

---

# Chatbot Arena: An Open Platform for Evaluating LLMs by Human Preference

---

Wei-Lin Chiang<sup>\*1</sup> Lianmin Zheng<sup>\*1</sup> Ying Sheng<sup>2</sup> Anastasios N. Angelopoulos<sup>1</sup> Tianle Li<sup>1</sup> Dacheng Li<sup>1</sup>  
Banghua Zhu<sup>1</sup> Hao Zhang<sup>3</sup> Michael I. Jordan<sup>1</sup> Joseph E. Gonzalez<sup>1</sup> Ion Stoica<sup>1</sup>

## Abstract

Large Language Models (LLMs) have unlocked new capabilities and applications; however, evaluating the alignment with human preferences still poses significant challenges. To address this issue, we introduce Chatbot Arena, an open platform for evaluating LLMs based on human preferences. Our methodology employs a pairwise comparison approach and leverages input from a diverse user base through crowdsourcing. The platform has been operational for several months, amassing over 240K votes. This paper describes the platform, analyzes the data we have collected so far, and explains the tried-and-true statistical methods we are using for efficient and accurate evaluation and ranking of models. We confirm that the crowdsourced questions are sufficiently diverse and discriminating and that the crowdsourced human votes are in good agreement with those of expert raters. These analyses collectively establish a robust foundation for the credibility of Chatbot Arena. Because of its unique value and openness, Chatbot Arena has emerged as one of the most referenced LLM leaderboards, widely cited by leading LLM developers and companies. The platform is publicly available at <https://chat.lmsys.org>.

## 1. Introduction

Recent advancements in large language models (LLMs) have significantly expanded their capabilities beyond traditional natural language processing boundaries, addressing a broad array of general tasks (OpenAI, 2023; Gemini et al., 2023; Touvron et al., 2023). These developments underscore the potential of LLMs but also have raised concerns with respect to performance evaluation. Current benchmarks often

<sup>\*</sup>Equal contribution <sup>1</sup>UC Berkeley <sup>2</sup>Stanford <sup>3</sup>UCSD. Correspondence to: Wei-Lin Chiang <weichiang@berkeley.edu>.

		Question Source	
		Static	Live
Evaluation Metric	Ground Truth	MMLU, HellaSwag, GSM-8K	Codeforces Weekly Contests
	Human Preference	MT-Bench, AlpacaEval	<b>Chatbot Arena</b>

Figure 1. Classification of LLM benchmarks: We categorize along two dimensions: whether the questions are from a static dataset or a live, fresh source, and whether the evaluation metric relies on ground truth or (approximated) human preferences. MMLU (Hendrycks et al., 2020), HellaSwag (Zellers et al., 2019), GSM-8K (Cobbe et al., 2021), MT-Bench (Zheng et al., 2023b), and AlpacaEval (Li et al., 2023) are common examples of static benchmarks. Chatbot Arena is the one introduced in this paper.

fail to capture the nuanced and diverse aspects of these models, particularly in assessing their alignment with human preferences in real-world, open-ended tasks.

To assess the performance of LLMs, the research community has introduced a variety of benchmarks. These benchmarks can be categorized based on two factors: the source of questions (either static or live) and the evaluation metric (either ground truth or human preference). According to these factors, benchmarks can be classified into four categories, as shown in Figure 1. While a range of benchmarks is beneficial, the most prevalent current method for evaluating LLMs remains a static, ground-truth-based evaluation, partly because such evaluations are inexpensive and reproducible.

However, these static, ground-truth-based benchmarks exhibit several limitations. Firstly, the questions within these benchmarks are not open-ended, hindering the ability to capture the flexible and interactive use found in real-world settings (Zheng et al., 2023b). Secondly, the test sets in these benchmarks are static, meaning they can become contaminated over time, which undermines the reliability of the evaluation results (Yang et al., 2023). Furthermore, for many complex tasks, establishing a definitive ground truth is not only challenging but sometimes unattainable. Consequently, current benchmarks fail to adequately address the needs of state-of-the-art LLMs, particularly in evaluating user preferences. Thus, there is an urgent necessity for an open, live evaluation platform based on human preference that can more accurately mirror real-world usage.

Creating such a benchmark platform entails significant challenges. It requires the collection of live, fresh, and diverse

user questions to accurately represent real-world scenarios. Additionally, developing scalable, incremental, and efficient ranking systems is essential for evaluating a large number of models. Moreover, ensuring the quality of human evaluations is crucial given the noisy nature of human preferences.

To this end, we introduce Chatbot Arena, a benchmarking platform for LLMs that features anonymous, randomized battles in a crowdsourced setting. Chatbot Arena is a free website open to all users.<sup>1</sup> On this website, a user can ask a question and get answers from two anonymous LLMs. Afterward, the user casts a vote for the model that delivers the preferred response, with the models' identities revealed only after voting. This crowdsourced method effectively gathers a diverse array of fresh user prompts, accurately reflecting real-world LLM applications. Armed with this data, we employ a suite of powerful statistical techniques, ranging from the statistical model of Bradley & Terry (1952) to the E-values of Vovk & Wang (2021), to estimate the ranking over models as reliably and sample-efficiently as possible. With these tools in hand, we have designed efficient sampling algorithms specifically to select model pairs in a way that accelerates the convergence of rankings while retaining statistical validity.

We conduct a thorough analysis of the collected data to ensure the credibility of our platform. We demonstrate that the user-generated questions are sufficiently diverse to encompass a wide range of LLM use cases and are sufficiently challenging to differentiate between models. Furthermore, we confirm that the crowd-sourced votes are highly consistent with expert evaluations.

We have been running our system since Apr 2023 and have received over 240K votes from about 90K users in over 100 different languages as of Jan 2024. To encourage user engagement, we have made over 50 state-of-the-art models available for free. We also collaborate with leading model developers such as OpenAI, Google, Anthropic, Mistral, Hugging Face, and various universities, incorporating their latest models into our platform. We keep the community engaged by routinely updating the leaderboard, publishing analytical blogs, releasing datasets, and sharing information via tweets. Because of its unique and significant value, our leaderboard has emerged as one of the most referenced in the LLM field and has become a benchmark for the industry. We commit to making our data and code available, ensuring that this platform is open-source and open-accessible.

We make the following contributions:

- We build the first large-scale crowd-sourced live LLM evaluation platform with over 1M users visit.<sup>2</sup>

---

<sup>1</sup><https://chat.lmsys.org>

<sup>2</sup>The number was estimated by Google Analytics as of March

- We conduct an in-depth analysis of the collected data, including prompt diversity, quality, vote quality, and insights on human feedback.
- We will publicly release a human preference dataset with over 100K pairwise votes collected from Chatbot Arena.
- We design an efficient sampling algorithm that actively chooses which model pairs to show, such that our sample efficiency improves, sometimes to a large degree.

## 2. Related Work

**LLM Benchmarks.** We briefly review the common LLM benchmarks, following the classification presented in Figure 1. The most prevalent benchmarks are static, ground-truth-based ones, typically in the form of multiple-choice questions or question-answering tasks with predefined answers and test cases. These benchmarks encompass a range of topics including language understanding, mathematics, coding, and logical reasoning. Prominent examples in this category are MMLU (Hendrycks et al., 2020), HellaSwag (Zellers et al., 2019), GSM-8K (Cobbe et al., 2021), BigBench (Srivastava et al., 2023), AGIEval (Zhong et al., 2023), and HumanEval (Chen et al., 2021). Benchmarks focusing on safety, such as ToxicChat (Lin et al., 2023), and comprehensive suites like HELM (Liang et al., 2022), also exist. In addition to closed-ended questions, benchmarks can include open-ended questions that are evaluated by human judgment, which can be rated by experts or crowd workers such as Amazon Mechanical Turk (Karpinska et al., 2021; Geng et al., 2023; Wang et al., 2023). The recent trend includes utilizing GPT-4 for approximating human judgment (Chiang & Lee, 2023), with notable instances being MT-Bench (Zheng et al., 2023b) and AlpacaEval (Li et al., 2023). In addition to static benchmarks, live benchmarks that include fresh questions are also available. These questions can be obtained from annual exams or weekly online contests such as Codeforces (Li et al., 2022; Huang et al., 2023). They can also be sourced from human interaction. Some studies have explored using live human interaction for reinforcement learning from human preference (Bai et al., 2022; Ouyang et al., 2022; Touvron et al., 2023). However, these studies are typically limited to specific organizations. In this paper, we introduce Chatbot Arena, the first open, large-scale, and crowdsourced benchmark platform that utilizes live human interaction.

**Risks of Static Benchmarks.** Static benchmarks have certain issues, including contamination, saturation, overfitting, and a lack of human alignment (Yang et al., 2023; Oren et al., 2023). DynaBench (Kiela et al., 2021) identifies these challenges and recommends the use of a live benchmark

2024. Note that user visit may not convert to votes as our website also offers "direct chat" mode.

that incorporates a human-in-the-loop approach for classical NLP benchmarks. Our system adopts a similar spirit. However, our focus is on chatting with LLMs, and we implement this on a significantly larger user scale.

**Ranking System.** Ranking systems have been a well-studied topic in statistics. Related topics include probability models (Hunter, 2004; Rao & Kupper, 1967), rank elicitation (Szörényi et al., 2015; Busa-Fekete et al., 2014a;b), and online experiment design (Chernoff, 1992; Karimi et al., 2021). The Elo rating system has also been used for LLMs (Bai et al., 2022; Boubdir et al., 2023). Contributing to this literature, we introduce techniques for accelerating ranking convergence and detecting abnormalities, specifically applied to large-scale, real-world settings of LLMs.

**Human Preference Dataset.** Owing to the significance of human preferences, several datasets and analyses exist that incorporate human preferences. These include OpenAssistant (Köpf et al., 2023), HH-RLHF (Bai et al., 2022), LMSYS-Chat-1M (Zheng et al., 2023a), and synthetic approximations of human preferences like UltraFeedback (Cui et al., 2023) and Nectar (Zhu et al., 2023). Our prior data release, LMSYS-Chat-1M (Zheng et al., 2023a), is similarly collected via crowdsourcing. However, LMSYS-Chat-1M comprises solely conversations and lacks human preference data, rendering it unsuitable for direct use in ranking studies. This paper focuses on the analysis of preference data for ranking purposes.

### 3. Human Preference Data Collection

In this section, we discuss our interface design to collect human preferences and present summary statistics.

#### 3.1. Interface

Chatbot Arena crowd-sources feedback from users for model evaluation. Our goal is to design an ease-of-use interface to reduce friction for users to contribute data. Since we collect feedback from many users, it is difficult to set a consistent grading rubric across different people. Hence, we adopt a pairwise comparison mechanism where users only need to compare two model responses and vote for the better one, instead of requiring users to provide an absolute score.

In each battle, two anonymous models are sampled. To encourage data diversity, we do not preset any input prompt on the website. Users are free to input any prompt to the two models. We believe this creates incentives for user engagement, particularly given that we offer a free service. It also helps us collect a diverse set of inputs representing real-world usage. After models provide their answers, user compare them side-by-side and vote for the preferred answer. If a user cannot choose in the first turn, the user can

continue chatting until identifying a winner. For those who are unsure, we also present two buttons, “tie” or “both are bad.” Figure 8 shows a screenshot of our interface. Before using our service, users are required to accept terms of use, which gives us their consent to release the data publicly.

