

# 000 001 002 003 004 005 BIASBUSTERS: UNCOVERING AND MITIGATING TOOL 006 SELECTION BIAS IN LARGE LANGUAGE MODELS 007 008 009

010 **Anonymous authors**  
011 Paper under double-blind review  
012  
013  
014  
015  
016  
017  
018  
019  
020  
021  
022  
023  
024  
025  
026

## ABSTRACT

027 Agents backed by large language models (LLMs) often rely on external tools  
028 drawn from marketplaces where multiple providers offer functionally equivalent  
029 options. This raises a critical point concerning fairness: *if selection is systematically biased, it can degrade user experience and distort competition by privileging some providers over others*. We introduce a benchmark of diverse tool categories,  
030 each containing multiple functionally equivalent tools, to evaluate tool-selection  
031 bias. Using this benchmark, we test seven models and show that unfairness exists  
032 with models either fixating on a single provider or disproportionately preferring  
033 earlier-listed tools in context. To investigate the origins of this bias, we conduct  
034 controlled experiments examining tool features, metadata (name, description, pa-  
035 rameters), and pre-training exposure. We find that: (1) semantic alignment be-  
036 tween queries and metadata is the strongest predictor of choice; (2) perturbing  
037 descriptions significantly shifts selections; and (3) repeated pre-training exposure  
038 to a single endpoint amplifies bias. Finally, we propose a lightweight mitigation  
039 that first filters the candidate tools to a relevant subset and then samples uniformly,  
040 reducing bias while preserving good task coverage. Our findings highlight tool-  
041 selection bias as a key obstacle for the fair deployment of tool-augmented LLMs.  
042  
043

## 1 INTRODUCTION

044 Large language models (LLMs) have transformed natural language processing, achieving near-  
045 human performance on tasks ranging from code generation to creative writing (Naveed et al., 2024;  
046 Luo et al., 2024). Yet LLMs cannot directly act in the world: they cannot query databases, fetch live  
047 information, or invoke external services. Additionally, their knowledge remains frozen at training  
048 time, leaving them prone to “hallucinations” when asked about events beyond their cutoff (Ji et al.,  
049 2023). Augmenting LLMs with external “tools” / APIs addresses these shortcomings by allowing  
050 models to delegate specialized functions to dedicated services (Qu et al., 2025). It endows LLMs  
051 with the ability to act, a core capability often associated with LLM *agents* (Chowa et al., 2025). A  
052 crucial step within the typical tool-usage pipeline is the multi-stage tool-selection process: given  
053 an instruction to the LLM, (i) retrieve a short list of the most relevant candidate tools based on the  
user query (with, e.g., highest semantic similarity) from a potentially large database of tools, (ii)  
insert their metadata into the prompt, (iii) have the LLM reason and pick one to solve (one of) the  
necessary user task(s) (see Appendix K covering the entire tool-usage pipeline). However, this pro-  
cess introduces a new challenge: bias (see Figure 1). An LLM may prefer certain tools not for their  
relevance or accuracy, but because of superficial metadata, i.e., tool names, descriptions, or prompt  
ordering. Such bias can degrade user experience by repeatedly selecting slow or unreliable ser-  
vices and can, therefore, also inflate operational costs. Additionally, under pay-per-request pricing  
for tool use, consider the scenario where such biases are systematic across frontier LLMs: if they  
consistently favor tools from a single provider, this creates major market unfairness, disadvantag-  
ing competing providers offering functionally equivalent services (RapidAPI, 2025a; Economize,  
2024). **See Appendix L covering the consequences of tool-selection bias in more detail.**

054 In this paper, we make three key contributions. **First**, we present a large-scale benchmark for  
055 measuring tool selection bias, which we use to conduct the first empirical study of tool-selection bias  
056 in LLMs. We introduce total variation-based metrics to quantify selection imbalances, assemble a  
057 novel benchmark of clusters of tools with equivalent functionality, and conduct extensive experi-  
058 ments across multiple models and configurations. These tell us that bias exists to a certain extent

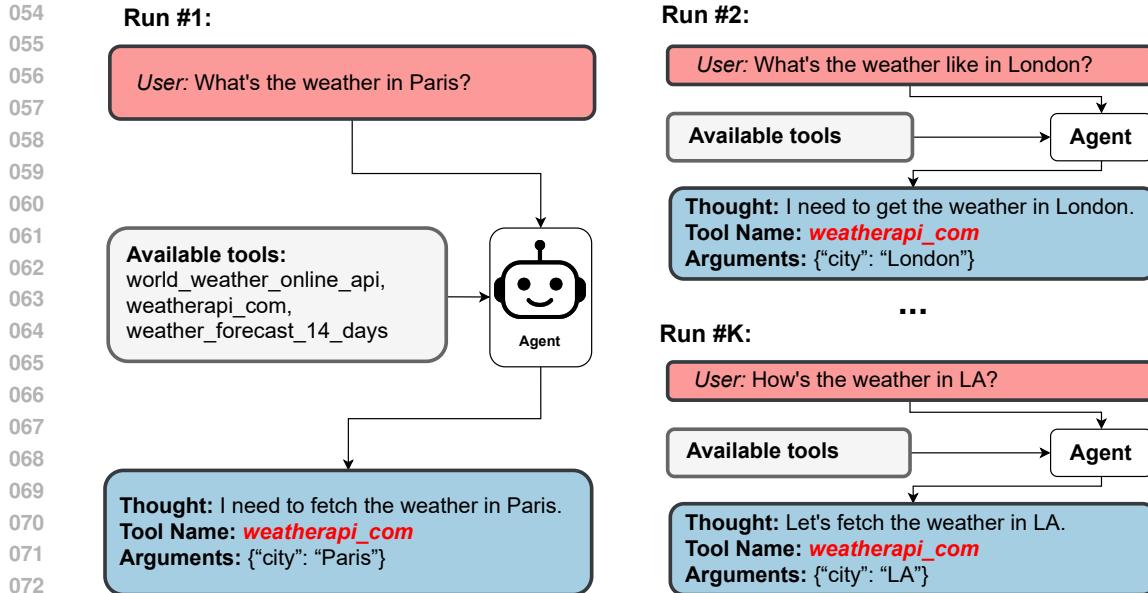


Figure 1: Tool-calling enables LLMs to act through external services, but the selection process introduces bias. Models may favor certain tools based on superficial metadata or position rather than relevance (here “weatherapi.com” is preferred), leading to a potential degraded user experience and unfair concentration of calls. If such biases are systematic across frontier LLMs, they risk distorting entire tool marketplaces, disadvantaging functionally equivalent competitors.

for *all* models tested; models either fixate on a single provider or disproportionately prefer earlier-listed tools in context. **Second**, after confirming the existence of bias, we investigate the root causes of these biases. We define a rich set of tool-level features and use regression and permutation-importance analyses to identify which factors predict selection. We further perturb metadata fields to test their influence and probe whether continued pre-training on biased data can induce persistent preferences. Our findings show that semantic alignment between query and tool description is the strongest interpretable driver of tool choice, metadata interventions on descriptions reliably steer choices, and biased pre-training amplifies preferences. However, none of these fully explain the behavior. **Third**, we propose a lightweight mitigation strategy: filtering the supplied tools to a subset of relevant ones and sampling uniformly among them. This approach substantially reduces bias while still ensuring the user task can be solved.

By uncovering, explaining, and addressing tool-selection bias, our work illuminates a critical blind spot in tool-augmented LLM research. Our contributions go beyond diagnosis: we provide reproducible resources and a simple mitigation strategy that practitioners can adopt immediately. More broadly, we aim to set a foundation for fairer, more reliable tool-calling systems and a precedent to judge tool-calling LLM applications not only by their accuracy but also by the equity of their interactions with external ecosystems. All resources are available online<sup>1</sup>.

## 2 RELATED WORK

**Fairness & Bias in LLMs.** Bias in large language models emerges from their opaque training on vast web-scale data, often carrying forward and amplifying human prejudices and reasoning shortcuts (Gallegos et al., 2024). Much of the literature has focused on identifying and mitigating these undesirable behaviors (Schick et al., 2021; Gallegos et al., 2024; Bouchard, 2024). Unwanted behaviors in LLMs manifest as social bias, where stereotypes shape outputs and lead to unequal treatment of protected groups, or as cognitive bias, reflecting human-like shortcuts such as anchoring or confirmation effects (Lou & Sun, 2024; Itzhak et al., 2024; O’Leary, 2025). Measuring these phenomena requires domain-specific metrics. For example, to measure how likely a model is to associate a

<sup>1</sup>All resources are available on the public GitHub repository.

108 neutral attribute with one group over another, one can utilize the normalized log-probability bias  
 109 score (Kurita et al., 2019). Mitigation strategies are diverse and can be deployed at multiple stages  
 110 (e.g., at pre-/post-inference via augmentation of training data or rewriting of outputs or during training  
 111 via fairness-aware objectives) (Gallegos et al., 2024). However, no single approach suffices for  
 112 every scenario; effective debiasing requires interventions matched to the task’s fairness goals and  
 113 deployment context. We are not aware of prior work on bias in the tool-use paradigm.

114 **Positional Bias in LLMs.** Prior research has revealed significant positional selection biases in how  
 115 LLMs answer multiple-choice questions. Pezeshkpour & Hruschka (2024) and Zheng et al. (2024)  
 116 demonstrated that LLMs exhibit sensitivity to the ordering of answer options, with performance  
 117 varying systematically based on option position. This was corroborated in the investigation of Wei  
 118 et al. (2024) who also found token sensitivity, where specific token characteristics at different pos-  
 119 itions affect selection. Recently Wang et al. (2025), analyses the underlying model mechanisms that  
 120 give rise to position bias and proposed a mechanistic approach to eliminating it. In this work we  
 121 study a investigate a broad range of potential biases in the critical context of tool selection.

122 **Fairness in Tool Selection.** Research on tool-selection bias is limited; however, some works in  
 123 information retrieval address related issues such as positional or ranking bias (Dai et al., 2024;  
 124 Ziems et al., 2024). Similarly, in code generation, favoritism for certain providers has been observed  
 125 (Zhang et al., 2025). However, these studies do not examine the specific setting of tool selection  
 126 among providers. Recent studies do show that LLMs can develop systematic preferences in this  
 127 setting, for example, for selecting tools with manipulated metadata (Mo et al., 2025; Faghah et al.,  
 128 2025). Analogously, one study shows an advanced attack on tool selection which splits malicious  
 129 tool entries into fragments optimized for retrieval and selection, achieving high success rates while  
 130 evading defenses (Shi et al., 2025). These findings reveal that tool-selection mechanisms are highly  
 131 fragile and susceptible to both naive and advanced exploitation. In contrast to overt adversarial  
 132 attacks, our study focuses on subtler sources of bias: small, non-adversarial differences in phrasing  
 133 or metadata that can still distort fairness in tool selection.

### 134 3 BIASBUSTERS: HOW WE UNCOVER, EXPLAIN, AND MITIGATE BIAS.

135 In this section, we present our end-to-end framework for uncovering and explaining tool-selection  
 136 bias in LLMs. We start in Section 3.1 by introducing our formal definition of this bias. In Section  
 137 3.2, we describe how we generate a comprehensive benchmark of API clusters and user queries that  
 138 enables systematic measurement of selection behavior. Section 3.3 details our analysis pipeline for  
 139 explaining bias.

#### 140 3.1 BIAS DEFINITION

141 Tool-selection bias captures the extent to which an LLM systematically favors certain APIs over  
 142 others for reasons unrelated to their true utility. Formally:

143 **Definition 3.1** *Tool-selection bias is the systematic tendency of a model to favor certain APIs over  
 144 others for reasons unrelated to the APIs’ true relevance or utility for the task.*

145 To quantify this bias, we compare a model’s empirical selection rates against an ideal uniform  
 146 choice. Consider a cluster of  $K$  APIs that can all solve a given query  $q$ . A perfectly unbiased  
 147 model would select each API with probability  $1/K$ , forming a uniform distribution  $U$ . Let  
 148  $P^{\text{API}} = (P_1^{\text{API}}, \dots, P_K^{\text{API}})$  denote the expected empirical selection-rate distribution over the cluster  
 149 under all orderings of the API list. We measure cluster-level bias via the total variation distance

$$150 \delta_{\text{API}} = \frac{1}{2} \sum_{i=1}^K \left| P_i^{\text{API}} - \frac{1}{K} \right| = \text{TV}(P^{\text{API}}, U).$$

151 Prior work has shown that list order itself can introduce bias: when multiple identical tools ap-  
 152 pear in sequence, the first position is favored Faghah et al. (2025). To capture this effect, we  
 153 also record the expected selection-rate distribution over absolute positions,  $P^{\text{pos}}$ , and compute  
 154  $\delta_{\text{pos}} = \frac{1}{2} \sum_{i=1}^K \left| P_i^{\text{pos}} - \frac{1}{K} \right|$ . Combining both yields an overall bias metric:  $\delta_{\text{model}} = \frac{\delta_{\text{API}} + \delta_{\text{pos}}}{2}$ .  
 155 Note that a high positional bias  $\delta_{\text{pos}}$  can be mitigated by randomizing API order at prompt time,  
 156 whereas API-level bias  $\delta_{\text{API}}$  requires deeper intervention.

