Leaderboard Hacking: Preference-Based Model Evaluations are Vulnerable to Manipulation

Eve Fleisig
UC Berkeley*

Joachim Baumann
Stanford University

Dylan Hadfield-Menell
MIT

Dirk Hovy
Università Bocconi

Abstract

Arena-style rankings of language models are a widely used evaluation framework in leaderboards like LMArena, research papers, and model evaluation in production. These rankings aim to realistically assess model performance using pairwise comparisons of anonymous models on user prompts, aggregating votes via the Bradley-Terry model to produce the final ranking. However, we find that these leaderboards are highly vulnerable to several manipulation strategies that unscrupulous providers could use to boost a model's rank: *strategic voting* (individual votes that boost a model's rank, including votes on irrelevant models), *strategic prompting* (choosing prompts that favor a model), and *strategic nomination* (boosting the rank of a model by inserting irrelevant models). We demonstrate these effects on public leaderboard data, analyze the circumstances that cause these vulnerabilities, and describe approaches to safeguard against each attack. Notably, these issues hold for *any* pairwise preference-based evaluation that uses the Bradley-Terry model. These manipulations boost the rank of nearly every tested model; each attack can boost a model by up to 15-21 places in a ranking and, combined, they can boost a model by up to 45 places.

1 Introduction

How can we compare the performance of large language models (LLMs)? To address concerns that static benchmarks do not evaluate real-world usage, arena-style evaluations offer a more flexible alternative: users post a prompt of their choice and indicate their preference between two anonymous models' responses. These pairwise preferences are then aggregated and transformed into rankings on prominent online leaderboards, such as Artificial Analysis (`artificialanalysis.ai`), DesignArena (`designarena.ai`), and LMArena (`lmarena.ai`). However, as these preference-based rankings gain influence—they affect users' decisions on which model to use, company publicity, funding decisions, and ultimately model developers' revenue—incentives increase to win these competitions even by unscrupulous means. We consider: how much, and how easily, could a bad actor fraudulently boost a model's rank?

Most preference-based evaluations transform a set of pairwise preferences into a unified ranking using the Bradley-Terry model, which implicitly reduces to the Borda count [1], a type of rank-choice voting with known vulnerabilities. Because of this, we hypothesized that known weaknesses of the Borda count also hold for LLM rankings that use Bradley-Terry. We test this hypothesis and indeed find three simple strategies that allow fraudulent actors to boost the rankings of an arbitrary *target model* A.

In *strategic voting*, users change their votes on competitor models to boost A's rank. Because the strength of A's competitors affects A's ranking, even strategic votes on matches that do not involve A can influence the ranking in its favor. In *strategic nomination*, users insert copies of irrelevant models to improve the rank of A. By inserting copies of models against which A does particularly well

*Corresponding author: `efleisig@berkeley.edu`

Preprint.

relative to its competitors, this method boosts the target model's ranking compared to other models. Finally, in *strategic prompting*, users provide prompts that favor A, e.g., prompts in a language on which A excels. Because win rates vary widely when splitting the preference data into subsets, A's rank can be boosted by prompting in any definable subcategory where A does well. As a corollary, we find that it is surprisingly easy to identify which models are presented, even though anonymity is one of the cornerstones of these evaluations.

The methods we find for manipulating model rankings are easy to implement, and can be combined to further magnify the boosting effect. We describe each method in detail and demonstrate their effects on publicly available data from LMArena. Our contributions are: (1) we demonstrate the effect of strategic voting, nomination, and prompting on preference rankings and link to the social choice literature (equivalence to Borda count); (2) we introduce an efficient strategic voting heuristic (that does not require inefficient Elo recomputations); and (3) we propose defenses against each attack.

2 Background and Related Work

2.1 Preference-based evaluation and its vulnerabilities

The Bradley-Terry model (BT) estimates the relative strength of items based on individual preferences between pairs of items. Let (A, B) denote a comparison between candidates A and B, and let $A \succ B$ denote that A is preferred to B: BT assumes that human preferences obey $P_{BT}(A \succ B) = \frac{e^{\xi_A}}{e^{\xi_A}+e^{\xi_B}}$.

The coefficients ξ serve as scores for the competitors (i.e., models on a leaderboard). Boubdir et al. [2] and Bai et al. [3] discuss using preference-based evaluations for language models. Chiang et al. [4] build on this to rank LLMs from user preferences; they estimate BT scores efficiently using maximum likelihood estimation on logistic regression, which we follow in this work.[2]

Singh et al. [5] raise issues regarding the reliability of LMArena's rankings due to pre-release testing that results in uneven data access for model providers, uneven sampling, and silent model deprecation; they argue that these differences benefit some model providers over others.[3]. In contrast, the evaluation vulnerabilities we discuss here are not limited to any particular leaderboard. Leaderboards designed impartially and in good faith remain vulnerable to the issues we present so long as they rely on the Bradley-Terry model.

Some previous work has explored methods of adversarial attacks on preference-based leaderboards. Zhao et al. [6] and Huang et al. [7] consider strategic voting attacks based on identifying the target model and then voting strategically on examples that involve that model. Min et al. [8] expand this to consider strategic voting on a stream of random votes on pairs (B, C) by considering what a target model's Elo score would be after voting in favor of B or C. Relative to this work, our contributions are as follows:

- We introduce two completely new attacks: strategic nomination and strategic prompting.
- We introduce an efficient strategic voting heuristic, with no inefficient Elo recomputation per vote.
- We explain why strategic voting, nomination, and prompting are effective by linking to the social choice literature (equivalence to Borda count).
- We show that the new attacks we discuss result in large boosts to many models' rankings.
- We propose defenses against each attack.

2.2 Parallels to voting vulnerabilities

Siththaranjan et al. [1] found that evaluations using the Bradley-Terry model implicitly optimize for the Borda count, a positional voting rule in which candidates in an election are scored based on how many candidates are ranked below them. In the traditional Borda count formulation, for an election with n candidates, each voter ranks every candidate in order of preferences. The first-place candidate for that voter receives n points; second place $n-1$, etc. These points are summed across voters, and candidates are ranked in order of points. This is equivalent to ranking candidates in

[2]On LMArena, these scores are sometimes referred to as Elo scores. The Elo model is a linear scaling of the Bradley-Terry model, but more recent comparisons are weighted more heavily when fitting the model (unlike Bradley-Terry, which weights comparisons equally regardless of when they occurred). However, the current LMArena implementation uses Bradley-Terry scaled to Elo's scoring range (models are assumed to be static), not Elo; see `github.com/lmarena/arena-rank`.

[3]The extent of these concerns is debated; see `lmarena.ai/blog/our-response`

order of the average probability (over voters) that the candidate is preferred to other candidates in the election. Siththaranjan et al. [1] find that any utility function that optimizes for the BT objective implicitly aggregates individual voters' utilities according to the Borda count. However, in the social choice literature, the Borda count is known to be vulnerable to several kinds of strategic manipulation [9]. Thus, we hypothesize that **preference-based leaderboards that implicitly optimize for the Borda count inherit these weaknesses.** In particular, we hypothesize that one can manipulate these leaderboards to raise a model A's rank relative to a model B using:

Strategic voting. Suppose that A and B are the top two candidates for an election, and supporters of a candidate A wish for it to win over candidate B. Under the Borda voting rule, they can use *strategic voting*, in which supporters vote that they prefer a third candidate $C \succ B$ in order for A to win (even though they truly believe $B \succ C$). In LLM leaderboards, users can raise A's rank over B by strategically voting on comparisons with unrelated models. They can also abstain from votes on pairings that do not involve A even though they do truly have a preference, in order to increase the fraction of votes in which A wins. In practice, strategic voting requires deanonymizing models in order to vote in favor of A.

Strategic nomination. Fraudulent actors can also engage in *strategic nomination*, by introducing a candidate C such that $A \succ C \succ B$. For voters who initially preferred $A \succ B$, their new ranking is $A \succ C \succ B$, lowering B in the ranking; for voters who initially preferred $B \succ A$, their new ranking is $B \succ A \succ C$, leaving A's placement unchanged. Repeating this process with many candidates C can substantially lower B's Borda score relative to A. In elections, this is most often discussed in the context of cloning, in which C is very similar to A but slightly worse [9]. Users can add clones A' of a model A, or copies of unrelated models C, in ways that raise model A's rank relative to B. If the voters simply want one of the models that resembles A to win, we may also relax the restriction that A' be similar to A: this means that it is to a faction's advantage to nominate many similar candidates. More broadly, inserting copies of any model that usually wins over B but loses to A will improve A's rank.