#### 3.2. Data Statistics

We began collecting data in April 2023. As of Jan 2024, we have received around 240K votes from over 90K users. Our data involves more than 50 models, including both proprietary models like GPT-4, Claude, and Gemini, as well as open models such as LLaMA and Mistral. These conversations cover more than 100 languages, with 77% being in English, 5% in Chinese, and the remaining languages, such as Russian, German, Spanish, French, and Japanese, each representing less than 2% of the total. Each data point includes multi-turn conversations between the user and two LLMs, and a vote to indicate which model the user prefers. We summarize statistics in Table 1 along with other existing human preference datasets.

Figure 10 in the Appendix shows the vote count per model. On average, 8K votes are collected for each model. In Figure 2, we select a set of representative models and present their win rate and the number of battles. Note that we employ non-uniform sampling to concentrate votes on model pairs that have similar performance due to higher uncertainty. This helps us reduce the number of votes required to reach stable results. We later develop an adaptive sampling method and demonstrate its effectiveness against random sampling. See Section 5 for further analysis.

To ensure anonymity, we use keywords to filter out conversations containing model identity such as model name (e.g., GPT, Claude) or companies (e.g., OpenAI, Anthropic). To avoid misuse, we adopt OpenAI moderation API to flag conversations that contain unsafe content. The flagged user requests account for 3% of the total requests. Figure 9 in the Appendix shows the number of valid user votes over time, where we get 1-2K votes per day in recent months and spikes as we introduce new models or leaderboard updates.

### 4. From Pairwise Comparisons to Rankings

Our data consists of pairwise comparisons—but how can we use these comparisons to recover a ranking over all  $M$  models? This is a well-studied topic in the literature on learning to rank (Liu et al., 2009), and we present our perspective here. We let  $\mathcal{A} = \{(m, m') : m < m' \text{ and } m, m' \in [M]\}$  denote our comparative data set.

We consider a sequential setting, where at time  $t \in \mathbb{N}$ , we serve the human a pair of models  $A_t \in \mathcal{A}$  (which we pick), and in turn we observe the human’s response  $H_t \in [0, 1]$ . As an example, we might have that  $A_t = (1, 2)$  and  $H_t = 1$ ,

Table 1. Statistics of human preference datasets, including Anthropic HH (Bai et al., 2022), OpenAssistant Conversations (Köpf et al., 2023), and Chatbot Arena (as of 2024/1/21). The tokens are counted by Llama2’s tokenizer. “Conv” = Conversation. “Lang” = Language.

Dataset	# Convs	# Models	# Users	# Langs	Avg. # Turns per Sample	Avg. # Tokens per Prompt	Avg. # Tokens per Response
Anthropic HH	338,704	-	143	1	2.3	18.9	78.9
OpenAssistant	66,497	-	13,500	35	-	36.9	214.2
Chatbot Arena (20240121)	243,329	50	90,051	149	1.3	94.9	269.0

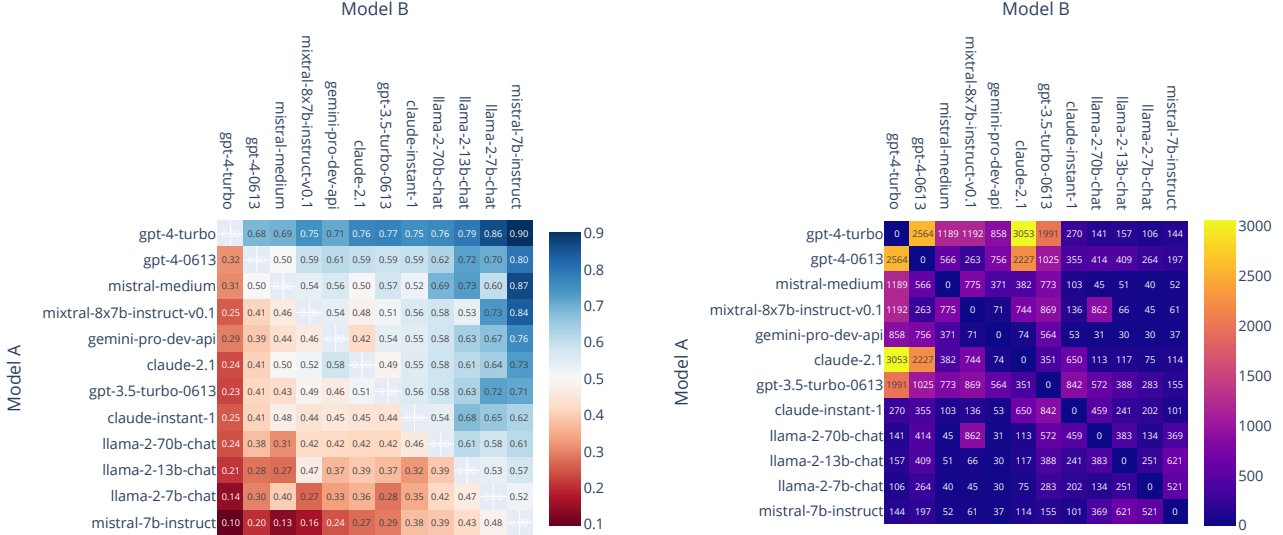


Figure 2. Win-rate (left) and battle count (right) between a subset of models in Chatbot Arena.

indicating that the human prefers model 2 over model 1. In the ensuing text, we will primarily focus on the binary case—where  $H_t \in \{0, 1\}$ —but our approach will generalize to any form of feedback, including the possibility of allowing the human to express different degrees of preference or to say the models are tied.

One critical goal is to estimate the *win matrix*:  $\theta^*(a) = \mathbb{E}[H_t \mid A_t = a]$ , for all  $a \in \mathcal{A}$ ; see the left panel of Figure 2 for an illustration of the (empirical) win matrix. In the binary case, the  $a$  entry in the win matrix corresponds to the probability the human prefers model  $a_2$  to  $a_1$  when shown the pair  $a$ . Finding the win matrix is a relatively straightforward mean-estimation problem; we will provide details in Section 5.

Formally, consider a *score*  $s(\mathbb{P}) \in \mathbb{R}^M$ , where  $\mathbb{P}$  is a joint distribution over  $A$  and  $H$  (by default, we will target a uniform distribution over  $\mathcal{A}$ ). Each model has a true score  $s(\mathbb{P})_m$ , and better models will have higher scores. In particular, we have the rank of model  $m$ :

$$\text{rank}(\mathbb{P})_m = 1 + \sum_{m' \in [M]} \mathbb{1} \{s(\mathbb{P})_{m'} > s(\mathbb{P})_m\}. \quad (1)$$

The best model has rank 1. If there is another model tied for best, they will both get assigned rank 1.

**Picking a score.** A standard score function in this setting

is the vector of Bradley-Terry (BT) coefficients (Bradley & Terry, 1952). In the Bradley-Terry model,  $H_t \in \{0, 1\}$ , and the probability model  $m$  beats model  $m'$  is modeled via a logistic relationship:

$$\mathbb{P}(H_t = 1) = \frac{1}{1 + e^{\xi_{m'} - \xi_m}}, \quad (2)$$

where  $\xi$  is an  $M$ -length vector of so-called BT coefficients. Without loss of generality, we take  $\xi_1 = 0$  (since the model is invariant to addition in  $\xi$ ). Our goal is to estimate the population Bradley-Terry coefficients, i.e., those that minimize the binary cross-entropy:

$$s(\mathbb{P}) = \underset{\xi}{\text{argmin}} \mathbb{E}_{(A,H) \sim \mathbb{P}} \left[ \ell \left( H, \frac{1}{1 + e^{\xi_{A_2} - \xi_{A_1}}} \right) \right], \quad (3)$$

where  $\ell$  is the binary cross-entropy loss,  $\ell(h, p) = -(h \log(p) + (1 - h) \log(1 - p))$ .

Although the BT model technically assumes a parametric form for the model win rates, the seminal results of Huber et al. (1967); White (1982) show that maximum likelihood estimators are still asymptotically normal even when these assumptions *do not* hold, so long as the so-called “sandwich” covariance matrix is used; see Section 5 for details, and see Appendix B for a nonparametric extension of the Bradley-Terry model. Finally, we remark that previous evolutions of

our online interface have reported different ranking scores, such as the Elo score (Elo, 1967) instead of the BT coefficients. We made this change because the BT coefficients are better for the purpose of statistical estimation.

## 5. Efficient Approximate Ranking

In Section 4 we described how to calculate the win matrix, score, and rank. Now we describe our estimation procedures.

**Win matrix estimation.** Estimation of the win matrix is relatively straightforward. Define  $X_t(a) = \frac{1}{P_t(a)} H_t \mathbb{1}\{A_t = a\}$ , where  $P_t(a)$  is the probability of sampling pair  $a$  at time  $t$ , and  $X_t$  as the according vector. Then the estimator is

$$\hat{\theta}_T = \frac{1}{T} \sum_{t=1}^T X_t. \quad (4)$$

Note that  $\mathbb{E}[X_t(a)] = \theta^*(a)$  for all  $t$ , and thus  $\hat{\theta}_T$  is an unbiased estimator of  $\theta^*$ . We will furthermore estimate the covariance matrix as

$$\hat{\Sigma}_T = \frac{1}{T} \sum_{t=1}^T (X_t - \hat{\theta}_T)(X_t - \hat{\theta}_T)^\top. \quad (5)$$

Under the appropriate regularity conditions, we have that

$$\sqrt{T} \hat{\Sigma}^{-1/2} (\hat{\theta} - \theta^*) \rightarrow \mathcal{N}(0, I_d), \quad (6)$$

and we construct confidence intervals accordingly. For an understanding of the appropriate regularity conditions, see Durrett (2019), Theorem 8.2.8, where condition (ii) is trivially satisfied so long as  $P_t(a) > \epsilon > 0$ , and condition (i) is implied by the almost-sure convergence of  $P_t(a)$  to a limiting distribution  $P(a)$ .

**Estimating the BT scores.** To estimate the BT coefficients, mirroring (3), we perform (reweighted) maximum likelihood estimation on our data points:

$$s(\hat{\mathbb{P}}) = \underset{\xi}{\operatorname{argmin}} \sum_{t=1}^T \frac{1}{P(A_t)} \ell \left( H_t, \frac{1}{1 + e^{\xi_{A_t,2} - \xi_{A_t,1}}} \right), \quad (7)$$

where  $A_t \sim P$ . We perform the inverse weighting by  $P(A_t)$  because this allows us to target a score with a uniform distribution over  $A$ .

To compute confidence intervals on the BT coefficients, we employ two strategies: (1) the pivot bootstrap (DiCiccio & Efron, 1996), and (2) the “sandwich” robust standard errors outlined in Huber et al. (1967) (see also Freedman (2006) for an outline of the necessary technical assumptions). Ultimately, based on the results of a simulation study described in Appendix A, we choose to deploy the sandwich intervals due to their smaller size in large samples.