162 3.2 DATASET GENERATION  
163

164

165 We build on the ToolLLM pipeline (Qin et al., 2024), which provides a large repository of APIs  
 166 scraped from RapidAPI (RapidAPI, 2025b). To measure tool-selection bias, we construct a bench-  
 167 mark of functionally interchangeable APIs. Specifically, we cluster APIs into groups performing  
 168 the same task (e.g., weather forecasting or translation), and generate balanced, provider-agnostic  
 169 user queries that all APIs in a cluster can answer. The final benchmark consists of 10 clusters, each  
 170 containing 5 APIs and 100 queries, yielding 1,000 total cluster-query pairs. Running LLMs on this  
 171 benchmark produces empirical API-selection distributions, from which we compute our bias metrics  
 172  $\delta_{\text{API}}$ ,  $\delta_{\text{pos}}$ , and  $\delta_{\text{model}}$ . See Appendix A for clustering and query-generation details.  
 173

174

175

176 3.3 EXPLAINING BIAS  
177

178

179 To pinpoint what drives tool-selection bias, we pursue three complementary analyses.

180

**Attribute-Level Analysis.** To test whether intrinsic API characteristics explain model preferences,  
 181 we extract seven descriptive features for each API. Examples include semantic similarity between  
 182 queries and descriptions, number of parameters, description length, readability, and promotional  
 183 wording (See Table 3 for the full list). Our benchmark contains 10 clusters with 5 APIs each, giv-  
 184 ing 50 APIs in total. For each API, we measure its empirical selection rate and pair this with its  
 185 feature values. This produces a dataset of 50 (API, features, selection rate) entries per model. We  
 186 then probe relationships between features and selection behavior in three ways. First, we compute  
 187 *Pearson correlations* between tool features and selection rate to capture linear and monotonic as-  
 188 sociations. Second, we fit a *linear regression* per model to quantify the aggregate explanatory power  
 189 (reported as  $R^2$ ) and inspect coefficients to understand the direction and relative weight of each  
 190 feature. Third, we train *random-forest regressors* with cross-validation to allow for non-linear in-  
 191 teractions and obtain alternative measures of feature importance. This pipeline reveals which API  
 192 attributes most strongly influence the model’s choices.

193

**Metadata Perturbation Experiments.** To isolate the superficial cues that drive tool-selection pref-  
 194 erences, we apply a series of controlled perturbations to tool metadata. Specifically, we utilize  
 195 the following manipulations: (1) *Full name scramble*. Replace every tool’s name with a fresh  
 196 20-character random string, destroying any learned association tied to the literal name; (2) *Name*  
 197 *shuffle*. Permute the tool names among APIs within each cluster so that names no longer align  
 198 with their original endpoints; (3) *Single-tool perturbation*. Identify the most frequently chosen tool  
 199 in each cluster and replace only its name with a random string; (4) *Description and parameter*  
 200 *scramble*. Randomize each tool’s descriptive text and parameter description(s) (but keep the orig-  
 201 inal names intact) to test whether the semantic content beyond the name influences selection; (5)  
 202 *Description-only / Parameter-only scramble*. Randomize only the tool descriptions (keeping pa-  
 203 rameter descriptions intact), or only the parameter descriptions (keeping the tool description intact),  
 204 to disambiguate their individual contributions; (6) *Targeted description scramble (most-selected)*.  
 205 Identify the most frequently chosen API in each cluster and scramble only *its* description to test  
 206 whether degrading its semantics reduces its selection share; (7) *Description transfer (most → least)*.  
 207 Swap the most-selected API’s description with the least-selected API while leaving other metadata  
 208 untouched, assessing whether swapping semantic “advantage” transfers selection probability; and  
 209 (8) *Full scramble*. Randomize each tool’s descriptive text, parameter description(s), and tool names  
 210 to test the effect of having minimal semantic signal in API metadata on bias.

211

212 By re-running our selection experiments under these alterations, we quantify how much of the ob-  
 213 served bias is attributable to literal names, to deeper semantic content in descriptions and parameters,  
 214 and to relative contrast between a clean and corrupted endpoint.

215

**Biased Continued Pre-Training.** We also test whether pre-training data itself can induce tool-  
 216 selection bias. To verify this hypothesis, we perform biased continued pre-training (CPT) on Qwen3-  
 217 8B using  $\sim 3.5$ M tokens deliberately saturated with a single API’s metadata. See Appendix C for  
 218 additional details on corpus construction and training setup.

216	Available tools:	217	Available tools:	218	Available tools:
218	* API A	219	* API C	220	* API B
219	* API B	220	User:	221	User:
220	* API C	221	What is the weather in Paris?	222	What is the weather in Paris?
222		223		224	

Figure 2: Cyclic rotations of one fixed tool list; each API appears at the top once.

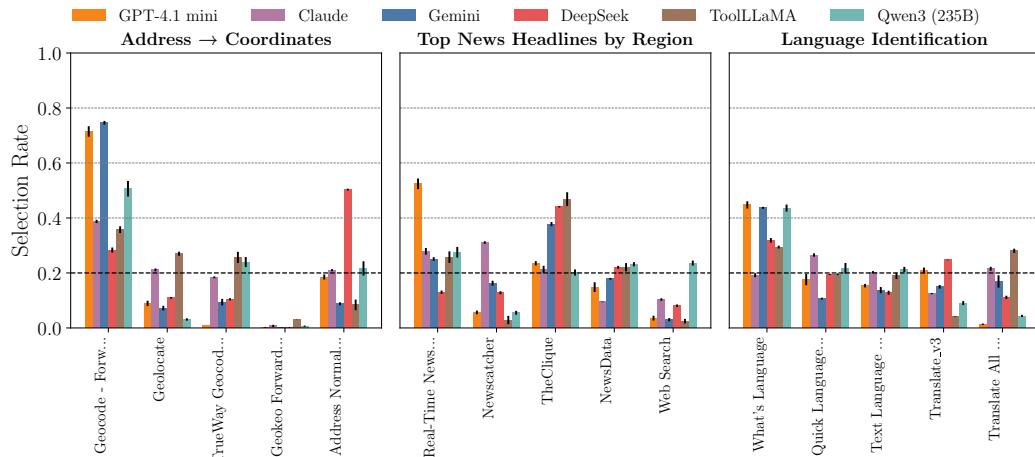


Figure 3: Selection distributions for six LLMs across three clusters of functionally equivalent APIs. Each subplot corresponds to one cluster, with the x-axis indicating the API in the cluster and the y-axis showing the (mean) fraction of times each model chose that API over 500 inference runs; error bars indicate the standard deviation across three independent experimental runs. The optimal uniform selection rate is highlighted.

## 4 EXPERIMENTS

This section reports the empirical results from our experiments. We first describe the experimental setup (Section 4.1), then characterize tool-selection behavior and quantify bias using our metrics (Section 4.2). Next, we investigate the drivers of bias (Section 4.3), and finally, we evaluate an approach to mitigate any observed bias in Section 4.4.

### 4.1 EXPERIMENTAL SET-UP

**Models.** We evaluate a diverse set of LLMs, including GPT-3.5-turbo and GPT-4.1-mini by OpenAI (2025), Claude 3.5 Sonnet by Anthropic (2025), DeepSeek-V3.2-Exp (DeepSeek-AI et al., 2025), Gemini 2.5 Flash (Google, 2023), ToolLLaMA-2-7B (Qin et al., 2024), and Qwen3 models spanning 1.7B to 235B parameters (Bai et al., 2023).

**Reasoning.** Each model is prompted to produce a short chain of thought (Wei et al., 2022) followed by at most one tool call. This design makes outputs efficient to generate and easy to analyze.

**Parameter Setup & Dataset.** Experiments use our benchmark of 10 clusters, each containing 5 interchangeable APIs and 100 distinct user queries (Section 3.2 and Appendix A). Unless otherwise noted, decoding uses a temperature of 0.5 and a top-p of 1.0.

**API ordering.** As noted in previous work (Faghah et al. (2025)), LLMs prefer tools that appear earlier in the prompt. To control for this, we execute each query five times, each with a different cyclic rotation of a fixed API ordering (see Figure 2). This ensures that every API appears at the top for a given query exactly once.

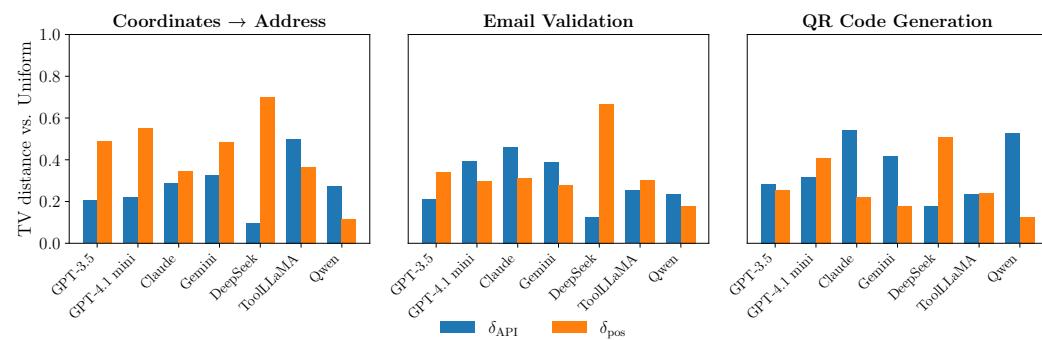


Figure 4: API- vs. positional bias by model for three clusters. Bars show total-variation deviation from uniform, where higher values indicate stronger bias.

Table 1: Average cluster-level API bias  $\delta_{\text{API}}$ , positional bias  $\delta_{\text{pos}}$ , and combined bias  $\delta_{\text{model}}$  (mean across clusters and runs  $\pm$  std across runs).

Model	$\delta_{\text{API}}$	$\delta_{\text{pos}}$	$\delta_{\text{model}}$
Gemini 2.5 Flash	$0.365 \pm 0.003$	$0.306 \pm 0.005$	$0.335 \pm 0.002$
Claude 3.5 Sonnet	$0.370 \pm 0.005$	$0.325 \pm 0.005$	$0.347 \pm 0.001$
DeepSeek-V3.2-Exp	$0.249 \pm 0.003$	$0.504 \pm 0.003$	$0.377 \pm 0.001$
Qwen3 235B	$0.330 \pm 0.006$	$0.168 \pm 0.004$	$0.249 \pm 0.001$
ToolLaMA	$0.277 \pm 0.002$	$0.391 \pm 0.002$	$0.334 \pm 0.002$
GPT-3.5-turbo	$0.320 \pm 0.022$	$0.336 \pm 0.012$	$0.328 \pm 0.005$
GPT-4.1 mini	$0.331 \pm 0.008$	$0.423 \pm 0.002$	$0.377 \pm 0.004$

#### 4.2 HOW DO LLMs SELECT AMONG FUNCTIONALLY EQUIVALENT APIs?

Figure 3 shows our first results: the empirical selection distributions of six LLMs over three clusters of functionally equivalent APIs (see Appendix G.2 for the full figure including all clusters). Choices are far from uniform. In some clusters (e.g., geocoding), all models concentrate heavily on a single API, whereas in others (e.g., language identification) the distributions are flatter. We also observe that different models do not always prefer the same API, a point we explore further when analyzing alignment in selection behavior.

**Positional bias persists even when API-level preference is weak.** Figure 4 summarizes API- vs. positional-bias (measured as total variation distance from uniform) across models and clusters. We notice two regimes: high API bias with low positional bias, or low API bias with high positional bias; a few exhibit both elevated. This pattern indicates that when no API clearly dominates, models rely more on positional cues.

**The extent of LLM bias in tool selection.** To get an idea of *how* biased each model is in their tool-selection behavior, we compute the positional bias  $\delta_{\text{pos}}$ , API bias  $\delta_{\text{API}}$ , and their average  $\delta_{\text{model}}$  for seven different models (Section 3.1). Table 1 reports these metrics averaged across all clusters and three runs. All models exhibit substantial bias:  $\delta_{\text{model}}$  values are around 0.3–0.4, meaning that roughly 30–40% of the selection probability mass would have to be redistributed to achieve fairness. GPT-4.1 mini is especially biased, with a combined metric of  $\sim 0.38$ . The least biased model in our suite is Qwen 3 (235B), which attains the lowest  $\delta_{\text{model}}$ .

**LLMs are (mostly) aligned in their bias.** To examine the similarity of selection behavior of the different models, we represent each model by a vector obtained by concatenating its empirical selection distributions over APIs across all clusters, then compute pairwise Pearson correlations between these vectors; the resulting matrix is shown in Figure 15. It shows that many of the models share

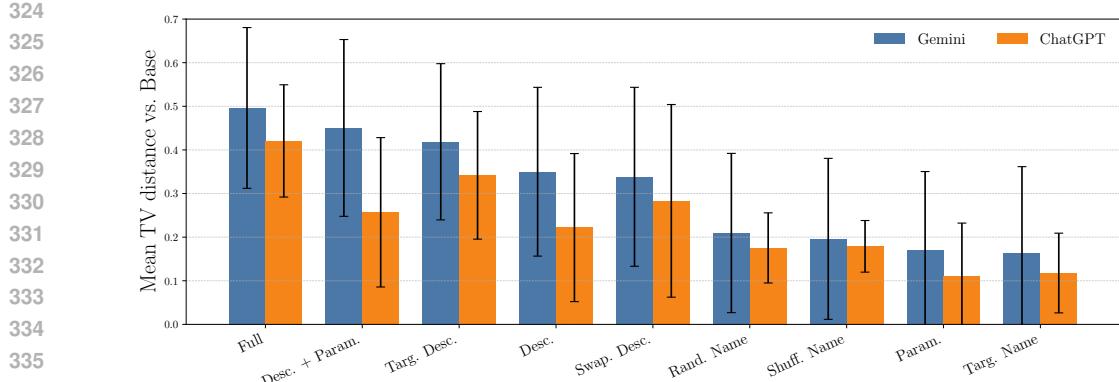


Figure 5: Mean total-variation (TV) distance from the base selection distribution (no perturbation) to the distribution pertaining to each metadata perturbation (higher = larger shift). Blue bars show results for Gemini and orange bars for GPT-3.5. The error bars denote standard deviation across clusters. The run-to-run standard deviation is left out; max run-to-run variability of per-run mean TV was 0.084.

similar bias patterns: GPT 4.1-mini, Claude, Gemini, DeepSeek, and Qwen3 (235B) tend to favor and disfavor the same APIs. GPT-3.5 and ToolLLaMA stand apart with consistently lower correlations, suggesting qualitatively different selection behavior. This clustering of high correlations points to common drivers of bias. This could be due to a shared set of similar implicit decision rules. The relative divergence of GPT-3.5 and ToolLLaMA highlights that model architecture, capacity, or training objective can alter these tendencies, but overall alignment underscores that tool-selection bias patterns are not isolated quirks but mostly reproducible phenomena across LLMs.

