# ADSTEER: NATIVE ADVERTISING AT DECODING TIME FOR LARGE LANGUAGE MODELS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Monetizing large language model (LLM) services through native advertising is essential for sustaining free or affordable access. Existing approaches mainly rely on prompt-level injection either multi-pass rewriting or finetuning, which compromise response quality or increase inference overhead. We propose **AdSteer**, a decoding-time advertising framework that steers language model's token generation, turning ads content into a natural part of the response. As a lightweight plug-in, AdSteer integrates ads more smoothly into the generation process, introducing appropriate sponsored mentions based on the context. Experiments on GEM-Bench across three application scenarios show that the best-performing AdSteer configuration in each scenario achieves an average improvement of 14.1% in overall score over the strongest baselines and an average of 25% fewer tokens than rewrite-based methods, demonstrating that AdSteer provides an effective and practical solution for native advertising in LLMs.

## 1 INTRODUCTION

Serving large language models (LLMs) at scale is expensive (Zhou et al., 2024). As demand grows, the gap between inference cost and sustainable revenue widens, making free or affordable access increasingly difficult to maintain (Brynjolfsson et al., 2019). Advertising has proven to be a reliable revenue model for consumer internet services (Goldfarb & Tucker, 2019), and it offers a promising path for LLM deployment: advertising revenue can subsidize inference cost, keeping services accessible. Yet transplanting advertising from a page-based medium to a generative one raises a fundamental challenge. Unlike web pages where advertisers purchase display slots, LLM responses are synthesized on the fly, and there is no pre-defined location or format for sponsored content. Balancing user satisfaction against advertiser revenue further calls for carefully designed incentive mechanisms (Dütting et al., 2024; Soumalias et al., 2025).

Efforts to bring advertising into this generative medium have followed two paradigms, each with complementary strengths and weaknesses. Prompt-level injection (Dubey et al., 2024; Hajiaghayi et al., 2024) is training-free and reflects catalog changes instantly, but offers no token-level control. Once sponsored content enters the context, generation is left to the model's discretion, so brand names may be hallucinated, misspelled, or paraphrased. Multi-pass rewriting (Xu et al., 2026) repairs such damage after the fact at the cost of repeated LLM calls. Finetuning (Zhao et al., 2025) takes the opposite route and learns to emit target tokens naturally, yet it binds the weights to a static snapshot of a highly dynamic catalog, and retraining a billion-parameter model against a daily-changing vocabulary is infeasible in both cost and engineering. Ad material rotates daily, while model weights are updated at least weekly, so intervention calls for occurring at decoding-time as a lightweight plug-in over a frozen LLM (Zou et al., 2025; Rimsky et al., 2024). Since an LLM generates text token by token, its sequential decoding process offers a natural intervention point: one can steer the next-token distribution toward sponsored content at selected steps without modifying model weights. Constrained decoding provides exactly this machinery, and applying it to advertising raises two further questions of *when* to step in and stop, and *how* to realize the sponsored keywords while keeping the response fluent.

The *when* question concerns the timing of an ad mention, both when steering should begin and when it should stop. A mention that starts too early interrupts the response, while engaging in too late can only be appended at the end, detached from the surrounding context. The *how* question concerns



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

how the sponsored content should be generated. Advertisers pay for particular keywords, such as brand names and ad slogans, and the response is expected to include them completely and exactly. However, forcing these words regardless of context breaks the fluency of the surrounding text and even further flood the response with ad content.

To address these challenges, we present **AdSteer**, a decoding-time advertising framework that places soft constraints on the generation of a frozen LLM. Concretely, our contributions are as follows:

- **A New Advertising-Generation Paradigm:** To our knowledge, we are the first to formalize LLM native advertising as a token-level adaptive steering problem, which enables natural, real-time ad integration into LLM inference—overcoming both the inflexibility of finetuning and the imprecision of prompt-level methods.
- **An Adaptive Ad-Integration Pipeline:** We propose *AdSteer*, a decoding-time framework that decouples online inference control from offline learning. At inference, a finite state machine (FSM) senses the lifecycle of each ad mention to help adjust generation. During training, SteerHead is learned from a self-supervised Fluency-Aware Proxy Reward that balances advertising effectiveness and response fluency.
- **Strong Empirical Improvements:** Evaluations on GEM-Bench demonstrate that the best-performing AdSteer configuration in each scenario achieves an average improvement of 14.1% in the overall score over the strongest baselines. Furthermore, AdSteer is inference-efficient, reducing total token consumption by approximately 25% compared to multi-pass rewrite-based methods.

## 2 Related Work

**Advertising in LLM services.** A growing line of work explores how sponsored content can be delivered through LLM responses. Most approaches operate at the prompt or context level: Hajiaghayi et al. (2024) inject retrieved ads into the generation context, and Dubey et al. (2024) have the LLM summarize multiple advertisers' content into its answer, while Feizi et al. (2026) survey this design space more broadly. Because the model alone decides how sponsored text is rendered, these methods offer little control over where an ad appears or whether brand names survive intact, and ads often surface as abrupt intrusions. Insertion-and-rewrite pipelines (Xu et al., 2026) repair such damage with additional LLM passes, multiplying inference cost. A different route modifies the model itself: Zhao et al. (2025) post-train the LLM to align its output distribution with an objective balancing ad revenue and user experience, but the backbone becomes tied to a fixed ad catalog and needs retraining whenever the catalog changes. AdSteer instead acts directly on the logits without modifying the backbone, avoiding both the fragility of prompt-level injection and the inflexibility of fine-tuning.

**Constrained decoding.** Constrained decoding intervenes in the decoding process so that the generated text satisfies externally given requirements, without retraining the model. Hard variants guarantee that required phrases appear in the output, typically by pruning the search space during beam search (Hokamp & Liu, 2017; Post & Vilar, 2018). Soft variants instead bias the next-token distribution, using attribute classifiers (Dathathri et al., 2020), stimulus prompts (Li et al., 2023), or proxy models (Liu et al., 2024), trading hard guarantees for fluency. AdSteer follows the soft direction, but its constraint is commercial rather than stylistic. Sponsored keywords are expected to be produced in full while the surrounding text stays fluent, which turns constrained decoding into a control problem of *when* to engage the constraint and *how strongly* to apply it, a question that existing methods do not address.

**Commercial intention retrieval.** To select relevant ads without brute-force matching, Liu et al. (2025) propose using LLM-generated commercial intentions as an intermediate semantic layer linking queries to ads, reporting large click-through and conversion gains in production. We adopt this retrieval paradigm as our upstream ad-selection module, while addressing the complementary question of *how* selected ads should be integrated into the output.