**Approximate rankings.** Finally, we report an approximate ranking for each model that accounts for the uncertainty in the estimation of the score. Given an  $M$ -dimensional confidence set  $\mathcal{C}$  satisfying

$$\mathbb{P}(s(\mathbb{P}) \in \mathcal{C}) \geq 1 - \alpha, \quad (8)$$

we extract an approximate ranking  $R_m = 1 + \sum_{m' \in [M]} \mathbb{1}\{\inf \mathcal{C}_{m'} > \sup \mathcal{C}_m\}$ . The uniform validity of  $\mathcal{C}$  directly implies that  $\mathbb{P}(\exists m : R_m > \operatorname{rank}(\mathbb{P})_m) \leq \alpha$ —i.e., with high probability, no model’s performance is understated. A guarantee on the other side—that no model’s performance is overstated—is possible by interchanging the inf and sup. To get the uniform confidence set, we construct the chi-squared interval implied by the central limit theorem using the sandwich estimate of the variance. In other words, we construct the interval  $\{\xi : T \|\hat{V}^{-1/2}(\hat{\xi} - \xi)\| \leq \chi_{1-\alpha, M-1}^2\}$ , where  $\hat{\xi}$  is our MLE of the BT coefficients and  $\hat{V}_\xi$  is the sandwich variance of the logistic regression.

**Active sampling rule.** Our sampling rule was to choose the model pair  $a \in \mathcal{A}$  proportionally to the reduction in confidence interval size by sampling that pair:

$$P_t(a) \propto \sqrt{\frac{\hat{\Sigma}_{t,a,a}}{|\{t : A_t = a\}|}} - \sqrt{\frac{\hat{\Sigma}_{t,a,a}}{|\{t : A_t = a\}| + 1}}. \quad (9)$$

### 5.1. Detecting Anomalous Users

On a different note, we take a first step towards identifying anomalous IP addresses in our dataset. In a dataset of  $U$  unique IPs, we let  $\text{IP} = \{1, \dots, U\}$  be the set of all IP addresses. Consider a “test” user, outside this database, who gives ratings  $H'_1, \dots, H'_n$  when presented actions  $A'_1, \dots, A'_n$ . The idea of our procedure is to compare the distribution of ratings for the new user to the historical distribution of ratings for a given action. We let  $\mathcal{H}_a = \{H_t : A_t = a\}$  and every time a user submits a vote, we calculate the following number:

$$p_i = \frac{1}{|\mathcal{H}_{A'_i}| + 1} \left( 1 + \sum_{h \in \mathcal{H}_{A'_i}} \mathbb{1}\{h \geq H'_i\} \right). \quad (10)$$

Under the null hypothesis that  $\mathcal{H}_{A'_i}$  is exchangeable with  $H'_i$ ,  $p_i$  is a valid p-value (see Appendix C for a proof). Furthermore, the dependence of these p-values asymptotically is negligible.

With this p-value in hand, we can test against this null hypothesis sequentially by using Fisher’s combination test (Fisher, 1928) along with a variant of the Bonferroni correction. In particular, for each user, after their  $j$ th vote, we compute  $M_j = -2 \sum_{i=1}^j \log(p_i)$ . At 5 randomly chosen values of  $j$  between 1 and 100, we identify a user as

anomalous if  $M_j \geq \chi_{2j, 1-\alpha/5}^2$ . (The times are randomly chosen, as to avoid anomalous users strategizing to hack this p-value.) Despite the heuristic application of this procedure, it seems to work well in our small-scale tests reported in Table 5.

## 6. Data Analysis

To examine whether Arena’s crowdsourced data reflects real-world use cases, we conduct topic modeling on the user prompts. We show how effective are these prompts in distinguishing models. Lastly, we validate the vote quality by relabeling data with experts.

### 6.1. Topic Modeling on User Prompts

To study the prompt diversity, we build a topic modeling pipeline with BERTopic<sup>3</sup> (Grootendorst, 2022). We start with transforming user prompts into representation vectors using OpenAI’s text embedding model (text-embedding-3-small). To mitigate the curse of dimensionality for data clustering, we employ UMAP (Uniform Manifold Approximation and Projection) (McInnes et al., 2020) to reduce the embedding dimension from 1,536 to 5. We then use the hierarchical density-based clustering algorithm, HDBSCAN, to identify topic clusters with minimum cluster size 32. Finally, to obtain topic labels, we sample 10 prompts from each topic cluster and feed into GPT-4-Turbo for topic summarization.

The pipeline identifies 600 clusters covering a wide range of topics including poetry writing, coding, math, and medical queries. We present the top-16 topic clusters in Figure 3. We observe that the largest cluster only accounts for 1% of the entire set and the rest quickly drop to <0.5%, and the similarity between clusters is small, showing a long-tail and diverse distribution. Due to space limit, we present the similarity matrix and cluster hierarchy of top-64 clusters in Figure 11 and 12 in Appendix.

### 6.2. Can Arena Prompts Distinguish Models?

Next, we study how effective are these topic clusters in distinguishing models strengths. Constructing challenging prompts has become increasingly difficult due to LLMs’ fast growing capabilities. For example, open models such as Llama-2-70b-chat can likely answer inquiries about movie or travel recommendation as good as GPT-4, but not in other domains such as reasoning or coding. To demonstrate, we sample 30 prompts from seven topic clusters and compare the performance of Llama-2-70b-chat and GPT-4. To control variables, we factor out user votes and consider LLM-as-judge (Zheng et al., 2023b) to evaluate model re-

<sup>3</sup><https://github.com/MaartenGr/BERTopic>

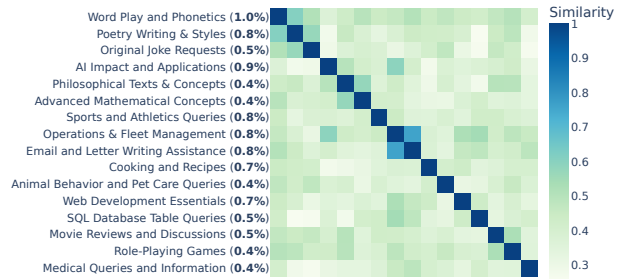


Figure 3. Similarity matrix of top-16 topic clusters. The number followed by the topic label represents the cluster size in percentage. Note that similarity is computed by cluster’s centroid embeddings, hence diagonals are always one.

Table 2. GPT-4-0613’s win-rate against Llama-2-70b-chat on 30 sample prompts from various topic clusters. We use GPT-4-turbo as judge to evaluate model responses in pairwise comparison.

Topic Cluster	Win-rate	Size
Python Game Programming Challenge	96.7%	0.2%
C/C++ Process Multi-Threading	86.7%	0.3%
SQL Query Database Assistance	73.3%	0.2%
Poetry Writing Prompts	66.7%	1.1%
Python Coding Basics	65.0%	0.2%
Linguistic Analysis & Wordplay	58.3%	0.7%
Travel Itinerary Planning	58.3%	0.4%
Movie Recommendations & Ratings	53.3%	0.2%

sponse. Results are shown in Table 2, where we see GPT-4 has significantly higher win-rate (up to 97%) in clusters that require coding and reasoning skills. On the other hand, for clusters with less problem-solving tasks, GPT-4 win-rate drops to below 60%. We show examples in Appendix D.1. This result shows models may exhibit varying strengths in different areas, but also highlights some of the topic clusters in Chatbot Arena are effective in differentiate models.

**Building Challenging Benchmark.** To further demonstrate the prompt quality, we show it is possible to construct a challenging benchmark with crowd-sourced user prompts. To ensure both topic coverage and quality, we first run the topic modeling pipeline and follow a similar procedure in Zheng et al. (2023a) to select challenging questions sampled from each topic cluster. Examples prompts and evaluation procedures can be found in the Appendix D.2 and Appendix D.3, respectively. We observe the selected prompts are highly effective in differentiating models. In Figure 4, we compare Arena bench against a widely used LLM benchmark, MT-Bench (Zheng et al., 2023b). We can see that Arena Bench effectively reveals a significant gap in performance between proprietary and the strongest open models.

### 6.3. Validating Vote Quality

To assess the quality of crowdsourced votes, we randomly selected 160 battles between GPT-4-Turbo and Llama-2-13B, as well as GPT-4-Turbo and GPT-3.5-Turbo-0613. We

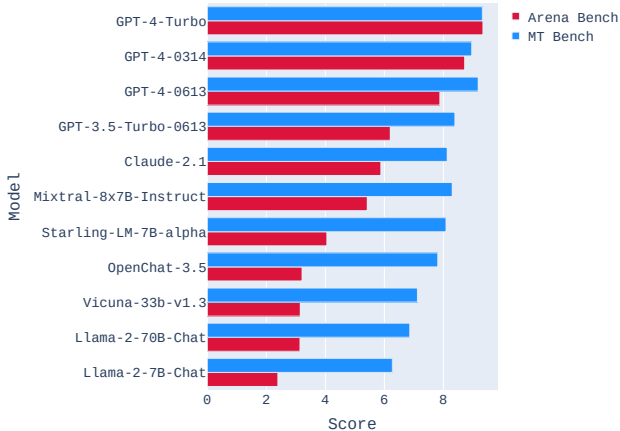


Figure 4. Model’s performance between Arena Bench and MT-Bench, showing an increased gap between open and proprietary models. Both uses GPT-4 as judge.

then asked experts<sup>4</sup> to label their preference per comparison. The experts were given the prompts and answers blindly, and asked to carefully fact-check model’s answer with external resources like search engine. Manually labeling each data point took on average 3-5 minutes. For reference, we also use GPT-4 as a judge for pairwise comparisons. The agreement rate between crowd-users, experts, and GPT-4-judge are presented in Table 3. The corresponding win-rate are shown in Table 4.

To summarize, we observe high agreement rates (72% to 83%) between Arena crowd-user and experts in both setup. Note that agreement rates between two experts are around similar levels (79.4% and 89.8%). As for the 10%-20% disagreement between experts, it is mostly due to some user prompts don’t have a ground truth answer. Depending on the preference of the evaluator, sometimes both answers can be argued as being better than the other one, such as the examples in Appendix D.4. The gap between crowd-vs-expert agreement rate and expert-vs-expert agreement rate (5%-10%) is mostly attributed to crowd user making mistakes or overlooking factual errors in model’s response. Overall, the agreement rates presented in Table 3 validate the decent quality of crowd-sourced votes in Chatbot Arena.

## 7. Experiments

### 7.1. Ranking system

**Computing the rank on real data.** In this section, we report results from our experiments on approximate ranking. For this experiment, we ran a replay of  $T = 213,576$  historical votes from our online platform and calculate the BT coefficients using our earlier-described estimation algorithm

<sup>4</sup>The laborers are graduate students at UC Berkeley.

Table 3. Pairwise agreement rate between crowd-user, gpt-4 judge, and experts on pairwise battles. The top part of the table is between GPT-4-Turbo and Llama-2-13b-chat. The bottom is between GPT-4-Turbo and GPT-3.5-Turbo-0613.

Llama-2-13b	Expert 1	Expert 2	GPT-4
Crowd	72.8%	77.8%	75.6%
Expert 1	-	89.8%	81.0%
Expert 2	-	-	78.5%

GPT-3.5-Turbo	Expert 1	Expert 2	GPT-4
Crowd	73.8%	83.1%	75.6%
Expert 1	-	79.4%	76.3%
Expert 2	-	-	79.3%

Table 4. GPT-4-Turbo’s win-rate across crowd-user, gpt-4 judge, and experts on pairwise battles against Llama-2-13b and GPT-3.5-Turbo-0613.

Baseline	Arena User	Expert 1	Expert 2	GPT-4
Llama-2-13b	81.2%	89.4%	86.9%	78.8%
GPT-3.5-Turbo	76.3%	82.5%	89.4%	79.4%

with confidence intervals; see Figure 5 for these intervals (with and without multiplicity correction; the formal notion of approximate ranking technically requires multiplicity correction, but it makes the intervals looser).