**Ablation and Sensitivity Analysis.** We also analyze the robustness of tool-selection bias across several factors (full results in Appendix E). *Temperature*: raising temperature reduces bias modestly by softening extreme preferences. *Top-p*: increasing top-p has only a negligible effect. *Model size*: larger models exhibit noticeably less bias than smaller ones. *API ordering*: cyclic vs. random permutations produce very similar outcomes, indicating intrinsic preferences dominate. *System prompts*: rewording or restructuring prompts shifts which tools are favored, but does not remove bias. *Toolset size*: the composition of the toolset given in context is the primary driver of selection behavior, more than the size of the toolset alone (see Appendix J).

#### 4.3 ARE TOOL-SELECTIONS DRIVEN BY HUMAN-INTERPRETABLE HEURISTICS?

We investigate the drivers of bias along three complementary axes: (i) feature-level correlations between API attributes and selection rates, (ii) direct interventions on API metadata, and (iii) biased continued pre-training (CPT) to test whether exposure alone can plant preferences. We summarise our results below, with full details in Appendix F.

**Which API-level features predict selection rates?** The results show a consistent pattern: (1) Semantic similarity between queries and API / tool descriptions is the strongest predictor of selection (Table 4). By contrast, structural or stylistic attributes (e.g., parameter count, promotional wording) exhibit little consistent influence. (2) Linear regression reveals that surface-level semantic alignment is the primary signal but leaves a lot unexplained ( $R^2 < 0.4$  as can be seen in Figure 13), and (3) Random forests fail to offer meaningful improvement.

**How do metadata interventions affect API selection?** Figure 5 shows the TV distance from the base selection distribution to the distribution obtained after a certain metadata perturbation, averaged over clusters and experimental runs. It shows a clear hierarchy: logically minimizing semantic signal by scrambling the description, name, and parameters causes the biggest shift. For Gemini, corrupting descriptions+parameters yields the second largest, most reliable shifts in selection ( $0.450 \pm 0.203$  TV), followed by scrambling the description of the most popular tool ( $0.419 \pm 0.179$ ), and even description swaps meaningfully steer choices ( $0.338 \pm 0.205$ ). By contrast, name-only per-

turbations are smaller and have higher variance, and parameters-only are the least impactful. We see similar results for GPT; description perturbation is more impactful than name/parameter perturbations. Overall sensitivity is higher for Gemini than GPT (mean TV 0.310 vs 0.234), indicating greater responsiveness to metadata changes.

We show similar results across clusters in Figure 14; however, we observe that manipulating the description of certain tools has mixed effects from cluster to cluster. It tells us that the impact of description tampering is context-dependent: the same intervention can invert, redistribute, or barely change preferences, underscoring that tool choice emerges from multiple interacting cues.

Together, these patterns indicate that description-level semantics are the primary cues models use to discriminate among functionally similar APIs. Name perturbation alone tends to inject noise without consistent effects. Finally, bias persists under minimal semantic signal (only leaving certain parameter schema fields intact), implying selection behavior sometimes relies on residual, non-obvious priors rather than solely on coherent, human-interpretable heuristics. For an analysis of the impact of metadata perturbations on bias, see Appendix H.

**Does additional pre-training exposure to one endpoint change selection distribution?** We run biased continued pre-training on Qwen3-8B in which the training corpus is saturated with metadata for a single target endpoint (Text Language by API-Ninjas)<sup>2</sup>. All inference settings, prompts, and decoding parameters are held fixed; only the checkpoint changes (see full training and inference hyperparameters in Appendix I). We then compare selection rates within the same Language Identification cluster before CPT and after  $\sim 1/3$ ,  $\sim 2/3$ , and one full epoch of biased training (See Figure 17 for full figure). We observe that biased CPT substantially increases the selection of the exposed endpoint but does not fully determine choice. The target endpoint’s share rises from 0.006 (base) to 0.122 after 1/3 epoch, remains 0.122 at 2/3 epoch, and nudges to 0.128 at one epoch. This is an absolute gain of  $\sim 12$  percentage points (over  $20\times$  relative). Most of the shift is realized early, suggesting a quickly saturating response. This demonstrates that biased exposure during training can directly shape tool-selection preferences in favor of the exposed endpoint. However, since the target never approaches a dominant share, pre-training exposure explains only part of the bias, and additional factors still shape tool choice.

#### 4.4 CAN WE MITIGATE THE OBSERVED BIAS?

After demonstrating the existence and causes of bias, we seek to mitigate it. Our approach is simple: models often recognize which APIs can solve a certain task, but still exhibit skewed preferences among interchangeable endpoints. Hence, we propose to decouple recognition from selection via a lightweight debiasing module. This module uses a smaller LLM (Qwen3-14B), prompted to return only the subset of APIs from the candidate list that can solve the given query. We then choose uniformly at random from this subset, ensuring that each valid API has the same expected probability of selection. This eliminates positional or metadata-based favoritism while maintaining task coverage. See Appendix D for benchmark details and evaluation metrics. This section showcases the potential of the method and outlines how it can help.

**Subset selection avoids spurious inclusions of incorrect APIs while maintaining high coverage of correct APIs, thereby mitigating bias without sacrificing performance.** Table 2 summarizes results for Qwen3-14B as the subset selector. Overall micro-precision is  $\sim 1.00$  (0.9964), meaning the selector almost never adds tools to the candidate set that cannot solve the user’s task. This is a desirable property as it means that our mitigation module is unlikely to output an incorrect tool, and therefore, performance will not suffer. Note that since all ground-truth set size classes have nearly the same number of queries, micro- and macro-precision are effectively equivalent; we report micro-precision for simplicity. Micro-recall is  $\sim 0.89$  (0.8856), so on average the selector itself is not biased and retains most ground-truth tools, with an exact set match of 0.69 across all instances. Broken out by the size of the ground-truth set  $K$ , precision remains essentially perfect across the board, while recall varies: it is strongest at  $K=4$  (0.9633) and somewhat lower at  $K=2$  (0.7717) and  $K=5$  (0.8610). The corresponding exact-match rates reflect the same pattern (notably 0.9100 at  $K=4$ ). In practice, this means the subset filter very rarely introduces distracting tools (good for performance), but it can occasionally omit a true option when  $K$  is small or large.

<sup>2</sup>We use this model because its smaller size makes training tractable on modest hardware.

Table 2: Subset-selection performance for Qwen3 (14B) (overall and by ground-truth set size  $K$ ).

Overall		By ground-truth set size $K$				
		$K$	n	Precision	Recall	Exact Set Match
Micro-Precision	0.9964	2	300	1.0000	0.7717	0.5433
Micro-Recall	0.8856	3	200	0.9925	0.8850	0.7350
Exact Set Match Rate	0.6900	4	300	0.9940	0.9633	0.9100
		5	200	1.0000	0.8610	0.5350

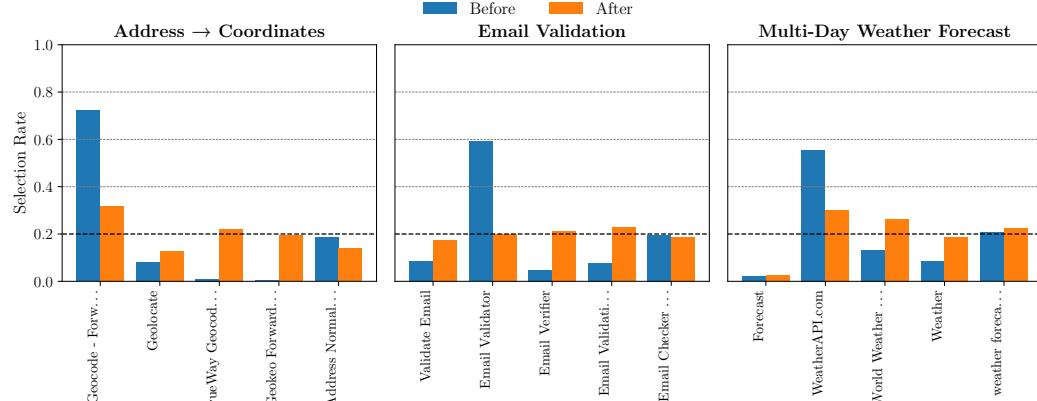


Figure 6: Selection distributions for GPT-4.1 mini with and without utilizing our mitigation method across three clusters. Each subplot corresponds to one cluster, with the x-axis indicating the API in the cluster and the y-axis showing the fraction of times that API was chosen using the corresponding setup over 500 inference runs.

Taken together, these results suggest subset selection is a promising first line of defense: it does not lead to spurious inclusions (high precision) and maintains high coverage of correct APIs on average, and thus, less bias downstream (strong recall).

**Our mitigation method successfully alleviates tool-selection bias.** Figure 6 shows the empirical selection distributions of GPT-4.1 mini before and after applying our mitigation method over three clusters of functionally equivalent APIs. Where choices were far from uniform before, after the mitigation method is applied we notice an even spread of selection share. However, in the weather forecasting cluster (right), we see that one API is still being underutilized, perhaps being an indication that the model has a difficult time using this API correctly even when it is the only API available. The effectiveness of the mitigation method is further exemplified in Table 5, where a steep decrease in all our bias metrics after applying the mitigation method can be observed.

## 5 CONCLUSION

In this paper, we introduced the first benchmark for evaluating tool selection bias in LLMs. Our results establish tool selection bias as a real and potentially significant issue for tool-augmented (agentic) LLMs, with implications for user experience, operational cost, and marketplace fairness. This study offers a concrete starting point for understanding and mitigating this bias.

**Limitations:** This study is limited by using 100 synthetically generated user queries per cluster and a narrow seven-feature set for selection-rate predictions, limited model/repeat coverage, and a focus on APIs from RapidAPI and English queries. Although our choices were constrained by compute (our setup already required  $\sim 500,000$  inference runs), these factors may induce variance and restrict generality.

486

**Future Work:** Future work should scale queries and clusters and enrich features with deeper semantic and structural signals to raise predictive power beyond the modest observed  $R^2$ . In addition, deploying more expressive models (e.g., boosted trees or deep nets) with cross-validation could capture higher-order interactions between tool features, further increasing their explanatory power. Lastly, broader replication across LLMs and runs would aid in quantifying variability.

491

492

493

494

495

496

497

498

499

500

501

502

503

504

505

506

507

508

509

510

511

512

513

514

515

516

517

518

519

520

521

522

523

524

525

526

527

528

529

530

531

532

533

534

535

536

537

538

539

## ETHICS STATEMENT

This research did not involve identifiable human data or animals and therefore did not require approval from an institutional ethics committee or review board. All experiments are conducted for scientific purposes only. The work does not involve or target any sensitive attributes such as gender, race, nationality, or skin color. Our study focuses on identify and mitigate tool-selection bias in LLM agents, with the aim of improving the trustworthiness and safety of LLM agents deployment.

## REPRODUCIBILITY STATEMENT

We have made every effort to ensure the reproducibility of our work. We provide detailed descriptions of data and experimental setup in Section 4. We have already included an anonymous version of our code in the public GitHub repository to facilitate replication.

**Additional note.** During initial experiments we observed an unusually large standard deviation in DeepSeek’s selection distributions for certain clusters. When we re-ran the experiments under the same configuration, this anomaly disappeared. Since we use the generic `deepseek-chat` endpoint for our LLM calls, which always serves the latest model version, one of the later runs was executed after a silent model update and thus, ran with a different model. All DeepSeek results reported in the current version of the paper are based on a fresh set of runs collected using the same model, which removes the anomalous error bars.

## REFERENCES

Anthropic. Meet Claude Anthropic, 2025. URL <https://www.anthropic.com/clause>.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Dylan Bouchard. An actionable framework for assessing bias and fairness in large language model use cases. *arXiv preprint arXiv:2407.10853*, 2024.

Sadia Sultana Chowdhury, Riasad Alvi, Subhey Sadi Rahman, Md Abdur Rahman, Mohaimenul Azam Khan Raian, Md Rafiqul Islam, Mukhtar Hussain, and Sami Azam. From language to action: A review of large language models as autonomous agents and tool users. *arXiv preprint arXiv:2508.17281*, 2025.

Sunhao Dai, Chen Xu, Shicheng Xu, Liang Pang, Zhenhua Dong, and Jun Xu. Bias and unfairness in information retrieval systems: New challenges in the ILM era. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2024.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng

594 Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing  
 595 Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen  
 596 Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong  
 597 Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu,  
 598 Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xi-  
 599 aoshua Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia  
 600 Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng  
 601 Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong  
 602 Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong,  
 603 Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou,  
 604 Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying  
 605 Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda  
 606 Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu,  
 607 Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu  
 608 Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforce-  
 609 ment learning. *arXiv preprint arXiv:2501.12948*, 2025.

610 Economize. AWS API Gateway Pricing and 5 Strategies for Cost Reduction, 2024. URL <https://www.economize.cloud/blog/aws-api-gateway-pricing/>.

