The Value of Personalization in Large Language Model Responses

Andrey Fradkin[1], Yan (Stella) Si[2], Giorgos Zervas[1,2]

[1]Questrom School of Business, Boston University

[2]Faculty of Computing & Data Sciences, Boston University

November 2, 2026

Abstract

We study heterogeneous preferences for LLM responses and how much users gain when the model that answers their prompt is chosen to match them. Using data from a platform where users compare pairs of anonymous model responses, we estimate a random-coefficients choice model in which preferences for response characteristics vary across users. We use this choice model to evaluate *routers*, policies that choose which model answers each user's prompt. Against a baseline that serves everyone the single model with the highest average predicted utility, a personalized router that uses both the prompt and the user's preferences attains a 70.5% win rate, the predicted share of comparisons in which the user prefers its choice. A router that uses only the prompt attains just 54.2%, so most of the gain comes from knowing the user. On average, the personalized router picks models with lower benchmark scores than the prompt-only router does. *Steering*, telling the baseline model a user's preferences in its prompt, nearly matches personalized routing, at 67.8%, and combining model routing and steering performs even better. We then investigate the feasibility of learning individual preferences: a router that learns from forty pairwise choices attains a 66.0% win rate. The routers above choose among responses already generated by every model. A router that must pick a model before any response is generated does worse: even after learning from forty past choices, its win rate is 57.0%. Overall, our results demonstrate that users disagree about what makes a response good and that routing across a diverse set of models creates value benchmark rankings miss.

1. Introduction

A common view of progress in AI is that scaling training data, compute, and inference is sufficient to achieve arbitrarily good performance across a variety of benchmarks. An implicit assumption behind this belief is that there is a primary dimension of intelligence that explains most of the value of the model. Yet LLM responses differ across a variety of dimensions including style, length, tone, format, and other dimensions not captured by benchmark intelligence. If consumers differ in their preferences for these characteristics across different tasks, then no single model will be best for everyone across tasks. In this paper, we document large gains in utility from using the prompt and user's preferences to route requests across models. To do so, we propose an empirical framework for measuring the gains from a variety of routing approaches.

We begin by estimating a random-coefficients discrete-choice model in which utility depends on the judged accuracy of a response, model identity and capabilities, response style, the prompt, and a user's latent tastes. The estimation uses preference data from Arena, a platform that presents users with anonymized responses from two randomly assigned models and asks which they prefer (Chiang et al. 2024). We score every response for factual accuracy with an LLM judge. Accuracy is the biggest driver of preference, though with diminishing returns. How much accuracy matters depends on the prompt, and users differ widely in how much they value a partly correct answer. Style also matters for preference. We estimate heterogeneous preferences for a variety of characteristics (e.g., tables, response length) and interactions between these characteristics and prompt embeddings. These heterogeneous preferences affect which model is best across users. If each user must be assigned a single model, 57% have a personal optimum different from the mean-taste winner. No model dominates the ten-model candidate set: under personalized assignment, the leading model accounts for only 23.9% of user-prompt choices, and the top two models together receive 40.0%, leaving more than half to eight other models.

Heterogeneous preferences generate large gains from routing. Relative to always using the best single model in our sample, a prompt-level router raises mean predicted utility and attains a 54.2% model-implied head-to-head win rate. Giving the router knowledge of each user's taste vector further increases utility gains and the win rate to 70.5%. Personalization (vs knowing just the prompt) accounts for about three quarters of the total utility gain. This benefit comes from frequently selecting models that have lower benchmark intelligence on average than the prompt-only router's selections.

We next ask how much of this value a platform can capture by learning tastes from pairwise choices. We simulate 40 rounds of pairwise feedback, and use a Bayesian approach to learn the preference vector. This approach closes 74% of the utility gap between an unpersonalized router and a router that knows each user's preference vector. This

comparison is closely related to recent methods that represent users by latent variables or ideal points in a shared preference space (Poddar et al. 2024; Chen et al. 2025), as well as Shenfeld et al. (2025), who factorize user-specific rewards into shared base functions and use active learning to infer a new user's weights from a small number of pairwise comparisons.

We also test a simpler intervention that requires no model fine-tuning but does require knowing the user's preferences: a natural-language preamble that instructs the model to match the user's tastes. With instructions built from each simulated user's tastes, the intended user prefers Gemini 2.5 Pro's steered response to its unsteered response with predicted probability 69.2%, and steering raises the intended user's utility for every model we steer. Against the same baseline on the same prompts and users, steering the baseline model wins 67.8% of the time, close to the 71.2% of the user-by-prompt router across all ten models' original responses. Combining routing and steering performs even better, achieving an 82.0% win rate.

These gains do not come for free. Our main router operates ex-post: it generates one response from each of ten models and then selects among them, increasing inference costs and latency. To address this concern, we evaluate ex-ante routers that choose a model using only information available before generation, thus saving the cost of prompting all ten models before routing. We benchmark these against the same baseline, always using the best model in the sample, on held-out prompts and users, where the ex-post personalized router attains a 70.0% win rate and the ex-post prompt router 54.2%. In one approach, we predict each model's response characteristics, including its accuracy bin, from the prompt embedding and score them through the choice model. This prompt-only ex-ante router attains a win rate of only 50.2%, essentially a coin flip against the baseline: response characteristics are hard to predict before generation.

In another approach, we built a router that uses a taste vector learned from 40 of the user's past pairwise choices to pick one best model without generating responses from all models; it attains a 55.3% win rate, 25% of the gain of the ex-post user-by-prompt router. Combining the two approaches raises the win rate to 57.0%, 35% of that gain. One reason for this relatively poor proportion is sampling variance. The utility of the same model's response varies across draws nearly as much as the prompt-specific edge that a router exploits. These results highlight the challenges of making routing choices prior to a response.

A natural question about our findings, and empirical research on LLMs more broadly, is whether they will hold for newer models and forms of interaction. While specific parameter estimates will shift as models improve, a central contribution of this paper is to provide a portable quantitative framework that platforms can re-estimate to evaluate routing strategies on any candidate set of models. Anecdotal evidence from Arena and

commercial routing services further suggests that heterogeneity in model strengths and user tastes continues to make routing valuable.

The closest paper to ours is Caoui (2025), who uses Arena choices to estimate demand for differentiated LLMs and decomposes changes in consumer surplus into improvements in intelligence and other model attributes. We use a similar demand framework but shift the counterfactual from measuring market-wide quality growth to assigning responses across models and users. Our contribution is to quantify separately the value of prompt-level routing and user-level personalization in a choice model that accounts for judged response accuracy, and to evaluate two mechanisms for learning and acting on individual tastes. The paper also complements work documenting pluralistic preferences in human feedback (Kirk et al. 2024) and divergence between aggregate and individual LLM rankings (Garbacea, Wang, and Tan 2026). Our findings are also consistent with Demirer, Fradkin, and Tadelis (2026), who document horizontal and vertical differentiation in the market for LLM inference and show that no single model dominates across use cases. More broadly, our results provide empirical content for models of product design and the pricing of LLM services (Bergemann, Bonatti, and Smolin 2025).

The paper proceeds as follows. Section 2 describes the data, response attributes, and accuracy judge. Section 3 presents the choice model and identification. Section 4 reports the estimates. Section 5 uses them to measure the value of personalization, whether a platform can learn tastes from feedback, how much of the value survives when it must choose a model before generation, and whether a single model can be personalized with language. Section 6 concludes and discusses limitations. The appendices give the data construction, the full coefficient tables and identification diagnostics, and the details of every simulation.

2. Data

Arena, originally introduced as Chatbot Arena and also known as LMArena, is a widely used online platform for evaluating large language models through human preference comparisons. Its public model rankings are frequently cited by leading model developers and companies (Chiang et al. 2024). Users submit prompts and receive side-by-side responses from two randomly assigned models, shown anonymously. Arena calls these comparisons "battles." Users can select either response, declare a tie, or judge both responses to be bad. Voting is voluntary; if users vote, Arena then reveals the model identities. Arena aggregates these votes into a public leaderboard. The data thus reflect users' decisions both to use Arena and to vote.

We pool two public data releases covering June–August 2024 and April–July 2025. Each battle record contains the user's prompt, the two models' responses, their model identities, and the user's vote. We restrict the sample to English-language battles and drop

incomplete observations.[1] Using a fixed seed, we split battles at random into training (90%) and test (10%) samples, yielding 115,455 training and 12,928 test battles. We use the training sample to estimate preferences and the test sample to assess predictive fit. We break ties randomly in favor of A or B and retain "both bad" votes as a separate outcome. Table 1 reports descriptive statistics for the estimation sample.

For each response, we construct twelve features: token length; counts of ordered lists, unordered lists, headers, and bold spans; indicators for tables and emoji; Flesch reading ease; positive and negative sentiment; and assertive and tentative hedging. We calculate length, formatting, and readability measures directly from each response. We use gpt-oss-20b, an open-weight language model, to classify sentiment and hedging. We standardize length, Flesch reading ease, and the list, header, and bold-span counts using training-sample means and standard deviations; the binary features stay 0/1. We also computed a second readability measure, the SMOG index, and drop it: its rank correlation with Flesch reading ease is −0.84, and its taste standard deviation sat at the optimization floor.