**Evaluating the coverage of the intervals.** A natural follow-up question is whether or not the intervals are doing their job correctly: whether they cover the true BT coefficients with probability at least (and almost exactly)  $1 - \alpha$ . Of course, this cannot be evaluated on real data, so we run a simulation. A vector of BT coefficients is drawn, with each coordinate sampled i.i.d. from a distribution  $\text{beta}(1/\gamma, 1/\gamma)$ ; we take  $\gamma = 2$  in Figure 6 (and we vary  $\gamma$  in Appendix A). Given these coefficients, a dataset is synthesized, and the coverage and average width are computed for each of 20 trials. The results can be seen in Figure 6 for the uncorrected intervals. The coverage of the intervals behaves as expected, centering around  $1 - \alpha$ , regardless of the number of models. Meanwhile, the more models are included, the larger the intervals become.

**Evaluating the active sampling rule.** Next, we discuss the evaluation of our active sampling rule as Equation (9) for win matrix estimation. We evaluate this sampling rule by taking the best fit BT coefficients to our 213,576 point sized holdout set, and then sampling from that distribution using our active sampling algorithm. The results are displayed in Figure 7. It is hard to tell by looking at plots, but the improvement is substantial: To estimate  $\theta^*$  to a precision of 0.2, random needs 6,800 samples and adaptive needs 4,400 samples; meanwhile to estimate the score to a precision of 0.3, random needs 17,200 samples and adaptive needs 16,400 samples. Thus, the random baseline requires 54% and 5% more data to achieve the same level of precision,

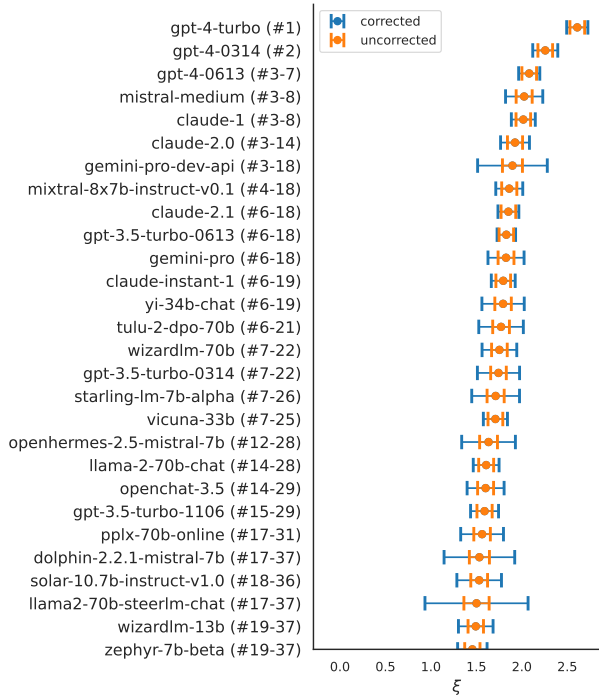


Figure 5. Intervals for the BT coefficients with and without multiplicity correction. The multiplicity correction, in this case a chi-square CLT interval, is technically required for the purpose of calculating the ranking, because it ensures *all* scores are simultaneously contained in their intervals (and the ranking is a function of all the scores). However, it induces extra conservatism, so we report both intervals.

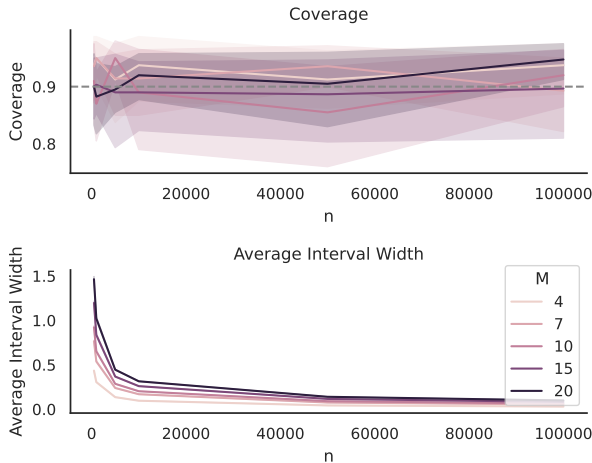


Figure 6. Intervals for the BT coefficients as a function of the number of samples and the number of models  $M$ .

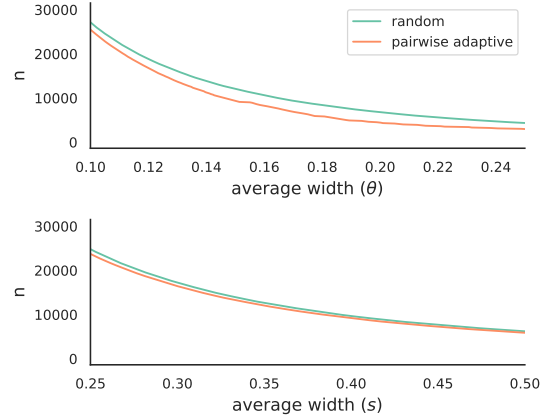


Figure 7. Interval widths on the win matrix (upper figure) and on the BT coefficients (lower figure) as a function of the number of samples, for random sampling and also adaptive sampling. Improvements from adaptive sampling can be seen in both cases, although they are more subtle on the scale of the score.

Table 5. Confusion matrix of different  $\alpha$ . “Pred.” means predicted. Positive means anomalous and negative means normal.

$\alpha = 0.1$	Pred. Positive	Pred. Negative
Actual Positive	13/14	12/36
Actual Negative	1/14	24/36
$\alpha = 0.3$	Pred. Positive	Pred. Negative
Actual Positive	21/29	4/21
Actual Negative	8/29	17/21

respectively. One can see from the plots in Figure 7 that these results are not cherry-picked: the sample-efficiency of our method is better at all values on the horizontal axis.

### 7.2. Anomalous Users Detection

We evaluate the outlier detection method in Section 5.1. We construct the evaluation set by manually identifying 25 anomalous users whose inputs are highly repetitive or meaningless (e.g., asking “hi” for 100 times or inputting garbled texts). We randomly sample 25 normal users with at least 50 votes, and inspect their input prompts to ensure no abnormal behaviors. As mentioned in Section 5.1, per user we compute five  $M_j$  and identify the user as anomalous if  $M_j \geq \chi_{2j, 1-\alpha/5}^2$ . We present results of two different  $\alpha$  (i.e., the significance level) in Table 5. We find the detection method effective (e.g., reaching 90% true positive and 60-70% true negative rate). We inspect the false negative errors and find those are from users do not always behave abnormally, making them harder to detect.

## 8. Discussion

**Limitations.** Although our user base is extensive, we anticipate that it will primarily consist of LLM hobbyists and



researchers who are eager to experiment with and evaluate the latest LLMs. This inclination may result in a biased distribution of users. Additionally, despite the wide array of topics encompassed by the prompts discussed in previous sections, the data predominantly comes from our online chat interface. This source might not accurately reflect the real-world usage of LLMs in production environments or specialized domains, potentially leading to a skewed prompt distribution. Moreover, our study concentrates on assessing the helpfulness of LLMs but overlooks their safety aspects. We recognize the possibility and necessity of a parallel mechanism to evaluate the safety of these models.

**Future Directions.** In our future work, we plan to develop comprehensive topic leaderboards and establish a dedicated section for multimodal and agent-based LLMs in more dynamic, gamified settings, catering to more complex tasks. We also believe our approach to detecting harmful users could be improved and made more formally rigorous by using the theory of nonnegative supermartingales and E-values (Howard et al., 2020; Waudby-Smith & Ramdas, 2020; Vovk & Wang, 2021; Ramdas et al., 2023); this would deal with the dependence, but the variants we tried did not perform well in terms of power.

## 9. Conclusion

In this paper, we present Chatbot Arena, an open platform for evaluating LLMs through crowdsourced, pairwise human preferences. We conduct an in-depth analysis of the crowdsourced user prompts and preference votes to validate the diversity and quality. We develop an efficient model sampling and ranking algorithm. Our dataset including 100K pairwise preference votes will be released for future research.

## Acknowledgments

This project is partly supported by sponsorship from Kaggle, MBZUAI, a16z, Together AI, Anyscale, and HuggingFace. This project is also partly supported by Accenture, AMD, Google, IBM, Intel, Microsoft, Samsung SDS, SAP, Uber, and VMware. Lianmin Zheng is supported by a Meta Ph.D. Fellowship. The authors would like to thank Siyuan Zhuang for insightful discussion and Tijana Zrnić for helpful feedback on the manuscript.

## Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

## References

- Bai, Y., Jones, A., Ndousse, K., Askell, A., Chen, A., Das-Sarma, N., Drain, D., Fort, S., Ganguli, D., Henighan, T., et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.
- Boubdir, M., Kim, E., Ermis, B., Hooker, S., and Fadaee, M. Elo uncovered: Robustness and best practices in language model evaluation, 2023.
- Bradley, R. A. and Terry, M. E. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.
- Busa-Fekete, R., Huellermeier, E., and Szörényi, B. Preference-based rank elicitation using statistical models: The case of mallows. In Xing, E. P. and Jebara, T. (eds.), *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pp. 1071–1079, Beijing, China, 22–24 Jun 2014a. PMLR. URL <https://proceedings.mlr.press/v32/busa-fekete14.html>.
- Busa-Fekete, R., Huellermeier, E., and Szörényi, B. Preference-based rank elicitation using statistical models: The case of mallows. In Xing, E. P. and Jebara, T. (eds.), *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pp. 1071–1079, Beijing, China, 22–24 Jun 2014b. PMLR. URL <https://proceedings.mlr.press/v32/busa-fekete14.html>.
- Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. d. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.
- Chernoff, H. *Sequential Design of Experiments*, pp. 345–360. Springer New York, New York, NY, 1992. ISBN 978-1-4612-4380-9. doi: 10.1007/978-1-4612-4380-9\_27. URL [https://doi.org/10.1007/978-1-4612-4380-9\\_27](https://doi.org/10.1007/978-1-4612-4380-9_27).
- Chiang, C.-H. and Lee, H.-y. Can large language models be an alternative to human evaluations? In Rogers, A., Boyd-Graber, J., and Okazaki, N. (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15607–15631, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.870. URL <https://aclanthology.org/2023.acl-long.870>.
- Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano,