611

612 Kazem Faghah, Wenxiao Wang, Yize Cheng, Siddhant Bharti, Gaurang Sriramanan, Sriram Bal-  
 613 subramanian, Parsa Hosseini, and Soheil Feizi. Gaming tool preferences in agentic llms. *arXiv*  
 614 *preprint arXiv:2505.18135*, 2025.

615

616 Isabel O. Gallegos, Ryan A. Rossi, Joe Barrow, Md. Mehrab Tanjim, Sungchul Kim, Franck Der-  
 617 noncourt, Tong Yu, Ruiyi Zhang, and Nesreen K. Ahmed. Bias and fairness in large language  
 618 models: A survey. *Comput. Linguistics*, 2024.

619

620 Google. Introducing Gemini: our largest and most capable AI model. 2023. URL <https://blog.google/technology/ai/google-gemini-ai/>.

621

622 GrandViewResearch. API Marketplace Market (2025 - 2030), 2025. URL  
 623 [https://www.grandviewresearch.com/industry-analysis/  
 624 api-marketplace-market-report](https://www.grandviewresearch.com/industry-analysis/api-marketplace-market-report).

625

626 Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang,  
 627 and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *The Tenth Interna-  
 628 tional Conference on Learning Representations, ICLR*, 2022.

629

630 Itay Itzhak, Gabriel Stanovsky, Nir Rosenfeld, and Yonatan Belinkov. Instructed to bias: Instruction-  
 631 tuned language models exhibit emergent cognitive bias. *arXiv preprint arXiv:2308.00225*, 2024.

632

633 Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang,  
 634 Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM*  
 635 *Computing Surveys*, 2023.

636

637 Keita Kurita, Nidhi Vyas, Ayush Pareek, Alan W Black, and Yulia Tsvetkov. Measuring bias in  
 638 contextualized word representations. In *Proceedings of the First Workshop on Gender Bias in  
 639 Natural Language Processing*, 2019.

640

641 Jiaxu Lou and Yifan Sun. Anchoring bias in large language models: An experimental study. *arXiv*  
 642 *preprint arXiv:2412.06593*, 2024.

643

644 Xiaoliang Luo, Akilles Rechardt, Guangzhi Sun, Kevin K. Nejad, Felipe Yáñez, Bati Yilmaz,  
 645 Kangjoo Lee, Alexandra O. Cohen, Valentina Borghesani, Anton Pashkov, Daniele Marinazzo,  
 646 Jonathan Nicholas, Alessandro Salatiello, Ilia Sucholutsky, Pasquale Minervini, Sepehr Razavi,  
 647 Roberta Rocca, Elkhan Yusifov, Tereza Okalova, Nianlong Gu, Martin Ferianc, Mikail Khona,  
 648 Kaustubh R. Patil, Pui-Shee Lee, Rui Mata, Nicholas E. Myers, Jennifer K. Bizley, Sebas-  
 649 tian Musslick, Isil Poyraz Bilgin, Guiomar Niso, Justin M. Ales, Michael Gaebler, N. Apurva  
 650 Ratan Murty, Leyla Loued-Khenissi, Anna Behler, Chloe M. Hall, Jessica Dafflon, Sherry Dongqi  
 651 Bao, and Bradley C. Love. Large language models surpass human experts in predicting neuro-  
 652 science results. *Nature Human Behaviour*, 2024.

648 Kanghua Mo, Li Hu, Yucheng Long, and Zhihao Li. Attractive metadata attack: Inducing llm agents  
 649 to invoke malicious tools. In *Advances in Neural Information Processing Systems (NeurIPS)*,  
 650 2025.

651

652 Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman,  
 653 Naveed Akhtar, Nick Barnes, and Ajmal Mian. A comprehensive overview of large language  
 654 models. *arXiv preprint arXiv:2307.06435*, 2024.

655 OpenAI. ChatGPT, 2025. URL <https://openai.com/chatgpt/overview/>.

656

657 Daniel E. O’Leary. Confirmation and specificity biases in large language models: An explorative  
 658 study. *IEEE Intelligent Systems*, 2025.

659

660 Pouya Pezeshkpour and Estevam Hruschka. Large language models sensitivity to the order of op-  
 661 tions in multiple-choice questions. *NAACL*, 2024.

662

663 Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru  
 664 Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein,  
 665 Dahai Li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master  
 666 16000+ real-world apis. In *The Twelfth International Conference on Learning Representations*,  
 667 *ICLR*, 2024.

668

669 Chang Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and  
 670 Ji-rong Wen. Tool learning with large language models: a survey. *Frontiers of Computer Science*,  
 671 2025.

672

673 RapidAPI. Monetizing your API on Rapidapi.com, 2025a. URL <https://docs.rapidapi.com/docs/monetizing-your-api-on-rapidapicom>.

674

675 RapidAPI. Nokia acquires Rapid technology and team!, 2025b. URL <https://rapidapi.com/>.

676

677 Timo Schick, Sahana Udupa, and Hinrich Schütze. Self-diagnosis and self-debiasing: A proposal for  
 678 reducing corpus-based bias in nlp. *Transactions of the Association for Computational Linguistics*,  
 679 2021.

680

681 Jiawen Shi, Zenghui Yuan, Guiyao Tie, Pan Zhou, Neil Zhenqiang Gong, and Lichao Sun. Prompt  
 682 injection attack to tool selection in llm agents. *arXiv preprint arXiv:2504.19793*, 2025.

683

684 Ziqi Wang et al. Eliminating position bias of language models: A mechanistic approach. *ICLR*,  
 685 2025.

686

687 Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi,  
 688 Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language  
 689 models. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh  
 690 (eds.), *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

691

692 Sheng-Lun Wei et al. Unveiling selection biases: Exploring order and token sensitivity in large  
 693 language models. *ACL*, 2024.

694

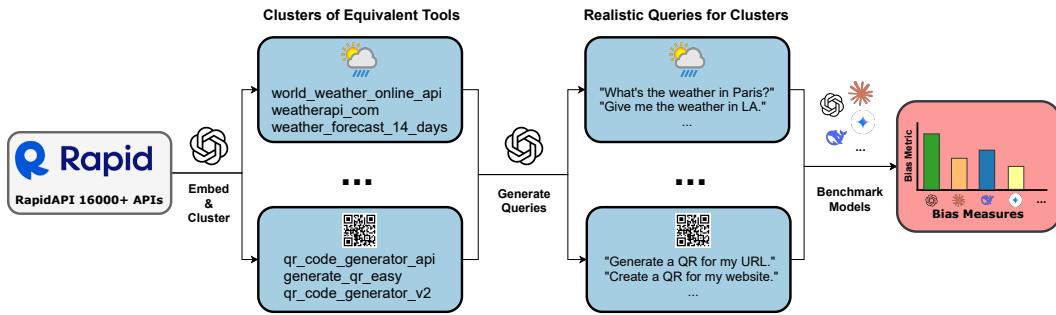
695 Xiaoyu Zhang, Juan Zhai, Shiqing Ma, Qingshuang Bao, Weipeng Jiang, Qian Wang, Chao Shen,  
 696 and Yang Liu. The invisible hand: Unveiling provider bias in large language models for code  
 697 generation. *Association for Computational Linguistics ACL 2025*, 2025.

698

699 Chujie Zheng et al. Large language models are not robust multiple choice selectors. *ICLR*, 2024.

700

701 Caleb Ziems, William Held, Jane Dwivedi-Yu, and Diyi Yang. Measuring and addressing indexical  
 702 bias in information retrieval. In *Findings of the Association for Computational Linguistics ACL*  
 703 2024, 2024.

702 A MORE DETAILS ON DATA GENERATION  
703  
704716 Figure 7: An overview of our clustering and query generation pipeline.  
717

718 We build on the “tool-usage” evaluation pipeline introduced by Qin et al. (2024), hereafter referred  
719 to as ToolLLM, owing to its wide adoption and extensibility. At its core, ToolLLM assembles a large  
720 catalog of real-world APIs scraped from RapidAPI spanning 49 functional categories (RapidAPI,  
721 2025b). For each API, ToolLLM provides a JSON file containing the API’s human-readable  
722 name, detailed description, and full parameter schema. In this work, we leverage exactly that API  
723 repository but restrict our attention to the stages in which ToolLLM selects among a short list of  
724 retrieved candidates. Note that in ToolLLM, a set of closely-related APIs is called a ‘tool’. For  
725 example, a geocoding tool could offer both forward geocoding and reverse geocoding APIs<sup>3</sup>.

726 We assemble our benchmark in two stages: (1) clustering APIs into functionally equivalent  
727 groups, and (2) generating realistic user queries for each group (see Figure 7).  
728

730 **Algorithm 1** Generation of Functionally-Equivalent API Clusters

731 **Require:** API id to metadata map  $\pi$ ,  
732 Precomputed embeddings  $E$ ,  
733 List of “general” APIs  $G = \{(tool, tool\_desc, api\_name, api\_desc)\}$ ,  
734 neighbor count  $K$ ,  
735 max outlier loops  $R$   
736 1:  $\mathcal{C} \leftarrow \emptyset$   
737 2: **for all**  $(tool, tool\_desc, api\_name, api\_desc) \in G$  **do**  
738 3:     Construct query text  $q = “tool:tool\_desc | api\_name:api\_desc”$   
739 4:     Embed  $q$  with ADA:  $\mathbf{v}_q \leftarrow \text{Embed}(q)$   
740 5:     Compute cosine similarities:  $s_i \leftarrow \cos(\mathbf{v}_q, E_i) \quad \forall i$   
741 6:     Select top- $K$  unique tools with largest similarity and store in set  $\text{TOP}_K$   
742 7:     candidate  $\leftarrow \{\pi[i] \mid i \in \text{TOP}_K\}$   
743 8:     **for**  $r \leftarrow 1$  **to**  $R$  **do**  
744 9:         Prompt GPT-4 to detect outliers: outliers  $\leftarrow \text{DetectOutliers}(\text{candidate})$   
745 10:        **if** outliers  $= \emptyset$  **then break**  
746 11:        **end if**  
747 12:        Remove outliers from candidate: candidate  $= \text{candidate} \setminus \text{outliers}$   
748 13:     **end for**  
749 14:     **if**  $|\text{candidate}| > 3$  **then**  
750 15:          $\mathcal{C} \leftarrow \mathcal{C} \cup \{\text{candidate}\}$   
751 16:     **end if**  
752 17: **end for**  
753 18: **return**  $\mathcal{C}$

754 <sup>3</sup>Geocoding is the process of converting between human-readable addresses and geographic coordinates:  
755 “forward” geocoding maps an address (e.g., “1600 Amphitheatre Parkway”) to its latitude/longitude, while  
“reverse” geocoding maps a given coordinate pair back to a structured postal address.

```

756 You are a prompt-writing assistant. I will give you a set of API
757 endpoints (tool name + description, endpoint name + description,
758 and potentially the required parameters) that all perform the same
759 underlying task. Please generate exactly  $\{n\}$  distinct, natural-
760 language user queries that could be satisfied by ALL of these
761 endpoints. **Include realistic sample values** for any required
762 parameters (e.g. use "https://example.com" for a URL, or "Hello
763 World" for a text field). Return them as a JSON array of strings,
764 with no extra commentary.
765
766 User:
767 Here are the endpoints:
768 - Tool: WeatherNow - Provides current weather information
    Endpoint: Current Weather - Returns temperature, humidity,
    and conditions for a given location.
769 Required parameters:
770     * city (string) - name of the city
    * country (string) - ISO country code
771 - ...
772
773

```

Figure 8: Example prompt used for query generation. The model outputs  $n$  natural-language queries that all listed endpoints can satisfy.

**API clustering.** We begin by embedding every endpoint’s metadata (tool name, API name, descriptions, etc.) into a shared vector space using a pre-trained text encoder (OpenAI’s text-embedding-ada-002 model). We then curate a small set of “seed” APIs whose descriptions span a number of “general” tasks, such as text translation or weather forecasting. For each seed, we retrieve its top- $K$  nearest neighbors in embedding space to form a candidate cluster. To ensure true functional equivalence, we iteratively prompt GPT-4 to flag any outlier endpoints that cannot perform the same task as the rest; flagged APIs are removed and the check repeats for up to a pre-defined number of rounds. Any cluster that stabilizes with more than three members is retained. See Algorithm 1 for an overview of our clustering approach. Lastly, we manually inspect and refine these clusters, yielding 10 high-quality groups of five APIs each.

**Query generation.** For each cluster, we prompt GPT-4 (see Figure 8) to produce natural-language queries that all members can satisfy. In batches of ten, the model generates candidate queries until we collect 100 unique queries per cluster, filtering out duplicates. In cases where freeform generation exhibits provider-specific bias (e.g., mentioning a particular vendor’s feature), we switch to a template-filling workflow: we design a small set of generic templates with placeholders (e.g. “Get the latest news headlines for  $\{\text{country}\}$  about  $\{\text{topic}\}$ .”), and ask GPT-4 to instantiate each template multiple times with realistic sample values.

**Final curation.** All 1,000 generated queries are then reviewed by hand to remove any that inadvertently favor a single provider or rely on specialized parameters. The resulting dataset consists of 10 clusters with 5 APIs each, and 100 balanced, provider-agnostic queries for each cluster.

Running each model over these prompts yields empirical selection distributions over APIs and list positions, from which we compute our total-variation-based bias metrics  $\delta_{\text{API}}$ ,  $\delta_{\text{pos}}$ , and  $\delta_{\text{model}}$ . This rigorously grounded benchmark enables precise measurement and comparison of tool-selection bias across models and settings.