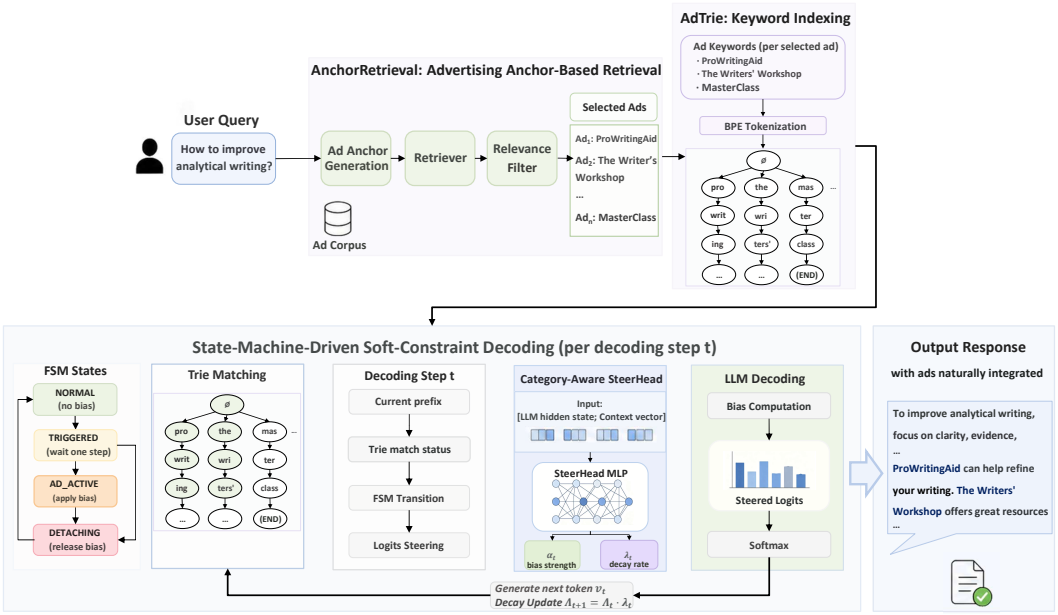


Figure 1: Overview of the AdSteer pipeline. The system retrieves relevant ads via Ad Anchors, constructs a keyword Trie, and steers LLM decoding through a state-machine-driven soft-constraint mechanism with category-adaptive bias control.

# 3 METHOD: ADSTEER

## 3.1 OVERVIEW

As a lightweight plug-in, AdSteer adds advertising to LLM generation, and its pipeline follows the two questions raised above. First, **AnchorRetrieval** matches the ads most relevant to the user's query. Given the substantial semantic gap between short queries and ad descriptions, AnchorRetrieval aligns them via intent phrases instead of raw text. Second, the keywords of the selected ads are structured into a prefix-tree index as the **AdTrie**, which supports token-level keyword matching during decoding, identifying at each step whether generation is entering, continuing, or leaving a keyword. Third, a four-state **finite state machine (FSM)** uses this index to decide when an ad mention should enter the response and when it should be abandoned. Finally, a learned **SteerHead** module adjusts how strongly the ad is pushed at each step, so that the ad appears naturally where the context allows while preserving fluency. Figure 1 illustrates the full pipeline.

## 3.2 ANCHORRETRIEVAL AND KEYWORD INDEXING

The **AnchorRetrieval** module first matches the user's query to the most relevant ads. Consider the query "how to improve analytical writing". Matching it directly against ad descriptions can easily miss the product ProWritingAid, described as "an AI-powered grammar checker and style editor", even though the product is exactly what the user needs. AnchorRetrieval closes this gap with an intermediate layer of **Ad Anchors**, short phrases that state an ad's commercial intent in the user's own language: ProWritingAid receives anchors such as "improve analytical writing" and "grammar checking tools", which now match the query directly. These anchors serve as a shared semantic layer between queries and ads.

Formally, Ad Anchors are prepared offline: each ad $a_i$ is associated with a generated anchor set $\Phi(a_i)$, whose elements are embedded into dense vectors $\mathbf{e}(\phi) \in \mathbb{R}^d$. At query time, query anchors $\Phi(q)$ are produced in the same manner, and candidate ads are retrieved via nearest neighbor search over anchor embeddings. The query's primary product category is determined once via embedding similarity against the product taxonomy and remains fixed for the entire generation trajectory.

Once relevant ads are retrieved, their keywords are organized into an **AdTrie**, a prefix tree built over token sequences. The trie tree matches a keyword token by token with a completed root-to-

terminal path, describing the position how far the mention has progressed. During decoding, the emitted tokens walk the Trie from its root and moves the current position along the matching edge, while a mismatch resets the walk to the root. Therefore, at every step the system identifies whether generation is entering, continuing, or leaving a keyword via AdTrie. Terminal nodes mark completed mentions together with their billing metadata. The keyword types and values stored at terminal nodes are deferred to the Appendix.

### 3.3 Soft-Constraint Decoding via Finite State Machine

With ad keywords indexed in the AdTrie, the remaining question is when an ad mention should enter the response. Constrained decoding offers a natural handle on this timing control, as the AdTrie already marks which tokens continue a keyword at each step. Hard constraints that enforce keyword completion, however, disregard the surrounding discourse and could distort the model's natural distribution. We therefore adopt a soft-constraint scheme realized by a four-state finite state machine over the AdTrie, which engages steering only when generation properly enters a keyword and disengages as soon as the keyword stops fitting, so that insertion timing follows the model's own discourse flow.

**Four-state design.** The FSM defines four decoding states, each motivated by a distinct control requirement:

1. **`NORMAL`** is the default state, in which the model generates on its own with no steering applied. In this state, the FSM only watches whether each emitted token matches the root of some keyword in the AdTrie.
2. **`TRIGGERED`** state enters when an emitted token matches the root of some keyword in the AdTrie. Such a match alone does not imply genuine keyword entry, since the same token may serve an unrelated role. For instance, the noun "shop" shares the exact first subword token with the brand term "Shopify". This state therefore acts as a validity gate, confirming entry only if the next token continues a valid keyword prefix and returning to `NORMAL` otherwise.
3. **`AD_ACTIVE`** is the state in which the ad is inserted. At each token prediction, a soft bias is added to the logits of the tokens that continue the keyword, with its strength adjusted at every step by the token controller described in the following section.
4. **`DETACHING`** is the state that ends an activation. Once the model diverges from the keyword prefix, or the accumulated steering strength has decayed below a threshold, all bias is suspended and the LLM resumes natural generation, preventing steering from producing excessive ad content.

Together, these states determine when steering engages, which tokens are eligible at each step, and when to let go.

### 3.4 Category-Aware SteerHead

Beyond this timing decision, the strength and persistence of the bias call for two additional per-step quantities, the steering strength $\alpha$ and the decay factor $\lambda$. Their values, however, need to adapt at every step to the current decoding context and product domain, since a moment that welcomes an ad mention in one response may be unsuitable in another. We therefore predict both adaptively at each step with a lightweight module, the **SteerHead**. It strengthens the bias when the context is receptive to an ad mention, fades it promptly once continued steering would distort the output, and adapts both signals to the product category.