Strategic prompting. Finally, LLMs offer a domain-specific way to raise model A's rank by exploiting the fact that Borda count aggregates over all votes evenly: *strategic prompting*. If voters know that there is a particular definable area in which A excels relative to other models, they can simply choose prompts in that area to explicitly favor A over B. This takes advantage of high variance in model win rates between different areas, defined broadly: e.g., different skills (coding vs. writing), or different languages (English vs. Korean vs. Amharic). For example, if model A does very well in Amharic but other models cannot process it, a voter can simply always prompt the model in Amharic.

We can consider strategic prompting as a type of strategic voting specific to language models. Live strategic voting requires deanonymizing models in order to know which model to vote for. Instead, strategic prompting uses subcategory performance as a proxy for model identification: if we expect that model A typically performs well on prompts in a subcategory, then prompting in that subcategory and always voting for the better model is a strategy that makes it more likely to vote in favor of A.

For each of the three attacks, we consider how feasible, likely, and preventable the attack is along several axes, leading to three research questions:

1. How many strategic votes, prompts, or inserted models are needed to shift a ranking?
2. How easy is it to gather the information required (e.g., identify anonymized models, create a suitable model clone, or determine which way to vote strategically)?
3. How can leaderboard developers secure their evaluations against these attacks?

3 Methods

To examine the effects of leaderboard manipulation on real data, we conduct our experiments on a public leaderboard dataset of over 100,000 human preferences from LMArena.[4]

3.1 Strategic Voting

We consider two strategic voting attacks. Both strategies take advantage of the fact that Borda count and Bradley-Terry lack independence of irrelevant alternatives: votes on model pairs that do not involve the model A can nonetheless boost A's rank.

[4]The data is at `huggingface.co/datasets/lmarena-ai/arena-human-preference-100k` under a CC-BY-4.0 license.

Greedy Best-Vote We consider an attack in which the manipulator votes strategically on a stream of new comparisons between random pairs of models (many of which do not involve A). We see that votes on unrelated models do shift A's Elo score, since they shift the scores of A's competitors. For this attack, we run a greedy best-strategic-vote algorithm: given a set of models S that includes a target model A to move up in the rankings, and a stream of n new comparisons to perform between any pair of models $(B, C) \in S$, check whether voting for model B or model C results in the best improvement for model A.[5] Then, calculate the improvement for A under n strategic votes. For these experiments, we set n to be at most 10K votes ($\sim$10% of votes in the dataset).

Min et al. [8] explore a greedy best-vote strategy in prior work (for each new preference pair, vote for the choice that best improves the target model's Elo). However, the greedy best-vote strategy is computationally expensive: it requires recalculating Elo scores after every vote.

Heuristic Best-Vote Instead, we introduce a simple heuristic that allows for strategic voting with far lower computational cost—no live Elo computations required. In initial experiments on the greedy best-vote strategy, we observe that flipping the winner of a preference is typically optimal for boosting a model's rank. Based on this, we implement a simple voting rule for a pair (B, C), where the voter's true preference is $B \succ C$: If $A \in \{B, C\}$, vote for A; if not, vote for the worse model C.

Identifying models in order to vote strategically. In practice, strategic voting requires the manipulator to *know* which pair of models they are voting on. Extensive prior work focuses on detecting LLM-generated text, typically by training classifiers to distinguish human-written from model-generated content or to attribute text to a specific model [10, 11, 12]. Our setting is much simpler, since the evaluator has full control over the prompt. The simplest identification strategy is to prompt the model to reveal its own identity. However, LMArena filters out responses that include the model name [7]. We therefore consider an approach that is undetectable with output filtering. For open-weight models, we use a likelihood-based approach. Given a prompt-response pair randomly sampled from the LMArena data,[4] we compute the average token-level log-likelihood of the response for a set of candidate models by measuring the probability assigned to each next answer token conditioned on all previous tokens. We then identify models based on the highest average log-probability. For proprietary API-based models, where token-level log-probabilities are unavailable, we use an even simpler approach. We generate a single response for a candidate model and score similarity to the observed response using character-level sequence similarity.

3.2 Strategic Nomination

In strategic nomination, the objective is to find a model C for which adding copies of C boosts A's ranking. To do so, we (1) identify the model C that best benefits model A, and (2) add new copies C' whose performance is assumed to be identical to C (thus, its outcomes in battles with existing models are likely the same). Computing the exact solution for which model to add is cheap, since there are only 55 candidate models in our dataset (and about 300 on the current live version of LMArena). For each target model A, we find the model C for which adding n copies of model C and its battles results in the best rank improvement for A; then, we add n copies of model C and recompute model A's rank. For these experiments, we set n at $\leq$10K votes ($\sim$10% of votes in the dataset).

3.3 Strategic Prompting

We consider how to manipulate votes simply by restricting prompts to a subset on which model A outperforms other models. As proof-of-concept, we use language to define our prompt subsets, for several reasons: the language of the prompt is readily available categorical data in the LMArena dataset, it is easy for a manipulator to restrict their prompts to that subset, and it is easy to create new prompts within any given subset by simply prompting in that language. However, we note that any easy-to-define split under which A performs well relative to other models would work here (e.g., performance on specialized domains); see Appendix I for an example.

For this attack, we first split the data into prompt subsets (in our case, different languages). We identify the language in which model A is ranked highest relative to the other models[6] and insert n random battles with known results in that language (i.e., copies of existing battles). For these

[5] We initially experimented with allowing ties, but found that voting for one winner was almost always better at shifting model ranks; thus, we restrict the options considered by this strategy to voting for one winner, reducing computational cost.

[6] We restrict to languages with at least 20 battles.

experiments, we set n to be at most 10K votes (approximately 10% of votes in the dataset). As a side effect, we note that strategic prompting also poisons later evaluations that use the prompt set: e.g., if prompts from an arena are later used for offline evaluation of new models, but strategic prompting in favor of a model A was used on evaluations on the arena, then those later evaluations will also favor models similar to A. (This issue does not apply to strategic voting or nomination.)

4 Results

4.1 Strategic Voting

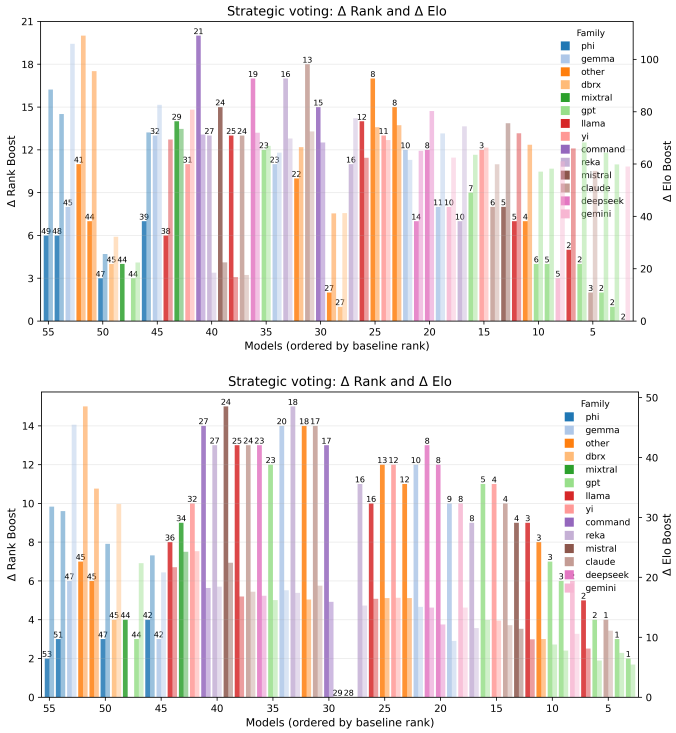

Figure 1: *Top:* Highest model rank (dark bars) and Elo (light bars) achieved under **greedy** strategic voting, capped at 10K votes ($\sim$ 10% of total votes). Numbers above bars indicate the best absolute rank that a model achieves. Models are boosted up to 20 places in the rankings; 23 models (42% of the dataset) move over 10 places up, and eight models move into the top 5. *Bottom:* Highest model rank & Elo achieved under **heuristic** strategic voting, under same conditions. Models are boosted up to 15 places in the rankings; 21 models (38% of the dataset) move over 10 places up, and 15 models move into the top 5.

Strategic Voting (Greedy Strategy) Adding strategic votes substantially boosts target models' ranks (Figure 1, top). We find that **models are boosted up to 20 places** in the ranking using this method, and that every tested model below 2nd place can move up at least one place. Boosting a model higher in the rankings requires more votes (Figure 6), and models close to the very top of the leaderboard (2nd-4th place) are harder to move than models further down, in part because they have participated in more battles. However, 23 models (42% of the dataset) move over ten places upwards using this method, and rankings near the top of the leaderboard are still swayed relatively easily: 11 models can move into the top 10 places, and eight models can move into the top 5 places.