- R., et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- Cui, G., Yuan, L., Ding, N., Yao, G., Zhu, W., Ni, Y., Xie, G., Liu, Z., and Sun, M. Ultrafeedback: Boosting language models with high-quality feedback, 2023.
- DiCiccio, T. J. and Efron, B. Bootstrap confidence intervals. *Statistical science*, 11(3):189–228, 1996.
- Durrett, R. *Probability: theory and examples*, volume 49. Cambridge university press, 2019.
- Elo, A. E. The proposed uscf rating system, its development, theory, and applications. *Chess Life*, 22(8):242–247, 1967.
- Fisher, R. A. *Statistical methods for research workers*. Number 5. Oliver and Boyd, 1928.
- Freedman, D. A. On the so-called “huber sandwich estimator” and “robust standard errors”. *The American Statistician*, 60(4):299–302, 2006.
- Gemini, T., Anil, R., Borgeaud, S., Wu, Y., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.
- Geng, X., Gudibande, A., Liu, H., Wallace, E., Abbeel, P., Levine, S., and Song, D. Koala: A dialogue model for academic research. Blog post, April 2023. URL <https://bair.berkeley.edu/blog/2023/04/03/koala/>.
- Grootendorst, M. Bertopic: Neural topic modeling with a class-based tf-idf procedure. *arXiv preprint arXiv:2203.05794*, 2022.
- Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2020.
- Howard, S. R., Ramdas, A., McAuliffe, J., and Sekhon, J. Time-uniform chernoff bounds via nonnegative supermartingales. 2020.
- Huang, Y., Lin, Z., Liu, X., Gong, Y., Lu, S., Lei, F., Liang, Y., Shen, Y., Lin, C., Duan, N., et al. Competition-level problems are effective llm evaluators. *arXiv preprint arXiv:2312.02143*, 2023.
- Huber, P. J. et al. The behavior of maximum likelihood estimates under nonstandard conditions. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pp. 221–233. Berkeley, CA: University of California Press, 1967.
- Hunter, D. R. MM algorithms for generalized Bradley-Terry models. *The Annals of Statistics*, 32(1):384 – 406, 2004. doi: 10.1214/aos/1079120141. URL <https://doi.org/10.1214/aos/1079120141>.
- Karimi, M. R., Gürel, N. M., Karlaš, B., Rausch, J., Zhang, C., and Krause, A. Online active model selection for pre-trained classifiers. In *International Conference on Artificial Intelligence and Statistics*, pp. 307–315. PMLR, 2021.
- Karpinska, M., Akoury, N., and Iyyer, M. The perils of using Mechanical Turk to evaluate open-ended text generation. In Moens, M.-F., Huang, X., Specia, L., and Yih, S. W.-t. (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 1265–1285, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.97. URL <https://aclanthology.org/2021.emnlp-main.97>.
- Kiela, D., Bartolo, M., Nie, Y., Kaushik, D., Geiger, A., Wu, Z., Vidgen, B., Prasad, G., Singh, A., Ringshia, P., et al. Dynabench: Rethinking benchmarking in nlp. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4110–4124, 2021.
- Köpf, A., Kilcher, Y., von Rütte, D., Anagnostidis, S., Tam, Z.-R., Stevens, K., Barhoum, A., Duc, N. M., Stanley, O., Nagyfi, R., et al. Openassistant conversations—democratizing large language model alignment. *arXiv preprint arXiv:2304.07327*, 2023.
- Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.
- Li, X., Zhang, T., Dubois, Y., Taori, R., Gulrajani, I., Guestrin, C., Liang, P., and Hashimoto, T. B. Alpaca-eval: An automatic evaluator of instruction-following models. [https://github.com/tatsu-lab/alpaca\\_eval](https://github.com/tatsu-lab/alpaca_eval), 2023.
- Li, Y., Choi, D., Chung, J., Kushman, N., Schrittwieser, J., Leblond, R., Eccles, T., Keeling, J., Gimeno, F., Dal Lago, A., et al. Competition-level code generation with alpha-code. *Science*, 378(6624):1092–1097, 2022.
- Liang, P., Bommasani, R., Lee, T., Tsipras, D., Soylu, D., Yasunaga, M., Zhang, Y., Narayanan, D., Wu, Y., Kumar, A., et al. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*, 2022.

- Lin, Z., Wang, Z., Tong, Y., Wang, Y., Guo, Y., Wang, Y., and Shang, J. ToxicChat: Unveiling hidden challenges of toxicity detection in real-world user-AI conversation. In Bouamor, H., Pino, J., and Bali, K. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 4694–4702, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.311. URL <https://aclanthology.org/2023.findings-emnlp.311>.
- Liu, T.-Y. et al. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3): 225–331, 2009.
- McInnes, L., Healy, J., and Melville, J. Umap: Uniform manifold approximation and projection for dimension reduction, 2020.
- OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- Oren, Y., Meister, N., Chatterji, N., Ladhak, F., and Hashimoto, T. B. Proving test set contamination in black box language models. *arXiv preprint arXiv:2310.17623*, 2023.
- Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R. Training language models to follow instructions with human feedback, 2022.
- Ramdas, A., Grünwald, P., Vovk, V., and Shafer, G. Game-theoretic statistics and safe anytime-valid inference. *Statistical Science*, 38(4):576–601, 2023.
- Rao, P. V. and Kupper, L. L. Ties in paired-comparison experiments: A generalization of the bradley-terry model. *Journal of the American Statistical Association*, 62(317): 194–204, 1967. doi: 10.1080/01621459.1967.10482901.
- Srivastava, A., Rastogi, A., Rao, A., Shoeb, A. A. M., Abid, A., Fisch, A., Brown, A. R., Santoro, A., Gupta, A., Garriga-Alonso, A., et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023.
- Szörényi, B., Busa-Fekete, R., Paul, A., and Hüllermeier, E. Online rank elicitation for plackett-luce: A dueling bandits approach. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL [https://proceedings.neurips.cc/paper\\_files/paper/2015/file/7eacb532570ff6858afd2723755ff790-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2015/file/7eacb532570ff6858afd2723755ff790-Paper.pdf).
- Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
- Vovk, V. and Wang, R. E-values: Calibration, combination and applications. *The Annals of Statistics*, 49(3):1736–1754, 2021.
- Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N. A., Khashabi, D., and Hajishirzi, H. Self-instruct: Aligning language models with self-generated instructions. In Rogers, A., Boyd-Graber, J., and Okazaki, N. (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13484–13508, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.754. URL <https://aclanthology.org/2023.acl-long.754>.
- Waudby-Smith, I. and Ramdas, A. Estimating means of bounded random variables by betting. *arXiv preprint arXiv:2010.09686*, 2020.
- White, H. Maximum likelihood estimation of misspecified models. *Econometrica: Journal of the econometric society*, pp. 1–25, 1982.
- Yang, S., Chiang, W.-L., Zheng, L., Gonzalez, J. E., and Stoica, I. Rethinking benchmark and contamination for language models with rephrased samples. *arXiv preprint arXiv:2311.04850*, 2023.
- Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, 2019.
- Zheng, L., Chiang, W.-L., Sheng, Y., Li, T., Zhuang, S., Wu, Z., Zhuang, Y., Li, Z., Lin, Z., Xing, E. P., Gonzalez, J. E., Stoica, I., and Zhang, H. Lmsys-chat-1m: A large-scale real-world llm conversation dataset, 2023a.
- Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., Zhang, H., Gonzalez, J. E., and Stoica, I. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023b. URL <https://openreview.net/forum?id=ucCHPGDlao>.

Zhong, W., Cui, R., Guo, Y., Liang, Y., Lu, S., Wang, Y., Saied, A., Chen, W., and Duan, N. Agieval: A human-centric benchmark for evaluating foundation models. *arXiv preprint arXiv:2304.06364*, 2023.

Zhu, B., Frick, E., Wu, T., Zhu, H., and Jiao, J. Starling-7b: Improving llm helpfulness & harmlessness with rlaiif, November 2023.

## ✂ Chatbot Arena: Benchmarking LLMs in the Wild

[Blog](#) | [GitHub](#) | [Paper](#) | [Dataset](#) | [Twitter](#) | [Discord](#)

### 📋 Rules

- Ask any question to two anonymous models (e.g., ChatGPT, Claude, Llama) and vote for the better one!
- You can continue chatting until you identify a winner.
- Vote won't be counted if model identity is revealed during conversation.

### 🏆 Arena Elo [Leaderboard](#)

We collect 300K+ human votes to compute an Elo-based LLM leaderboard. Find out who is the 🏆 LLM Champion!

### 👉 Chat now!

The screenshot shows the Chatbot Arena interface. At the top, there is a search bar with the text "Expand to see the descriptions of 36 models". Below this, there are two panels for "Model A" and "Model B". Both panels display the same prompt: "i found a property i would like to buy. please draft a letter to the owners expressing my interest to purchase and convincing them of completing a private sale with me. list the benefits of private sale and how it will be done with our lawyers." Below the prompt, each model's response is shown. Model A's response is a letter with a subject line "Subject: Expression of Interest in Your Property for a Private Sale" and a body that discusses the benefits of a private sale. Model B's response is a letter with a subject line "Subject: Expression of Interest to Purchase Your Property - Private Sale Proposal" and a body that discusses the benefits of a private sale. Below the responses, there are four buttons for voting: "A is better", "B is better", "Tie", and "Both are bad". At the bottom, there is a text input field with the placeholder "Enter your prompt and press ENTER" and a "Send" button.

Figure 8. Screenshot of Chatbot Arena.

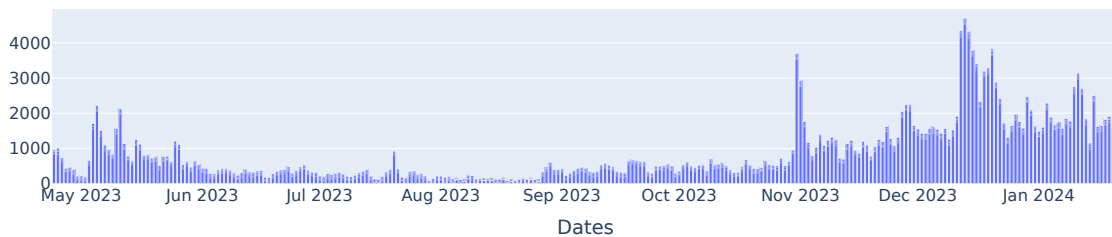


Figure 9. The number of votes over time



### Hierarchical Clustering

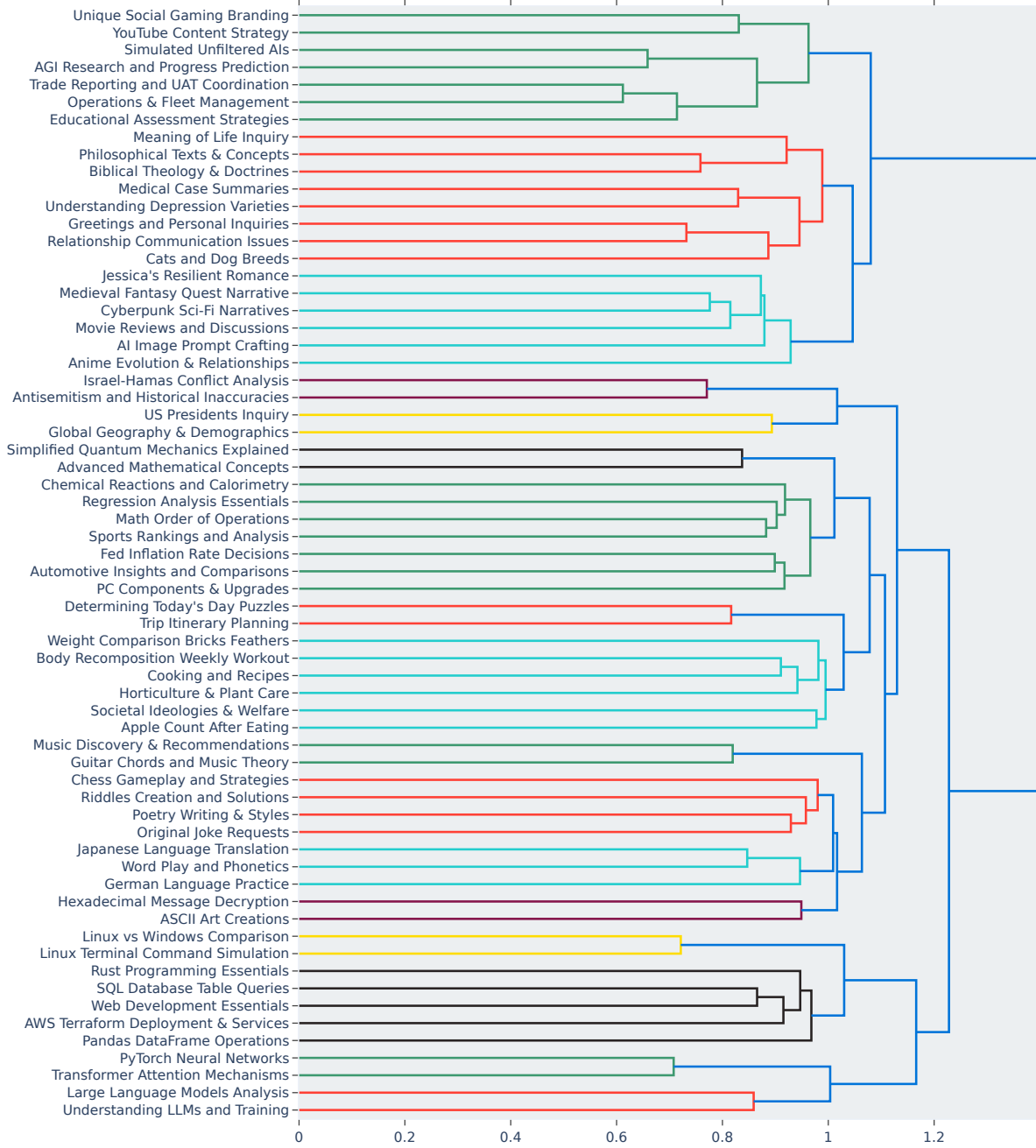


Figure 12. Top-64 clusters visualized in hierarchy. x-axis represents the cosine similarity distance. y-axis shows the topic title per cluster summarized by gpt-4-turbo.