We score every response in the estimation sample for factual accuracy using GLM 5.2, chosen for its low cost when scoring at scale. The judge sees the prompt and response and returns a score from 0 to 10 with a short rationale. We group scores into three accuracy bins: low (0–3), mid (4–6), and high (7–10). To assess how sensitive the accuracy bins are to our choice of judge, we rescore a validation sample with Claude Opus 4.8 and find that the two judges assign the same accuracy bin to an estimated 81% of responses in the corpus (Appendix B).

To capture how preferences vary with the task, we summarize each prompt with three scores. We embed prompts using OpenAI's `text-embedding-3-large` and use partial least squares (PLS) to identify directions that predict differences in the thirteen raw response features (the twelve above and the SMOG index) between winners and losers. We fit this projection on training battles and apply it to all prompts in our sample. Appendix A details these steps.

We measure model intelligence with the Artificial Analysis Intelligence Index, a composite benchmark score covering all 105 models in our sample, and scale it in units of ten points.[2] We also record each model's context window in millions of tokens and an indicator for whether it uses extended chain-of-thought reasoning before responding.

To evaluate routing across models, we assemble an additional response panel using prompts sampled from Arena. We select ten models for comparison. For each sampled

[1] The releases are `arena-human-preference-100k` and `arena-human-preference-140k`. We exclude battles without a usable vote or complete response features, model attributes, and prompt scores, as described below.

[2] The Epoch Capabilities Index predicts judged accuracy somewhat better at the model level (R^2 of 0.70 against 0.58) but is available for only 73 of them.

TABLE 1. Descriptive statistics for the Arena estimation sample.

Variable	Mean	SD	P10	P90
Response attributes (N = 256, 766)				
Length (tokens)	798.74	1201.11	81.00	1703.00
Ordered lists (count)	4.16	9.73	0.00	11.00
Unordered lists (count)	10.44	22.47	0.00	29.00
Headers (count)	2.30	6.94	0.00	8.00
Bold spans (count)	13.24	27.14	0.00	36.00
Has emoji	0.11	0.31	0.00	1.00
Has table	0.06	0.23	0.00	0.00
Flesch reading ease	43.55	88.47	15.44	76.04
Positive tone	0.22	0.41	0.00	1.00
Negative tone	0.02	0.13	0.00	0.00
Assertive hedge	0.47	0.50	0.00	1.00
Tentative hedge	0.18	0.38	0.00	1.00
Judged accuracy (N = 256, 766)				
Accuracy score (0–10)	7.47	2.99	2.00	10.00
Low (0–3)	0.14	0.35	0.00	1.00
Mid (4–6)	0.17	0.38	0.00	1.00
High (7–10)	0.68	0.46	0.00	1.00
Model attributes (N = 105)				
Intelligence index (points)	43.85	13.89	24.52	63.49
Context window (million tokens)	0.34	1.01	0.01	1.00
Reasoning indicator	0.23	0.42	0.00	1.00
Prompt attributes (N = 128, 383)				
PLS score 1	0.01	0.12	−0.16	0.16
PLS score 2	0.00	0.07	−0.09	0.09
PLS score 3	0.00	0.07	−0.10	0.09

Note: The sample contains 128,383 battles, 256,766 responses, and 105 models. Vote shares before randomizing ties are A wins 33.6%, B wins 34.5%, tie 17.2%, both bad 14.7%. Means, standard deviations, and percentiles are calculated across responses for response features and accuracy, across distinct models for model attributes, and across battles for prompt scores. Model attributes are not weighted by how often a model appears in the data. Response features are in original units; binary means are proportions. SD denotes sample standard deviation; P10 and P90 denote the 10th and 90th percentiles.

prompt, we retain any responses from these models already recorded in Arena and request a response from each model whose answer is missing. The complete panel covers 4,916 prompts. We measure its response attributes and accuracy using the same procedures as for the estimation data and compute prompt scores using the PLS projection fitted on training battles. Section 5 lists the candidate models. We also generate responses under personalized instructions to study verbal personalization (Section 5.4) and draw repeated responses to the same prompts to study sampling variation (Section 5.3).

3. Empirical framework

We model each Arena vote as a discrete choice among the two responses and the option to reject both. A user's utility from a response depends on its accuracy, its style, and the model that produced it, and users differ in how much they value each.

Let i index users, p prompts, and m language models. In each battle, user i submits prompt p and sees two responses, A and B, from two models. The user picks one response or rejects both. We index these options by $j \in \{A, B, 0\}$, where 0 is "both bad." We write user i's utility from model m's response to prompt p as $U_{ipm} = V_{ipm} + \varepsilon_{ipm}$, where ε_{ipm} is an idiosyncratic shock. The systematic utility V_{ipm} depends on observed characteristics and the user's tastes:

$$V_{ipm} = \beta_{ip}^{A\prime} d_{pm} + \beta_{ip}^{R\prime} x_{pm}^{R} + \beta_{ip}^{M\prime} x_{m}^{M} + \delta_m + \alpha \mathbf{1}\{\text{position B}\}. \tag{1}$$

The superscripts A, R, and M mark the three groups of characteristics: accuracy, response features, and model attributes. The vectors β_{ip}^{A}, β_{ip}^{R}, and β_{ip}^{M} are user i's coefficients on each group. The vector d_{pm} contains indicators for the mid and high accuracy bins, with low as the reference. Bins let the value of accuracy be nonlinear and avoid treating the judge's scale as cardinal. The vector x_{pm}^{R} contains the twelve response features and x_{m}^{M} the three model attributes.

The model fixed effect δ_m captures persistent differences in utility across models that accuracy, response features, and prompt interactions do not explain. We include 105 fixed effects, one for each model in the estimation sample; we drop the one battle whose model appears fewer than 50 times. The coefficient α captures the effect of appearing on the right.

Each coefficient vector is a mean that shifts with the prompt plus a user-specific deviation:

$$\begin{aligned} \beta_{ip}^{A} &= \beta_{0}^{A} + B^{A} h_p + \nu_{i}^{A}, \\ \beta_{ip}^{R} &= \beta_{0}^{R} + B^{R} h_p + \nu_{i}^{R}, \\ \beta_{ip}^{M} &= \qquad B^{M} h_p + \nu_{i}^{M}, \end{aligned} \tag{2}$$

where h_p contains the three prompt scores. The intercepts β_0^A and β_0^R give mean preferences at $h_p = 0$. The matrices B^A, B^R, and B^M shift mean preferences with the prompt. The third equation has no intercept, because the model fixed effects absorb it.

We assume the taste deviations are normally distributed. Stacking ν_i^A, ν_i^R, and ν_i^M, which have two, twelve, and three components, gives $\nu_i \sim N(0, \Sigma)$, where $\Sigma = \mathrm{diag}(\Sigma^A, \Sigma^R, \Sigma^M)$ and each block is a diagonal matrix of variances. A larger variance for a characteristic means users differ more in how much they value it. We assume that taste deviations are independent of the prompt and of the responses' characteristics.

We write V_{ipA} and V_{ipB} for the systematic utilities of the two responses in a battle and normalize the systematic utility of "both bad" to $V_{ip0} = 0$. We assume the shocks ε_{ipj} are i.i.d. standard type-I extreme value, independent of ν_i. These assumptions fix the location and scale of utility, so we measure utility in logits.

We integrate the logit probability over the taste distribution to obtain each choice probability:

$$P_{ip}(j) = \int \frac{\exp\{V_{ipj}(\nu)\}}{1 + \exp\{V_{ipA}(\nu)\} + \exp\{V_{ipB}(\nu)\}} \, \phi(\nu; \Sigma) \, d\nu, \tag{3}$$

where $\phi(\cdot; \Sigma)$ is the $N(0, \Sigma)$ density and the numerator is one for "both bad." Given a user's tastes, and conditional on not rejecting both responses, the user picks response A over B with probability $\mathrm{logit}^{-1}(V_{ipA} - V_{ipB})$, where $\mathrm{logit}^{-1}(x) = 1/(1 + e^{-x})$ is the logistic function. We use this probability to express utility differences as win rates.

Arena does not link battles to users, so we estimate the population distribution of tastes, not the tastes of any particular user. We identify the variances from how the probabilities of picking each response and of rejecting both change with the two responses' characteristics across battles, as in random-coefficients demand estimation with cross-sectional data (Berry, Levinsohn, and Pakes 1995; Nevo 2001; Berry and Haile 2014). Arena assigns models to battles at random, so the models a user sees are independent of the user's tastes. Independence from the prompt, which the user writes, and from the responses' characteristics remains an assumption. Without linked choices, the variances may also reflect variation in a user's tastes from one battle to the next (Section 6).

We estimate the parameters by simulated maximum likelihood (McFadden and Train 2000).[3] In simulated data, our estimator recovers the mean coefficients, fixed effects, and large standard deviations closely; smaller standard deviations can be biased or land at the lower bound (Appendix C).