Strategic Voting (Heuristic) The strategic voting heuristic also substantially boosts target models' ranks (Figure 1, bottom). Models are **boosted up to 15 places**, and 21 models move over ten places upward. This method is particularly effective for models high up in the ranking, moving fifteen models into the top 5 places. This suggests that, while the greedy voting strategy results in the largest boost for the most models, the heuristic strategy results in larger boosts for models near the top of the rankings, at a far lower computational cost.

Figure 6 (Appendix D) compares the number of votes needed under each strategy to boost a model's rank. Small improvements to rank (2-3 places) are relatively easy to achieve, requiring under 2% of total votes (2K out of 100K total) to be manipulated and achievable by a relatively large subset of models. Larger improvements (15+ places) are achievable by a smaller subset of models with more manipulated votes. We find that while the maximum rank boost achieved across models is higher for the greedy strategy, more models are able to achieve smaller boosts under the heuristic strategy (and typically with fewer votes). We also find that, in practice, **the identification methods for open-weight and proprietary models suffice for strategic voting** (Appendix E). Likelihood-based attribution identifies open-weight models with high precision ($> 80\%$), and similarity-based scoring for API-access models performs above chance (>40-50%).

4.2 Strategic Nomination

Strategic nomination **boosts model ranks by up to 15 places** (Figure 2), though boosts are typically more moderate (1-7 places). Curiously, the effect of strategic nominations varies widely by model; for example, the 19th place model jumps to 4th place, whereas other models with similar rankings display more moderate effects. We examine these factors in more detail in Section 4.5.

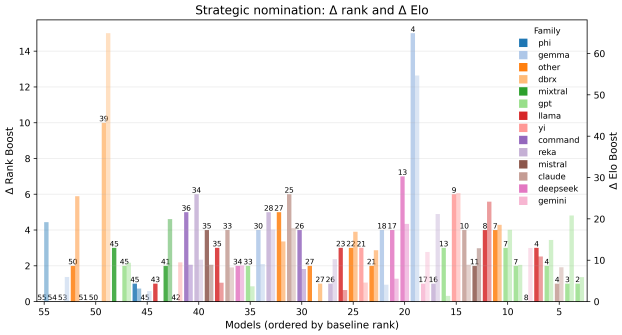

Figure 2: Change in model rank and Elo achieved when adding strategic nominations (10 copies of the most favorable model). Change in rank is the left-hand bar of each double-bar; change in Elo is the right-hand (lighter) bar. Numbers above bars indicate the best absolute rank achieved by the model (1=best). For example, the model originally in 19th place (light blue) jumps 15 places to 4th place.

Nomination effects in current leaderboards. We examine whether these effects might already affect current leaderboards by examining a counterfactual (Figures 7 and 8, Appendix D): what would a model's rank be if a particular family of models were included or excluded? A more stable ranking would be immune to irrelevant alternatives, i.e., in a set of models S, A's rank relative to $S \setminus B$ should not depend on whether B is included. For each model A, we find the family B of models whose removal increases A's rank the most, and the family that decreases A's rank the most. We also calculate A's rank in the full (original) dataset, skipping members of family B for its ranking. We compare each model A's rank among $S \setminus B$ when family B is excluded from the Elo calculation, versus its rank among $S \setminus B$ when family B is included. We thus measure whether the ranking among a set of models is affected by the inclusion of unrelated models.

Effects are small but consistent: **including a model family B can shift a model's rank among $S \setminus B$ up to 1-4 places in a desired direction (up or down)**. Claude models tend to have the strongest effect on model ranks, causing the most movement downwards for 12 models and upwards for 10 models: this is likely because the strongest Claude model participates in more battles than any other model (14,000, compared to 10,000 for the model with the second-most battles). Several models are also boosted by the inclusion of models from their own family: the inclusion of other Gemma models moves `gemma-2-9b-it-simpo` up two places, and the inclusion of other Claude models moves Claude 3 Sonnet up two places as well. Furthermore, **42% of models among $S \setminus B$ are boosted up one or more places in the ranking due to a single model**. Claude 3.5 Sonnet, the model involved in the most battles, has the strongest effects: it moves six models up 1-3 ranks when it is included compared to when it is excluded.

4.3 Strategic Prompting

Strategic prompting boosts model ranks almost as effectively as strategic voting: models move up to 15 places upward in the rankings, including moving several models up to the top 5 models (Figure 3). For example, `reka-core-20240501` moves from 33rd place to 18th place by prompting in Hungarian, `llama-3.1-8b-instruct` moves from 38th place to 23rd place by prompting in Croatian; and `phi-3-medium-4k-instruct` moves from 46th place to 32nd place by prompting in Scots. This method is particularly effective with small numbers of prompts, moving model ranks with as few as 500 strategic prompts added (Figure 9, Appendix D).

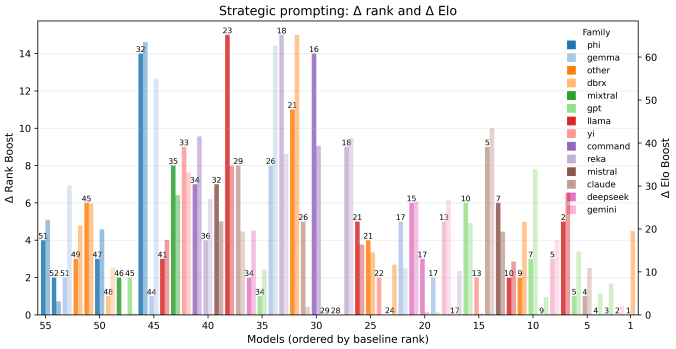

Figure 3: Change in model rank and Elo achieved by adding strategic prompts, capped at 10K prompts ($\sim 10\%$ of total battles). Models are boosted up to 15 places in the ranking.

In principle, leaderboard developers could spot and filter out unusual clusters of prompts in a language. However, the use of "language" as the axis for strategic prompting is arbitrary. This strategy works for any definable subcategories, meaning it is easy to pick subcategories that are harder to spot. As a demonstration (Appendix I), we repeated the prompt experiments with prompts are split into subcategories based on whether they involve complexity, creativity, domain knowledge, and/or math.[7] Under this version of strategic prompting, we see rank boosts of up to 14 places, comparable to those from strategic prompting using language for prompt subcategories.

4.4 Combined Nomination and Prompt Attacks

We examine the effects of combining strategic nomination and prompting (Figure 4). To combine the attacks, for each model, we follow the strategic nomination strategy to add copies of the most favorable model; then, we run the strategic prompting algorithm on the dataset with those model copies added. We find that **models are boosted up to 45 places in the ranking**, with 9 models moving over 10 places upwards.

[7]These are preexisting LMArena labels, defined by having a model classify whether a prompt has each of these traits.

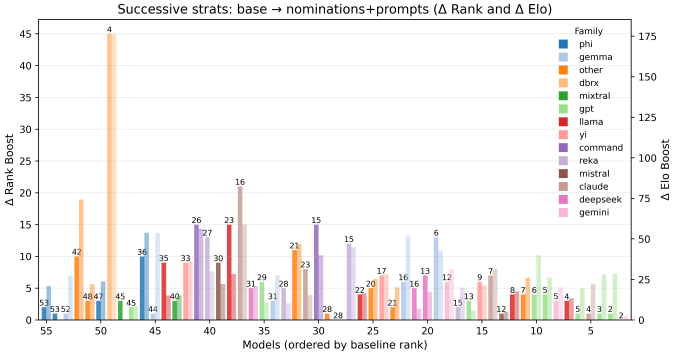

Figure 4: Change in rank and Elo when performing strategic nominations followed by strategic prompting. Models are boosted up to 45 places in the rankings, with 9 models moving over 10 places upwards.

4.5 Factors Contributing to Ease of Manipulation

The results across strategic voting, nomination, and prompting indicate that **some models are much easier to move than others**. For example, whereas a few models only move 1-2 ranks under all strategies, `dbrx-instruct-preview` moves from 49th to 39th place with strategic nominations; `command-r` moves from 41st to 21st place under strategic voting; and `reka-core-20240501` moves from 33rd to 18th place under strategic prompting.

We run regressions to understand what factors make it easier to manipulate a model's rank (Appendix F). The factors we consider are A's variance in win rate against different models; A's starting rank and Elo; the number of models in A's model family; A's win rate, gap in Elo, and total battles against the model one rank higher; the number of battles in A's best language for strategic prompting, and A's number of total battles and tied battles. To capture if the model could easily swap places with a model close above it, we define the "Elo density" as the number of models with Elo 0-50 points higher than A. For strategic voting (R^2=0.660), the top feature is Elo density, followed by win rate variance. For strategic prompting (R^2=0.518), the top features are the number of battles in the best prompting language and the gap in Elo score vs. the model one rank higher. For strategic nomination (R^2=0.557), the top features are Elo density and gap in Elo score vs. the model one rank higher. These features suggest that models placed below a cluster of models with slightly higher Elo scores are easier to shift in the rankings, since smaller changes in Elo result in larger changes in model rank.