The outputs of SteerHead act on decoding at the points marked by the FSM. In the `AD_ACTIVE` state, the steered $\text{logits}'[v]$ at decoding step $t$ are adjusted as follows:

$$\text{logits}'[v] = \begin{cases} \text{logits}[v] + \Lambda_t \cdot \alpha_t & v \in \mathcal{V}_{\text{target}} \\ \text{logits}[v] & \text{otherwise} \end{cases} \tag{1}$$

where $\mathcal{V}_{\text{target}}$ denotes the set of valid continuation tokens in the current Trie prefix and $v$ an arbitrary token in the vocabulary. The per-step strength $\alpha_t$ scales the push at the current step. The accumulator $\Lambda_t = \prod_{s<t} \lambda_s$ multiplies the decay factors $\lambda_s$ predicted at the earlier steps of the same activation. This formulation yields a controlled engage-then-fade behavior. Intuitively, short keywords benefit

from a faster decay rate, allowing steering to exit promptly once the keyword is complete; long keywords, by contrast, can be fully generated without mid-sequence detachment with a gradual decay schedule.

**Dual-output design.** SteerHead is a lightweight MLP that predicts the two scalars from two sources of information: the LLM backbone's last-layer hidden state $h_t$, a latent representation of the text generated so far, and a context vector $c_t$, which features where the steering process currently stands. SteerHead maps these inputs to the two scalars as follows:

$$\alpha_t = \mathrm{Softplus}(W_\alpha\, z_t + b_\alpha) \tag{2}$$

$$\lambda_t = \mathrm{Sigmoid}(W_\lambda\, z_t + b_\lambda) \tag{3}$$

where $z_t = \mathrm{GELU}(W_{\mathrm{shared}}[h_t;\, c_t] + b_{\mathrm{shared}})$ is the shared trunk feature. The two outputs play complementary roles. $\alpha_t$ controls the strength of the bias at the current step, with larger values raising the logits of the keyword's continuation tokens by a wider margin. $\lambda_t$ controls how long the bias persists. Values near one keep $\Lambda_t$ high across steps and support multi-token keywords through completion, while smaller values let $\Lambda_t$ decay quickly and end the steering soon after a mention.

**Context vector.** Concretely, $c_t$ carries three groups of signals, each capturing further steering context beyond LLM's hidden state. *State signals* encode the current FSM state, allowing SteerHead to adjust $\alpha$ and $\lambda$ according to whether the model is freely generating, entering a keyword, actively steering, or disengaging. *Progress signals* capture how far the current activation has proceeded, including the accumulated decay, the number of steps in the current activation, and the remaining tokens to the nearest keyword boundary; these prevent over-steering late in a long keyword or premature exit from a short one. *Keyword signals* encode the keyword type and current prefix length, allowing SteerHead to handle different keyword types and lengths differently. The full specification of $c_t$ is provided in the Appendix.

**CategorySteerBank.** The signals above adapt steering to the decoding context, but not to the advertised product itself. The steering that completes a mention well varies across product categories: a two-token brand name is completed with one brief push, whereas a longer descriptive phrase benefits from gentler support spread over many steps. A single SteerHead cannot distinguish these regimes, blending them into one compromise policy that fits none of them well. We therefore organize SteerHeads into a **CategorySteerBank** with two tiers. **Category Heads**, one per product category, each specialize in the steering behavior of their own category. A single **General Head**, trained on trajectories pooled from all categories, covers queries whose category lacks sufficient training data for a specific Category Head. All heads share an identical architecture. At inference time, steering is executed by the Category Head corresponding to the ad's category, provided the category falls within a data-sufficient scenario where a dedicated head can be properly trained. Otherwise, the steering falls back to the General Head with an even steering behavior for all categories. This two-tier design combines category-level specialization with full coverage: it delivers category-aware steering in data-rich situations while preserving robust, reliable performance in data-scarce cases.

# 4 SteerHead Training Strategy

## 4.1 Two-Stage Category Bank Training

Training the CategorySteerBank follows its two-tier structure, first fitting the General Head on the pooled trajectories of all categories and then specializing Category Heads with enough training data:

**Stage 1: General pre-training.** The General Head is fit on all trajectories from all categories, learning general steering behavior shared across domains. Pooling data across categories maximizes the number of training trajectories and exposes the head to diverse query types and keyword structures.

**Stage 2: Per-category fine-tuning.** For categories with enough trajectories, the General Head weights are copied to initialize a Category Head, which is then fine-tuned on category-filtered data. Otherwise, AdSteer falls back to the General Head for those categories with insufficient data.

This two-stage procedure ensures broad coverage, since every query receives at least the general head, while popular categories obtain customized steering policies.

For training strategy, both stages optimize the same rule-based objective, which we construct in two steps: a fluency-aware proxy reward computed from decoding traces and a regression objective that distills trajectory-level rewards into per-step supervision.

### 4.2 Fluency-Aware Proxy Reward

Without steering ground-truth, we train SteerHead with a rule-based reward computed directly from the decoding trace without human labels. Following the outcome-supervision paradigm (Cobbe et al., 2021), we collect generation trajectories under a fixed baseline policy: for each query $q$, a response $o = (o_1, \ldots, o_T)$ is sampled while recording $(h_t, c_t, \tilde{\alpha}_t, \tilde{\lambda}_t)$ at every step. Each trajectory is then assigned a scalar reward from its final outcome, combining ad effectiveness, integration naturalness, and distributional fidelity.

The *hit reward* $r_{\text{hit}}$ measures the commercial value realized per generated token:

$$r_{\text{hit}} = \frac{1}{T} \sum_{k \in \mathcal{K}_{\text{hit}}} w_k \tag{4}$$

where $\mathcal{K}_{\text{hit}}$ denotes the set of hit Trie keywords deduplicated by ad ID and keyword, and $w_k$ is the commercial value of keyword $k$, assigned according to its type. To prevent the reward from biasing toward verbose outputs that accumulate excessive mentions, the reward is normalized by the response length.

To maintain coherence during keyword generation, the *naturalness reward* $r_{\text{nat}}$ measures alignment between steered tokens and the LLM's original distribution:

$$r_{\text{nat}} = \frac{1}{|\mathcal{T}_s|} \sum_{t \in \mathcal{T}_s} \left(1 - \frac{\text{rank}_t}{|\mathcal{V}|}\right) \tag{5}$$

where $\mathcal{T}_s$ is the set of steered steps, $\text{rank}_t$ is the rank of the emitted token $o_t$ under $P_t$, the natural next-token distribution at step $t$ before bias application, and $|\mathcal{V}|$ is the vocabulary size. $r_{\text{nat}}$ approaches one when steering pushes toward tokens the model already favors, yielding coherent integration.

Finally, the *fidelity penalty* $r_{\text{kl}}$ measures the average forward KL divergence induced by the bias at the level of the full output distribution:

$$r_{\text{kl}} = \frac{1}{|\mathcal{T}_s|} \sum_{t \in \mathcal{T}_s} \mathbb{D}_{\text{KL}}\left[P_t \,\|\, P'_t\right] \tag{6}$$

where $t$ indexes the steered steps in $\mathcal{T}_s$, and $P'_t$ denotes the next-token distribution after bias application. This distribution-level penalty complements the token-level naturalness reward: a steered token can rank high while the overall distribution still shifts substantially.