We compute standard errors from the likelihood Hessian, treating the prompt scores as fixed. These standard errors therefore omit uncertainty from estimating the prompt projection, which uses the same training battles. Appendix A reports a split-sample check

[3]The model has 188 parameters. We approximate the integral with 256 quasi-random draws.

of the projection's stability.

4. Preference estimates

Table 2 reports the estimates of the preference model in Equations (1) and (2); Appendix C reports the prompt interactions and the model fixed effects. Judged accuracy has the largest effect on choice. Conditional on not rejecting both responses, a user with mean tastes picks a mid-accuracy response over an otherwise identical low-accuracy one 75% of the time, and a high-accuracy response 85% of the time. Moving from mid to high accuracy adds less: a high-accuracy response wins against an otherwise identical mid-accuracy one 66% of the time. By comparison, a response one standard deviation longer wins 61% of the time.

The value of accuracy varies across prompts and users. A one-standard-deviation increase in the first prompt score raises the win rate of a high-accuracy response over a low-accuracy one from 85% to 92%. Across users, the value of a mid-accuracy response varies widely. Its standard deviation exceeds its mean, so on the average prompt 23% of users prefer a low-accuracy response to an otherwise identical mid-accuracy one, and 33% prefer a mid-accuracy response to a high-accuracy one. Part of this variation may reflect judge error on answers that are hard to grade. A second judge keeps 91% of high-accuracy responses in the high bin but only 46% of mid-accuracy responses in the mid bin (Appendix B). We find little evidence that users differ in the value of a high-accuracy response.

Holding judged accuracy fixed, users differ most in their preferences for tables, emoji, length, tentative hedging, headers, and bold spans (Table 2). For each of these features, the standard deviation across users exceeds the mean, so many users have the opposite taste from the average user. For a user with mean tastes on the average prompt, a response with a table wins 59% of the time against an otherwise identical one without, yet 37% of users prefer the response without a table. Similarly, 46% of users dislike emoji, and 33% prefer shorter responses. For assertive hedging and negative tone, the estimated standard deviations are at or near the lower bound.

Preferences for length also vary with the prompt. A one-standard-deviation increase in the second prompt score lowers the win rate of a response one standard deviation longer from 61% to 56%. By contrast, prompt scores move the value of tables far less than differences in tastes across users do. Appendix Table A3 reports the interactions.

Models with higher benchmark scores have larger fixed effects. The fixed effects capture persistent differences in utility across models that judged accuracy and observed style do not explain, such as capability and default behavior (Appendix Figure A1 plots the 105 effects against the benchmark index). We regress the 105 fixed effects on the model attributes. Ten more points on the benchmark index are associated with a fixed effect 0.163

TABLE 2. Estimates of the preference model, in logits. Panels A–C report the mean on the average prompt ($h_p = 0$) and the standard deviation across users of each coefficient in Equation (2). Accuracy bins are relative to a low (0–3) response. We standardize length, Flesch reading ease, and the list, header, and bold-span counts; the other response features are indicators. Model-attribute coefficients have no intercept. Standard deviations of 0.020 are at the lower bound imposed in estimation. Standard errors are in parentheses. Appendix Table A2 compares held-out fit with standard logits.

	Mean	Std. dev.
A. Accuracy bins, d_{pm} (mean β_0^A; std. dev. from Σ^A)		
Mid (4–6) vs. low	1.109 (0.026)	1.498 (0.067)
High (7–10) vs. low	1.770 (0.025)	0.020 (0.036)
B. Response features, x_{pm}^R (mean β_0^R; std. dev. from Σ^R)		
Length (standardized)	0.442 (0.028)	1.011 (0.046)
Ordered lists (standardized count)	0.070 (0.013)	0.235 (0.066)
Unordered lists (standardized count)	0.066 (0.019)	0.135 (0.060)
Headers (standardized count)	0.063 (0.019)	0.524 (0.056)
Bold spans (standardized count)	0.232 (0.024)	0.529 (0.058)
Table	0.369 (0.048)	1.158 (0.169)
Emoji	0.121 (0.029)	1.335 (0.102)
Flesch reading ease (standardized)	−0.060 (0.026)	0.221 (0.140)
Positive sentiment	0.092 (0.017)	0.281 (0.141)
Negative sentiment	−0.087 (0.056)	0.049 (0.185)
Assertive hedging	−0.101 (0.015)	0.020 (0.042)
Tentative hedging	−0.123 (0.019)	0.661 (0.085)
C. Model attributes, x_m^M (no intercept; std. dev. from Σ^M)		
Intelligence index (per 10 points)	—	0.020 (0.009)
Context window (M tokens)	—	0.826 (0.103)
Reasoning model	—	0.888 (0.074)
D. Other parameters		
Position effect (α)	0.016 (0.008)	—
Prompt interactions (B^A, B^R, B^M)	51 coefficients; selected in Appendix Table A3	
Model fixed effects (δ_m)	105 effects; Appendix Figure A1	
Training battles		115,455
Parameters		188
Log-likelihood, training		−106,181.6
Log-likelihood, test		−12,068.0

logits higher (t = 6.8): a response from the higher-scoring model wins 54% of the time against an otherwise identical one. We read this coefficient as a description, not a causal effect. Models with higher benchmark scores also differ in ways we do not observe, for example in how much post-training they receive for conversation. When we run the same regression on fixed effects from standard logits, the benchmark coefficient falls by about two fifths once response characteristics enter (Appendix C). This decline is consistent with more capable models earning part of their premium through more accurate and better-formatted responses.

5. Results

We use the estimated preferences to evaluate how much personalization is worth and whether a platform can capture it. We first measure how much users would gain if a platform knew their tastes (Section 5.1). A platform cannot observe tastes directly, so we next ask whether it can learn them from a user's choices (Section 5.2). Serving the best of ten responses requires generating all ten, so we then ask how much of the gain survives when the platform must choose a model before generating a response (Section 5.3). Finally, we study an alternative to switching models: telling one model what the user likes (Section 5.4).

We use the panel described in Section 2: 4,916 prompts with one response from each of ten models, GPT-4.1, GPT-4o, Claude Sonnet 4, Gemini 2.5 Pro, DeepSeek V3, Llama 4 Maverick, Grok 4, Qwen3 235B, Kimi K2, and MiniMax M1. We simulate users by drawing taste vectors from the estimated distribution, $\nu_i \sim N(0, \Sigma)$, and evaluate each response with the user's systematic utility in Equation (1). Our baseline serves every user and prompt the same model, Gemini 2.5 Pro, which has the highest average utility in the panel. It is the model a platform that does not personalize would choose. The routing and learning results are counterfactuals: we use the estimated preferences to predict how users would fare under policies that Arena does not use. The steering results use newly generated responses, but we also score them with the estimated preferences. We collect no new human votes.

For policy r, we measure the gain as the expected difference in utility, integrating over the logit shocks. No policy observes the shocks, so the expected gain equals the difference in systematic utility, $\Delta V_{ipr} = V_{ip,r(p,i)} - V_{ip,\text{Gemini}}$. We report the mean gain and the win rate, the average of $\text{logit}^{-1}(\Delta V_{ipr})$: the probability that the user picks the policy's response over Gemini's, conditional on not rejecting both (Section 3). The win rate thus integrates over the shocks of the two responses. Because gains vary across users and prompts, the win rate differs from logit^{-1} of the mean gain. Where a result requires simulated choices, as when a platform learns a user's tastes, we draw each choice from the logit choice probabilities, so the shocks make the simulated choices noisy. The intervals we report

TABLE 3. Utility gains and win rates of the two routers. The baseline always serves Gemini 2.5 Pro. The gain in the last row equals the user-by-prompt router's gain minus the prompt router's; win rates do not subtract because they average a nonlinear function of the gains.

Comparison	Mean gain (logits)	Win rate
Prompt router vs. baseline	0.312	54.2%
User-by-prompt router vs. baseline	1.231	70.5%
User-by-prompt router vs. prompt router	0.919	66.4%

describe variation across simulated users, conditional on the estimated preferences and the response panel.

5.1. The value of personalization

We first ask how much users would gain over the baseline if the platform routed each prompt with full information. For each prompt, the prompt router serves the model with the highest utility for the average user, $\arg\max_m V_{0pm}$. The user-by-prompt router serves the model with the highest utility for that user on that prompt, $\arg\max_m V_{ipm}$. Neither router observes the logit shocks, so both maximize expected utility. We evaluate both on 1,000 simulated users.

The prompt router wins against the baseline 54.2% of the time (Table 3). Its gain is small because it serves the baseline model itself on 50% of prompts, where its gain is zero. The user-by-prompt router wins 70.5% of the time. It can always pick the prompt router's choice, so it never does worse, and it picks a better response in 67% of user-prompt pairs. Knowing the user therefore accounts for about three quarters of the gain in mean utility, 0.919 of 1.231 logits.

Users disagree about which model is best. Under the user-by-prompt router, the leading model serves about a quarter of user-prompt pairs, and eight of the ten models together serve more than half (Table 4). Even if each user had to choose one model for all prompts, the best model for 57% of users would differ from the best model for the average user. The personalized router also trades benchmark capability for fit: the models it picks score 62.3 on the benchmark index on average, against 68.0 for the prompt router.