5 Attack Feasibility

Table 1: Comparison of attack strategies and their properties.

	Voting (greedy) *also see [8]*	**Voting (heuristic)**	**Prompting**	**Nomination**	**Prompting + nomination**
Max rank boost	20 places ↑	15 places ↑	15 places ↑	15 places ↑	45 places ↑
Must identify models?	All models	Only target model A	–	–	–
Cost of Elo calculations (m models, n votes; $n \gg m$)	$O(n)$	0	$\leq O(m)$; depends on desired # of prompt subsets	$O(m)$	$O(m)$
Possible defenses	Block bots & prompting w/o voting	Block bots & prompting w/o voting	Stratify/cluster prompts	Harmonic Borda count	Harmonic Borda count
	Harmonic Borda count	Harmonic Borda count			Stratify/cluster prompts

All attacks we describe are highly feasible, as shown in Table 1. First, identifying the best way to boost a target model A is computationally cheap. Strategic nomination for a target model requires iterating once over a set of candidate models (55 in our dataset; 300 on the live version of LMArena). Strategic prompting requires iterating over a set of candidate subsets ($\sim$40 languages in our dataset; easy to restrict to the top n languages on live data). Optimally computing which model to nominate or language to prompt in is a one-time computation per model; computing it for all 55 models in this dataset takes less than 5 minutes on one CPU. Greedy strategic voting on a random stream is more expensive (for each vote, the change in Elo if one model or the other were selected must be calculated), but using the heuristic strategy is much cheaper and the cost per vote does not depend on the number of models or battles in the dataset. In addition, as we have shown, identifying the models in an anonymized battle is relatively straightforward via multiple methods.

How do the manipulations scale as dataset size increases? We expect our manipulation effects to be feasible for LMArena's leaderboards that are similar in number of votes ($\sim$100K) to our dataset (e.g., code and search). However, LMArena's text leaderboard has $\sim$5 million votes; we expect that a higher absolute number of manipulated votes/prompts/nominations would be necessary to change the text rankings (limitations discussed further in Appendix G). Our scaling experiments in Appendix H show that the number of manipulations needed to boost a model **scales at most linearly** with dataset size. Though larger arenas are more secure, there is no exponential advantage to a larger leaderboard.

6 Leaderboard Defenses

While leaderboards currently implement some mitigations against strategic voting, such as bot detection [7], we do not know of any protections against strategic nomination (in which the attack is carried out by those submitting models, not those voting) or strategic prompting (in which users' votes indeed align with their "true" preferences, but the prompts are skewed to favor the target model). We also find that properties of how the leaderboards are currently designed (e.g., ease of prompting to deanonymize models; ability to abstain from voting) facilitate the attacks we describe. Bearing in mind the potential impacts of studying leaderboard attacks (Appendix C), we discuss operational and algorithmic defenses to protect against each attack we demonstrate:

Strategic voting defenses. The ease of identifying models facilitates strategic voting. Since models are still readily identifiable through likelihood-based attribution, one defense is to check for voters who exhibit suspicious behavior, e.g. votes that always make one model's ranking improve on the leaderboard, since a true stream of votes on unrelated models should have mixed effects. This might require voters to be logged in and verified, to prevent rapid, anonymous manipulated votes from a single person. Strategic voting would then require many voters to coordinate, which would make strategic voting significantly more difficult. Finally, strategic voting is easiest when attackers need only vote on comparisons that involve the target model, which makes the strategic votes have a more concentrated effect. Thus, leaderboards should prevent users from viewing many comparisons without voting on any of them. Strategic voting still remains difficult to detect if spread across a user group or if a user only manipulates a subset of their votes.

Strategic nomination defenses. Unlike strategic voting, strategic nomination involves manipulation by those submitting models to the leaderboard (rather than those evaluating them). A simple defense against this method is to prevent the submission of many similar models at once. A more robust defense to both strategic voting and strategic nomination is to change the aggregation function of the Bradley-Terry model to put less weight on a voter V's preferences among models that V ranks poorly overall. This solution draws on the solution to the parallel problem in the social choice literature: whereas the Borda count (equivalent to the Bradley-Terry objective) gives $n - r$ points to the model at rank r in a list of length n (+1 point for each step it goes up in a voter's ranking), we can instead use the Dowdall or *harmonic* Borda count, which gives $\frac{1}{r}$ points to the model at rank r. In real election settings, this change hampers strategic voting and makes strategic nomination require enough new nominations as to be broadly infeasible.[8] We propose a BT-compatible implementation of harmonic Borda in Appendix A and find it reduces strategic nomination's effect on boosting Elo scores.

Strategic prompting defenses. Strategic prompting allows individual votes to correctly align with broader preferences on that comparison, but shifts the prompt distribution so more prompts favorable to a particular model come up. When comparisons are clustered, model results become skewed. Approaches to address this could involve stratifying or clustering prompts and then weighting these groupings evenly so that overall performance is not unduly influenced by any single grouping (e.g., prompt language). As a theoretical contribution in Appendix B, we propose a modified version of Bradley-Terry that replaces individual scores with kernel-smoothed scores to make it more difficult to use clusters of comparisons to manipulate model rankings.

7 Conclusion

As model evaluations become targets for highly incentivized actors, such as model providers who benefit from improving their models' perceived performance, the likelihood and potential impact of ranking manipulations increases. We drew upon known weaknesses of the Borda count in social choice, which translates to the Bradley-Terry model used in most LLM leaderboards, and tested the equivalents of strategic voting, strategic nomination, and strategic prompting on LMArena data. We found that preference-based model rankings are vulnerable to all three attacks. Concerningly, these effects can move models substantially in the rankings and are relatively simple, cheap, and fast to compute and implement. We discussed defenses for each manipulation technique, including interface changes and variants of current aggregation techniques. Our findings highlight the need to secure widely used leaderboards against such attacks to ensure trust in preference-based model rankings.

[8]The government of Kiribati uses classic Borda count for some elections; Nauru uses the harmonic Dowdall variant. Reilly [13] finds that strategic voting and nomination are both documented (and effective) in Kiribati, but he and Fraenkel and Grofman [14] find that while the same strategies are used in Nauru, they seem less effective than under classic Borda count.

References

[1] Anand Siththaranjan, Cassidy Laidlaw, and Dylan Hadfield-Menell. Distributional preference learning: Understanding and accounting for hidden context in rlhf. In *The Twelfth International Conference on Learning Representations*, 2024.

[2] Meriem Boubdir, Edward Kim, Beyza Ermis, Sara Hooker, and Marzieh Fadaee. Elo uncovered: Robustness and best practices in language model evaluation. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 106135–106161. Curran Associates, Inc., 2024. doi: 10.52202/079017-3367. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/bfba8efb806a970455b83b852c9cf846-Paper-Conference.pdf.

[3] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. Training a helpful and harmless assistant with reinforcement learning from human feedback, 2022. URL https://arxiv.org/abs/2204.05862.

[4] Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios N. Angelopoulos, Tianle Li, Dacheng Li, Banghua Zhu, Hao Zhang, Michael I. Jordan, Joseph E. Gonzalez, and Ion Stoica. Chatbot arena: an open platform for evaluating llms by human preference. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

[5] Shivalika Singh, Yiyang Nan, Alex Wang, Daniel D'Souza, Sayash Kapoor, Ahmet Üstün, Sanmi Koyejo, Yuntian Deng, Shayne Longpre, Noah A. Smith, Beyza Ermis, Marzieh Fadaee, and Sara Hooker. The leaderboard illusion, 2025. URL https://arxiv.org/abs/2504.20879.

[6] Wenting Zhao, Alexander M Rush, and Tanya Goyal. Challenges in trustworthy human evaluation of chatbots. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 3359–3365, 2025.

[7] Yangsibo Huang, Milad Nasr, Anastasios Angelopoulos, Nicholas Carlini, Wei-Lin Chiang, Christopher A Choquette-Choo, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Ken Ziyu Liu, et al. Exploring and mitigating adversarial manipulation of voting-based leaderboards. *arXiv preprint arXiv:2501.07493*, 2025.

[8] Rui Min, Tianyu Pang, Chao Du, Qian Liu, Minhao Cheng, and Min Lin. Improving your model ranking on chatbot arena by vote rigging. *arXiv preprint arXiv:2501.17858*, 2025.

[9] Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D Procaccia. *Handbook of computational social choice*. Cambridge University Press, 2016.

