main results.

Table 30: YooChoose: inference usage by recorded arm block; the pooled A+B and A80+B80 logs cannot be split by arm. Tokens are in millions, output includes reasoning, and counts include retries.

Block	Calls	Input M	Output M	Reason M	Reason %	Out/call	s/call	USD
GPT-5.6								
A+B	1,129	1.384	0.198	0.079	40.1	175	3.9	5.14
A80+B80	1,129	1.381	0.204	0.082	40.3	180	3.9	5.21
C	80,001	16.672	6.103	3.927	64.3	76	3.6	106.58
D	2,000	8.180	1.789	0.102	5.7	895	11.7	37.83
Qwen3.8								
A+B	1,135	1.674	1.840	1.613	87.6	1,621	21.6	14.39
A80+B80	1,133	1.650	1.733	1.515	87.4	1,530	21.5	13.70
C	80,003	20.356	34.203	30.860	90.2	428	7.2	245.93
D	2,002	9.389	9.783	6.964	71.2	4,886	52.6	77.47
Gemini3.8								
A+B	1,131	1.770	0.317	0.129	40.6	280	3.5	2.52
A80+B80	1,130	1.765	0.312	0.131	42.0	276	3.1	2.49
C	80,069	18.735	5.137	2.597	50.6	64	1.8	33.31
D	2,002	10.454	2.645	0.000	0.0	1,321	8.0	17.76
GLM5.3								
A+B	1,131	1.451	0.188	0.043	23.0	166	4.4	2.86
A80+B80	1,129	1.429	0.196	0.053	26.9	174	3.8	2.86
C	80,000	17.638	3.068	0.873	28.4	38	1.1	38.19
D	2,059	8.776	2.207	0.467	21.2	1,072	13.3	22.00
Sonnet5								
A+B	1,129	1.934	0.183	0.000	0.0	162	2.7	5.70
A80+B80	1,129	1.931	0.182	0.000	0.0	161	2.7	5.68
C	80,000	22.878	3.007	0.003	0.1	38	1.9	75.82
D	2,000	10.974	2.412	0.003	0.1	1,206	8.5	46.06

Table 31: Gain from comparison against the informativeness of one entry: the rank correlation between observed rates and a cross-fitted supervised score from a single session or summary of each product (Table 32), not defined for calls whose entries all belong to one product. Gain: mean paired difference over five models from the isolated counterpart.

	Informativeness of one entry			Gain from comparison		
Entries in one call	Cart	Purch.	YooCh.	Cart	Purch.	YooCh.
Sessions of one product	one product per call			-0.144	-0.161	-0.104
One session from each of 40 products	0.18	0.06	0.36	-0.044	-0.040	$+0.062$
Summaries of 80 sessions, 40 products	0.52	0.41	0.61	$+0.148$	$+0.162$	$+0.207$
Summaries of full histories, 40 products	0.63	0.54	0.70	$+0.260$	$+0.297$	$+0.250$

Standardization adds 0.086, 0.087 and 0.061 to a single comparative summary call (Table 22), less when five groupings are averaged, and changes Batch session, which averages 80 calls, by -0.006, -0.005 and 0.004 on average, with every interval containing zero; for individual models, 2 of the 15 intervals exclude zero, one in each direction, at -0.017 (GLM, Coveo cart) and 0.019 (GPT, YooChoose).

Batch session against Session at equal numbers of judgments. In Table 34, we recompute both session methods from the same number X of stored judgments per product, as in Figures 4 and 3, and report their difference by model, together with the loss from batching a product's own sessions into one call (Table 21). On Coveo the difference stays within 0.035 of zero at every depth for GPT, Qwen and Gemini, and widens with X for GLM and Sonnet. With 80 rounds we cannot tell whether it would close with more. The loss from batching a product's own sessions is smallest for GPT and largest for Sonnet on both Coveo outcomes.

Table 32: Informativeness of a single entry. Sessions: between-product share of the cross-fitted session score's variance, and rank correlation with the observed rate of one session's score and of the 80-session average; "No product-level fields" drops fields constant within a product (Coveo: price bucket; YooChoose: price and click context). Summaries: rank correlation of the cross-fitted score and of the best single column.