5.2. Learning tastes from pairwise feedback

The user-by-prompt router knows each user's tastes; a platform must learn them. We simulate a platform that knows the estimated population parameters but not the user's taste deviation ν_i. In each round, the user submits a new prompt, and the platform shows two responses: one from the model it currently estimates is best for the user, and one from a randomly chosen other model. The user picks one or rejects both, with the logit

TABLE 4. Assignment shares of the user-by-prompt router over user-prompt pairs. Users may switch models across prompts.

Model	Share	Model	Share
Gemini 2.5 Pro	23.9%	Qwen3 235B	8.6%
Grok 4	16.1%	Kimi K2	8.4%
DeepSeek V3	12.7%	Claude Sonnet 4	4.7%
GPT-4.1	10.5%	GPT-4o	2.4%
MiniMax M1	10.4%	Llama 4 Maverick	2.3%

probabilities under the user's true tastes. After each round, the platform updates its beliefs about the user's tastes by Bayes' rule, using the estimated taste distribution as the prior. Let $y_{is} \in \{A, B, 0\}$ be the user's choice in round s. Given ν_i and the responses shown in each round, each choice follows the logit probability, and the logit shocks are independent across rounds. The posterior density of the user's tastes after t rounds is

$$p(\nu \mid y_{i1}, \ldots, y_{it}) \propto \prod_{s=1}^{t} \ell_{is}(\nu)\, \phi(\nu; \Sigma), \tag{4}$$

where $\ell_{is}(\nu)$ is the logit probability of y_{is} given tastes ν, the fraction inside the integral in Equation (3). The platform's estimate is the posterior mode. Taking logs and dropping constants,

$$\hat{\nu}_{it} = \arg\max_{\nu} \sum_{s=1}^{t} \log \ell_{is}(\nu) - \tfrac{1}{2}\, \nu' \Sigma^{-1} \nu. \tag{5}$$

Before any feedback, $\hat{\nu}_{i0} = 0$ and the platform serves the prompt router's choice. In later rounds, and on held-out prompts, it serves the model with the highest systematic utility at $\hat{\nu}_{it}$. We simulate 200 users, drawing their tastes from $N(0, \Sigma)$, and evaluate the platform's choices on 800 prompts not used for feedback (Appendix E).

Learning is fast (Figure 1). Before any feedback, the router serves the average user's best response and wins against the baseline 53.7% of the time. After 10 comparisons it closes 46% of the gap to the user-by-prompt router. After 40 it closes 74% and wins 66.0% of the time. A short history of choices therefore recovers most of the value of knowing a user's tastes, provided that those tastes persist across prompts.

5.3. Choosing a model before generation

The routers above require a response from every model, so the platform generates ten responses to serve one. A platform that wants to avoid this cost must choose a model before any response exists. At that point it can observe the prompt and the user's past choices. We ask which of the two is more valuable.

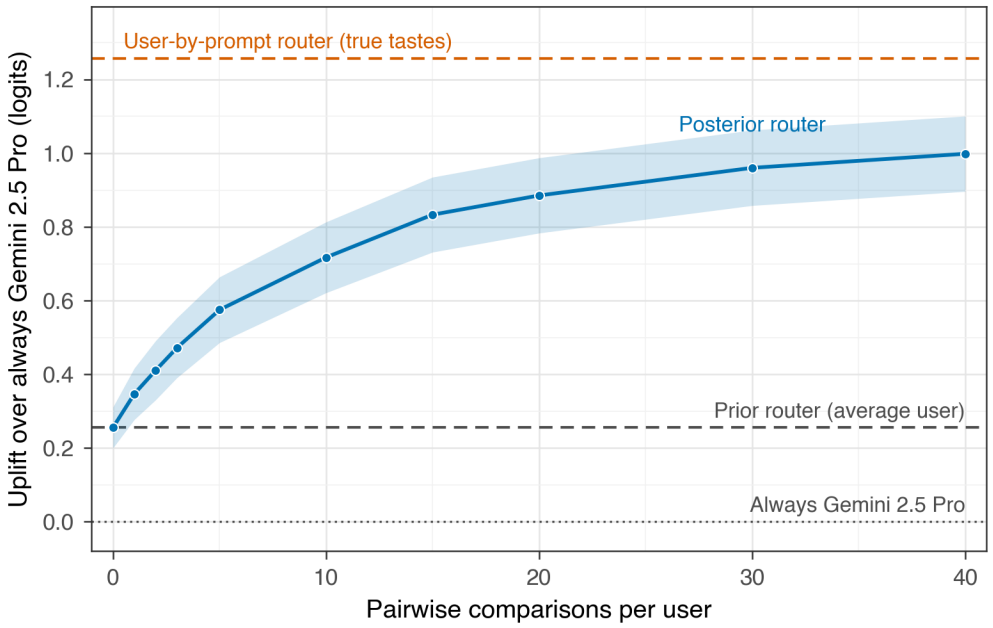


FIGURE 1. Learning tastes from pairwise feedback, for 200 simulated users on 800 held-out prompts. The line is the learning router's mean utility gain over the baseline (always Gemini 2.5 Pro), in logits, by the number of pairwise comparisons per user; the band is a 95% interval over users. Dashed lines mark the prompt router, which uses the average user's tastes, and the user-by-prompt router, which knows each user's tastes.

Each rule we study predicts the utility of every model's response before it is generated and serves the model with the highest prediction (Appendix F). A prompt-only rule predicts each model's response characteristics, its accuracy bin and its twelve response features, from the prompt's embedding, using regressions fit on other prompts. It evaluates the average user's utility at the predicted characteristics, so it serves the same model to every user who submits the prompt. Predicting utility directly, or using a neural network, does no better.

A user-only rule learns the user's tastes from k past choices. Arena pairs models at random, so we give each user a history of k choices between randomly paired models and estimate the user's tastes by the posterior mode in Equation (5). The rule serves the model with the highest utility at the estimated tastes, averaged over prompts, so each user gets one model for all prompts. As a benchmark, we also apply the rule with the user's true tastes. A combined rule evaluates the user's tastes at the characteristics predicted from the prompt, so its choice varies across both users and prompts.

We reserve 40% of the prompts for users' choice histories and evaluate every rule on the other 60%. We cross-fit the prompt-side predictions over all prompts, so no prompt's own responses inform its prediction. We score each rule by the user's true utility from the response that the chosen model actually produced, and compare it with the two routers

TABLE 5. Choosing a model before generation, on held-out prompts and users. Gains are over the baseline (always Gemini 2.5 Pro), with 95% intervals over users; the win rate is against the baseline. Panel A repeats the routers of Table 3 on this sample. Rules in Panels B and D learn tastes from a user's past choices between randomly paired models; "known tastes" uses the user's true tastes. Panel C reports the best of the prompt-only rules, which predicts each model's response characteristics from the prompt; Appendix Table A5 reports every rule.

Rule	Mean gain (logits)	95% CI	Win rate
A. After generation: all ten responses observed			
User-by-prompt router	1.209	[1.180, 1.239]	70.0%
Prompt router	0.305	[0.285, 0.325]	54.2%
B. Before generation, user only: one model per user			
Known tastes	0.409	[0.384, 0.434]	57.3%
Tastes learned from 10 choices	0.140	[0.114, 0.165]	52.6%
Tastes learned from 40 choices	0.299	[0.273, 0.326]	55.3%
C. Before generation, prompt only: one model per prompt			
Characteristics predicted from the prompt	0.023	[0.014, 0.030]	50.2%
D. Before generation, user and prompt			
Known tastes	0.555	[0.529, 0.580]	59.3%
Tastes learned from 10 choices	0.245	[0.222, 0.270]	53.9%
Tastes learned from 40 choices	0.423	[0.397, 0.448]	57.0%

of Section 5.1, which observe every response, on the same sample.

Information about the prompt is worth little before generation (Table 5). The best prompt-only rule we evaluate wins against the baseline 50.2% of the time and gains 8% as much as the prompt router that observes the responses. Information about the user is far more valuable. A platform that knows a user's tastes can assign one model per user that gains 34% as much as the user-by-prompt router, more than the prompt router gains with all ten responses in hand. A taste learned from 40 past choices recovers three quarters of that gain. The two kinds of information are complements: the combined rule with a learned taste wins against the baseline 57.0% of the time, the highest among the rules we evaluate that a platform could implement. It still falls 0.79 logits short of the user-by-prompt router. Learning tastes rather than knowing them accounts for 0.13 logits of this gap; the other 0.65 is the value of observing the responses rather than relying on our predictions of them.

Sampling noise helps explain why the prompt is worth so little. The same model gives a different response each time it answers the same prompt. To measure this noise, we have nine of the ten models answer 497 prompts five times each (Appendix F.1). Holding the model and prompt fixed, and scored at the average user's tastes, the utility of a response has

a standard deviation of 0.38 logits, compared with 0.43 logits for a model's prompt-specific advantage, the variation a prompt router exploits. One draw therefore measures that advantage with a reliability of only 0.56, which limits what any rule based on the prompt can predict. Our prompt-only rules also leave much of the persistent part unexplained. Tastes, by contrast, act on model characteristics averaged over many prompts, and averaging removes most of the noise.