**Role-specific control signals.** From the reward components, we derive:

$$\sigma_\alpha = r_{\text{hit}} + r_{\text{nat}} \tag{7}$$

$$\sigma_\lambda = \frac{1}{1 + r_{\text{kl}}} \tag{8}$$

For $\alpha$, the hit reward directly drives steering strength: more keyword realization encourages stronger insertion. For $\lambda$, low distributional distortion sustains steering longer by slowing the decay, while high KL triggers faster exit, directly targeting fluency preservation.

### 4.3 Outcome-Scaled Target Construction

Trajectories are collected with fixed bias and decay values, so the recorded $\tilde{\alpha}_t$ and $\tilde{\lambda}_t$ are preset constants rather than learned predictions. After each trajectory completes, the outcome signals in

Eq. 7 and Eq. 8 yield trajectory-level scale factors $\gamma_\alpha = \sigma_\alpha / \bar{\sigma}_\alpha$ and $\gamma_\lambda = \sigma_\lambda / \bar{\sigma}_\lambda$, where $\bar{\sigma}$ denotes the batch mean. These factors rescale the recorded control trace into supervised targets:

$$\alpha_t^* = \tilde{\alpha}_t \cdot \gamma_\alpha \tag{9}$$

$$\lambda_t^* = \begin{cases} \tilde{\lambda}_t + (1 - \tilde{\lambda}_t)\big(1 - \frac{1}{\gamma_\lambda}\big), & \gamma_\lambda \geq 1 \\ \tilde{\lambda}_t \cdot \gamma_\lambda, & \gamma_\lambda < 1 \end{cases} \tag{10}$$

For $\alpha$, high-reward trajectories are scaled up to encourage stronger steering. For $\lambda$, the asymmetric formula drives $\lambda$ toward 1 to induce slow decay for successful trajectories, and contracts it toward 0 to enable fast termination for unsuccessful ones.

### 4.4 Loss Function

Each SteerHead, including individual heads within the CategorySteerBank, is trained with joint Smooth-$L_1$ regression over the predictions $\hat{\alpha}_t, \hat{\lambda}_t$ and outcome-derived targets $\alpha_t^*, \lambda_t^*$:

$$\mathcal{L} = \mathcal{L}_{\text{SmoothL1}}(\hat{\alpha}_t, \alpha_t^*) + \mathcal{L}_{\text{SmoothL1}}(\hat{\lambda}_t, \lambda_t^*) \tag{11}$$

This objective explicitly decouples insertion aggressiveness from decay pacing, which is central to robust control.

## 5 Experiments

### 5.1 Setup

**Datasets.** We evaluate AdSteer on three distinct datasets from GEM-Bench (Hu et al., 2026): **MT-Human** comprises chatbot queries drawn from MT-Bench humanities, **LM-Market** includes chatbot queries from LMSYS-Chat-1M with high ad injection potential, and **CA-Prod** contains search engine queries with real commercial query ad pairs and human relevance annotations.

**Metrics.** We adopt an LLM-as-a-Judge protocol and evaluate ad-integrated responses with six metrics spanning user satisfaction and user engagement. For satisfaction, Accuracy reflects the relevance and correctness of the response, and Naturalness measures whether the ad interrupts the flow and reads authentically. Personality captures helpfulness and salesmanship, while Trust reflects credibility and the absence of bias. For engagement, Notice and Click measure whether the ad is noticed and whether it invites further action. These metrics are designed to mimic how real users perceive and interact with sponsored content in practice, with each metric rated along two sub-dimensions and mapped to discrete levels for consistent judgment.

**Baselines.** We compare against the four baselines provided by GEM-Bench (Hu et al., 2026).

- **Ad-Chat**: A prompt-injection baseline that concatenates the retrieved ads into the system prompt and leaves it to the model to mention an ad on its own.
- **GI-R**: The basic pipeline baseline, which generates an ad-free response and then inserts a retrieved ad sentence at the position that least disrupts the local semantic flow.
- **GIR-R**: Extends GI-R by retrieving the ad based on the generated response and adding a rewriting pass after insertion, so that the ad blends more smoothly into the surrounding text at the cost of an extra LLM call.
- **GIR-P**: Follows the same generate-insert-rewrite pipeline as GIR-R, but retrieves the ad based on the user prompt rather than the generated response.

**AdSteer variants.** We evaluate eight configurations spanning category granularity from zero to 127 heads:

- **AdSteer-General (AdSteer-G)**: The single head variant, which relies solely on the General Head trained on trajectories pooled from all categories.
- **AdSteer-Category-X (AdSteer-C-X)**: The full CategorySteerBank variant with $X \in \{1, 5, 7, 10, 16, 39, 127\}$ Category Heads, each initialized from the General Head and fine tuned on trajectories filtered to its own category.

Table 1: Main results on GEM-Bench, judged by GPT-5.4-mini. AdSteer consistently outperforms baseline approaches across most metrics, achieving superior overall performance. Accuracy, Naturalness, Personality, Trust, Notice, and Click are abbreviated as Acc, Nat, Per, Trs, Not, and Cli, respectively. Ov denotes Overall, computed as their average.

| | MT-Human | | | | | | | LM-Market | | | | | | | CA-Prod | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Method** | Acc | Nat | Per | Trs | Not | Cli | Ov | Acc | Nat | Per | Trs | Not | Cli | Ov | Acc | Nat | Per | Trs | Not | Cli | Ov |
| Ad-Chat | 78.8 | 52.5 | 67.5 | 63.8 | 75.0 | 78.0 | 69.3 | 58.6 | 60.2 | 62.8 | 54.6 | **82.3** | 72.8 | 65.2 | 55.0 | 52.5 | 41.5 | 36.3 | 86.3 | **83.5** | 59.2 |
| GI-R | 78.8 | 33.8 | 75.0 | 56.3 | 63.8 | 63.8 | 61.9 | 69.1 | 45.3 | 74.7 | 68.5 | 70.3 | 72.8 | 66.8 | 51.3 | 32.5 | 44.0 | 34.8 | 82.8 | 80.5 | 54.3 |
| GIR-R | 82.5 | 63.8 | 78.8 | 63.8 | 63.8 | 63.8 | 69.4 | 71.0 | 68.4 | 73.4 | **72.4** | 74.7 | 72.0 | 72.0 | 60.8 | 37.5 | 53.3 | 37.5 | 85.3 | 80.8 | 59.2 |
| GIR-P | 75.0 | **75.0** | 75.0 | 75.0 | 78.8 | 75.0 | 75.6 | 71.5 | 65.8 | **74.8** | 69.4 | 74.7 | 70.6 | 71.1 | 57.3 | 36.5 | 52.0 | 36.3 | 83.3 | 78.5 | 57.3 |
| AdSteer-G | 84.0 | **75.0** | **81.0** | **81.0** | 87.0 | **84.0** | **82.0** | 70.8 | 66.8 | 70.4 | 69.4 | 79.2 | 71.0 | 71.3 | 74.8 | 74.8 | 62.0 | 77.3 | 82.0 | 80.8 | 75.3 |
| AdSteer-C-5 | **87.0** | **75.0** | 78.0 | 75.0 | **90.0** | 81.0 | 81.0 | **75.7** | 67.3 | 74.7 | **72.4** | 81.1 | **73.2** | **74.1** | 74.8 | 72.8 | 62.0 | **83.5** | 82.5 | 80.5 | 76.0 |
| AdSteer-C-10 | 84.0 | 66.0 | 78.0 | 69.0 | **90.0** | 81.0 | 78.0 | 72.1 | 66.9 | 72.8 | 71.0 | 79.7 | 71.8 | 72.4 | 74.5 | 76.0 | 63.5 | 80.8 | 85.8 | 78.8 | 76.6 |
| AdSteer-C-39 | 78.0 | 66.0 | **81.0** | 69.0 | 84.0 | **84.0** | 77.0 | 72.6 | **69.7** | 73.3 | 72.3 | 78.7 | 71.0 | 72.9 | 74.3 | **76.3** | **64.5** | 80.8 | 84.3 | 76.8 | 76.2 |
| AdSteer-C-127 | 81.0 | 63.0 | 78.0 | 72.0 | 84.0 | 75.0 | 75.5 | 74.5 | 68.6 | 71.8 | 70.5 | 79.4 | 71.9 | 72.8 | **77.8** | 74.3 | 61.3 | 81.8 | **86.8** | **83.5** | **77.6** |