## 802 B ATTRIBUTE-LEVEL ANALYSIS FEATURE TABLE

804 See Table 3 for the list of features used in the analysis of Section 4.3.

## 806 C MORE DETAILS ON THE BIASED CPT EXPERIMENT

808 We test whether pre-training data can cause tool-selection bias by doing biased continued pre-
809 training (CPT) on a single model. That is, we do additional next-token training on raw text using

810 Table 3: API-level predictor features.  
811

812 <b>Feature</b>	813 <b>Description</b>
814 avg_similarity_tool_desc	815 Mean text similarity between cluster queries and the tool’s de- 816 scription.
817 avg_similarity_api_desc	818 Mean text similarity between cluster queries and each API’s 819 description.
820 age_days	821 Days since the API was first published.
822 desc_name_length_sum	823 Total character count of the API’s name plus description.
num_params	824 Number of required and optional parameters.
flesch_reading_ease	825 Flesch reading-ease score of the combined descriptions.
positive_word_count	826 Count of positive or promotional words (e.g. “efficient,” “ro- 827 bust”).

828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000  
1001  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1020  
1021  
1022  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079  
1080  
1081  
1082  
1083  
1084  
1085  
1086  
1087  
1088  
1089  
1090  
1091  
1092  
1093  
1094  
1095  
1096  
1097  
1098  
1099  
1100  
1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118  
1119  
1120  
1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1130  
1131  
1132  
1133  
1134  
1135  
1136  
1137  
1138  
1139  
1140  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1160  
1161  
1162  
1163  
1164  
1165  
1166  
1167  
1168  
1169  
1170  
1171  
1172  
1173  
1174  
1175  
1176  
1177  
1178  
1179  
1180  
1181  
1182  
1183  
1184  
1185  
1186  
1187  
1188  
1189  
1190  
1191  
1192  
1193  
1194  
1195  
1196  
1197  
1198  
1199  
1200  
1201  
1202  
1203  
1204  
1205  
1206  
1207  
1208  
1209  
1210  
1211  
1212  
1213  
1214  
1215  
1216  
1217  
1218  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278  
1279  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296  
1297  
1298  
1299  
1300  
1301  
1302  
1303  
1304  
1305  
1306  
1307  
1308  
1309  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387  
1388  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1396  
1397  
1398  
1399  
1400  
1401  
1402  
1403  
1404  
1405  
1406  
1407  
1408  
1409  
1410  
1411  
1412  
1413  
1414  
1415  
1416  
1417  
1418  
1419  
1420  
1421  
1422  
1423  
1424  
1425  
1426  
1427  
1428  
1429  
1430  
1431  
1432  
1433  
1434  
1435  
1436  
1437  
1438  
1439  
1440  
1441  
1442  
1443  
1444  
1445  
1446  
1447  
1448  
1449  
1450  
1451  
1452  
1453  
1454  
1455  
1456  
1457  
1458  
1459  
1460  
1461  
1462  
1463  
1464  
1465  
1466  
1467  
1468  
1469  
1470  
1471  
1472  
1473  
1474  
1475  
1476  
1477  
1478  
1479  
1480  
1481  
1482  
1483  
1484  
1485  
1486  
1487  
1488  
1489  
1490  
1491  
1492  
1493  
1494  
1495  
1496  
1497  
1498  
1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508  
1509  
1510  
1511  
1512  
1513  
1514  
1515  
1516  
1517  
1518  
1519  
1520  
1521  
1522  
1523  
1524  
1525  
1526  
1527  
1528  
1529  
1530  
1531  
1532  
1533  
1534  
1535  
1536  
1537  
1538  
1539  
1540  
1541  
1542  
1543  
1544  
1545  
1546  
1547  
1548  
1549  
1550  
1551  
1552  
1553  
1554  
1555  
1556  
1557  
1558  
1559  
1560  
1561  
1562  
1563  
1564  
1565  
1566  
1567  
1568  
1569  
1570  
1571  
1572  
1573  
1574  
1575  
1576  
1577  
1578  
1579  
1580  
1581  
1582  
1583  
1584  
1585  
1586  
1587  
1588  
1589  
1590  
1591  
1592  
1593  
1594  
1595  
1596  
1597  
1598  
1599  
1600  
1601  
1602  
1603  
1604  
1605  
1606  
1607  
1608  
1609  
1610  
1611  
1612  
1613  
1614  
1615  
1616  
1617  
1618  
1619  
1620  
1621  
1622  
1623  
1624  
1625  
1626  
1627  
1628  
1629  
1630  
1631  
1632  
1633  
1634  
1635  
1636  
1637  
1638  
1639  
1640  
1641  
1642  
1643  
1644  
1645  
1646  
1647  
1648  
1649  
1650  
1651  
1652  
1653  
1654  
1655  
1656  
1657  
1658  
1659  
1660  
1661  
1662  
1663  
1664  
1665  
1666  
1667  
1668  
1669  
1670  
1671  
1672  
1673  
1674  
1675  
1676  
1677  
1678  
1679  
1680  
1681  
1682  
1683  
1684  
1685  
1686  
1687  
1688  
1689  
1690  
1691  
1692  
1693  
1694  
1695  
1696  
1697  
1698  
1699  
1700  
1701  
1702  
1703  
1704  
1705  
1706  
1707  
1708  
1709  
1710  
1711  
1712  
1713  
1714  
1715  
1716  
1717  
1718  
1719  
1720  
1721  
1722  
1723  
1724  
1725  
1726  
1727  
1728  
1729  
17210  
17211  
17212  
17213  
17214  
17215  
17216  
17217  
17218  
17219  
17220  
17221  
17222  
17223  
17224  
17225  
17226  
17227  
17228  
17229  
17230  
17231  
17232  
17233  
17234  
17235  
17236  
17237  
17238  
17239  
17240  
17241  
17242  
17243  
17244  
17245  
17246  
17247  
17248  
17249  
17250  
17251  
17252  
17253  
17254  
17255  
17256  
17257  
17258  
17259  
17260  
17261  
17262  
17263  
17264  
17265  
17266  
17267  
17268  
17269  
17270  
17271  
17272  
17273  
17274  
17275  
17276  
17277  
17278  
17279  
17280  
17281  
17282  
17283  
17284  
17285  
17286  
17287  
17288  
17289  
17290  
17291  
17292  
17293  
17294  
17295  
17296  
17297  
17298  
17299  
172100  
172101  
172102  
172103  
172104  
172105  
172106  
172107  
172108  
172109  
172110  
172111  
172112  
172113  
172114  
172115  
172116  
172117  
172118  
172119  
172120  
172121  
172122  
172123  
172124  
172125  
172126  
172127  
172128  
172129  
172130  
172131  
172132  
172133  
172134  
172135  
172136  
172137  
172138  
172139  
172140  
172141  
172142  
172143  
172144  
172145  
172146  
172147  
172148  
172149  
172150  
172151  
172152  
172153  
172154  
172155  
172156  
172157  
172158  
172159  
172160  
172161  
172162  
172163  
172164  
172165  
172166  
172167  
172168  
172169  
172170  
172171  
172172  
172173  
172174  
172175  
172176  
172177  
172178  
172179  
172180  
172181  
172182  
172183  
172184  
172185  
172186  
172187  
172188  
172189  
172190  
172191  
172192  
172193  
172194  
172195  
172196  
172197  
172198  
172199  
172200  
172201  
172202  
172203  
172204  
172205  
172206  
172207  
172208  
172209  
172210  
172211  
172212  
172213  
172214  
172215  
172216  
172217  
172218  
172219  
172220  
172221  
172222  
172223  
172224  
172225  
172226  
172227  
172228  
172229  
172230  
172231  
172232  
172233  
172234  
172235  
172236  
172237  
172238  
172239  
172240  
172241  
172242  
172243  
172244  
172245  
172246  
172247  
172248  
172249  
172250  
172251  
172252  
172253  
172254  
172255  
172256  
172257  
172258  
172259  
172260  
172261  
172262  
172263  
172264  
172265  
172266  
172267  
172268  
172269  
172270  
172271  
172272  
172273  
172274  
172275  
172276  
172277  
172278  
172279  
172280  
172281  
172282  
172283  
172284  
172285  
172286  
172287  
172288  
172289  
172290  
172291  
172292  
172293  
172294  
172295  
172296  
172297  
172298  
172299  
172300  
172301  
172302  
172303  
172304  
172305  
172306  
172307  
172308  
172309  
172310  
172311  
172312  
172313  
172314  
172315  
172316  
172317  
172318  
172319  
172320  
172321  
172322  
172323  
172324  
172325  
172326  
172327  
172328  
172329  
172330  
172331  
172332  
172333  
172334  
172335  
172336  
172337  
172338  
172339  
172340  
172341  
172342  
172343  
172344  
172345  
172346  
172347  
172348  
172349  
172350  
172351  
172352  
172353  
172354  
172355  
172356  
172357  
172358  
172359  
172360  
172361  
172362  
172363  
172364  
172365  
172366  
172367  
172368  
172369  
172370  
172371  
172372  
172373  
172374  
172375  
172376  
172377  
172378  
172379  
172380  
172381  
172382  
172383  
172384  
172385  
172386  
172387  
172388  
172389  
172390  
172391  
172392  
172393  
172394  
172395  
172396  
172397  
172398  
172399  
172400  
172401  
172402  
172403  
172404  
172405  
172406  
172407  
172408  
172409  
172410  
172411  
172412  
172413  
172414  
172415  
172416  
172417  
172418  
172419  
172420  
172421  
172422  
172423  
172424  
172425  
172426  
172427  
172428  
172429  
172430  
172431  
172432  
172433  
172434  
172435  
172436  
172437  
172438  
172439  
172440  
172441  
172442  
172443  
172444  
172445  
172446  
172447  
172448  
172449  
172450  
172451  
172452  
172453  
172454  
172455  
172456  
172457  
172458  
172459  
172460  
172461  
172462  
172463  
172464  
172465  
172466  
172467  
172468  
172469  
172470  
172471  
172472  
172473  
172474  
172475  
172476  
172477  
172478  
172479  
172480  
172481  
172482  
172483  
172484  
172485  
172486  
172487  
172488  
172489  
172490  
172491  
172492  
172493  
172494  
172495  
172496  
172497  
172498  
172499  
172500  
172501  
172502  
172503  
172504  
172505  
172506  
172507  
172508  
172509  
172510  
172511  
172512  
172513  
172514  
172515  
172516  
172517  
172518  
172519  
172520  
172521  
172522  
172523  
172524  
172525  
172526  
172527  
172528  
172529  
172530  
172531  
172532  
172533  
172534  
172535  
172536  
172537  
172538  
172539  
172540  
172541  
172542  
172543  
172544  
172545  
172546  
172547  
172548  
172549  
172550  
172551  
172552  
172553  
172554  
172555  
172556  
172557  
172558  
172559  
172560  
172561  
172562  
172563  
172564  
172565  
172566  
172567  
172568  
172569  
172570  
172571  
172572  
172573  
172574  
172575  
172576  
172577  
172578  
172579  
172580  
172581  
172582  
172583  
172584  
172585  
172586  
172587  
172588  
172589  
172590  
172591  
172592  
172593  
172594  
172595  
172596  
172597  
172598  
172599  
172600  
172601  
172602  
172603  
172604  
172605  
172606  
172607  
172608  
172609  
172610  
172611  
172612  
172613  
172614  
172615  
172616  
172617  
172618  
172619  
172620  
172621  
172622  
172623  
172624  
172625  
172626  
172627  
172628  
172629  
172630  
172631  
172632  
172633  
172634  
172635  
172636  
172637  
172638  
172639  
172640  
172641  
172642  
172643  
172644  
172645  
172646  
172647  
172648  
172649  
172650  
172651  
172652  
172653  
172654  
172655  
172656  
172657  
172658  
172659  
172660  
172661  
172662  
172663  
172664  
172665  
172666  
172667  
172668  
172669  
172670  
172671  
172672  
172673  
172674  
172675  
172676  
172677  
172678  
172679  
172680  
172681  
172682  
172683  
172684  
172685  
172686  
172687  
172688  
172689  
172690  
172691  
172692  
172693  
172694  
172695  
172696  
172697  
172698  
172699  
172700  
172701  
172702  
172703  
172704  
172705  
172706  
172707  
1

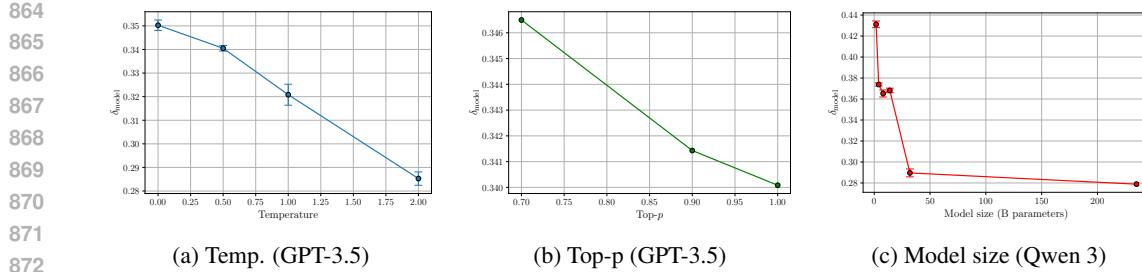


Figure 9: Sensitivity of the combined bias metric  $\delta_{\text{model}}$  to model hyperparameters. Each point is the mean over three independent runs (except for the top-p subplot); vertical bars show one standard deviation where available.

To evaluate this approach, we build a 1000-query benchmark with 8 API candidates per query and a ground truth set indicating which  $K \in \{2, 3, 4, 5\}$  APIs are sufficient ( $\sim 250$  items each). We report subset quality: precision, recall, and exact-set match. Note that the formula for exact-set match is given by  $\frac{1}{N} \sum_{i=1}^N \mathbf{1}[S_i = G_i]$  where  $G$  denotes the set of ground truth sets,  $S$  the set of selected subsets, and  $N$  is the number of queries. Additionally note that bias can persist if the subset selector itself is biased; underselecting viable tools (false negatives) or selecting irrelevant ones (false positives). Measures of recall and precision will tell us whether this is the case.