[10] Adaku Uchendu, Thai Le, Kai Shu, and Dongwon Lee. Authorship attribution for neural text generation. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8384–8395, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.673. URL https://aclanthology.org/2020.emnlp-main.673/.

[11] Vivek Verma, Eve Fleisig, Nicholas Tomlin, and Dan Klein. Ghostbuster: Detecting text ghostwritten by large language models. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1702–1717, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.95. URL https://aclanthology.org/2024.naacl-long.95/.

[12] Vinu Sankar Sadasivan, Aounon Kumar, Sriram Balasubramanian, Wenxiao Wang, and Soheil Feizi. Can AI-generated text be reliably detected? stress testing AI text detectors under various attacks. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=0OgsAZdFOt.

[13] Benjamin Reilly. Social choice in the south seas: Electoral innovation and the borda count in the pacific island countries. *International Political Science Review*, 23(4):355–372, 2002. doi: 10.1177/0192512102023004002. URL https://doi.org/10.1177/0192512102023004002.

[14] Jon Fraenkel and Bernard Grofman. The borda count and its real-world alternatives: Comparing scoring rules in nauru and slovenia. *Australian Journal of Political Science*, 49(2):186–205, 2014. doi: 10.1080/10361146.2014.900530.

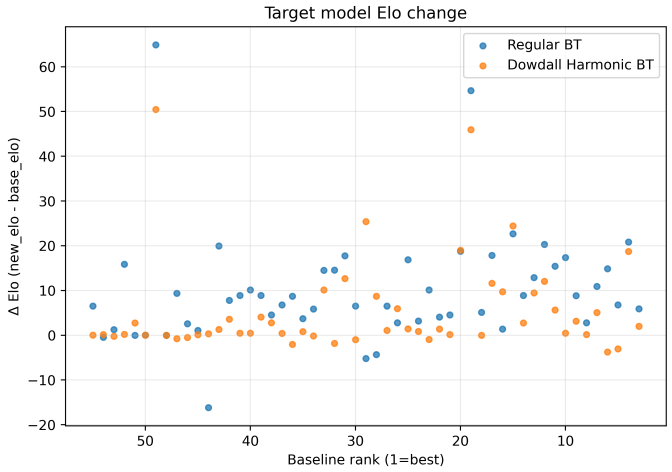

Figure 5: Change in Elo when using the standard Borda count (blue) vs. the harmonically weighted Borda count (orange).

A Harmonic Bradley-Terry Objective

Here, we propose one implementation of a harmonically-weighted Borda count that maintains compatibility with the Bradley-Terry objective. The Borda count gives $n - r$ points to the model at rank r in a list of length n (+1 point for each model it beats in a voter's ranking). The Dowdall or *harmonic* Borda count instead gives $\frac{1}{r}$ points to the model at each voter's rank r.

Finding an equivalence to the harmonic Borda count for Bradley-Terry introduces additional challenges. First, the definition of harmonic Borda hinges on the notion of a complete ranking of all candidates for each voter (so that models can be reweighted based on their rank in that voter's ballot). However, typical preference learning datasets do not contain a complete ranking for each voter; only some pairwise preferences are known per voter.

To account for this, we instead interpret the harmonic Borda to use the ranking for *voter sets* instead of individual voters. We require voter sets to be large enough to contain preferences involving all models in the dataset. That is: for a set of models $m_1 \dots m_n$ and preferences (m_i, m_j), each voter defines a comparison graph with models as the vertices and the voter's preferences as the edges. Each voter set, consisting of the union of voters' edges, must be a connected comparison graph with vertices $m_1 \dots m_n$. We partition the judges into clusters that maximize the number of clusters while keeping the comparison graphs connected.

For each voter set v, we calculate the ranking among those voters using the standard BT objective. Let $r(m, v)$ be the rank of model m for voter set v. We assign the weights $w_{m,v} = \frac{1}{r(m,v)}$, which adds more weight to wins by models that perform well within that voter set. We interpolate between the standard BT objective (all weights are 1) and the harmonic one using $w_{m,v,\beta} = (1 - \beta) + \beta w_{m,v}$.

At $\beta = 0$, every battle gets weight 1 (recovering standard BT); at $\beta = 1$, battles are weighted only by the harmonic weight. This permits smooth tuning between the standard and harmonic BT objectives. We then reweight the matrix of pairwise wins by $w_{m,v,\beta}$ and aggregate using the standard BT objective.

We experiment with $\beta = [0.2, 0.5, 1]$, finding that $\beta = 1$ performs best, and the maximum number of voter sets (145 for this dataset). We find that the harmonically-reweighted BT **reduces sensitivity to strategic nomination**: whereas standard Borda count permits a mean change in Elo of +9.98 points, the harmonic variant reduces this to +5.50 points, cutting the strategic nomination effect nearly in half (Figure 5). Similarly, it reduces the mean boost in rank from 3 places to 1.5 places.

Studying the effects of the harmonic Borda count on strategic prompting, as well as a more detailed understanding of how it may change rankings overall in LLM settings, remains an important area for future work.

B Density-smoothed Bradley–Terry / Elo for LM-arena-style model rankings

This note describes a simple modification of standard Bradley–Terry (BT) / Elo fitting for pairwise preference data. The modification replaces point scores with kernel-smoothed (KDE-like) "regional mass" scores. The intent is to make repeated or highly clustered comparisons produce diminishing returns by coupling similar prompt–response instances through a similarity kernel.

We focus on model-level Elo parameters (one scalar per model), using LM-arena / LMR-style data: each comparison is between two model responses to a prompt.

B.1 Standard Bradley–Terry / Elo (model-level)

Data and notation. A dataset D consists of N pairwise comparisons. Each comparison i has:

- prompt p_i,
- model m_i^A producing response r_i^A,
- model m_i^B producing response r_i^B,
- label $y_i \in \{0, 1\}$, where $y_i = 1$ means A is preferred to B.

We write the row as $(p_i, m_i^A, r_i^A;\ m_i^B, r_i^B;\ y_i)$.

In model-level Elo/BT, the learned parameter is one scalar per model:

$$\theta_m \in \mathbb{R}, \qquad m \in \{1, \dots, M\}.$$

Define the positive model strength

$$w_m \ = \ \exp(\theta_m).$$

Choice model. Standard Bradley–Terry assumes the probability that A beats B depends only on the two models:

$$P_i^{BT} := \Pr(m_i^A \succ m_i^B) = \frac{w_{m_i^A}}{w_{m_i^A} + w_{m_i^B}} \tag{1}$$

$$= \sigma\big(\theta_{m_i^A} - \theta_{m_i^B}\big), \tag{2}$$

where $\sigma(t) = 1/(1 + e^{-t})$ is the logistic function.

Objective. The standard regularized negative log-likelihood is

$$L_{BT}(\theta) = -\sum_{i=1}^{N} \Big[y_i \log P_i^{BT} + (1 - y_i) \log\big(1 - P_i^{BT}\big) \Big] \tag{3}$$

$$+ \frac{\lambda}{2} \sum_{m=1}^{M} \theta_m^2. \tag{4}$$

The fitted θ values are then reported (after an affine rescaling) as Elo.

Why repetition has linear leverage. In standard BT/Elo, each match depends only on $(\theta_{m_i^A}, \theta_{m_i^B})$. Repeating the same comparison adds the same loss term again; influence is essentially linear in the amount of repetition (until regularization or finite-data effects dominate).

B.2 Density-smoothed (kernel) Bradley–Terry / Elo

Key change: "items" are prompt–response instances. Define the two *items* compared in row i as the prompt–response pairs

$$x_i^A = (p_i, r_i^A), \qquad x_i^B = (p_i, r_i^B).$$

Let $\mathcal{I}$ be the set of all unique items $x = (p, r)$ that appear in the dataset, and let $T = |\mathcal{I}|$. Each item $x_t \in \mathcal{I}$ is produced by some model $m(t)$.

Kernel on embeddings. Define a similarity kernel between items

$$K(x, z) \geq 0,$$

where x and z are prompt–response pairs. Intuitively, K is large when two prompt–response pairs are similar (e.g., same/similar prompt and similar response) and small otherwise. Practically, K would be computed from embeddings of prompts/responses or a joint (prompt+response) embedding.

Kernel-smoothed (regional mass) score. Define the kernel-smoothed score (regional mass) for item x as

$$\tilde{s}(x; \theta) = \sum_{z \in \mathcal{I}_{multiset}} \exp(\theta_{m(z)}) K(x, z). \tag{5}$$

Here $\mathcal{I}_{\text{multiset}}$ is the multiset of items as they appear in the dataset: if an item z appears multiple times across comparisons (on either side), it appears multiple times in the sum.