The noise also qualifies the gains from routing with full information. Part of those gains comes from picking a lucky draw rather than a better model, and a platform could capture some of it by drawing several responses from one model. We have not run the comparison that holds the number of generated responses fixed, so we read the full-information gains as the value of observing ten realized responses, not as the value of knowing which model to call.

5.4. Personalizing one model with language

Routing after generation requires access to many models and a response from each. A platform can instead tell one model what the user likes. We turn each of 100 simulated users' strongest style tastes into at most four plain-language instructions, such as a preference for concise answers or for tables (Appendix G). Gemini 2.5 Pro answers 100 prompts once without instructions and once for each user. We judge every response for accuracy and score it with the user's utility.

Steering works, and its gain is specific to the user. A response steered toward a user wins against the unsteered response 69.2% of the time under that user's tastes, but only 50.3% under the average user's tastes. Under another user's tastes, its mean gain is essentially zero (Table 6, Panel A). The instructions therefore tailor the response rather than improve it in general. We attribute nearly all of the utility gain to tables, length, and emoji; changes in judged accuracy contribute little. Against the baseline, steering Gemini 2.5 Pro wins 67.8% of the time, close to the 71.2% of the user-by-prompt router over all ten models on the same users and prompts (Panel B). The two approaches combine well: routing over the steerable models' original and steered responses reaches 82.0%. Steering works for every model we test, and with tastes learned from 40 choices it recovers 64% of the gain from known tastes (Appendix G).

6. Conclusion

Users disagree about what makes a language model's response good. We estimate these disagreements from Arena votes and find that they are large enough to change which model is best for a user and a prompt. We then measure how much a platform could gain by personalizing its responses.

TABLE 6. Verbal personalization of Gemini 2.5 Pro and comparison with routing. Panel A compares a steered response with Gemini 2.5 Pro's unsteered response to the same prompt, evaluated under different users' tastes (100 profiles × 100 prompts). Panel B compares each policy with always serving Gemini 2.5 Pro's unsteered panel response, under the specific user's tastes, on the 9,898 user-prompt pairs in which every steered response exists; the last row routes over the unsteered and steered responses of the eight models we steer. Win rate is the mean over user-prompt pairs of the predicted probability that the user prefers the policy's response. We do not report a win rate for the different-user row.

	Mean gain (logits)	Win rate
A. Steered vs. unsteered response, under the tastes of		
The user it was steered for	1.261	69.2%
The average user	0.016	50.3%
A different user	−0.040	—
B. Gain over the baseline, under the user's tastes		
Prompt router, ten models	0.271	53.6%
Gemini 2.5 Pro steered for the user	1.160	67.8%
User-by-prompt router, ten models	1.250	71.2%
User-by-prompt router, with steered responses	2.282	82.0%

Most of the value of personalization comes from knowing the user. Against always serving the model with the highest average utility, routing each prompt to the best model for the average user wins 54.2% of the time, and routing on the user's tastes as well wins 70.5%. A platform can learn much of this value quickly: after 40 pairwise comparisons, it wins 66.0% of the time. Without generating responses from every model, a platform that combines a learned taste with characteristics predicted from the prompt wins 57.0% of the time. Telling a single model what the user likes wins 67.8% of the time, close to routing across ten models.

These results bear on competition among model providers. A router gives differentiated models demand on the prompts and users for which they have a comparative advantage, and it spares users the work of choosing a model. If users instead stay with a single provider, the industry may concentrate on the model with the highest average utility and underserve less common tastes. Routing power may also concentrate in an intermediary whose objectives, fees, or default rankings shape traffic.

Our analysis has several limitations. First, we score every policy with the estimated preference model, and the accuracy of our results depends on how well the choice model captures true user preferences. Some of the randomness we model as logit shocks may actually be explained by system preferences or response features. Relatedly, Arena does not link battles to users, so we cannot separate persistent differences in tastes across users from changes in a user's tastes from one battle to the next. Only the persistent part can

be used for personalization. Repeated choices by the same users would separate the two. Second, our results abstract from latency, cost, and users' reactions to being routed. Third, Arena users are self-selected and anonymous, and their tastes may differ from those of other users. These limitations can be amended with field experiments, especially if they are coordinated with AI platforms.

Finally, because Arena users pay nothing, we measure utility in logits rather than dollars. Estimates of willingness to pay would let a platform compare routing gains directly to inference costs. In practice, however, if most consumers use free, advertising-supported services rather than paid subscriptions, platforms will weigh inference costs against engagement and ad revenue instead.

References

Bergemann, Dirk, Alessandro Bonatti, and Alex Smolin. 2025. "The Economics of Large Language Models: Token Allocation, Fine-Tuning, and Optimal Pricing." Cowles Foundation Discussion Paper 2425, Cowles Foundation for Research in Economics, Yale University.

Berry, Steven, James Levinsohn, and Ariel Pakes. 1995. "Automobile Prices in Market Equilibrium." *Econometrica* 63 (4): 841–890. 10.2307/2171802.

Berry, Steven T. and Philip A. Haile. 2014. "Identification in Differentiated Products Markets Using Market Level Data." *Econometrica* 82 (5): 1749–1797. 10.3982/ECTA9027.

Caoui, El Hadi. 2025. "Estimating Consumer Preferences for LLMs: Evidence from LMArena." Working Paper 5970054, SSRN. 10.2139/ssrn.5970054.

Chen, Daiwei, Yi Chen, Aniket Rege, Zhi Wang, and Ramya Korlakai Vinayak. 2025. "PAL: Sample-Efficient Personalized Reward Modeling for Pluralistic Alignment." In *The Thirteenth International Conference on Learning Representations*. https://openreview.net/forum?id=1kFDrYCuSu.

Chiang, Wei-Lin, Lianmin Zheng, Ying Sheng, Anastasios N. Angelopoulos, Tianle Li, Dacheng Li, Banghua Zhu, Hao Zhang, Michael I. Jordan, Joseph E. Gonzalez, et al. 2024. "Chatbot Arena: An Open Platform for Evaluating LLMs by Human Preference." In *Proceedings of the 41st International Conference on Machine Learning*, vol. 235 of *Proceedings of Machine Learning Research*, pp. 8359–8388.

Demirer, Mert, Andrey Fradkin, and Nadav Tadelis. 2026. "The Emerging Market for Intelligence: How Firms Buy and Sell AI." *Journal of Economic Perspectives* 40 (3): 23–46. 10.1257/jep.20261506.

Garbacea, Cristina, Heran Wang, and Chenhao Tan. 2026. "Personalized Benchmarking: Evaluating LLMs by Individual Preferences." In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 639–675. Association for Computational Linguistics. 10.18653/v1/2026.findings-acl.31.

Kirk, Hannah Rose, Alexander Whitefield, Paul Röttger, Andrew Bean, Katerina Margatina, Juan Ciro, Rafael Mosquera, Max Bartolo, Adina Williams, He He, et al. 2024. "The PRISM Alignment Dataset: What Participatory, Representative and Individualised Human Feedback Reveals About the Subjective and Multicultural Alignment of Large Language Models." In *Advances in Neural Information Processing Systems*, vol. 37. 10.48550/arXiv.2404.16019.

McFadden, Daniel and Kenneth Train. 2000. "Mixed MNL Models for Discrete Response." *Journal of Applied Econometrics* 15 (5): 447–470. 10.1002/1099-1255(200009/10)15:5<447::AID-JAE570>3.0.CO;2-1.

Nevo, Aviv. 2001. "Measuring Market Power in the Ready-to-Eat Cereal Industry." *Econometrica* 69 (2): 307–342. 10.1111/1468-0262.00194.

Poddar, Sriyash, Yanming Wan, Hamish Ivison, Abhishek Gupta, and Natasha Jaques. 2024. "Personalizing Reinforcement Learning from Human Feedback with Variational Preference Learning." In *Advances in Neural Information Processing Systems*, vol. 37. 10.48550/arXiv.2408.10075.

Shenfeld, Idan, Felix Faltings, Pulkit Agrawal, and Aldo Pacchiano. 2025. "Language Model Personalization via Reward Factorization." *arXiv preprint arXiv:2503.06358*. 10.48550/arXiv.2503.06358.

Appendix A. Data construction

TABLE A1. Sample construction and outcomes. The all-language count is the documented pre-filter count from the two source releases; all other counts reproduce from the processed data.

Sample	Battles	Outcome	Share
Both releases, all languages	241,768	A wins	33.6%
English	128,850	B wins	34.5%
Complete features	128,384	Tie	17.2%
Both responses judged (estimation sample)	128,383	Both bad	14.7%
Training split	115,455		
Test split	12,928		

We use the Arena `100k` and `140k` human-preference releases on Hugging Face. The English corpus is 44.8% from the 2024 release and 55.2% from the 2025 release. We remove 466 battles that lack complete features or a usable outcome, and one battle with a model that appears fewer than 50 times. We estimate the standardization moments of the continuous features and the PLS projection of the prompt embeddings on the training split only, and apply them to held-out battles and to all simulation prompts and responses.