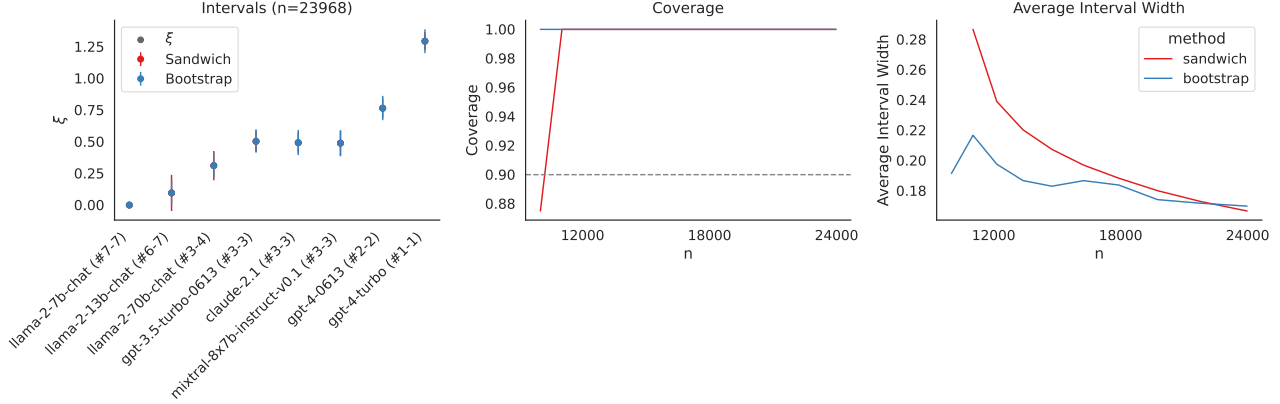


Figure 13. Replay experiment showing the intervals, coverage, and average interval sizes of the bootstrap and of the sandwich intervals. The sandwich intervals, though larger in small samples, are more stable, and in large samples, they actually become smaller. We use the multiplicity corrected version of both intervals, so they both have a coverage of 1. (Coverage here is calculated with respect to the BT coefficient solution on the full dataset, so it is not as meaningful as in the simulation plot below.)

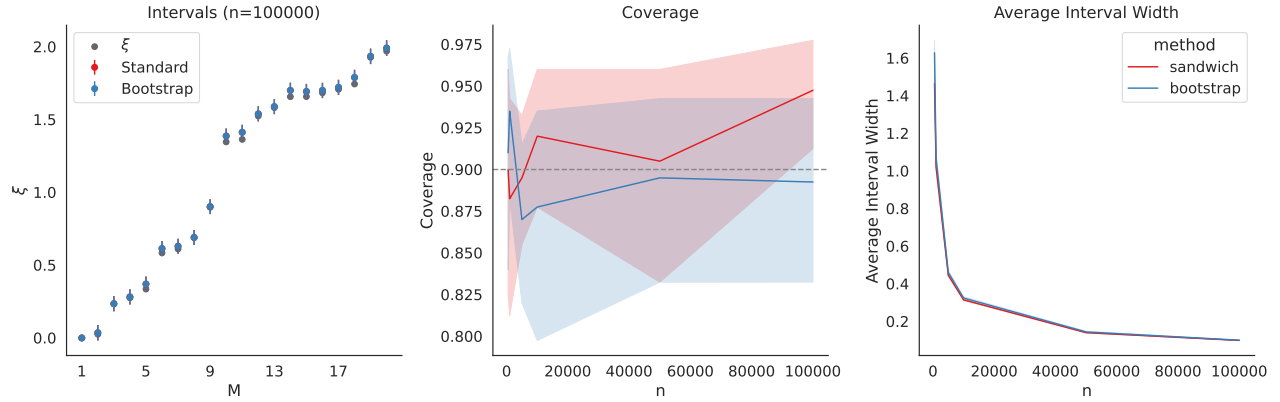


Figure 14. Synthetic experiment. Coefficients are drawn from the BT-coefficient distribution  $x$  on the left. Coverage of the uncorrected intervals is shown in the middle. Line plots of set width are shown on the right, and they almost perfectly match.

### A. Confidence Interval Simulation Study

We conduct a simulation study to evaluate the bootstrap confidence intervals versus the sandwich estimator. To a large extent, both intervals are the same—indeed, their intervals are often identical to the naked eye. Nonetheless, in our experiments, there are some differences. First, in Figure 13, we conduct a replay study using the same 213576 data points mentioned in the main text.

We also do a suite of experiments in simulation using the same beta generating process as in the main text, with  $\gamma = 2$ . The result is shown in Figure 14; results are similar across many choices of the parameter  $\gamma$  and the model strength, which indicates that both intervals will have good coverage and width in the practical conditions we would expose them to.

### B. The Nonparametric Bradley-Terry Model

**Nonparametric Bradley-Terry.** We next consider a nonparametric extension of the Bradley-Terry (BT) model (Bradley & Terry, 1952) to the case where the ranking is not necessarily transitive. Let  $\mathcal{G}(m)$  denote the set of all paths to the model  $m$ , i.e.,

$$\mathcal{G}(m) = \{g \in \mathcal{B}^{M-1} : g_{i,1} \neq g_{j,1}, \forall i \neq j, \text{ and } g_{M-1,2} = m\}, \tag{11}$$

where  $\mathcal{B} = \mathcal{A} \cup \{(a_2, a_1) : a \in \mathcal{A}\}$ . Each element of  $\mathcal{G}(m)$  is a chain of model pairings that leads to  $m$ ; for example, if  $m = 5$  and  $M = 6$ , one element of  $\mathcal{G}(m)$  is  $((1, 2), (2, 4), (4, 3), (3, 6), (6, 5))$ . Our score function is given by the average



path-sum of the log odds of the second model winning, over the entirety of  $\mathcal{G}(m)$ :

$$s(\theta)_m = \frac{1}{|\mathcal{G}(m)|} \sum_{g \in \mathcal{G}(m)} \left( \log \frac{\theta'((1, g_{1,1}))}{1 - \theta'((1, g_{1,1}))} + \sum_{a \in g} \log \frac{\theta'(a)}{1 - \theta'(a)} \right), \quad (12)$$

where  $\theta'(a) = \theta(a)\mathbb{1}\{a \in \mathcal{A}\} + (1 - \theta((a_2, a_1))\mathbb{1}\{a \notin \mathcal{A}\})$ , with the convention that  $\theta((m, m)) = 1/2$  for all  $m$ . Note that for any  $g \in \mathcal{G}(m)$  where  $a \in g$  and  $m \notin a$ , we also have some  $g' \in \mathcal{G}(m)$  such that  $(a_2, a_1) \in g'$ . Meanwhile, if  $a \in g$  and  $m \in a$ , then  $a = (m', m)$  for some  $m'$ . Thus, we can compute

$$s(\theta)_m = \sum_{\substack{a \in \mathcal{A} \\ m \notin a}} \frac{1}{2} \left( \log \frac{\theta'(a)}{1 - \theta'(a)} + \log \frac{\theta'((a_2, a_1))}{1 - \theta'((a_2, a_1))} \right) + \sum_{m' \in [M] \setminus \{m\}} \left( \log \frac{\theta'((m', m))}{1 - \theta'((m', m))} + \frac{\theta'((1, m'))}{1 - \theta'((1, m'))} \right) \quad (13)$$

$$= \sum_{\substack{a \in \mathcal{A} \\ m \notin a}} \frac{1}{2} \left( \log \frac{\theta(a)}{1 - \theta(a)} + \log \frac{1 - \theta(a)}{\theta(a)} \right) + \sum_{m' \in [M] \setminus \{m\}} \left( \log \frac{\theta'((m', m))}{1 - \theta'((m', m))} + \frac{\theta'((1, m'))}{1 - \theta'((1, m'))} \right) \quad (14)$$

$$= \sum_{m' \in [M] \setminus \{m\}} \left( \log \frac{\theta'((m', m))}{1 - \theta'((m', m))} + \frac{\theta'((1, m'))}{1 - \theta'((1, m'))} \right) \quad (15)$$

$$= \sum_{m' \in [M] \setminus \{m\}} \left( (1 - 2\mathbb{1}\{m' > m\}) \log \frac{\theta((m', m))}{1 - \theta((m', m))} + \frac{\theta((1, m'))}{1 - \theta((1, m'))} \right). \quad (16)$$

This score is always well-defined, and is a simple, smooth function of  $\theta$ . Its derivative is, for all  $a \in \mathcal{A}$ ,

$$\frac{\partial}{\partial \theta(a)} s(\theta)_m = \mathbb{1}\{a_2 = m\} (1 - 2\mathbb{1}\{a_1 > m\}) \frac{1}{\theta(a)(1 - \theta(a))} + \mathbb{1}\{a_1 = 1, a_2 \neq m\} \frac{1}{\theta(a)(1 - \theta(a))}. \quad (17)$$

How is the BT score related to the original Bradley-Terry model? In the original Bradley-Terry model,  $H_t \in \{0, 1\}$ , and the probability of model  $m$  beating model  $m'$  is assumed to be given by

$$\theta((m', m)) = \frac{e^{\xi_m}}{e^{\xi_m} + e^{\xi_{m'}}}, \quad (18)$$

for some unknown parameters  $\xi_1, \dots, \xi_M$ —the *Bradley-Terry coefficients*. The basic goal of the Bradley-Terry model is to estimate these parameters from the observed outcomes. In our setting, however, we use the outcomes to get a CLT on  $\theta$ , and then can immediately recover the coefficients. Taking without loss of generality  $\xi_1 = 0$ , we have that

$$\log \frac{\theta((1, m'))}{1 - \theta((1, m'))} + \log \frac{\theta((m', m))}{1 - \theta((m', m))} = \log \frac{\theta((1, m'))}{\theta((m', 1))} + \log \frac{\theta((m', m))}{\theta((m, m'))} \quad (19)$$

$$= \log \frac{e^{\xi_{m'}} (e^{\xi_{m'}} + 1)}{e^{\xi_{m'}} + 1} + \log \frac{e^{\xi_m} (e^{\xi_{m'}} + e^{\xi_m})}{e^{\xi_{m'}} (e^{\xi_{m'}} + e^{\xi_m})} \quad (20)$$

$$= \xi_{m'} + \xi_m - \xi_{m'} = \xi_m \quad (21)$$

Thus, all the sums over paths in (12) are equal to  $\xi_m - \xi_{g_{1,1}}$ .

$$\log \frac{\theta'((1, g_{1,1}))}{1 - \theta'((1, g_{1,1}))} + \sum_{a \in g} \log \frac{\theta'(a)}{1 - \theta'(a)} \quad (22)$$

$$= \xi_{g_{1,1}} + \xi_{g_{1,2}} - \xi_{g_{1,1}} + \xi_{g_{2,2}} - \xi_{g_{2,1}} + \dots + \xi_{g_{M-1,2}} - \xi_{g_{M-1,1}} \quad (23)$$

$$= \xi_{g_{M-1,2}} = \xi_m. \quad (24)$$

Thus, if the parametric BT model is well-specified, the nonparametric version will exactly recover the Bradley-Terry coefficients. However, our nonparametric analogue of the BT model has major advantages over the original: it will retain statistical validity even if  $H_t$  is not binary, if the win rate is non-transitive, and if the logistic model assumed by the BT model is misspecified. In practice, the nonparametric BT coefficient can be easily computed by (16).