BT probability, but using smoothed scores. For comparison i, we have

$$P_i^{KBT} := \Pr(x_i^A \succ x_i^B) = \frac{\tilde{s}(x_i^A; \theta)}{\tilde{s}(x_i^A; \theta) + \tilde{s}(x_i^B; \theta)}. \tag{6}$$

This has the same algebraic form as BT, but the scores are kernel-smoothed sums over (prompt, response) instances in the dataset.

Objective. Use the same Bernoulli log-loss form:

$$L_{KBT}(\theta) = -\sum_{i=1}^{N} \Big[y_i \log P_i^{KBT} \tag{7}$$

$$+ (1 - y_i) \log \left(1 - P_i^{KBT}\right) \Big] \tag{8}$$

$$+ \frac{\lambda}{2} \sum_{m=1}^{M} \theta_m^2. \tag{9}$$

Qualitative difference from standard BT. In standard BT, each match uses $(w_{m_i^A}, w_{m_i^B})$ directly. In kernel-smoothed BT, each match uses KDE-like regional masses that sum model scores over *similar* prompt–response items across the dataset. Therefore, adding (or repeating) a comparison changes not only the loss by adding another term, but also changes the *scores used by other terms* via the kernel sums. Evidence tends to accumulate at the level of regions of similarity, rather than enabling arbitrarily large separation among near-duplicate items within the same region.

Implementation simplification for per-model θ. Because $\text{score}(z)$ depends only on the model identity of z, each smoothed score is a fixed linear combination of $\exp(\theta_m)$:

$$w_m = \exp(\theta_m).$$

Define weights

$$A_{tm} := \sum_{z \in \mathcal{I}_{multiset}:\, m(z)=m} K(x_t, z). \tag{10}$$

Then for item t,

$$\tilde{s}(x_t; \theta) = \sum_{m=1}^{M} A_{tm}\, w_m. \tag{11}$$

We note that, as a limitation, computing A may be expensive. However, once A is computed, evaluating a single match probability only requires two dot products with w:

$$P_i^{KBT} = \frac{A_{t_i^A} \cdot w}{\left(A_{t_i^A} + A_{t_i^B}\right) \cdot w}.$$

Thus, after precomputing A, SGD-style optimization looks very similar to standard BT/Elo: each (mini)batch only requires the rows of A corresponding to the items in that batch.

B.3 Simple limiting behavior and intuition

Duplicates of a single comparison (same prompt, same responses). Suppose we repeatedly add copies of one labeled comparison $(p, a \succ b)$. In standard BT, this adds repeated identical terms and pushes $\theta_{m(a)} - \theta_{m(b)}$ upward roughly in proportion to repetition (until regularized).

In kernel-smoothed BT, repeating the comparison increases the regional mass around both $x = (p, a)$ and $x = (p, b)$ (and around nearby items) because both sides appear in the kernel sums. This yields diminishing returns: additional duplicates also increase the denominator of related comparisons through the KDE. This counteracts the incentive to increase the model score, since increases will also push nearby comparisons closer together.

Duplicates concentrated on a single model. An "attack" that makes one model appear and win many times is essentially linear in standard BT: more wins leads to more score movement. In the kernel-smoothed objective, concentrated wins inflate local regional mass in the neighborhoods where those wins occur. If wins are highly clustered (same prompt type / same response style), the marginal gain in moving θ diminishes. Large global movement requires wins spread across diverse regions of prompt–response space.

Duplicates concentrated in a single language. If the kernel uses prompt similarity (or prompt+response embeddings), then a single language typically forms a coherent region in embedding space. This means that a large number of comparisons in a single language will dilute the effect, because there will also be a large contribution from that model to the denominator.

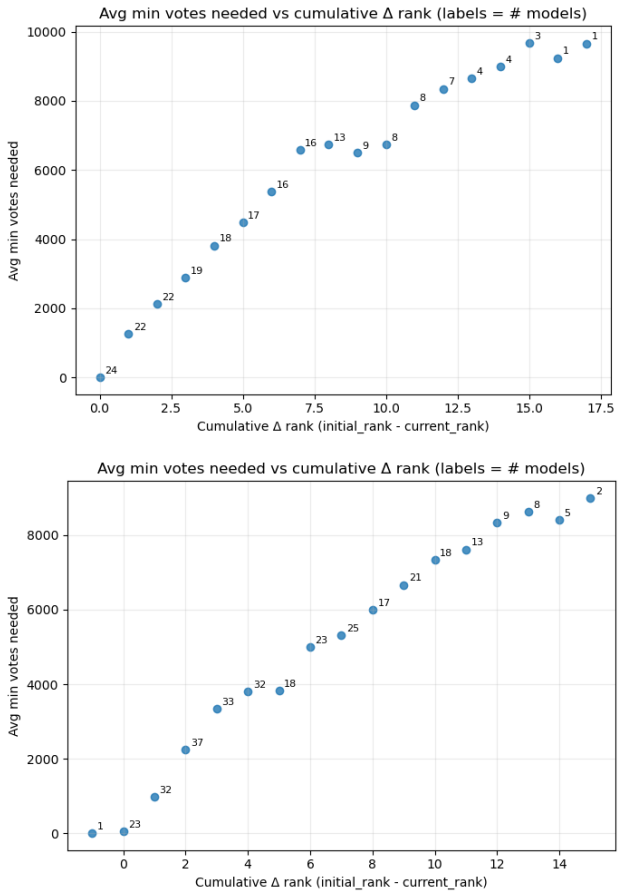

Figure 6: Number of votes under the **greedy strategy (top)** and **heuristic strategy (bottom)** needed to achieve greater improvements in leaderboard rank. Point labels indicate the number of models that can achieve that rank improvement with under 10K strategic votes. Small improvements to rank (2-3 places) are relatively easy to achieve, requiring under 2% of total votes (2K of 100K total) to be manipulated and achievable by a relatively large subset of models. Larger improvements (15+ places) are achievable by a smaller subset of models with more manipulated votes. While the maximum rank boost achieved is higher for the greedy strategy, more models are able to achieve smaller boosts under the heuristic strategy (and typically with fewer votes).

C Impact Statement

Research on potential methods for leaderboard "hacking" raises the concern that bad actors could use these methods in order to manipulate their model results. To mitigate this, we discuss defenses to each of these methods. We also plan to inform staff at major leaderboards before preprinting this work. Our goal is that, in making these simple methods widely known (rather than waiting for bad-faith actors to use them privately), this work raises awareness of the need to defend against manipulated evaluations.

D Results: Supporting Details

Figure 6 tracks the number of votes needed to move a model up a certain number of places in the rankings under greedy vs. heuristic voting strategies.

Figures 7 and 8 indicate the boost achieved in a model's rank by adding the model family that favors it most. Inclusion of the most favorable family can move models 1-4 places upward, if desired, or 1-4 places downward.

Figure 9 indicates the number of strategic prompts needed to achieve different changes in model rank.

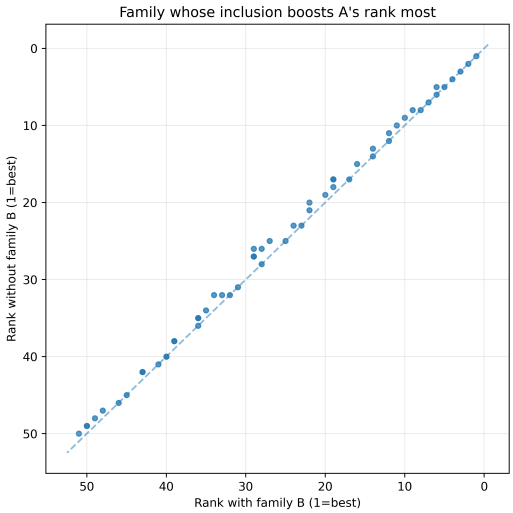

Figure 7: Counterfactual of model ranks with and without the family B of models that moves the target model furthest *up* when included. We plot each model's rank when family B is excluded from the Elo calculation, versus its rank (among models not in B) when family B is included in the Elo dataset, i.e., the original dataset.

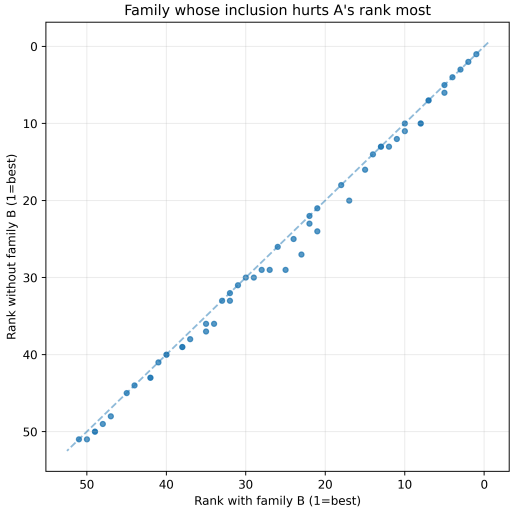

Figure 8: Counterfactual of model ranks with and without the family B of models that moves the target model furthest *down* when included. We plot each model's rank when family B is excluded from the Elo calculation, versus its rank (among models not in B) when family B is included in the Elo dataset, i.e., the original dataset.