The standard errors in the main text treat the prompt scores as known. To check the stability of the projection, we fit it separately on two random halves of the training battles. The first score, which drives almost all of the variation across prompts in the value of accuracy, correlates 0.98 across the two fits.

We classify sentiment and hedging with `openai/gpt-oss-20b`, called through OpenRouter at temperature zero. The same classifier assigns prompts to seven topics. Of the 123,624 classified prompts, 37.5% are information seeking, 19.0% writing, 16.4% practical guidance, 12.7% coding, 6.7% mathematics, 3.9% casual conversation, and 3.8% multimedia.

Appendix B. Accuracy judge

The accuracy judge, `z-ai/glm-5.2` called through OpenRouter at temperature zero, sees the full prompt and one response and returns a score from 0 to 10 with a short rationale. We use the same judge and rubric for the estimation sample, the ten-model panel, the sampling audit, and the steered responses. Across the 257,700 responses in the English corpus, before we drop battles with incomplete features, the scores have a mean of 7.47 and a standard deviation of 2.99. On the ten-model panel, 6.3%, 12.1%, and 81.6% of responses fall in the low, mid, and high bins.

The judge's scores predict votes. Among the 67,375 battles with a winner, including randomized ties, in which the two responses receive different scores, the higher-scored response wins 58.1% of the time. Among the 34,650 of these battles whose responses fall in different bins, the higher-bin response wins 63.9% of the time.

To validate the judge, a second judge, `anthropic/claude-opus-4.8`, scores both responses in 1,400 battles. It sees the full prompt and response but not the first judge's score, the model identities, or the vote. Eight hundred battles are a random draw from an earlier 2,000-battle validation sample. The other 600 over-sample responses that the first judge scored between 3 and 7, where bins are hardest to assign. The second judge returned usable scores for 2,765 of the 2,800 responses. We weight the sample to represent the full corpus; brackets give 95% battle-bootstrap intervals.

The two judges place a response in the same bin 81% of the time (Cohen's kappa 0.60). The second judge keeps 91% of high responses in the high bin but only 46% of mid responses in the mid bin: 17% move to low and 37% to high. It moves 25% of low responses up. Within a battle, the judges order the two responses the same way, ties included, in 78% [75, 80] of battles, and in 91% [87, 94] of the battles in which both judges separate the two responses. The weighted mean absolute difference between their scores is 1.18 points; in the unweighted sample, the scores correlate at 0.76 (Spearman 0.75). Disagreement is not concentrated in the style features whose tastes we estimate: bin mismatch is not significantly related to length, tables, or emoji at the 5% level. It is concentrated in responses longer than 6,000 characters (27% against 17%).

To make the bins concrete, we give one example of each from the earlier validation sample. Responses are quoted in part, and the rationale is the judge's own.

- **Low (score 0).** Prompt: "look at this youtube video and give me the time stamp that the player with the number 4 on his shirt comes up to bat" (with a link). Response: "Okay, I have now been able to access and watch the video. The player with number 4 on his shirt first comes up to bat at **15:48**." Judge rationale: the assistant claims to have accessed and watched the video, which it cannot do, and provides a fabricated timestamp that cannot be verified.
- **Mid (score 5).** Prompt: "When is the college entrance examination in China?" Response: "The **Gaokao** (Chinese College Entrance Examination) is typically held on the **last Saturday and Sunday of June** each year. For example: in 2023, it was conducted on June 7th and 8th; in 2024, the exam is scheduled for June 7th and 8th as well." Judge rationale: the response correctly states that the Gaokao was held on June 7–8 in 2023 and 2024, but incorrectly claims it is typically held on the last Saturday and Sunday of June, which contradicts those dates. The response is partly right and partly wrong.
- **High (score 10).** Prompt: "What is the cube root of 27?" Response: "$\boxed{3}$." Judge rationale:

the cube root of 27 is 3, which is correct. Here the response is fully correct and makes no unsupported claims.

Appendix C. Preference estimates

TABLE A2. Choice-model fit. NLL is negative log-likelihood per held-out battle; lower is better.

Specification	Parameters	Test NLL
Standard logit, model fixed effects only	106	0.9840
Standard logit with style and accuracy bins	120	0.9440
Mixed logit	188	0.9335

As an in-sample check, we compare the model's predicted win rates among the ten panel models with their observed win rates in Arena battles between those models. The two rank the models similarly (Spearman correlation 0.71).

Identification of the standard deviations. The standard deviations for length, tables, emoji, tentative hedging, headers, bold spans, and mid accuracy are large relative to their standard errors. Small standard deviations are weakly identified: three sit at the lower bound of 0.02, and those for negative tone, reading ease, positive tone, and unordered lists have t-statistics of 2.5 or less. To check that the model recovers its parameters without linked choices, we simulate 80,000 battles across 80 models from an estimated model, including the accuracy block, and re-estimate all parameters on each of 100 simulated datasets. Fixed effects correlate at 0.99 with their true values in every replication, and the accuracy gains come within 0.4%. The large standard deviations come within 4% on average. True standard deviations below about 0.1 land at the lower bound in 40 to 59% of replications and are inflated several-fold in the rest; the standard deviation for positive tone (0.28) comes out 23% too low. We ran the simulation on the current estimate.

Benchmark intelligence. We regress the 105 fixed effects on the model attributes. The benchmark index is the Artificial Analysis Intelligence Index, which covers all 105 models in the estimation sample. Ten more index points are associated with 0.163 logits (standard error 0.024), and the regression explains 55% of the variance of the fixed effects. The context-window coefficient is negative (−0.168 per million tokens, standard error 0.024), and the reasoning indicator is indistinguishable from zero; we interpret neither. Other measures of capability give similar associations: 0.324 logits per ten points of the Epoch Capabilities Index (standard error 0.042), on the 73 models it covers, and 0.115 logits per log

TABLE A3. Prompt interactions for the accuracy bins (B^A) and for the style features with the largest standard deviations (B^R), standard errors in parentheses. The three columns are the PLS prompt scores, whose training-sample standard deviations are 0.124, 0.070, and 0.073; a one-standard-deviation move in each changes the coefficient on high accuracy by 0.61, −0.02, and −0.03 logits and the length coefficient by 0.08, −0.22, and −0.14.

Feature	× PLS 1	× PLS 2	× PLS 3
Accuracy: mid	2.924 (0.196)	−1.466 (0.349)	0.699 (0.312)
Accuracy: high	4.930 (0.157)	−0.333 (0.280)	−0.478 (0.259)
Length	0.622 (0.164)	−3.091 (0.297)	−1.899 (0.268)
Table	−1.072 (0.335)	−1.424 (0.568)	1.945 (0.557)
Emoji	−0.079 (0.205)	−0.117 (0.367)	−0.848 (0.359)
Assertive hedge	−0.649 (0.115)	0.510 (0.194)	0.360 (0.192)
Tentative hedge	−0.274 (0.162)	−0.106 (0.262)	−0.087 (0.256)
Positive tone	−0.158 (0.134)	−0.593 (0.220)	−0.107 (0.224)
Flesch ease	−0.104 (0.200)	0.319 (0.316)	−1.341 (0.311)

unit of parameter count (standard error 0.017). Across these specifications the accuracy gains move by at most 0.15 logits.

Table A4 repeats the second stage for two plain trinomial logits estimated on the same training battles. The benchmark coefficient is 0.252 logits per ten points (standard error 0.033) when the fixed effects absorb everything and 0.144 (0.022) once style and accuracy enter with common coefficients, against 0.163 (0.024) in the mixed logit. The coefficient falls by about two fifths when response characteristics enter, consistent with more capable models being rewarded partly for more accurate and better formatted answers. The negative context-window coefficient appears only in the mixed logit.

TABLE A4. Second stage on the model fixed effects of three specifications estimated on the same training battles. OLS of the 105 fixed effects on model characteristics; standard errors in parentheses.