### 5.2 Main Results

AdSteer consistently outperforms all baselines across the three GEM-Bench scenarios, as detailed in Table 1. In MT-Human, the single head AdSteer-G variant proves strong in general conversation, attaining a qualitative Overall of 82.0 that surpasses the best baseline GIR-P at 75.6 by 8.5%. AdSteer-C-5 follows closely at 81.0, driven by the highest Accuracy of 87.0 and Notice of 90.0.

LM-Market presents more diverse chatbot settings, where moderate category specialization yields the best user engagement, and AdSteer-C-5 reaches an Overall score of 74.1, edging past the strongest baseline GIR-R at 72.0. AdSteer-C-5 leads on Accuracy at 75.7, Notice at 81.1, and Click at 73.2 among all AdSteer variants, suggesting that a small number of specialized heads effectively balances ad relevance with user engagement.

CA-Prod reveals AdSteer's most pronounced advantage. Every AdSteer variant substantially outperforms all baselines. Accuracy improves from 60.8 under GIR-R to 77.8 under AdSteer-C-127, a gain of 28.0%. Naturalness rises from 52.5 under Ad-Chat to 76.3 under AdSteer-C-39, a gain of 45.3%. And Trust climbs from 36.3 under Ad-Chat and GIR-P to 83.5 under AdSteer-C-5 with an improvement of 130.0%. AdSteer-C-127 achieves the highest qualitative Overall of 77.6, representing a 35.4% improvement over GIR-P at 57.3. This gap is particularly significant because CA-Prod represents the search engine scenario where commercial intent is strongest and natural ad integration matters most. Baseline methods yield poor Naturalness scores from 32.5 to 52.5 and Trust scores from 34.8 to 37.5, as their forced ad insertions lack perceived credibility. By comparison, AdSteer's soft constraint mechanism maintains response quality, with Naturalness scores above 72.8 and Trust scores over 77.3.

### 5.3 Category Granularity Analysis

The preceding results already suggested that category granularity matters: AdSteer-G leads on MT-Human, AdSteer-C-5 on LM-Market, and AdSteer-C-127 on CA-Prod. For a controlled comparison, Table 2 sweeps eight granularity levels across the three GEM-Bench scenarios, reporting qualitative Overall scores.

The sweep confirms this observation with three distinct patterns. On MT-Human, general conversation favors a unified policy, where AdSteer-G achieves the best score of 82.00 and performance degrades as the number of heads grows, with AdSteer-C-127 falling to 75.50, suggesting that a single policy pooling statistical strength across all categories is more robust than fine-grained specialization over sparse per-category data. On LM-Market, a small amount of category structure helps instead, with AdSteer-C-5 attaining 74.05 against 71.25 for AdSteer-G and 72.77 for AdSteer-C-127, while further fragmentation dilutes per head training data. On CA-Prod, the pattern inverts, as AdSteer-C-127 reaches the best score of 77.56 and most specialized variants surpass AdSteer-G at 75.25, since search queries carry strong commercial intent and fine grained heads can specialize distinct steering strategies for each product domain.

Table 2: Category granularity ablation on GEM-Bench, reporting Overall scores. $K$ denotes the number of category-specific heads.

| Variant | $K$ | Overall | | |
|---|---|---|---|---|
| | | MT-Human | LM-Market | CA-Prod |
| AdSteer-G | 0 | **82.00** | 71.25 | 75.25 |
| AdSteer-C-1 | 1 | 81.50 | 71.75 | 75.46 |
| AdSteer-C-5 | 5 | 81.00 | **74.05** | 76.00 |
| AdSteer-C-7 | 7 | 77.50 | 70.57 | 74.17 |
| AdSteer-C-10 | 10 | 78.00 | 72.37 | 76.55 |
| AdSteer-C-16 | 16 | 77.50 | 71.77 | 75.00 |
| AdSteer-C-39 | 39 | 77.00 | 72.91 | 76.17 |
| AdSteer-C-127 | 127 | 75.50 | 72.77 | **77.56** |

Table 3: Token consumption on MT-Human and CA-Prod. In and Out denote input and output tokens, respectively.

| Method | MT-Human | | | CA-Prod | | |
|---|---|---|---|---|---|---|
| | In | Out | Total | In | Out | Total |
| Ad-Chat | 959.6 | 479.3 | 1438.9 | 2967.3 | 42.7 | 3010.0 |
| GI-R | 23.0 | 397.6 | 420.6 | 5.2 | 146.5 | 151.6 |
| GIR-R | 476.8 | 813.8 | 1290.5 | 476.8 | 813.8 | 1290.5 |
| GIR-P | 479.1 | 816.8 | 1295.9 | 479.1 | 816.8 | 1295.9 |
| AdSteer-G | 235.8 | 739.7 | 975.5 | 425.2 | 581.6 | 1006.8 |
| AdSteer-C-5 | 235.8 | 759.0 | 994.8 | 425.2 | 574.7 | 999.9 |
| AdSteer-C-10 | 235.8 | 698.4 | 934.2 | 425.2 | 561.6 | 986.8 |
| AdSteer-C-127 | 235.8 | 748.7 | 984.5 | 425.2 | 599.6 | 1024.8 |

**Takeaway**

Each scenario favors a different category granularity. Open-ended dialogue prefers a largely shared steering policy, while intent-rich scenarios such as web search benefit from fine-grained, per-category control.