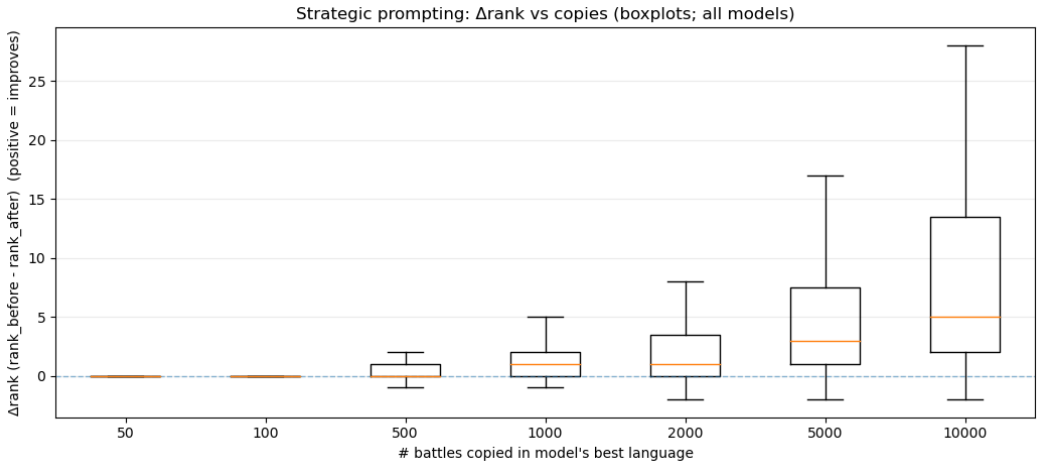

Figure 9: Average rank change achieved as more strategic prompts are added.

E Live model identification

We evaluate live model identification on the `lmarena-ai/arena-human-preference-100k` dataset. We consider seven open-weight models and three proprietary API-based models. The evaluated models include:

- `gpt-4o-2024-08-06`
- `gpt-4o-mini-2024-07-18`
- `gpt-3.5-turbo-0125`
- `llama-3.1-70b-instruct`
- `llama-3-70b-instruct`
- `llama-3-8b-instruct`
- `llama-3.1-8b-instruct`
- `qwen2-72b-instruct`
- `gemma-2-2b-it`
- `gemma-1.1-7b-it`

For each model, we sample $N = 200$ prompt-response pairs, of which 25% have been generated by the target model and the remaining 75% by other models. We then formulate the identification as a ranking task, where we make a guess for K outputs and we report Precision@K, where K/N corresponds to the fraction of total outputs guessed.

For open-weight models, we compute the average token-level log-probability of the observed response under each candidate model using teacher forcing, i.e., the log-probability assigned to each answer token conditioned on the prompt and all preceding answer tokens. For proprietary models, we generate a single response per prompt and compute character-level similarity to the observed response.

Figure 10 shows that open-weight models can be identified with high precision ($> 80\%$, except `qwen2-72b-instruct`, which has a slightly lower precision). Proprietary models achieve lower but above chance precision (approximately $40 - 50\%$).

The outputs contain 315 tokens, on average. Figure 17 demonstrates that for open-weight models, we do not need to score all generated tokens to achieve high precision. Scoring only the first few tokens could reduce the computational cost of real-world model identification.

F Factors Influencing Ease of Movement

We conducted regressions on the best Δ rank achieved from strategic voting, prompting, or nomination. We used ridge regression with the following factors for each model A: The factors we consider are A's variance in win rate against different models, A's starting rank and Elo, the percent of battles in which A tied, the number of models in A's model family, the gap in Elo between A and the model

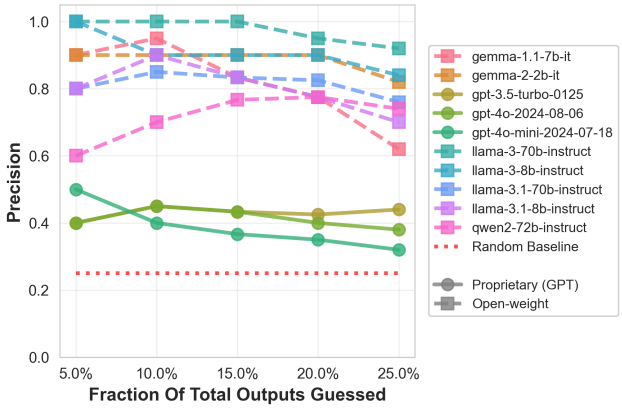

Figure 10: Model detection precision at different fractions of total outputs guessed for open-weight models (square markers, dashed lines) and proprietary GPT models (circle markers, solid lines). Random baseline shown at 0.25 (base rate).

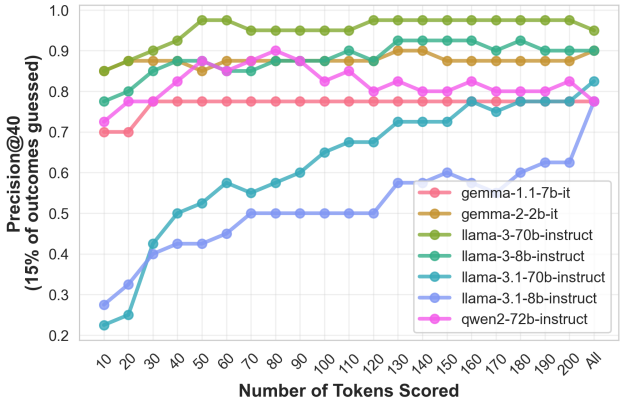

Figure 11: Precision@30 (15% of outputs) as a function of tokens scored for open-weight models. Many models reach near-optimal precision with as few as 30 tokens, suggesting substantial efficiency gains are possible by truncating log-probability calculations early.

one rank higher, A's win rate and total battles against the model one rank higher, the number of battles in A's best language for strategic prompting, the number of total battles involving A, and the "Elo density" (defined as the number of models with Elo 0-50 points higher than the current model).

Table 2: Regression coefficients for the best Δ rank achieved from strategic voting (RMSE=2.75).

Feature	Coefficient
elo density vs above	0.354919
winrate variance	0.079747
# total battles	-0.064431
elo gap vs model above	0.028544
winrate vs model above	-0.027852
model Elo	0.026598
# battles vs model above	-0.021019
tie rate	-0.015949
# battles in best prompt lang	-0.011382
model rank	-0.009996
family size	-0.009464
Train R^2	0.660

Table 3: Regression coefficients for the best Δ rank achieved from strategic prompting (RMSE=2.67).

Feature	Coefficient
# battles in best prompt lang	-0.337490
elo gap vs model above	-0.126945
elo density vs above	0.111206
winrate variance	0.069611
family size	-0.066190
winrate vs model above	0.040990
# battles vs model above	-0.039253
tie rate	0.036262
model rank	0.027060
# total battles	-0.016139
model Elo	-0.012417
Train R^2	0.518

Table 4: Revised regression coefficients for the best Δ rank achieved from strategic nomination (RMSE=1.85).

Feature	Coefficient
elo density vs above	0.231206
elo gap vs model above	-0.200973
winrate variance	0.179882
model Elo	0.157270
# battles in best prompt lang	0.140642
tie rate	0.089433
model rank	-0.077523
# battles vs model above	0.073610
# total battles	-0.036676
family size	0.017748
winrate vs model above	0.001865
Train R^2	0.557

G Limitations

The dataset we work with is smaller than the full LMArena rankings; we expect that larger numbers of absolute manipulations would be necessary to move models on larger leaderboards (though several of the LMArena leaderboards are of comparable size to our dataset). We also note that, for ethical reasons, we have not tried to actually move models on the live rankings themselves. Red-teaming for the effects of these sorts of manipulations on live rankings, and for effects on larger leaderboards, remain important areas for future work.

H Scaling Experiments

We conducted experiments varying the number of votes and found that when varying the total dataset size from 25%-200% of the current dataset size, manipulating the same percentage of total votes gives a near-uniform rank increase. This means that as the dataset size increases, the number of votes needed to achieve a similar boost increases linearly.