## E MORE ELABORATE ABLATION AND SENSITIVITY ANALYSIS

**Temperature.** Raising temperature reduces combined bias. As shown in Figure 9a, as temperature goes from 0 to 2, the mean  $\delta_{\text{model}}$  for GPT-3.5 drops from about 0.350 to 0.285, a 6.5% absolute reduction. Figure 10 makes clear why: the overall selection patterns remain similar across temperatures, but higher temperatures soften extreme preferences. This suggests that increased stochasticity slightly mitigates bias, but does not eliminate it.

**Top-p.** Figure 9b shows how the combined bias  $\delta_{\text{model}}$  for GPT-3.5 varies with the top-p cutoff. Increasing top-p from 0.7 to 1.0 yields a small decrease in bias (from  $\sim 0.346$  to  $\sim 0.340$ ), suggesting that less aggressive truncation of the probability distribution slightly softens extreme tool preferences. The effect is noticeably weaker than the temperature change.

**Model Size.** In Figure 9c, the combined bias  $\delta_{\text{model}}$  is depicted for Qwen 3 with varying model size. It seems that larger models exhibit less bias, with a notable drop at 32B. This pattern suggests that larger models develop more nuanced selection mechanisms which temper extreme preferences for certain APIs.

**API Ordering.** Figure 11 compares GPT-3.5’s API selection under two different ordering schemes: cyclic rotations versus random permutations. Across all clusters, the choice distribution is very similar: no API’s selection rate shifts more than about ten percentage points. This indicates that the ordering of the APIs has some influence, but the dominant signal is the model’s intrinsic preference. However, the small differences could also reflect the inherent noise from stochastic token sampling, and overall we argue that the tool-selection behavior is robust to either type of shuffling.

**System Prompts.** To evaluate how sensitive tool selection is to the phrasing and structure of the instructions given, we compare three variants of the system prompt: the original “Base” prompt, a lightly reworded “Similar” prompt, and a structurally different “Adjusted” prompt. Figure 12 shows the resulting distributions for GPT-3.5.

Prompt wording shifts model preferences but does not remove bias. Reworded prompts can amplify dominant choices and in some cases radically redistribute the selection shares. Elsewhere, effects are modest. Overall, framing and formatting can tilt the implicit ranking among functionally equivalent APIs, indicating that part of the observed bias is prompt-dependent even as a tendency to favor a subset of tools remains.



Figure 10: Selection distributions for GPT-3.5 using four different temperatures across ten clusters of functionally equivalent APIs. Each subplot corresponds to one cluster, with the x-axis indicating the API in the cluster and the y-axis showing the fraction of times that API was chosen by the respective model over 500 runs.

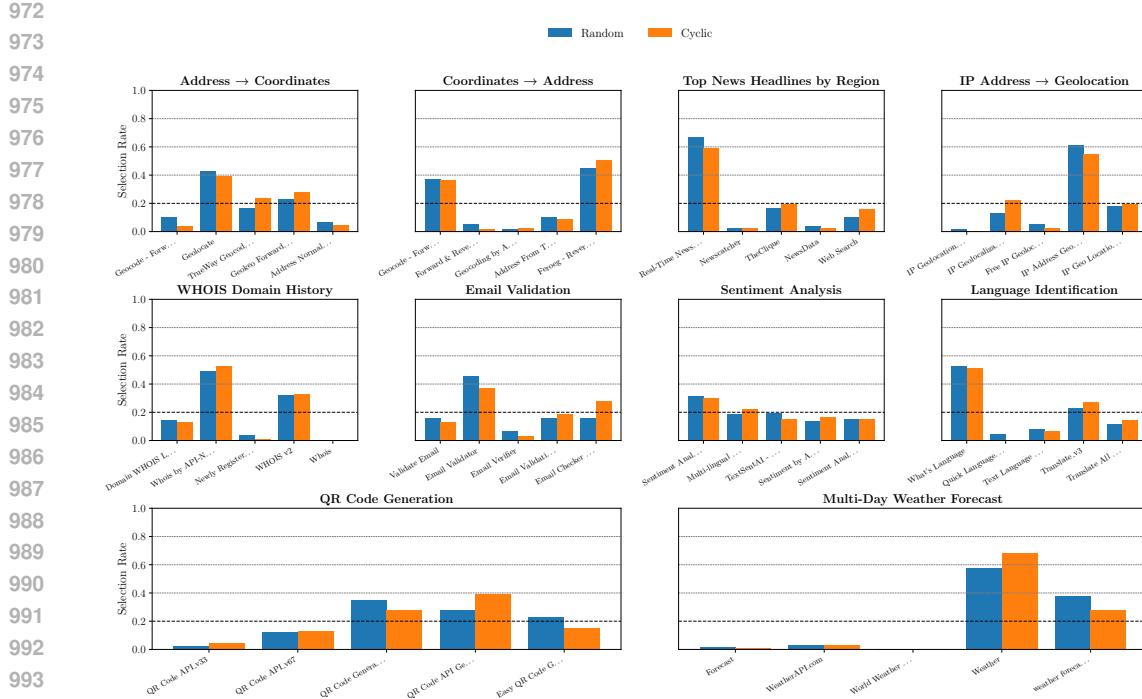


Figure 11: Selection distributions for GPT-3.5 using cyclic and random shuffling of the APIs across ten representative clusters. Each subplot corresponds to one cluster, with the x-axis indicating the API in the cluster and the y-axis showing the fraction of times that API was chosen for that specific API ordering method over 500 runs.

## F MORE ELABORATE DISCUSSION ON THE EXPLANATION OF BIAS

We now expand on the analysis given in the main text surrounding the investigation of bias. We expand on the feature-level analysis, where we try to predict selection rates according to intrinsic API attributes, and on the perturbation experiments that directly intervene on the API metadata to see which cues the models rely on during selection.

### F.1 WHICH API-LEVEL FEATURES PREDICT SELECTION RATES?

We extract a common set of descriptive features from every API (see Section 3.3) and mean-center them to investigate how being relatively high or low on a feature affects the API selection. These are then paired with the empirical selection rates yielding a dataset of 50 examples for each LLM. We then probe relationships between features and selection behavior in three ways. First, we compute Pearson correlations to capture linear and monotonic associations. Second, we fit a linear regression per model to quantify the aggregate explanatory power (reported as  $R^2$ ) and inspect coefficients to understand the direction and relative weight of each feature. Third, we train random-forest regressors with cross-validation to allow for non-linear interactions and obtain alternative measures of feature importance.

**Similarity between tool / API description and query is most correlated to selection rate.** As Table 4 makes clear, the most predictive feature of API selection is semantic similarity between the query and the tool/API descriptions. Both `avg_similarity_tool_desc` and `avg_similarity_api_desc` are consistently positively correlated with selection rates—especially strong for Qwen and clear for Gemini; GPT shows the same pattern, albeit weaker with higher p-values. By contrast, structural or stylistic attributes (e.g., parameter count, promotional wording) exhibit little consistent influence. Tool age (`age_days`) shows a modest, broadly consistent negative correlation across models.

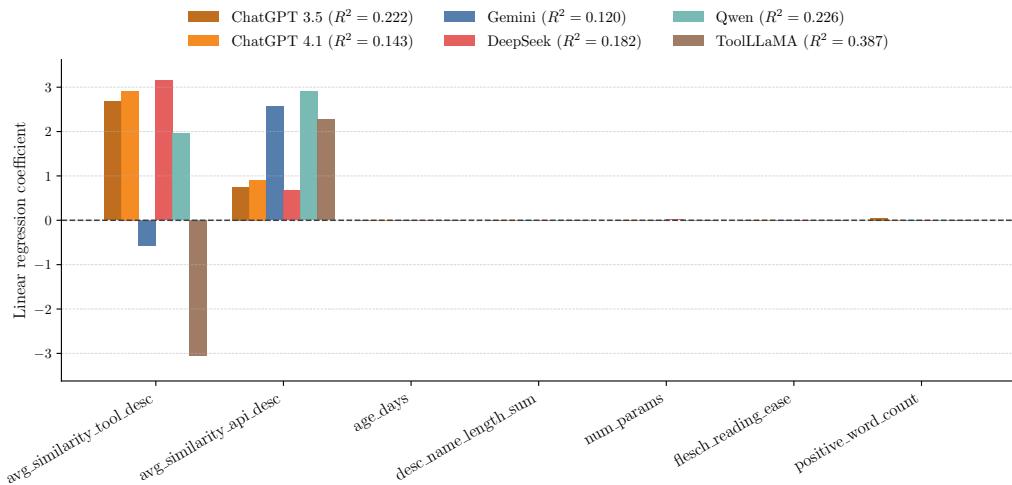


Figure 12: Selection distributions for GPT-3.5 using different system prompts across ten clusters. Each subplot corresponds to one cluster, with the x-axis indicating the API in the cluster and the y-axis showing the fraction of times that API was chosen using the corresponding system prompt over 500 runs.

1080 Table 4: Correlation between API-level features and model selection rates. Each entry shows Pearson  
 1081  $r$  with its  $p$ -value.  
 1082

1083 <b>Feature</b>	1084 <b>GPT-4.1 mini</b>	1085 <b>Gemini</b>	1086 <b>Qwen</b>
1087 avg_similarity_tool_desc	1088 +0.227( $p=0.113$ )	1089 $-0.092(p=0.526)$	1090 $+0.234(p=0.101)$
1091 avg_similarity_api_desc	1092 $+0.111(p=0.442)$	1093 $+0.330(p=0.019)$	1094 $+0.411(p=0.003)$
1095 age_days	1096 $-0.199(p=0.201)$	1097 $-0.144(p=0.356)$	1098 $-0.163(p=0.296)$
1099 desc_name_length_sum	1100 $+0.044(p=0.760)$	1101 $+0.103(p=0.477)$	1102 $+0.038(p=0.795)$
1103 num_params	1104 $-0.065(p=0.653)$	1105 $+0.038(p=0.793)$	1106 $-0.185(p=0.198)$
1107 flesch_reading_ease	1108 $+0.160(p=0.267)$	1109 $+0.176(p=0.222)$	1110 $+0.098(p=0.496)$
1111 positive_word_count	1112 $+0.126(p=0.384)$	1113 $+0.087(p=0.547)$	1114 $+0.093(p=0.521)$

1093 **Linear regression reveals that surface-level semantic alignment is the primary signal but leaves**  
 1094 **a lot unexplained.** Figure 13 tells us that linear models explain only part of the variance:  $R^2$   
 1095 is modest—0.143 for GPT-4.1 mini and 0.387 for ToolLLaMA—leaving substantial error. Coeffi-  
 1096 cients show surface-level semantic alignment dominates: similarity between the query and tool/API  
 1097 descriptions has the largest positive weights for most models, with Qwen weighting both most  
 1098 strongly and Gemini emphasizing API-level descriptions. Unexpectedly, ToolLLaMA gives a neg-  
 1099 ative weight to tool-description similarity. Other features contribute little. Hence, semantic align-  
 1100 ment is important in driving selection but still gives an incomplete explanation, implying nonlinear  
 1101 or omitted factors and motivating more flexible models (e.g., random forests).  
 1102



1115 Figure 13: Linear regression feature weights used to predict API selection rates for six LLMs. Each  
 1116 group of bars corresponds to one API-level feature; different colors denote models, with their  $R^2$   
 1117 shown in the legend. Larger positive weights indicate features that increase the predicted selection  
 1118 rate.  
 1119

1120 **Random forests fail to offer meaningful improvement.** We fitted random-forest regressors with  
 1121 the same mean-centered features using both cross-validation and a held-out split, but predictive per-  
 1122 formance was poor, meaning the forests often did worse than a trivial constant baseline. This sug-  
 1123 gests the available features, at least in their current form and scale, don’t contain enough signal or  
 1124 that some artifacts are overwhelming the gains from nonlinearity. Therefore, any feature-importance  
 1125 estimates from these trees would be unreliable and we do not lean on them for explanation. Future  
 1126 work could involve revisiting this with a richer feature set, more data, or alternative nonlinear mod-  
 1127 eling.  
 1128

1134 F.2 HOW DO METADATA INTERVENTIONS AFFECT API SELECTION?  
1135

1136 We saw that corrupting descriptions produces much larger and more stable effects on selection  
1137 behavior than name-level perturbations, which are sometimes noisy and unpredictable, in Section  
1138 4.3. Figure 14 corroborates this across clusters. Name perturbations often leave distributions near-  
1139 unchanged or can make them drift unpredictably (e.g., Cluster 1, where Cluster 1 is positioned at the  
1140 top-left and Cluster 6 at the bottom-right), whereas description/parameter scrambles frequently over-  
1141 haul rankings: sometimes amplifying the dominant API (Cluster 2), other times causing dramatic  
1142 re-ordering (Clusters 3). Name edits rarely produce comparably stable re-ranking.

1143 Together, these patterns indicate that description-level semantics (and, to a lesser extent, parameter  
1144 semantics) are the primary cues models use to discriminate among functionally similar APIs. Name  
1145 perturbation alone tend to inject noise without consistent effects. Finally, bias persists under minimal  
1146 semantic signal (only names and schema fields), implying selection behavior sometimes relies on  
1147 residual, non-obvious priors rather than solely on coherent, human-interpretable heuristics.

1148 **Manipulating the description of certain tools has mixed effects across clusters.** Figure 14 (lower  
1149 row) shows three behaviors when we manipulate descriptions. First, swapping the most- and least-  
1150 popular tools’ descriptions can invert their selection rates (Cluster 4), indicating description text  
1151 alone can dominate choice. Second, the same swap sometimes yields only a modest lift for the  
1152 least-popular tool while unexpectedly altering the selection shares of unaffected tools (Cluster 5),  
1153 suggesting the landscape is reconfigured rather than ranks simply exchanged. Third, in some clus-  
1154 ters the swap has minimal effect (Cluster 6), implying other cues—e.g., name priors or parameter  
1155 schemas—anchor preferences.