### C. Valid P-Value

Consider the p-value

$$p_i = \frac{1}{|\mathcal{H}_{A'_i}| + 1} \left( 1 + \sum_{h \in \mathcal{H}_{A'_i}} \mathbb{1}\{h \geq H'_i\} \right). \quad (25)$$

We will prove that this p-value is valid, i.e., that  $\mathbb{P}(p_i \leq t) \leq t$ , under the null hypothesis that the vector  $\mathcal{H}' = (H_t : A_t = A'_i) \parallel (H'_i)$  is exchangeable, where  $\parallel$  denotes the concatenation operator. First, notice that we can equivalently write

$$p_i = \frac{1}{|\mathcal{H}'|} \sum_{h \in \mathcal{H}'} \mathbb{1}\{h \geq H'_i\}. \quad (26)$$

We also have that  $\mathbb{P}(p_i \leq t) \leq \mathbb{P}(p_i \leq \frac{k}{|\mathcal{H}'|})$ , where  $k = \lfloor t|\mathcal{H}'| \rfloor / |\mathcal{H}'|$ . Then, since the data points are exchangeable, we have that  $H'_i$  is uniformly distributed among the ranks of  $\mathcal{H}'$ , so  $\mathbb{P}(p_i \leq \frac{k}{|\mathcal{H}'|}) \leq \frac{k}{|\mathcal{H}'|} \leq t$ , completing the proof.

## D. Sample Prompts

### D.1. Prompt Examples From Different Clusters

**Example 1:** An example user prompt from the cluster "Python Game Programming Challenge". This prompt distinctively separates GPT-4-0613 performance from Llama-2-70b-chat. The user request "higher quality code" from each assistant. While GPT-4-Turbo provides a correct implementation along with helpful explanation, Llama-2-70b-chat fail to provide any program and provided a less helpful response overall.

#### <|User Prompt|>

Develop a snowman game similar to the Hangman game but draws a snowman figure standing on the snow instead of a hangman. It also downloads (or uses library) the complete English dictionary to select words for three levels of difficulty. Think step-by-step, take a deep breath and choose a higher quality code for each step.

#### <|GPT-4-0613's Answer|>

Sure, here's a step-by-step guide on how to create a Snowman game similar to Hangman. We'll use Python for this example.

Step 1: Import necessary libraries

```
```python
import random
import urllib.request
```
```

Step 2: Download the English dictionary

```
```python
url = "https://raw.githubusercontent.com/dwyl/english-words/master/words.txt"
urllib.request.urlretrieve(url, 'words.txt')
```
```

Step 3: Load the words from the dictionary and categorize them into three levels of difficulty

```
```python
with open('words.txt') as word_file:
    english_words = word_file.read().split()
easy_words = [word for word in english_words if 4 <= len(word) <= 6]
medium_words = [word for word in english_words if 7 <= len(word) <= 9]
hard_words = [word for word in english_words if len(word) >= 10]
```
```

Step 4: Create a function to randomly select a word based on the difficulty level

```
```python
def get_random_word(difficulty):
    if difficulty == 'easy':
        word = random.choice(easy_words)
    elif difficulty == 'medium':
        word = random.choice(medium_words)
    else:
        word = random.choice(hard_words)
    return word.upper()
```
```

Step 5: Create a function to print the snowman

```

'''python
def print_snowman(guesses):
    snowman = [' _____ ', ' ( ) ', ' ( ( ) ) ', ' (__(__)_) ', ' (
        ) ', ' ( ) ', ' ( ) ', ' ( ) ', ' ( ) ', ' (
            ) ', ' _____ ']
    for i in range(guesses):
        print(snowman[i])
'''
Step 6: Create the main function for the game
'''python
def play_snowman():
    difficulty = input("Choose difficulty (easy, medium, hard): ")
    word = get_random_word(difficulty)
    guessed_letters = []
    guesses = 0
    print('The word contains', len(word), 'letters.')

    while guesses < 10:
        guess = input("Guess a letter: ").upper()
        if guess in guessed_letters:
            print("You already guessed that letter.")
        elif guess in word:
            guessed_letters.append(guess)
            print("Good job! The word contains that letter.")
        else:
            guesses += 1
            print_snowman(guesses)
            print("Sorry, the word does not contain that letter.")
    print(' '.join(letter if letter in guessed_letters else '_' for
        letter in word))

    if set(guessed_letters) == set(word):
        print("Congratulations! You've successfully guessed the word.")
        break
    else:
        print("Sorry, you didn't guess the word. The word was", word)
'''
Step 7: Run the game
'''python
play_snowman()
'''
This is a simple version of the game. You can enhance it by adding more
features like scoring, time limit, etc.
'''

```

<|Llama-2-70b-chat's Answer|>

Sure! Here's a step-by-step guide to developing a snowman game similar to Hangman, using the complete English dictionary for three levels of difficulty.

Step 1: Setting up the project

- \* Create a new Python project directory and initialize it with Git.
- \* Install the necessary libraries:
  - + 'numpy' for generating random numbers
  - + 'nltk' for accessing the English dictionary
  - + 'pygame' for creating the game graphics
  - + 'pydub' for playing sound effects (optional)
- \* Download the English dictionary dataset (e.g., from Kaggle) and store it in a file named 'dictionary.txt'.

Step 2: Loading the dictionary and generating words

- \* Use the 'nltk' library to read the dictionary file and store the words in a list.
- \* Split the list into three parts: easy, medium, and hard words, based on their length and frequency of use.
- \* Shuffle the lists to randomize the order of the words.

Step 3: Creating the game graphics

- \* Use 'pygame' to create a window for the game, with a white background to represent the snow.
- \* Draw a snowman figure on the background, using black rectangles for the body, hat, and scarf.
- \* Add buttons for the player to submit their guesses.

Step 4: Implementing the game logic

- \* Create a function to generate a random word from the appropriate difficulty level and display it on the screen with missing letters (e.g. ., \_ \_ \_ \_ \_).
- \* Create a function to check if a guessed letter appears in the word, and update the display accordingly (e.g., \_ \_ \_ X \_ \_).
- \* Create a function to handle incorrect guesses, such as displaying a warning message or playing a sound effect.
- \* Create a function to win the game, such as displaying a congratulations message or playing a victory sound effect.

Step 5: Adding sound effects (optional)

- \* Use 'pydub' to play sound effects for various actions in the game, such as guessing a letter correctly or incorrectly.
- \* Consider using a library like 'pygame.mixer' to simplify the process of playing sound effects in pygame.

Step 6: Testing and refining the game

- \* Test the game thoroughly to ensure that it works as expected, with no bugs or errors.
- \* Refine the game mechanics and graphics as needed, based on player feedback or personal preference.
- \* Consider adding additional features, such as keeping track of the player's score or implementing a time limit for each guess.

Step 7: Packaging and distributing the game

- \* Package the game as a standalone executable file using a tool like PyInstaller.
- \* Distribute the game to friends, family, or online communities, and gather feedback for future improvements.

This is a high-level overview of the steps involved in creating a snowman game similar to Hangman. Of course, the details will depend on your specific implementation and the tools you choose to use. Good luck with your project!

<|GPT-4-Turbo's Judgment Verdict|>

GPT-4-0613 Wins

**Program Output:** Below is the terminal output of the snowman game programmed by GPT-4-0613, demonstrating its correctness. It is clear that GPT-4-0613's response is more helpful and relevant to the user's request.

```
>>> python snowman.py
Choose difficulty (easy, medium, hard): easy
The word contains 6 letters.
Good job! The word contains that letter.
_ A _ _ _ _
Guess a letter: e
Good job! The word contains that letter.
_ A _ _ _ E
Guess a letter: d

  _____
 (         )
Sorry, the word does not contain that letter.
_ A _ _ _ E
Guess a letter: g

  _____
 (         )
 ( (     ) )
```

**Example 2:** An example user prompt from the cluster "Python Game Programming Challenge". This prompt distinctively separates GPT-4-0613 performance from Llama-2-70b-chat. Due to limited space, the code from each assistant's response is omitted.

**<|User Prompt|>**

Write a python function that prints a christmas tree in the form of a triangle of asterisks, where the only input is size.

**<|GPT-4-0613's Code Output|>**

```
>>> print_christmas_tree(5)
```

```
      *
     ***
    *****
   *********
  ***********
 *****
```

**<|Llama-2-70b-chat's Code Output|>**

```
>>> print_christmas_tree(5)
```

```
* * * * *
* * * *
* * *
* *
*
```

**<|GPT-4-Turbo Verdict|>**

GPT-4-0613 Wins

**Example 3:** An example user prompt from the cluster "Movie Recommendations & Rating". Both assistant retrieve the correct movie information and gave an almost identical explanation. Therefore, this prompt cannot distinguish GPT-4-0613's performance from Llama-2-70b-chat.

**<|User Prompt|>**

Which film came out first, Blind Shaft or The Mask Of Fu Manchu?

**<|GPT-4-0613's Answer|>**

The Mask of Fu Manchu was released in 1932, and Blind Shaft was released in 2003. Therefore, The Mask of Fu Manchu came out first.

**<|Llama-2-70b-chat's Answer|>**

The Mask Of Fu Manchu came out first. It was released in 1932, while Blind Shaft was released in 2003.

**<|GPT-4-Turbo Verdict|>**

Tie

**Example 4:** An example user prompt from the cluster "Travel Itinerary Planning". Both assistant provided an equally helpful itinerary. Therefore, this prompt cannot distinguish GPT-4-0613's performance from Llama-2-70b-chat.

**<|User Prompt|>**

Can you make a list of things I could do with my girlfriend in Lille, France?