We manipulated a constant $\sim 10\%$ of battles as the dataset size increases. For experiments over 100K examples, we resampled battles from the existing 100K dataset. Then, we plotted dataset size vs. maximum rank boost when manipulating that constant proportion of votes. For heuristic voting, greedy voting, and combined prompting + nomination, the slope is near-constant with $p >> 0.05$ (that is, not significantly different from a line with slope 0). This indicates that the number of manipulations needed to achieve similar rank boosts scales linearly with dataset size. For prompting and nominations, the slope is also near-constant, though we do find that manipulations become slightly easier as dataset size increases (though these attacks depend on the proportion of models or prompts to total votes, so this result may vary by leaderboard). Figures 12-16 show the effects of scaling dataset size on the number of manipulations needed. Across experiments, the number of manipulations scales linearly or close to linearly.

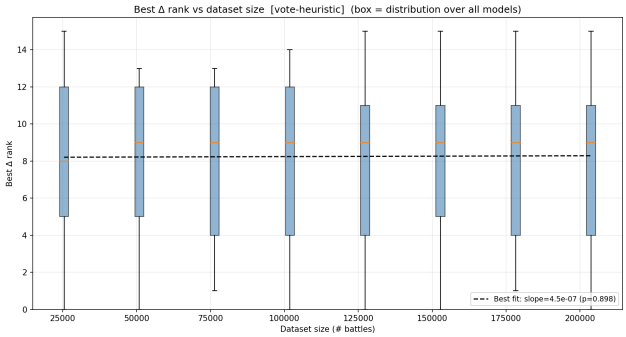

Figure 12: Best delta rank achieved when manipulating a constant 10% of battles on datasets of different sizes, on the *heuristic voting* strategy. When plotting dataset size (x) against best rank achieved with $\leq 10\%$ battles modified (y), the slope is near-constant with p $>>$ 0.05 (not significantly different from a line with slope 0). This indicates that the number of manipulations needed to achieve similar rank boosts scales linearly with dataset size.

I Strategic Prompting: Changing Subcategories

Here, we test strategic prompting with a different subcategory split as proof-of-concept that strategic prompting can use any definable, consistent subcategories. We find rank boosts of up to 14 places, comparable to those from strategic prompting using language for prompt subcategories.

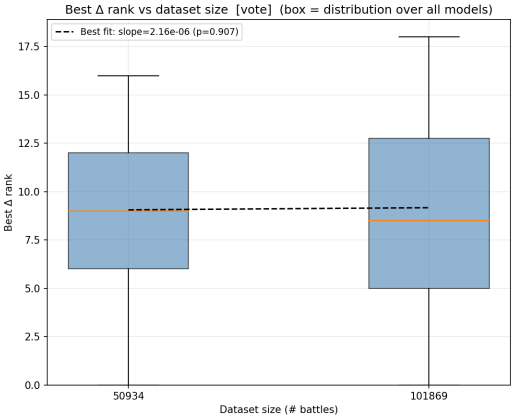

Figure 13: Best delta rank achieved when manipulating a constant 10% of battles on datasets of different sizes, on the *greedy voting* strategy. When plotting dataset size (x) against best rank achieved with $\leq 10\%$ battles modified (y), the slope is near-constant with $p >> 0.05$ (not significantly different from a line with slope 0). This indicates that the number of manipulations needed to achieve similar rank boosts scales linearly with dataset size.

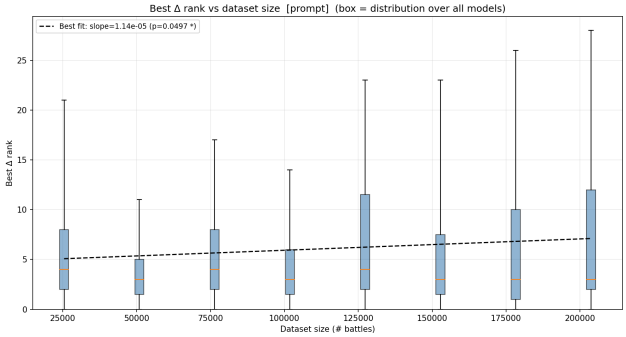

Figure 14: Best delta rank achieved when manipulating a constant 10% of battles on datasets of different sizes, with the *strategic prompting* strategy. When plotting dataset size (x) against best rank achieved with $\leq 10\%$ battles modified (y), the slope is very slightly positive ($p < 0.05$), meaning that the manipulation becomes slightly easier as dataset size increases.

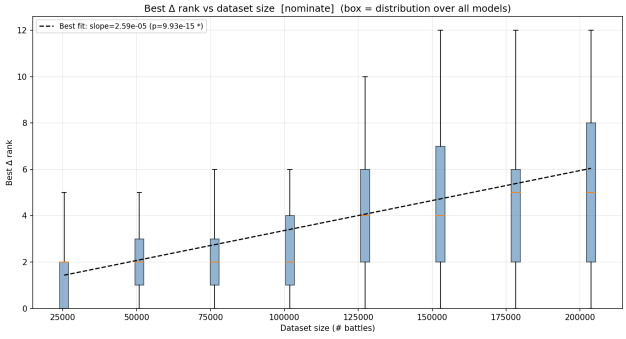

Figure 15: Best delta rank achieved when manipulating a constant fraction of models on datasets of different sizes, using the *strategic nomination* strategy. When plotting dataset size (x) against best rank achieved (y) with ≤ 10 model copies inserted (number of battles per model copy thus scales with dataset size), the slope is slightly positive ($p < 0.05$), meaning the attack is somewhat easier as dataset size increases with the same proportion of models changed (though note that the proportion of models to total votes may vary by leaderboard).

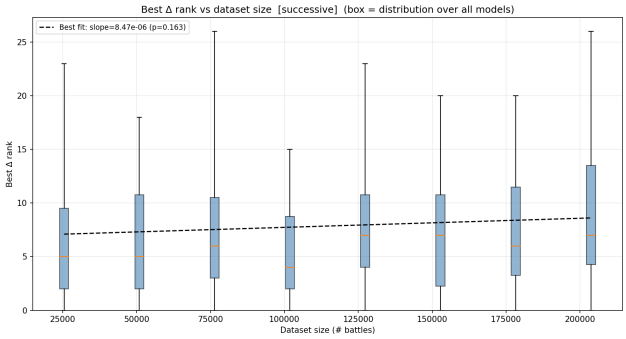

Figure 16: Best delta rank achieved when manipulating a constant 10% of battles on datasets of different sizes, when combining the strategic nomination and prompting strategies in succession. When plotting dataset size (x) against best rank achieved with $\leq 10\%$ battles modified (y), the slope is near-constant with $p >> 0.05$ (not significantly different from a line with slope 0). This indicates that the number of manipulations needed to achieve similar rank boosts scales linearly with dataset size.

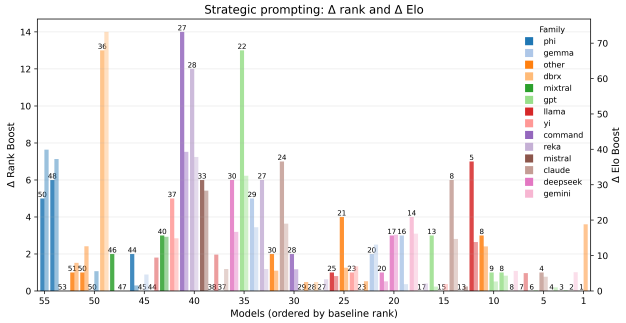

Figure 17: Proof-of-concept that strategic prompting can use any definable, consistent subcategories: best delta rank and Elo achieved under strategic prompting, where prompt categories are based on whether the prompt is labeled as involving complexity, creativity, domain knowledge, and/or math. Rank boosts (up to 14 places) are comparable to those from strategic prompting using language for prompt subcategories.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Yes, the abstract and introduction state the claims made in the paper regarding leaderboard manipulation.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Yes, limitations are discussed in Appendix G.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: Yes, for the Density-smoothed Bradley–Terry, the proof is provided in Appendix B.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 3 provides a link to the (public) data we use and a detailed description of our methods, which are fairly straightforward to implement. We also plan to release code with the camera-ready.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: The data is publicly available. We will release the code with the camera-ready.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We specify the details of our experimental methodology; none of the experiments involved training new models, so there are no training-related hyperparameters to report.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report p-values for the scaling experiments in Appendix H and RMSE for the regressions in Appendix F. For Figures 1-4, the main plots already show the full distribution of attack effect sizes (i.e., the range of observed boosting effects) over all tested models.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Yes, we describe computational feasibility in section 5.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Yes, we have read and adhere to the code of ethics. Defenses to mitigate security risks from leaderboard manipulation research are discussed in Section 6.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Yes, we discuss impacts in the conclusion and in Appendix C.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We do not release data or models.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Yes, we provide links to the LMArena data used and note its CC-BY-4.0 license in footnote 4.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: We do not introduce new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: No human subjects research was performed.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: No human subjects research was performed.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core methods do not involve LLMs as an important/original/non-standard component.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.