1156 Targeted corruption of the most-selected tool’s description has similarly inconsistent effects. In  
1157 Cluster 4, scrambling collapses its share to near zero as another tool absorbs the mass; in Clusters  
1158 5–6, corruption diffuses probability across competitors, producing a more even allocation. Over-  
1159 all, description tampering often yields substantial influence, but the impact is context-dependent:  
1160 the same intervention can invert, redistribute, or barely change preferences, underscoring that tool  
1161 choice emerges from multiple interacting cues.

1162 G ADDITIONAL FIGURES  
11631164 G.1 CORRELATION IN SELECTION BETWEEN MODELS  
1165

1166 Figure 15 shows that models exhibit varying degrees of correlation in their tool-selection patterns,  
1167 suggesting shared but non-identical biases across families.  
1168

1169 G.2 SELECTION DISTRIBUTIONS FOR ALL CLUSTERS  
1170

1171 This subsection provides a full version of the figure referenced in the main text (Figure 3). It expands  
1172 the subset plot to all ten clusters and keeps axes, run counts, and error-bar conventions identical to  
1173 the summary in Section 4.2. Use Figure 16 for detailed inspection of per-cluster behavior.  
1174

1175 G.3 FULL FIGURE RELATED TO THE CPT EXPERIMENT  
1176

1177 Figure 17 shows how biased continued pre-training gradually increases preference for the exposed  
1178 endpoint.  
1179

1180 G.4 TABLE RELATED TO THE REDUCTION OF BIAS DUE TO MITIGATION  
1181

1182 Table 5 demonstrates that our mitigation substantially flattens selection distributions and reduces  
1183 both API- and position-level bias.  
1184

1185 H EFFECT OF METADATA PERTURBATION ON BIAS  
1186

1187 Relative to the base distributions, both models move farther from uniform (get more biased)  
1188 when we lower semantic signal (see bars corresponding to the description + parameter and full

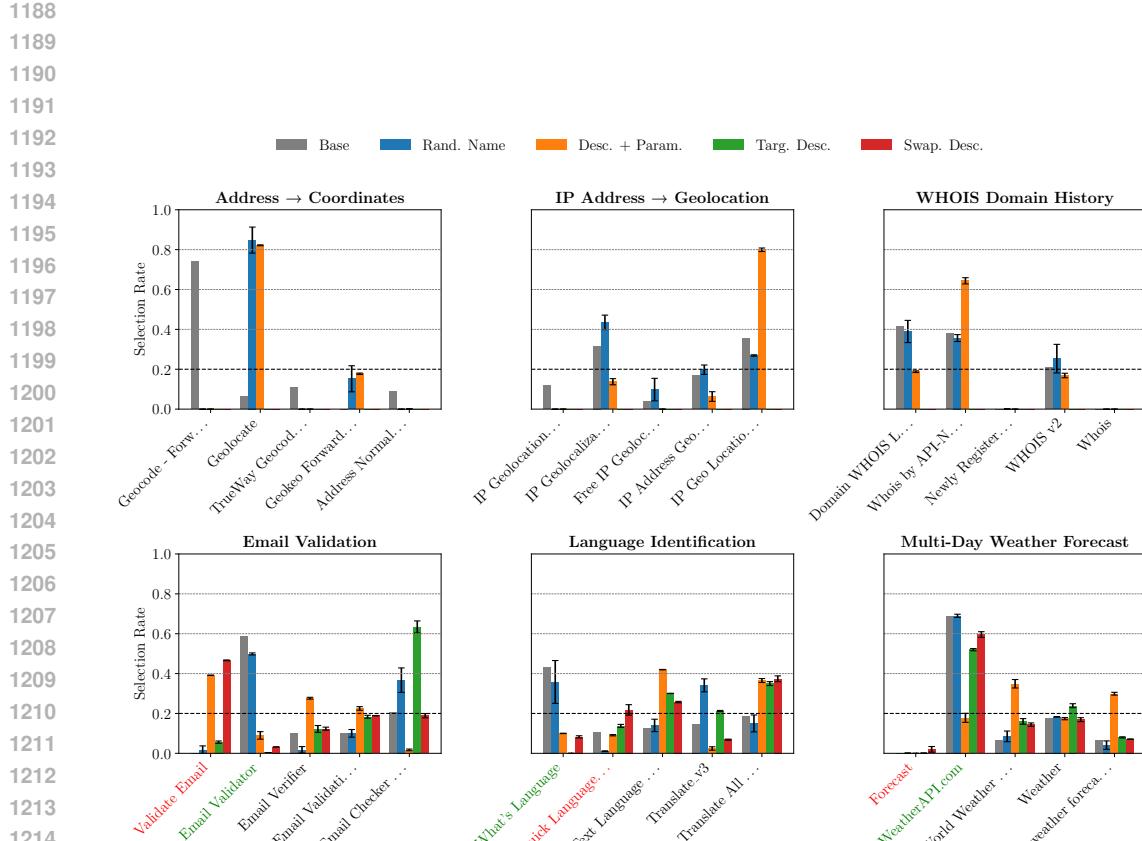


Figure 14: Selection distributions for Gemini under different name/ordering perturbations across six clusters of functionally equivalent APIs. Each subplot corresponds to one cluster; the x-axis lists the APIs and the y-axis shows the fraction of times the model under each condition selected that API, averaged over repeated runs. Error bars (when present) indicate the standard deviation across those repeats, making visible how robust or variable the preferences are under the different perturbations. Tools whose names are in green are the most selected by the baseline, and those in red are the least selected. These are the tools that are targeted for the swapping and selected scramble experiments.

Table 5: Average cluster-level API bias  $\delta_{\text{API}}$ , positional bias  $\delta_{\text{pos}}$ , and combined bias  $\delta_{\text{model}}$  before and after mitigation.

Setup	$\delta_{\text{API}}$	$\delta_{\text{pos}}$	$\delta_{\text{model}}$
<b>Before</b>	0.338	0.422	0.380
<b>After</b>	0.108	0.079	0.094

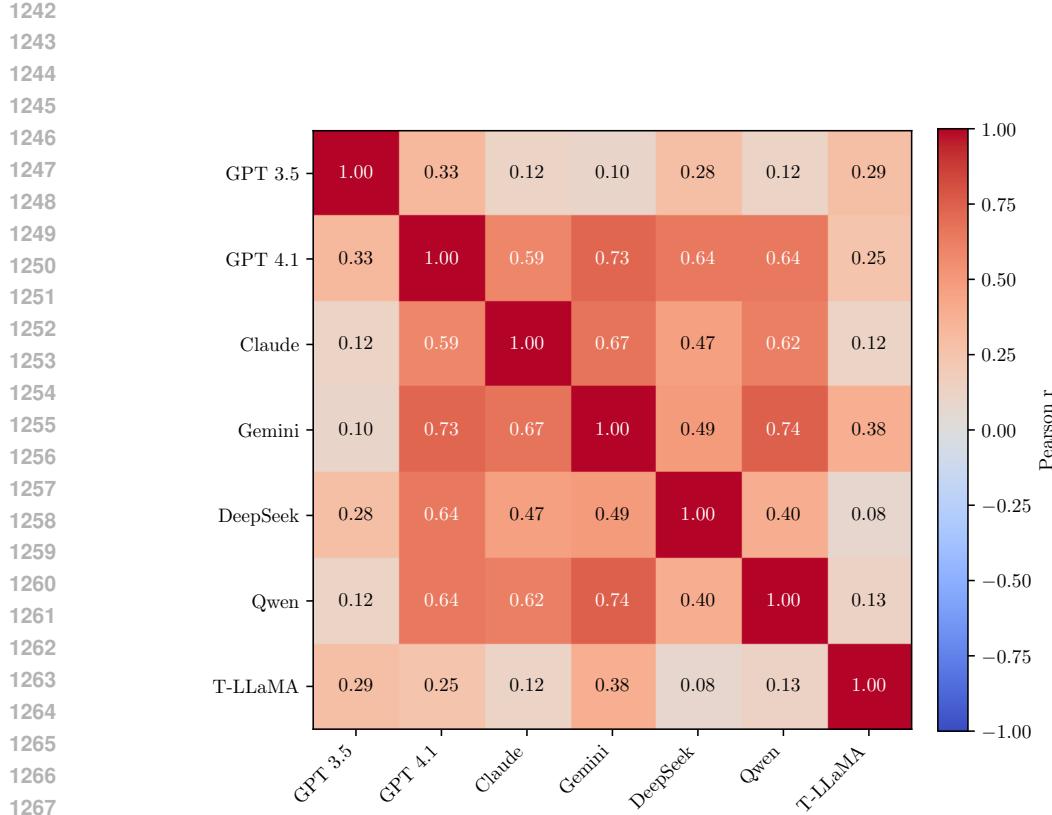


Figure 15: Pearson correlation matrix between models’ tool-selection bias patterns.

perturbations in Figure 18). For Gemini, these manipulations yield the largest TV distances to uniform ( $\approx 0.42$ – $0.43$ ); GPT shows similar results. This suggests that when descriptions/parameters are corrupted, models amplify bias rather than flatten choices.

Conversely, targeted edits to the most popular API tends to decrease bias. For Gemini, targeting the description of the most popular API leads to an average TVD slightly below the baseline and swapping the description between most- and least popular API leads to one that is substantially lower, indicating that weakening or transferring the strongest semantic cue moves the selection distribution toward uniform. Name-only manipulations have similar effects, but name scrambling does not increase bias as much.

## I DETAILS ON CPT SETUP

**Model and adapters.** We continue pre-training Qwen3–8B–Base using Unslot with 4-bit loading. The maximum sequence length is 2048. We attach LoRA adapters with  $\sim 16.29\%$  trainable parameters. See Table 6 for more info.

Table 6: Model/adapter configuration.

Base model	unslot/Qwen3–8B–Base–unslot–bnb–4bit
Max seq. length	2048
Quantization	4-bit (bitsandbytes)
LoRA hyperparameters	$r = 128$ , $\alpha = 32$ , dropout= 0



Figure 16: Selection distributions for six LLMs across ten clusters of functionally equivalent APIs. Each subplot corresponds to one cluster, with the x-axis indicating the API in the cluster and the y-axis showing the (mean) fraction of times each model chose that API over 500 inference runs; error bars indicate the standard deviation across three independent experimental runs. This visualization highlights how different models exhibit systematic preferences for some APIs.

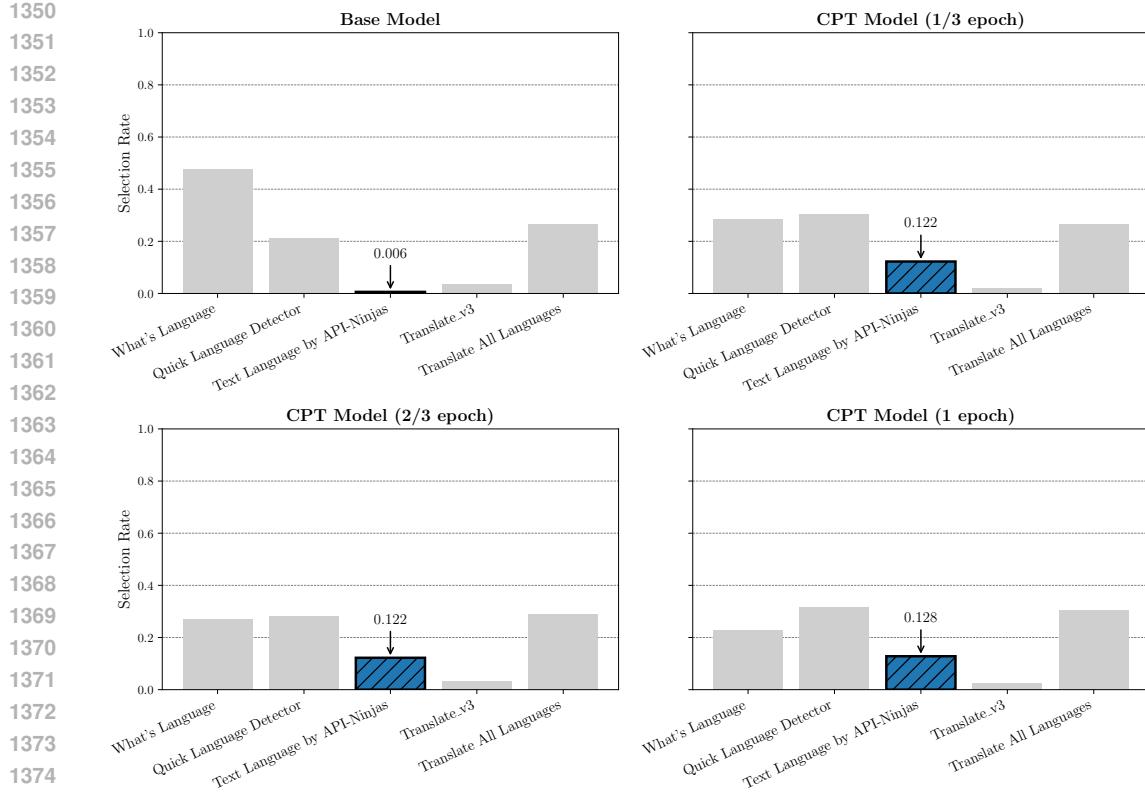


Figure 17: Selection rates for the Language Identification cluster across biased continued pre-training (CPT) checkpoints. Bars give the fraction that each endpoint was chosen across 500 inference runs. Panels show (top-left) base model, (top-right) CPT after 1/3 epoch, (bottom-left) 2/3 epoch, and (bottom-right) 1 full epoch. The Text Language by API-Ninjas endpoint is highlighted, with its exact selection rate printed above its bar. Differences across panels visualize how biased CPT shifts tool choice over training.