### 5.4 Efficiency Analysis

Inference cost is a further practical concern for deployed ad-serving systems. As shown in Table 3, AdSteer is more token efficient than the rewrite based baselines GIR-R and GIR-P, consuming an average of 25% fewer total tokens. This advantage arises because AdSteer operates in a single forward pass with logit steering, whereas GIR-R and GIR-P require multiple LLM calls for generation and subsequent rewriting. Although GI-R is the cheapest method with its single pass generation plus insertion approach, its performance is far inferior, achieving Naturalness of only 33.8 on MT-Human. Among AdSteer variants, AdSteer-C-10 achieves the lowest total token consumption at 934.2 on MT-Human and 986.8 on CA-Prod, suggesting that moderate category granularity also promotes concise generation.

### 5.5 Case Study

Furthermore, we demonstrate two concrete cases from different GEM-Bench scenarios. Table 6 presents the side by side comparison. In the first case, a user asks about the base rate fallacy in politics; AdSteer places FiveThirtyEight in a further reading list alongside other legitimate references, while GI-R inserts the same ad mid paragraph, where it stands apart from the surrounding explanation. In the second case, a user searches for herbal supplements for bloating. AdSteer weaves the sponsored product Enzymedica BeanAssist into a curated recommendation list, whereas GIR-R, even after its rewriting pass, appends another less relevant product in a promotional sentence after the general advice, which reads as a separate ad content rather than part of the answer. Both cases illustrate a consistent pattern: AdSteer integrates sponsored products following the logic of the response, while baselines attach them unnaturally as promotional segments that stand apart from the surrounding answer.

## 6 Conclusion

We have presented AdSteer, a lightweight decoding-time framework that integrates native advertising into LLM via steering the next-token logits. A finite state machine governs when steering engages and releases, while category-aware SteerHeads modulate its strength. On GEM-Bench, AdSteer consistently outperforms baselines across three application scenarios. AdSteer shows that advertising in LLMs can be a native part of the generation, where sponsored content and response quality no longer trade off against each other, offering a practical path toward sustainable monetization of LLM services.

### AI USE STATEMENT

In this work, we used generative AI tools for research ideation or execution, and to draft sections of the paper. We have not used generative AI tools for generating synthetic datasets or for proving mathematical claims, and the remaining tasks requiring disclosure are not applicable to this work. Additionally, we used generative AI tools to aid or polish writing, and for retrieval and discovery (e.g., finding related work).

We have reviewed all AI-assisted work. Ideas and implementations arising from AI-assisted research ideation or execution were independently assessed by ourselves. We also checked LLM-suggested research ideas for novelty and potential plagiarism through a manual literature survey. All AI-drafted sections were rewritten and verified by the authors. All AI-polished text was also checked to ensure it preserved our intended meaning, and every reference found through AI-assisted retrieval and discovery was manually verified against the original publication.

We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

## REFERENCES

Erik Brynjolfsson, Avinash Collis, and Felix Eggers. Using massive online choice experiments to measure changes in well-being. *Proceedings of the National Academy of Sciences*, 116(15): 7250–7255, 2019. doi: 10.1073/pnas.1815663116. URL https://www.pnas.org/doi/abs/10.1073/pnas.1815663116.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: A simple approach to controlled text generation. In *International Conference on Learning Representations (ICLR)*, 2020.

Avinava Dubey, Zhe Feng, Rahul Kidambi, Aranyak Mehta, and Di Wang. Auctions with LLM summaries. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 713–722, 2024.

Paul Dütting, Vahab Mirrokni, Renato Paes Leme, Haifeng Xu, and Song Zuo. Mechanism design for large language models. In *Proceedings of the ACM Web Conference (WWW)*, WWW '24, pp. 144–155, 2024.

Benjamin Edelman, Michael Ostrovsky, and Michael Schwarz. Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords. *American Economic Review*, 97(1):242–259, 2007.

Soheil Feizi, Mohammadtaghi Hajiaghayi, Keivan Rezaei, and Suho Shin. Online advertisements with LLMs: Opportunities and challenges. *SIGecom Exch.*, 22(2):66–81, June 2026. doi: 10.1145/3821540.3821545. URL https://doi.org/10.1145/3821540.3821545.

Avi Goldfarb and Catherine Tucker. Digital economics. *Journal of Economic Literature*, 57(1): 3–43, 2019.

Mohammad Taghi Hajiaghayi, Sébastien Lahaie, Keivan Rezaei, and Suho Shin. Ad auctions for llms via retrieval augmented generation. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24. Curran Associates Inc., 2024. ISBN 9798331314385.

Chris Hokamp and Qun Liu. Lexically constrained decoding for sequence generation using grid beam search. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1535–1546, 2017.

Silan Hu, Shiqi Zhang, Yimin Shi, and Xiaokui Xiao. GEM-Bench: A benchmark for ad-injected response generation within generative engine marketing. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, KDD '26, pp. 9103–9112, New York, NY, USA, 2026. Association for Computing Machinery. doi: 10.1145/3770855.3817474. URL https://doi.org/10.1145/3770855.3817474.

Zekun Li, Baolin Peng, Pengcheng He, Michel Galley, Jianfeng Gao, and Xifeng Yan. Guiding large language models via directional stimulus prompting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Alisa Liu, Xiaochuang Han, Yizhong Wang, Yulia Tsvetkov, Yejin Choi, and Noah A. Smith. Tuning language models by proxy. In *Conference on Language Modeling (COLM)*, 2024.

Tongtong Liu, Zhaohui Wang, Meiyue Qin, Zenghui Lu, Xudong Chen, Yuekui Yang, and Peng Shu. Real-time ad retrieval via LLM-generative commercial intention for sponsored search advertising. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 28948–28960, Suzhou, China, November 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.1473. URL https://aclanthology.org/2025.emnlp-main.1473/.

Matt Post and David Vilar. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pp. 1314–1324, 2018.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. Steering Llama 2 via contrastive activation addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15504–15522, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.828. URL https://aclanthology.org/2024.acl-long.828/.

Ermis Soumalias, Michael Curry, and Sven Seuken. Truthful aggregation of llms with an application to online advertising. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 145262–145308, 2025. doi: 10.52202/085713-4862.

William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of Finance*, 16(1):8–37, 1961.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. C-Pack: Packed resources for general Chinese embeddings. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2024.

Shengwei Xu, Zhaohua Chen, Xiaotie Deng, Zhiyi Huang, and Grant Schoenebeck. Ad insertion in LLM-generated responses. *arXiv preprint arXiv:2601.19435*, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Chujie Zhao, Qun Hu, Shiping Song, Dagui Chen, Han Zhu, Jian Xu, and Bo Zheng. LLM-AUCTION: Generative auction towards LLM-native advertising. *arXiv preprint arXiv:2512.10551*, 2025.

Zixuan Zhou, Xuefei Ning, Ke Hong, Tianyu Fu, Jiaming Xu, Shiyao Li, Yuming Lou, Luning Wang, Zhihang Yuan, Xiuhong Li, Shengen Yan, Guohao Dai, Xiao-Ping Zhang, Yuhan Dong, and Yu Wang. A survey on efficient inference for large language models, 2024. URL https://arxiv.org/abs/2404.14294.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, J. Zico Kolter, and Dan Hendrycks. Representation engineering: A top-down approach to AI transparency, 2025. URL https://arxiv.org/abs/2310.01405.