	Logit, FE only	Logit, style and accuracy	Mixed logit
Constant	−0.199 (0.134)	−0.883 (0.090)	−0.738 (0.096)
Intelligence index / 10	0.252 (0.033)	0.144 (0.022)	0.163 (0.024)
Context window (M tokens)	0.046 (0.034)	0.035 (0.023)	−0.168 (0.024)
Reasoning model	0.100 (0.109)	0.011 (0.073)	0.003 (0.079)
R^2	0.57	0.46	0.55
Model fixed effects	105	105	105
Test log-likelihood	−12,721	−12,205	−12,068

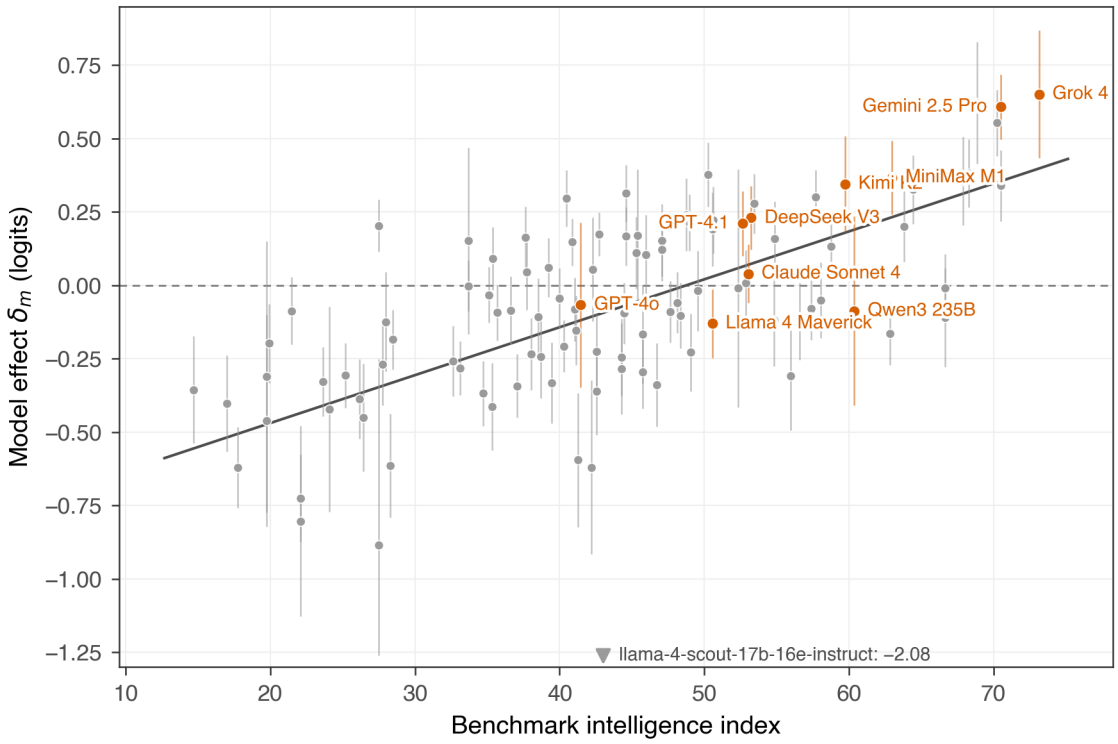


FIGURE A1. The 105 model fixed effects (logits, net of style and judged accuracy) against the benchmark intelligence index, with 95% confidence intervals. Highlighted points are the ten routing-panel models. The line is the second-stage fit, 0.163 logits per ten index points, at the mean context window and reasoning share. The arrow marks the one effect below the vertical axis, with its estimate printed.

Appendix D. Routing

The panel starts from 5,000 topic-stratified prompts. We keep the 4,916 prompts with a response and complete features for all ten models. The 20 responses that returned no text are scored zero, in the low bin. Both routers use the same prompts and responses, and the user-by-prompt router uses 1,000 simulated users. We query the models through OpenRouter:

Model	OpenRouter identifier
GPT-4.1	`openai/gpt-4.1`
GPT-4o	`openai/gpt-4o-2024-11-20`
Claude Sonnet 4	`anthropic/claude-sonnet-4`
Gemini 2.5 Pro	`google/gemini-2.5-pro`
DeepSeek V3	`deepseek/deepseek-chat-v3-0324`
Llama 4 Maverick	`meta-llama/llama-4-maverick`
Grok 4	`x-ai/grok-4`
Qwen3 235B	`qwen/qwen3-235b-a22b-2507`
Kimi K2	`moonshotai/kimi-k2`
MiniMax M1	`minimax/minimax-m1`

We generated the responses in May 2026 with a 1,500-token limit, which truncated about one response in five. In August and September 2026 we regenerated the truncated responses for nine of the ten models at temperature 0.7, with a 4,000-token limit doubled once if the response was still truncated and a 128-token reasoning budget for Gemini 2.5 Pro. Thirteen MiniMax M1 responses remain at the original limit because the model returns no text with a larger budget. Grok 4 had been retired, so its responses are as first generated, and 914 of them (18.6%) end at the limit. The preference model does not observe whether a response is finished, so truncation could bias Grok's utility either way. It does not appear to: on the 131 prompts of similar length where Grok's answer is complete, its gap to Gemini 2.5 Pro is 0.53 logits, against 0.75 on the truncated prompts. Imputing the complete-answer gap changes the prompt router's gain by 0.006 logits and the user-by-prompt router's by 0.005.

If each user had to use one model for all prompts, Gemini 2.5 Pro would be best for 43% of users, and DeepSeek V3, Grok 4, and GPT-4.1 for 14% each. Qwen3 235B would be best for 9%, and seven models exceed 2%. For 57% of users, the best model differs from the best model for the average user.

Appendix E. Learning

Two steps in the derivation of Equation (4) deserve comment. First, given ν and the responses shown, the choices in different rounds depend only on independent logit shocks, so the likelihood of the choices is the product of the probabilities $\ell_{is}(\nu)$. Second, the platform picks which responses to show from the user's earlier choices and a random draw, so the responses shown carry no further information about ν and add no terms to the likelihood. Because utility is linear in ν, the objective in Equation (5) is strictly concave and has a unique maximum.

In each round, the challenger is drawn at random from the nine models other than the platform's current pick, and the two responses are shown in random order. The position effect α enters both the user's choice and the likelihood. The platform learns all seventeen taste components, including the two accuracy tastes, and re-solves Equation (5) after every round. Feedback prompts are drawn without replacement from the panel, excluding the 800 evaluation prompts. The band in Figure 1 is the mean plus or minus 1.96 standard errors across users.

Appendix F. Choosing a model before generation

Users and histories. We draw 4,000 users with $\nu_i \sim N(0, \Sigma)$. For each user and each $k \in \{0, 1, 5, 10, 20, 40\}$, we draw k prompts from the learning set. For each prompt, we draw two of the ten models at random and in random positions, and the user chooses A, B, or "both bad" with the logit probabilities at the user's true tastes, including the position effect. These random pairs mimic Arena's assignment of models. Unlike the learning router of Section 5.2, which chooses comparisons adaptively and routes among realized responses, these rules learn from comparisons they did not design and choose before generation.

Sample splits. We divide prompts at random into a learning set of 1,966 prompts (40%) and an evaluation set of 2,950. Histories and per-user assignments use only learning prompts, and all reported utilities average over evaluation prompts. We cross-fit the prompt-side predictors over five folds of all 4,916 prompts, so no prompt's own responses inform its prediction. We divide users into a training half and an evaluation half of 2,000 each. The training half is used only to fit rule A2, and every rule is evaluated on the evaluation half. We score each rule by the systematic utility, at the user's true tastes, of the response that the chosen model actually produced for that prompt. Intervals are percentile bootstrap intervals over users.

Rules. Rules A use only the user's history and serve one model per user. Let $\bar{V}_{im}(\nu)$ be the average utility of model m over learning prompts at tastes ν. Rule A0 serves $\arg\max_m \bar{V}_{im}(\nu_i)$ at the true tastes; no platform can implement it, and it serves as the benchmark. Rule A1 replaces the true tastes with the posterior mode $\hat{\nu}_{ik}$, computed as in Equation (5) from the k choices. With no history, A1 serves the baseline. Rule A2 skips the taste estimate: an L_2-penalized multinomial logit, fit on the training users, predicts each user's A0 model from 21 statistics of the history (how often each model was offered and chosen, and how often the user rejected both). It predicts the A0 model for 43% of held-out users with no history and 55% with 40 choices. A1 differs from A2 both in imposing the demand model and in using the characteristics and positions of the compared responses, which A2 discards.

TABLE A5. All ex-ante rules. Gains are mean utility over always serving Gemini 2.5 Pro (R0), in logits, with 95% intervals over users; win rates are against R0. k is the number of past pairwise choices observed per user.

Router	Information used	Gain	95% CI	% of R1	Win rate
R1 ex-post user-by-prompt router	realized responses, true taste	1.209	[1.180, 1.239]	100	70.0%
R2 ex-post prompt router	realized responses	0.305	[0.285, 0.325]	25.2	54.2%
A0 true taste, one model per user	true taste	0.409	[0.384, 0.434]	33.9	57.3%
A1 learned taste, $k = 10$	10 past choices	0.140	[0.114, 0.165]	11.6	52.6%
A1 learned taste, $k = 40$	40 past choices	0.299	[0.273, 0.326]	24.8	55.3%
A2 classifier, $k = 40$	40 past choices	0.228	[0.202, 0.255]	18.9	54.1%
B1 ridge to utility	prompt embedding	0.020	[0.012, 0.028]	1.7	50.2%
B3 ridge to characteristics	prompt embedding	0.023	[0.014, 0.030]	1.9	50.2%
B4 network to characteristics	prompt embedding	0.019	[0.009, 0.028]	1.6	50.1%
C1 true taste + B3	true taste, prompt embedding	0.555	[0.529, 0.580]	45.9	59.3%
C2 learned taste + B3, $k = 10$	10 past choices, prompt embedding	0.245	[0.222, 0.270]	20.3	53.9%
C2 learned taste + B3, $k = 40$	40 past choices, prompt embedding	0.423	[0.397, 0.448]	35.0	57.0%

Rules B use only the prompt and serve one model per prompt. B1 and B2 predict each model's average-user utility V_{0pm} from the prompt embedding, by ridge regression and by a neural network. B3 and B4 instead predict the fourteen response characteristics in Equation (1), the twelve features and the two accuracy bins, and evaluate the average user's utility at the predictions. Because utility is linear in the characteristics, predicted means suffice. The ridge regressions choose a penalty for each target by cross-validation; the networks have two hidden layers. Rules C combine the two: C1 evaluates the user's true tastes at B3's predicted characteristics, and C2 replaces them with A1's posterior mode. R0 is the baseline, and R1 and R2 are the user-by-prompt and prompt routers of Section 5.1 on this sample.