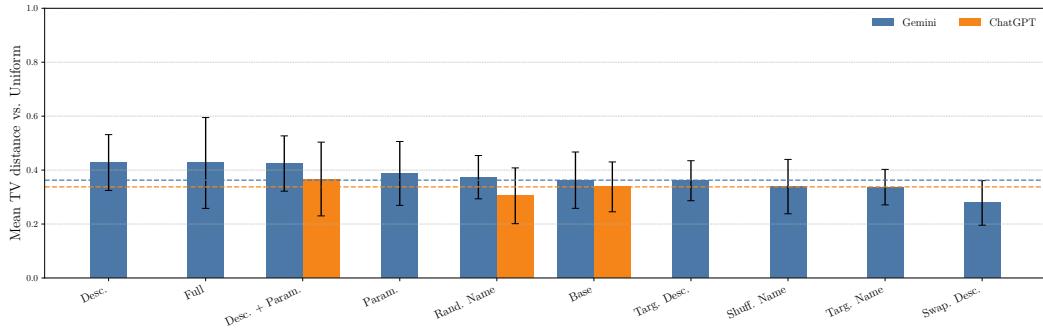


Figure 18: Mean total-variation (TV) distance from the uniform selection distribution to the distribution pertaining to each metadata perturbation (higher = more deviation from uniform). Blue bars show Gemini and orange bars show GPT; error bars denote standard deviation across clusters, and single bars indicate perturbations not run for GPT. Dashed horizontal lines (in the corresponding model colors) mark each model’s baseline TV-to-uniform without perturbations.

**Data.** We use our corpus saturated with metadata of a single target endpoint (Text Language by API-Ninjas). The corpus contains  $\sim 3.5$ M tokens.

1404  
 1405 **Training.** Training uses Unsloth’s trainer for one epoch with cosine LR scheduler and warmup.  
 1406 Optimizer is 8-bit AdamW. We also set a smaller LR for the embedding modules. See Table 7 for  
 1407 more info.

1408 Table 7: CPT training hyperparameters.  
 1409

1410	Epochs	1
1411	Total steps (epoch)	153
1412	Per-device batch size	2
1413	Grad. accumulation	8
1414	Effective batch size	16
1415	Learning rate	$5 \times 10^{-5}$ (embeddings $5 \times 10^{-6}$ )
1416	Scheduler / Warmup	cosine / warmup ratio 0.1
1417	Optimizer	adamw_8bit
1418	Weight decay	0.0
1419	Checkpoints used	step 0 (base), 52 ( $\approx 1/3$ ), 104 ( $\approx 2/3$ ), 153 (1 epoch)

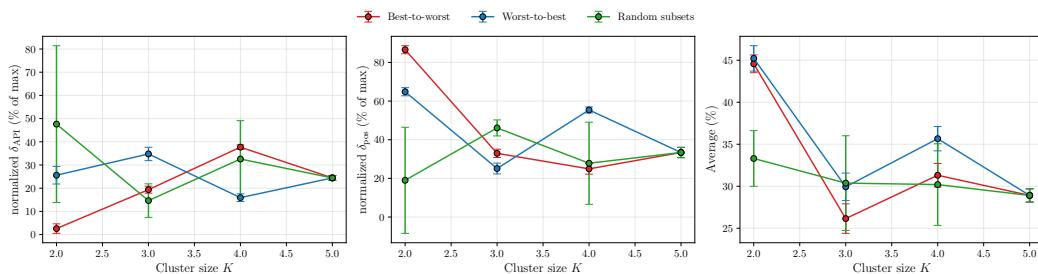
1420  
 1421 **Evaluation (inference).** For all checkpoints, we keep prompts and decoding fixed: temperature  
 1422 = 0.5, top-p = 1.0, and max\_new\_tokens=512. We evaluate with the Language Identification  
 1423 cluster under circular shifts, aggregating the selection rates over 500 inference runs per checkpoint.  
 1424

1425 J EFFECT OF TOOL COUNT ON BIAS  
 1426

1427 To investigate how the number of available tools affects selection bias, we conducted an additional  
 1428 experiment using the *Sentiment Analysis* cluster (which contains five functionally equivalent APIs).  
 1429 From this cluster, we constructed subsets of size  $K \in \{2, 3, 4\}$  using three selection strategies:  
 1430

- 1431 • **Best-to-worst:** select the  $K$  APIs that were most frequently chosen in our initial  $K = 5$   
 1432 experiments.
- 1433 • **Worst-to-best:** select the  $K$  least frequently chosen APIs.
- 1434 • **Random subsets:** uniformly sample  $K$  tools from the cluster.  
 1435

1436 For each subset configuration, we re-ran the corresponding queries three times using Qwen3 (235B)  
 1437 and computed the normalized API-level and positional bias metrics.  
 1438



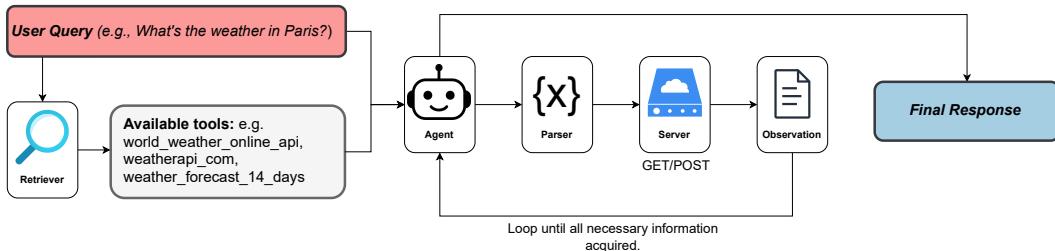
1448 Figure 19: Normalized API-level and positional bias as a function of cluster size  $K$  for the  
 1449 sentiment-analysis cluster. Each curve corresponds to one subset-selection strategy (best-to-worst,  
 1450 worst-to-best, random), with error bars indicating variability across three independent runs. Bias is  
 1451 reported as a percentage of its theoretical maximum ( $1 - 1/K$ ). The rightmost subplot shows the  
 1452 average of the normalized API-level and positional bias over  $K$ .  
 1453

1454 As can be seen in Figure 19, we find that bias is highly sensitive to the specific subset of tools shown  
 1455 to the model. At  $K = 2$ , both normalized  $\delta_{\text{API}}$  and  $\delta_{\text{pos}}$  vary substantially across selection strate-  
 1456 gies, revealing strong instability. At moderate subset sizes ( $K = 3, 4$ ), the variance decreases and  
 1457 the measured bias becomes more stable, although not uniformly smaller. Overall, the relationship

1458 between  $K$  and bias is non-monotonic: depending on *which* tools are included, reducing the number  
 1459 of available tools can either amplify or attenuate bias. These findings indicate that the composition  
 1460 of the toolset is the primary driver of selection behavior, more than its size alone.

1461 Finally, note that when  $K = 5$  all methods trivially converge, since they expose the full cluster and  
 1462 therefore share identical toolsets. To fully characterize tool-count effects, future work should extend  
 1463 this analysis to larger clusters (e.g.,  $K = 6\text{--}10$ ).

## 1465 K EXPLANATION OF LLM TOOL USAGE PIPELINE



1477 Figure 20: The overall workflow for tool usage with large language models. This figure illustrates  
 1478 the last three stages of tool usage: tool selection (via retriever and LLM selection), tool calling, and  
 1479 response generation (task planning is omitted). Note how the LLM’s preliminary outputs need to be  
 1480 parsed to create a functional API call.

1482 Large language models equipped with external tools typically follow a multi-stage decision and  
 1483 execution process. As summarized by prior work (Qu et al., 2025), the standard tool-usage pipeline  
 1484 consists of four stages: task planning, tool selection, tool calling, and response generation. Figure 20  
 1485 showcases the latter three of these stages.

1486 In the first stage, the LLM interprets the user query, identifies the user’s intent, and (if necessary)  
 1487 decomposes it into sub-tasks that can be handled by external tools. Next, in the tool selection stage,  
 1488 the model determines which tools are suitable for each sub-task. Modern systems often employ a  
 1489 retriever-based filtering step before LLM-based selection: rather than presenting the descriptions of  
 1490 hundreds or thousands of available tools to the model (which is the case with large API hubs like  
 1491 RapidAPI), a retriever identifies the top- $k$  most relevant candidates. This design is widely adopted  
 1492 in practical deployments due to context-length limitations (Qu et al., 2025).

1493 In the tool calling stage, the LLM chooses one of the retrieved tools and generates the corresponding  
 1494 API call, including the required parameters in the correct format. This step requires the model not  
 1495 only to select an appropriate tool but also to extract and structure the tool arguments accurately.  
 1496 Once the LLM generation is parsed, the tool executes and returns its output, the LLM may decide to  
 1497 proceed to the final stage or determine that more tools need to be called. In the final stage, the LLM  
 1498 integrates the returned tool results into a coherent final answer to the user.

## 1500 L CONSEQUENCES OF TOOL SELECTION BIAS

1502 We have briefly touched on the consequences of tool selection bias in the introduction. In this  
 1503 section, we will further elaborate our points on why tool selection bias matters to give the reader a  
 1504 comprehensive view of the the effects and significance of the bias we uncover.

### 1506 L.1 ECONOMIC CONSEQUENCES

1508 **Problem Statement.** A pay-per-request pricing model is common practice within tool marketplaces  
 1509 (see RapidAPI’s pricing model or that of BridgeAPI). This means that usage of an API directly  
 1510 corresponds to the amount of revenue the developer of that API makes. Ideally, when two or more  
 1511 tools offer identical functionality, usage (and thus revenue) should also be split equally. This is  
 not achieved when LLMs show consistent bias among functionality equivalent tools. This is not a

1512 hypothetical concern. As we show in our experiments, some tools gets selected up **10 times** more  
 1513 than others whilst being functionally identical (see the geocoding cluster in Figure 16). If revenue  
 1514 is directly correlated with usage, the developer of this tool can then also expect a 10 times higher  
 1515 revenue than the developer of the disadvantaged tool for seemingly no particular reason but more  
 1516 advantageous metadata phrasing (see section 4.3 investigating why tool-selection bias occurs).

1517 **Magnitude.** The size of the tool-usage market is hard to determine exactly. RapidAPI alone handles  
 1518 over 9 billion requests per month and recommends \$0.00003 per API call *at a minimum*, a quick  
 1519 calculation gives us at least \$270.000 in developer revenue per month or \$3.24 million per year  
 1520 on this platform alone (RapidAPI (2025a)). Looking more broadly, the global API marketplace  
 1521 market size was estimated at \$18.00 billion in 2024 and is projected to reach \$49.45 billion by 2030,  
 1522 growing at a CAGR of 18.9% from 2025 to 2030 (GrandViewResearch (2025)). From this, one can  
 1523 see that even small shifts in automated traffic can have a meaningful economic impact. It's unclear  
 1524 how much of this API traffic is currently due to LLM agents, but with the increasing ubiquity of  
 1525 agents, we expect this share to rise significantly. This makes tool selection bias a tangible and  
 1526 pressing economic issue.

## 1527 L.2 USER EXPERIENCE

1529 Tool-selection bias can directly degrade user experience when an LLM consistently favors an API  
 1530 that is objectively slower, less accurate, or more costly than its functionally equivalent alternatives.  
 1531 In such scenarios, end users may experience higher latency, lower-quality responses, or unnecessary  
 1532 costs, despite the existence of equally capable tools that would have delivered better performance. A  
 1533 more balanced usage distribution across equivalent APIs mitigates these issues by ensuring that no  
 1534 single suboptimal tool is disproportionately selected purely due to incidental metadata or positional  
 1535 biases. This leads to more stable, predictable, and higher-quality user outcomes.

## 1536 L.3 SAFETY & RELIABILITY

1539 Biased tool selection magnifies the system's vulnerability to manipulated metadata and adversarial  
 1540 tools. Recent work on metadata-poisoning attacks (Mo et al. (2025)) shows that adversaries can  
 1541 craft strategically misleading tool names or descriptions to lure LLM agents into invoking harmful  
 1542 or unreliable APIs. When an LLM already exhibits strong, unintended preferences toward superfi-  
 1543 cial metadata cues, such attacks become significantly easier to execute. In this sense, selection bias  
 1544 is not merely an efficiency problem. It increases the surface area for adversarial exploitation and  
 1545 directly undermines the reliability of agentic systems. A more uniform or semantics-invariant se-  
 1546 lection process would provide a stronger baseline defense by reducing the influence of manipulable  
 1547 metadata.

## 1548 L.4 EROSION OF TRUST & ECOSYSTEM EFFECTS

1550 A further, systemic consequence of tool-selection bias is the gradual erosion of trust in API market-  
 1551 places. If developers observe that LLM-mediated traffic does not meaningfully reflect the functional  
 1552 quality of their tools but instead hinges on arbitrary metadata preferences, they may perceive the  
 1553 marketplace as unfair or unpredictable. This creates incentives to bypass marketplaces entirely by  
 1554 hard-coding specific APIs into their applications. This outcome undermines the value proposition  
 1555 of marketplaces as neutral, competitive intermediaries. In the long term, this can reduce innova-  
 1556 tion, further distort competition, and create fragile ecosystems where a small number of arbitrarily  
 1557 preferred tools dominate traffic. Addressing tool-selection bias is therefore critical not only for indi-  
 1558 vidual user or developer outcomes, but for maintaining trust, participation, and healthy competition  
 1559 in the broader API economy.

1560  
 1561  
 1562  
 1563  
 1564  
 1565