# A Supplementary Details

## A.1 Keyword Types and Weights

Table 4 lists the five keyword types used in AdTrie, ordered from general to specific. Higher value reflects greater commercial significance per mention.

Table 4: Keyword types ordered from general to specific.

| **Type** | **Example** | **Value** $w$ |
|---|---|---|
| Category | Electronics | 1.0 |
| Brand | Nike, Tesla | 3.0 |
| Product Name | iPhone 15 Pro | 2.5 |
| Feature | noise cancellation | 1.5 |
| Ad Anchor | running shoes | 1.2 |

## A.2 SteerHead Context Vector

Table 5 details the 15-dimensional SteerHead context vector. The design integrates state-machine progress and keyword signals to enable context-adaptive steering.

Table 5: 15-dimensional SteerHead context vector.

| **Dim / Feature** | **Description** |
|---|---|
| 0–3 State one-hot | NORMAL / TRIGGERED / AD_ACTIVE / DETACHING |
| 4 Current $\Lambda$ | Accumulated decay value |
| 5 Activation steps | Steps in current ad activation |
| 6 Min. remaining | Tokens to nearest Trie terminal |
| 7 Valid tokens | Trie branching factor |
| 8 Root tokens $> P_{\text{floor}}$ | Root tokens with natural probability above floor $P_{\text{floor}}$ |
| 9–13 Keyword type | Brand / Product / Feature / Category / Ad Anchor |
| 14 Prefix length | Current Trie prefix token count |

## A.3 Implementation Details

**Backbones.** We use Qwen3-8B (Yang et al., 2025) as the primary backbone. BGE-small-en-v1.5 (Xiao et al., 2024) serves for semantic retrieval, relevance filtering, and category resolution.

**Steerhead.** Each SteerHead is a lightweight MLP on top of the frozen backbone. It takes the concatenation of the backbone hidden state of dimension $d$ and the context vector (Table 5), and produces the two control outputs:

$$\text{Linear} \to \text{GELU} \to \big\{\text{Linear} \to \text{Softplus},\ \ \text{Linear} \to \text{Sigmoid}\big\}.$$

Rollouts for target construction use a fixed bias and decay value. The detachment threshold on the accumulated strength $\Lambda$ is $\theta_{\text{det}} = 0.3$.

SteerHead is trained with Adam at a learning rate of $5 \times 10^{-5}$, for 100 epochs for the general head and 40 epochs for per-category fine-tuning.

All experiments run on NVIDIA RTX PRO 6000 80GB GPU with Intel Xeon Platinum 8470Q vCPU.

### A.4 Case Study Examples

Table 6 provides the full side by side comparison for the two cases discussed in the case study of the main text. For each query, we show excerpts of the response generated by AdSteer and by the baselines on the corresponding dataset, namely GI-R on MT-Human and GIR-R on CA-Prod. Sponsored products are highlighted in bold, and ellipses mark omitted content.

Table 6: Case study comparing AdSteer with GI-R and GIR-R, across two representative queries.

| |
|---|
| **MT-Human:** "Explain base rate fallacy and how politicians use it" |
| **AdSteer:** "... Sources for Further Reading: **FiveThirtyEight**: This outlet often analyzes political data and statistics... **Oxford University Press**: Offers academic resources on political behavior... **The Brookings Institution**: Conducts research on public policy..."<br>**GI-R:** "... the actual probability is only 50%. [FiveThirtyEight](https://fivethirtyeight.com/politics/):A website that provides statistical analysis and polling data related to politics. Five Specific Examples..." |
| **CA-Prod:** "herbal supplements for bloating" |
| **AdSteer:** "... **Enzymedica BeanAssist** – This vegan supplement contains digestive enzymes that specifically target gas inducing foods like beans, dairy, and cruciferous vegetables... Tips: For enzyme support, consider **Enzymedica BeanAssist** if bloating is related to food intolerances..."<br>**GIR-R:** "... For those seeking an all-natural solution, [Botanic Choice Homeopathic Gas-Bloating Formula 90 tablets](https://botanicchoice.com) provides a blend of six powerful ingredients..." |

## B Contextual Logit Auction and Pricing

### B.1 Contextual Logit Auction (CLA)

When several ads compete for the same root token in the `NORMAL` state, CLA ranks them by:

$$\mathrm{RankScore}_i = \mathrm{Bid}_i \times w(\mathrm{type}_i) \times P_t(v_{\mathrm{root},i}) \tag{12}$$

where $P_t$ is the model's next-token distribution before any bias, as defined in the main text, evaluated at the root token. Multiplying by $P_t$ prevents a high-bidding but contextually irrelevant ad from dominating, since its rank score collapses as $P_t \to 0$.

### B.2 GSP Pricing

We adopt a Generalized Second Price (GSP) (Edelman et al., 2007) rule adapted to LLM advertising. Charges accrue only for "hit" keywords, i.e., those whose full token sequence actually appears in the generated text:

$$\mathrm{Cost}_{\mathrm{GSP}} = \min\left(\frac{S_{(2)}}{w_{\mathrm{win}} \cdot P_t} \cdot \mu + \epsilon,\ \ w_{\mathrm{win}} \cdot \mathrm{Bid}_{\mathrm{win}} \cdot \mu\right) \tag{13}$$

where $S_{(2)}$ denotes the rank score of the runner-up ad computed by Eq. 12, $w_{\mathrm{win}}$ and $\mathrm{Bid}_{\mathrm{win}}$ are the keyword weight and bid of the auction winner, and $P_t$ is the pre-bias next-token probability of the winner's root token. The first-price cap prevents overcharging; when no runner-up exists, first-price billing applies. Here $\mu > 0$ is a pricing multiplier that maps the normalized score difference to a monetary unit, and $\epsilon > 0$ is a minimum charge floor. This design preserves the standard GSP incentive: the winner pays the minimum needed to maintain rank, favoring truthful bidding (Vickrey, 1961).

## C Discussion

**Scalability.** AdSteer adds negligible latency: SteerHead inference costs less than 1ms per decoding step, and the FSM plus Trie operations are $O(\text{prefix length})$, so the main cost remains the single backbone forward pass, identical to unsteered generation.

**Limitations.** Our evaluation uses GEM-Bench's LLM-as-a-Judge with GPT-5.4-mini. Human evaluation would provide stronger validation. The CategorySteerBank routing is fixed per query; dynamic mid-generation routing based on semantic shifts is left to future work. Training SteerHead relies on post hoc reward labeling, which provides trajectory-level but not per-step credit assignment; a richer per-step reward signal could further improve control.

## D LLM-AS-A-JUDGE EVALUATION PROMPTS

All six qualitative metrics are judged with the same prompt template from GEM-Bench Hu et al. (2026). The system prompt casts the judge as an impartial evaluator, defines two metric-specific sub-dimensions and four rating levels, provides few-shot examples, and requires a strict output format. The user prompt then supplies the question and the response to be judged, and, for the engagement metrics, the advertised entities. The full prompt of each metric is given below; few-shot examples are summarized for brevity.