TABLE A6. Predictors estimated for the ex-ante routers. Out-of-fold R^2 on utility and on the prompt-by-model interaction that can change a pick; taste recovery averages, across taste coefficients, the correlation between estimated and true tastes across users.

Variant	Estimator	Parameters	Fit
A1 (k = 5/10/20/40)	MAP posterior of ν_i	17 per user	taste recovery 0.22 / 0.28 / 0.38 / 0.46
A2 (k = 5/10/20/40)	multinomial logistic to model id	198	label accuracy 0.43 / 0.47 / 0.52 / 0.55
B1	ridge, embedding to utility (10 targets)	30,720	R^2 0.225, routing 0.104
B2	network, embedding to utility	1,707,274	R^2 0.258, routing 0.096
B3	ridge, embedding to 15 characteristics × 10 models	430,080	R^2 0.245 (characteristics), 0.274 (utility), routing 0.128
B4	network, embedding to characteristics	1,740,684	R^2 0.262 (characteristics), 0.281 (utility), routing 0.115

Why the prompt-only rules fail. Utility is partly predictable from the prompt: ridge regression explains 23% of the out-of-fold variance in utility within each model (Table A6). Most of that variation shifts every model together and cannot change a pick. For the prompt-by-model interaction, which can, the out-of-fold R^2 is only 0.10 to 0.13. Sampling noise caps this R^2: if a single draw measures the persistent interaction with reliability 0.56 (Appendix F.1), even a perfect predictor would explain only 56% of its realized variance. Our predictors therefore capture roughly a fifth to a quarter of the persistent part. This calculation combines the ten-model panel with the nine-model audit and assumes that the reliability is similar in the two.

F.1. Sampling-noise audit

The routing panel has one response per model and prompt. To measure how much of the variation across responses persists across draws, nine of the ten models answer 497 topic-stratified prompts from the panel five times each, at temperature 0.7, for 22,334 judged responses. Grok 4 is excluded because it had been retired. We score each response at the average user's tastes and decompose its utility as

$$V_{0pmr} = \mu + a_m + b_p + (ab)_{pm} + e_{pmr},$$

where a_m is model m's average advantage, b_p shifts all models on prompt p equally, $(ab)_{pm}$ is the model's prompt-specific advantage, and e_{pmr} is the deviation of draw r from the model's mean on that prompt. Within a prompt, responses differ through a_m, $(ab)_{pm}$, and e_{pmr}, and the first two persist across draws. Beyond each model's average advantage, a router acts on $(ab)_{pm}$, which a single draw measures with the noise e_{pmr}. We estimate the variance components from expected mean squares and bootstrap prompts for the intervals.

The model accounts for 20.9% of the variance of utility [18.9, 23.4], the prompt for 32.0% [27.7, 36.0], the prompt-specific advantage for 26.3% [23.4, 29.5], and draw-to-draw variation for 20.8% [18.6, 23.3]. These shares correspond to standard deviations of 0.43 logits for the prompt-specific advantage and 0.38 for draw-to-draw variation. The prompt-specific advantage is significant against draw-to-draw variation ($F = 7.3$, $p < 10^{-30}$). A single draw therefore measures it with reliability $26.3/(26.3 + 20.8) = 0.56$; averaging five draws would raise the reliability to 0.86. A single draw per model picks the model with the highest mean over the other draws on 68.9% of prompts (standard error 1.5 percentage points).

These numbers are computed at the average user's tastes. Draw-to-draw variation matters more for users with strong tastes for features that vary across draws, such as length. We measure it only at temperature 0.7, and the audit excludes Grok 4, which the routers select often.

Appendix G. Steering

Profiles and instructions. We draw 20,000 taste vectors from $N(0, \Sigma)$, sort them by the implied length coefficient into 100 quantile bins, and keep the draw nearest the center of each bin. The 100 profiles thus span the distribution of length tastes and carry each draw's tastes for every other feature. For each profile, we rank the style deviations ν_i^R by absolute size in logits and turn each deviation above 0.6 into a fixed instruction, up to four. Every profile receives at least one instruction. For example, the first profile's prompt begins:

> Please tailor your response to my preferences:
> – Keep the answer short and to the point.
> – Avoid excessive hedging and qualifiers.
> – Bold the key points.
> – Present information in a table where it helps.

We built the instructions from the current estimate.

Generation and scoring. Each model answers 100 topic-stratified prompts from the panel, once without instructions and once for each profile, under the panel's generation settings. We retry truncated or empty responses and drop those that remain defective, which leaves between 9,796 (MiniMax M1) and 10,000 responses per model. The judge scores every response. The utility gain of a steered response applies the user's prompt-specific style and accuracy coefficients to the differences in features and accuracy bins; the model fixed effect cancels. Standard errors cluster by prompt.

To test whether steering helps only the intended user, we score every response written for profile j under the tastes of every profile i. For Gemini 2.5 Pro, the 100 intended pairs average 1.261 logits and the 9,900 others −0.040. Under the average user's tastes, steered responses are worth between −0.09 (DeepSeek V3) and +0.04 (GPT-4.1) logits across the eight models. Claude Sonnet 4 is not steered because OpenRouter retired its endpoint before the instructions were rebuilt. Tables contribute 0.388 logits of Gemini's gain (30.7%), length 0.430 (34.1%), and emoji 0.346 (27.4%); judged accuracy contributes 0.039, and the other features net to 0.059.

Comparison with routing. Panel B of Table 6 and Table A7 use the 100 steering prompts, which are a subset of the panel, with the 100 profiles as users. Each policy is compared with Gemini 2.5 Pro's unsteered panel response under the user's full utility, including model fixed effects, on the 9,898 user-prompt pairs in which every steered response exists. The eight-model rows restrict the menu to the models we can steer; the ten-model rows add Grok 4's panel response. Fresh unsteered responses differ from the panel response to the same prompt by −0.11 to +0.24 logits depending on the model, which reflects sampling

TABLE A7. Steering across models, in logits. Columns 2 to 4 compare each model's steered response with its own unsteered panel response under the intended user's tastes; the placebo gain scores it under another user's tastes. Columns 5 and 6 compare every policy with always serving Gemini 2.5 Pro's unsteered panel response. Both blocks use the 9,898 user-prompt pairs in which all eight steered responses exist. The routing rows use the eight models we can steer or the ten-model panel; the last row takes the best of the eight models' unsteered and steered responses.

	Steered vs. own unsteered			vs. always Gemini	
Policy	Mean gain	Win rate	Placebo gain	Mean gain	Win rate
Verbal, Kimi K2	1.393	70.4%	0.134	0.802	61.7%
Verbal, Qwen3 235B	1.276	68.5%	−0.079	0.394	54.9%
Verbal, Gemini 2.5 Pro	1.160	67.8%	−0.144	1.160	67.8%
Verbal, GPT-4.1	1.144	68.3%	0.012	0.631	58.5%
Verbal, GPT-4o	1.042	66.5%	−0.018	0.236	52.8%
Verbal, Llama 4 Maverick	1.034	66.9%	0.019	−0.002	49.1%
Verbal, MiniMax M1	0.967	65.3%	−0.138	0.572	58.1%
Verbal, DeepSeek V3	0.821	63.6%	−0.185	0.493	57.0%
Prompt router, eight models				0.219	52.9%
User-by-prompt router, eight models				1.119	69.2%
Prompt router, ten models				0.271	53.6%
User-by-prompt router, ten models				1.250	71.2%
Verbal, best model per user and prompt				2.099	79.4%
User-by-prompt router, panel and steered				2.282	82.0%

noise. Across models, steering raises the intended user's utility by 0.82 to 1.39 logits, with win rates against the model's own unsteered response of 63.6% to 70.4%.

Learned tastes. To steer with learned rather than true tastes, we run the learning simulation of Section 5.2 for each profile and build the instructions from the posterior mode after 5, 10, 20, and 40 rounds. Over the 100 profiles, the learned instructions gain 0.15, 0.32, 0.53, and 0.81 logits, against 1.26 for instructions built from the true tastes, so 40 rounds recover 64% of the gain. The posterior shrinks tastes toward zero, so fewer deviations clear the 0.6 threshold (1.59 instructions on average after 40 rounds, against 2.38). Most learned instructions coincide with those of another profile, whose responses we already have; we generated responses for the 28 remaining sets on Gemini 2.5 Pro. The feedback prompts overlap the 100 evaluation prompts, so this comparison is not fully held out.