		Coveo cart	Coveo purchase	YooChoose
Session, all fields	between-product share	0.127	0.098	0.407
	one session	0.183	0.056	0.355
	80-session average	0.261	0.222	0.582
Session, no product-level fields	between-product share	0.097	0.125	0.099
	one session	0.124	0.048	0.150
Summary of 80 sessions	cross-fitted score	0.517	0.407	0.612
Summary of full history	cross-fitted score	0.629	0.544	0.699
	best single column	0.519	0.371	0.415

Table 33: Within-call standardization. Mean over five models of the standardized minus the unstandardized rank correlation, and the unstandardized Batch session−Session difference, with 95% paired-bootstrap intervals.

	Coveo cart	Coveo purchase	YooChoose
Batch summary (5 groupings)	0.039 $[0.025, 0.052]$	0.036 $[0.025, 0.048]$	0.053 $[0.040, 0.066]$
Batch summary-80	0.011 $[0.001, 0.022]$	0.017 $[0.007, 0.027]$	0.062 $[0.048, 0.075]$
Batch session (80 rounds)	-0.006 $[-0.020, 0.008]$	-0.005 $[-0.021, 0.011]$	0.004 $[-0.005, 0.013]$
Batch session (unstandardized)−Session	-0.038 $[-0.058, -0.016]$	-0.035 $[-0.058, -0.011]$	0.058 $[0.029, 0.086]$

Table 34: Batch session−Session when both use X judgments per product (averaged over random draws), and the loss from batching a product's 80 sessions into one call (Session−Same-product batch).

	Coveo cart, $X=$			Coveo purchase, $X=$			YooChoose, $X=$			Same-product batch loss		
Model	1	20	80	1	20	80	1	20	80	Cart	Purch.	YooCh.
GPT-5.6	$+.002$	$+.019$	$+.006$	$+.006$	$+.004$	$-.007$	$+.025$	$+.128$	$+.143$	.114	.135	.017
Qwen3.8	$+.013$	$+.011$	$+.000$	$+.006$	$-.009$	$-.020$	$-.003$	$+.029$	$+.029$	.124	.152	.182
Gemini3.8	$-.014$	$-.010$	$-.021$	$-.009$	$-.018$	$-.027$	$-.006$	$+.041$	$+.033$	.147	.163	.057
GLM5.3	$-.013$	$-.044$	$-.083$	$-.008$	$-.024$	$-.051$	$-.046$	$-.032$	$-.003$	.163	.156	.128
Sonnet5	$-.037$	$-.070$	$-.122$	$-.033$	$-.054$	$-.096$	$-.005$	$+.070$	$+.109$	.172	.200	.134

R SAME-PRODUCT AND CROSS-PRODUCT CONTEXT

In Section 5.2, we interpret task variation as the most direct way for context to inform how products compare. Here we state that interpretation and the evidence we have for it, which is consistent with it but does not establish it.

Interpretation. Ranking products correctly requires placing their characteristics relative to one another, for example knowing whether an average viewing time of 60 seconds or a given price is high or low for this store. Entries from different products show directly what is typical across products. A product's own sessions vary mainly across its shoppers, so we expect them to inform the product's own characteristics more than its place among other products. They are not uninformative about the environment, however: Coveo session records include the prices, viewing times and discovery order of competing products, and a model's prior may relate a product's records to the store as a whole. Sessions seen together can also reveal patterns no single session shows, such as the mix of shoppers a product attracts, and better inference about a product's own characteristics can improve its rank even when the model's view of the environment is unchanged. We therefore do not claim that same-product context cannot improve rankings, only that we expect it to contribute less than context that varies across products.

Evidence. Session bag places a product's 80 sampled sessions in one call and asks for a single product rate, so the model can use any pattern across the sessions. It ranks at 0.145, 0.137 and 0.250 on the three outcomes, against 0.085, 0.083 and 0.293 for Summary-80, the isolated summary of the same sessions, and 0.275, 0.304 and 0.423 for Session. The paired Session bag−Summary-

80 differences are 0.059 $[0.0185, 0.0991]$, 0.054 $[0.0168, 0.0943]$ and -0.044 $[-0.0846, -0.0019]$. This comparison of procedures is in line with the interpretation: a single judgment from one product's sessions together ranks much nearer an isolated product-level judgment than the average of separate session judgments. It does not isolate the reason, since Session bag also changes the requested output and the computation within the call. Same-product batch also carries context noise shared by all 80 judgments (Appendix J); with one call per product we cannot tell whether that shift is random or a systematic response to the particular sessions.