**<|GPT-4-0613's Answer|>**

Absolutely, here are some suggestions for activities you and your girlfriend can enjoy in Lille, France:

1. Visit the Palais des Beaux-Arts: This is one of France's largest museums and it's right in the heart of Lille. It houses collections of European paintings, a medieval and Renaissance collection, ceramics, French sculptures and more.
2. Explore the Vieux Lille: Stroll through the charming old town with its Flemish influenced architecture, chic boutiques, and delightful eateries.
3. Visit the Lille Cathedral (Basilique-cathedrale Notre-Dame-de-la-Treille): This beautiful cathedral is a mix of modern and traditional architecture.
4. Enjoy a picnic at the Citadel Park: This is a great place to relax, have a picnic, and enjoy the outdoors.
5. Visit the LaM (Lille Metropole Musee d'art moderne, d'art contemporain et d'art brut): This museum is home to a unique collection of modern and contemporary art.
6. Take a boat tour on the Deule River: This is a romantic way to see the city and its surroundings.
7. Visit the Maison Natale de Charles de Gaulle: This is the birthplace of Charles de Gaulle and it's now a museum dedicated to his life and work.
8. Explore the Marche de Wazemmes: This is one of the largest markets in France and it's a great place to buy local products.
9. Enjoy a meal at a local restaurant: Lille is known for its delicious cuisine, so make sure to try some local dishes.
10. Visit the Lille Zoo: This is a small but well-kept zoo that houses about 70 different species.
11. Take a romantic walk through the Jardin des Plantes de Lille: This botanical garden is a peaceful place to enjoy nature.
12. Visit the Musee de l'Hospice Comtesse: This museum is located in a former hospital and it's home to a collection of art and artifacts from the Middle Ages.
13. Enjoy a performance at the Lille Opera: Check the schedule and book tickets for a romantic night out.
14. Take a day trip to the nearby city of Bruges: This charming Belgian city is just a short train ride away.
15. Visit the Le Palais Rihour: This palace houses the Lille Tourist Office and it's also home to the Salle du Conclave, which has a beautiful Gothic fireplace.

Remember to check the opening times and any COVID-19 restrictions before you visit. Enjoy your trip!

**<|Llama-2-70b-chat's Answer|>**

Sure! Lille is a beautiful city in northern France that offers a variety of activities for couples to enjoy. Here are some ideas for things you could



do with your girlfriend in Lille:

1. Explore the city's charming streets and historic landmarks, such as the Grand Place, the Vieille Bourse, and the Cathedral of Lille.
2. Visit one of the many museums, such as the Palais des Beaux-Arts de Lille, the Musee d'Art Moderne, d'Art Contemporain et d'Art Brut, or the Musee de l'Hospice Comtesse.
3. Take a stroll along the canal and enjoy the views of the city from the waterfront.
4. Visit the famous Lille Christmas market, which takes place every year in December and features festive stalls, food, and drinks.
5. Go shopping at the city's many boutiques and galleries, or at the large shopping centers such as the Euralille or the Lille Europe.
6. Enjoy a romantic dinner at one of Lille's many excellent restaurants, such as the Michelin-starred Le Comptoir du Relais or the cozy Le Bistrot d'Alsace.
7. Take a day trip to the nearby city of Bruges, Belgium, which is known for its picturesque canals and medieval architecture.
8. Go to a show or concert at the Lille Opera or the Le Zenith concert hall.
9. Visit the nearby town of Roubaix, which is famous for its textile industry and its beautiful Art Deco architecture.
10. Take a walk in the beautiful Parc de la Citadelle, which offers a peaceful escape from the city and features a pond, a botanical garden, and a zoo.
11. Go on a bike ride or a hike in the surrounding countryside, such as the Parc Naturel Regional Nord-Pas de Calais.
12. Visit the famous Lille flea market, which takes place every Sunday and offers a wide range of antiques, vintage items, and second-hand goods.
13. Take a cooking class together and learn how to make some of the region's delicious specialties, such as carbonnade flamande (beef stew) or tarte au maroilles (cheese tart).
14. Go on a wine tasting tour and discover the region's excellent wines, such as the Chablis or the Champagne.
15. Take a hot air balloon ride over the city and enjoy a bird's-eye view of Lille's beautiful architecture and surroundings.

These are just a few ideas to get you started. Lille is a vibrant and romantic city that offers many opportunities for couples to have a great time together. Bon voyage !

<|GPT-4-Turbo Verdict|>

Tie

## D.2. Arena Bench Prompt

**Example 1:** An prompt from Arena Bench requiring a very high level of complex problem-solving skill and reasoning while adhering to real-world application.

Create a flutter app for habit tracking that tracks daily habits for a user. The user should be able create multiple daily tasks that he wants to track. And he should be able to group the habits into a set of groups. One the user completes a task, he is rewarded a number of points per task. The app should have a page that summarize the total score per group. And the aggregate score of all the groups of habits. This flutter app needs to be compilable for both andriod and iOS.

**Example 2:** An prompt from Arena Bench requiring a very high level of complex problem-solving skill and reasoning while adhering to real-world application.

I want to set up a remote raspberry pi zero, powered by a solar panel with simple wiring. I want to power a small 2W pump, a simple electet microphone, a custom python script running on the raspberry pi that is used to classify audio detected by the microphone. What components will I need to optimise for cost and minimise any electrical work (e.g. soldering)? What size solar panel will I need to power this whole system?

### D.3. Arena Bench System Prompt

The novel evaluation procedure is as follow: we prompt GPT-4-Turbo with the system prompt displayed below alongside a user prompt, a reference answer, and 2 assistant's answers. For reference answer, we present the user prompt with 3 assistants' answers, GPT-4-Turbo, GPT-4-0314, and Claude-1, to GPT-4-Turbo and ask GPT-4-Turbo to generate an answer to the prompt. To ensure consistent pairwise judgment, we set up GPT-3.5-Turbo-0301 as the baseline answer for all models to be compared against. To avoid positional bias, we conduct two judgments per prompt: the first judgment presents the baseline answer as Assistant A while the second judgment presents the baseline answer as Assistant B. In total, we conduct 700 pairwise comparisons between each model against GPT-3.5-Turbo-0301 across 350 user prompts to calculate a win-rate against the baseline. Then we project the win-rate on a scale from 0 to 10 by assigning wins with a score of 10, ties with a score of 5, and losses with a score of 0. Further, we assign a significant win or loss as 3 wins or 3 losses, respectively, and keeping the other verdicts as a single win, loss, or tie. Finally, we calculate the final score by averaging across the wins, losses, and ties.

**<|System Prompt|>**

Please act as an impartial judge and evaluate the quality of the responses provided by two AI assistants to the user prompt displayed below. Your job is to evaluate which assistant's answer is better.

When evaluating the assistants' answers, compare both assistants' answers. You must identify and correct any mistakes or inaccurate information.

Then consider if the assistant's answers are helpful, relevant, and concise. Helpful means the answer correctly responds to the prompt or follows the instructions. Note when user prompt has any ambiguity or more than one interpretation, it is more helpful and appropriate to ask for clarifications or more information from the user than providing an answer based on assumptions. Relevant means all parts of the response closely connect or are appropriate to what is being asked. Concise means the response is clear and not verbose or excessive.

Then consider the creativity and novelty of the assistant's answers when needed. Finally, identify any missing important information in the assistants' answers that would be beneficial to include when responding to the user prompt.

After providing your explanation, you must output only one of the following choices as your final verdict with a label:

1. Assistant A is significantly better: `[[A>>B]]`
2. Assistant A is slightly better: `[[A>B]]`
3. Tie, relatively the same: `[[A=B]]`
4. Assistant B is slightly better: `[[B>A]]`
5. Assistant B is significantly better: `[[B>>A]]`

Example output: "My final verdict is tie: `[[A=B]]`".

#### D.4. Arena Battles with No Ground Truth

**Example 1:** An example of an Arena Battle between GPT-4-Turbo and Llama-2-13B-chat, where depending on different preferences, either answer can be argued as better than the other. While both answers are correct, GPT-4-Turbo's answer is concise and appropriate to the format of multiple choice question. On the other hand, Llama-2-13B-chat provides helpful explanation for the multiple choice answer.

**<|User Prompt|>**

Multiple Choice Question

All plants have both a haploid and a diploid form. As the plant changes from one form to another, this is called:

Multiple choice question.

alternative life cycle

succession

an alternation of generations

**<|GPT-4-Turbo's Answer|>**

an alternation of generations

**<|Llama-2-13B-chat's Answer|>**

The correct answer is: an alternation of generations.

Plants, like other organisms, have different life cycle stages, and they can exist in two different forms: haploid and diploid. The haploid form is a single set of chromosomes, while the diploid form is two sets of chromosomes. As the plant changes from one form to another, this is called an alternation of generations. This process involves the alternation of haploid and diploid phases, and it is a fundamental aspect of the life cycle of plants.

Alternative life cycle and succession are not correct descriptions of this process, as they refer to different concepts. Alternative life cycle refers to the different stages of life an organism may go through, such as larva, pupa, and adult in insects, while succession refers to the process of one species replacing another in a particular environment or niche.

**Example 2:** An example of an Arena Battle between GPT-4-Turbo and Llama-2-13B-chat, where depending on different preferences, either answer can be argued as better than the other.

**<|User Prompt|>**

What are critical elements of a Product Mindset for startup founders?

**<|GPT-4-Turbo's Answer|>**

Adopting a Product Mindset is crucial for startup founders as it helps them focus on creating products that deliver value to customers while achieving business objectives. Here are some critical elements of a Product Mindset for startup founders:

1. Customer Centricity:

- Understand and empathize with your customers' needs, problems, and motivations.

- Continuously gather and incorporate customer feedback into the product development process.
  - Prioritize features and improvements that directly address customer pain points and enhance their experience.
2. Value Proposition:
    - Clearly define the unique value your product offers to the market.
    - Align your product's features and capabilities with its value proposition to ensure it meets the expectations of your target audience.
  3. Problem-Solving Focus:
    - Identify and focus on solving real problems for users rather than just creating features for the sake of innovation.
    - Validate assumptions and hypotheses through experimentation and user testing.
  4. Iterative Development:
    - Embrace an agile development approach, where you build, measure, and learn in iterative cycles.
    - Be open to pivoting or making significant changes based on feedback and market response.
  5. Data-Driven Decision Making:
    - Use data to inform decisions about product development, marketing, and growth strategies.
    - Establish key performance indicators (KPIs) to measure the success of the product and track progress over time.

...

By internalizing these elements, startup founders can develop a strong Product Mindset that guides their decisions and strategies, ultimately leading to a product that customers love and that contributes to the startup's success.

**<|Llama-2-13B-chat's Answer|>**

A product mindset is a way of thinking and approaching product development that is critical for startup founders. Here are some essential elements of a product mindset for startup founders:

1. Customer-centricity: A product mindset starts with understanding the customer's needs, desires, and pain points. Founders must be obsessed with their customers and be willing to put in the effort to understand their needs.
2. Iteration and experimentation: A product mindset is all about experimentation and iteration. Founders must be willing to try new things, test hypotheses, and iterate on their product based on customer feedback.
3. Focus on value: A product mindset is focused on delivering value to the customer. Founders must be able to identify and prioritize the most important features and functionality that deliver the most value to their customers.

4. Collaboration: A product mindset requires collaboration between cross-functional teams, including engineering, design, and product management. Founders must be able to work effectively with these teams to develop a product that meets customer needs.

5. Data-driven decision making: A product mindset is data-driven, and founders must be comfortable using data to make informed decisions about their product. This includes analyzing customer feedback, usage data, and other metrics to inform product development.

...

By embracing these elements of a product mindset, startup founders can develop a product that meets customer needs, delivers value, and sets their company up for long-term success.