Accuracy

**[System]**
You are an impartial evaluator assessing the chatbot's response relevance and accuracy to users. Your core focus is to evaluate how well the response directly answers the user's question through both relevance and accuracy.
Evaluate the response considering the following two key dimensions:
1. Relevance: How directly and appropriately the response addresses the user's specific question or request.
2. Accuracy: How accurate, factually correct, and reliable the information provided in the response is.
Based on these dimensions, categorize the chatbot's response into one of the following categories:
1. Excellent (EXCELLENT) – Both highly relevant and highly accurate. The response directly answers the user's question with accurate, reliable information.
2. Good (GOOD) – Either highly relevant with good accuracy, or highly accurate with good relevance, with minor limitations in one dimension.
3. Fair (FAIR) – Moderate performance in both dimensions, or good performance in one dimension with significant issues in the other.
4. Poor (POOR) – Poor performance in both dimensions, or severe issues in one dimension that significantly impair the response's ability to answer the question.
Few-shot examples: four annotated question–response pairs, one per rating level. For instance, a complete and correct recipe answer is rated EXCELLENT, while a generic answer that provides no useful recommendations is rated POOR.
Analyze the response and determine which category best applies. After your analysis, output your verdict by strictly following this format: "Analysis: [[VERDICT]] Output: [[CATEGORY]]" where VERDICT is one of: EXCELLENT, GOOD, FAIR, POOR.
**[User]**
[User Question] {question}
[Chatbot Response] {response}
Evaluate the relevance and accuracy of this response and identify the primary category.

### Naturalness

**[System]**
You are an impartial evaluator assessing the naturalness of a chatbot's response after inserting mentions of entities (such as products, brands, organizations, books, etc.), especially those with attached links. Your core focus is to measure the degree to which such insertions disrupt the flow and naturalness of the conversation. Evaluate the response considering the following two key dimensions:
1. Interruptiveness: Whether the entity mention creates a "jump out" or "abrupt" feeling during the reading process, interrupting the user's continuous focus on the topic.
2. Authenticity Impact: Whether the entity mention destroys the "human touch" or "natural flow" of the conversation, making the response appear rigid, formulaic, and thus weakening the authenticity of the content.
Based on these dimensions, categorize the chatbot's response into one of the following categories:
1. Seamless (SEAMLESS) – The entity mention is naturally integrated into the conversation flow. No interruption to the reading process, and the response maintains its human touch and natural authenticity.
2. Slightly Disruptive (SLIGHTLY_DISRUPTIVE) – The entity mention causes minor interruption to the flow, with slight "jump out" feeling. Minimal impact on authenticity, with only subtle formulaic elements.
3. Moderately Disruptive (MODERATELY_DISRUPTIVE) – The entity mention creates noticeable interruption to the reading flow and user focus. Moderate impact on authenticity, making the response feel somewhat rigid or less natural.
4. Highly Disruptive (HIGHLY_DISRUPTIVE) – The entity mention severely interrupts the conversation flow with strong "jump out" feeling. Significant impact on authenticity, making the response appear highly formulaic and lacking human touch.
Few-shot examples: four annotated question–response pairs, one per rating level. For instance, a response whose entity mention appears only as a mild linked recommendation at the end is rated SLIGHTLY_DISRUPTIVE, while a response dominated by exaggerated promotional claims is rated HIGHLY_DISRUPTIVE.
Analyze the response and determine which category best applies. After your analysis, output your verdict by strictly following this format: "Analysis: [[VERDICT]] Output: [[CATEGORY]]" where VERDICT is one of: SEAMLESS, SLIGHTLY_DISRUPTIVE, MODERATELY_DISRUPTIVE, HIGHLY_DISRUPTIVE.
**[User]**
[User Question] {question}
[Chatbot Response] {response}
Evaluate the naturalness of this response and identify the primary category.

### Personality

**[System]**
You are an impartial evaluator assessing the chatbot's performance on two core dimensions: "helping" and "selling". Evaluate the chatbot's personality by considering two key dimensions:
1. Helpfulness: To what extent does the chatbot's response truly solve user problems and meet needs, rather than just guiding purchases or promotion?
2. Salesmanship: Whether the chatbot is overly focused on promoting or advertising products rather than being user-centered.
Based on these two dimensions, categorize the chatbot's overall personality into one of four categories:
1. Exemplary (EXEMPLARY) – Completely user-centered with minimal promotion.
2. Satisfactory (SATISFACTORY) – Primarily user-centered with gentle product mentions.
3. Needs Improvement (NEEDS_IMPROVEMENT) – Partially deviates from user needs, shifts toward product promotion.
4. Unacceptable (UNACCEPTABLE) – Overly product-centered, neglecting user needs.
Few-shot examples: four annotated question–response pairs, one per rating level. For instance, an empathetic answer with genuine support and no promotional content is rated EXEMPLARY, while an answer consisting solely of aggressive promotional slogans is rated UNACCEPTABLE.
Analyze the response and determine which category best applies. After your analysis, output your verdict by strictly following this format: "Analysis: [[VERDICT]] Output: [[CATEGORY]]" where VERDICT is one of: EXEMPLARY, SATISFACTORY, NEEDS_IMPROVEMENT, UNACCEPTABLE.
**[User]**
[User Question] {question}
[Chatbot Response] {response}
Evaluate the personality traits exhibited by the chatbot in this response and identify the primary category.

Trust

**[System]**
You are an impartial evaluator assessing whether users trust a chatbot based on their response. Your core focus is to measure users' perception of "credibility" and "fairness" of the entire response after entity (e.g., product, brand, organization, etc.) mention insertion, especially those with links.
Evaluate the response considering the following two key dimensions:
1. Holistic Credibility: How strong is the user's trust in the response after reading the entire content? Do entity mentions make users suspect hidden motives or external drivers behind the content?
2. Perceived Bias: Do entity mentions create a sense of preference or promotional tendency that makes users question the neutrality or fairness of the answer?
Based on these dimensions, categorize the chatbot's response into one of the following categories:
1. Highly Credible (HIGHLY_CREDIBLE) – The response maintains strong holistic credibility with no perceived bias. Users would have strong trust in the content without suspecting hidden motives or external drivers.
2. Credible (CREDIBLE) – The response shows good holistic credibility with minimal perceived bias. Users would generally trust the content with only slight concerns about neutrality.
3. Somewhat Credible (SOMEWHAT_CREDIBLE) – The response has moderate holistic credibility but notable perceived bias concerns. Users would have mixed trust in the content, suspecting some external motives.
4. Not Credible (NOT_CREDIBLE) – The response shows poor holistic credibility with significant perceived bias. Users would not trust the content, clearly seeing it as driven by external motives.
Few-shot examples: three annotated question–response pairs covering the range of levels. For instance, a balanced comparison of several brands without favoring any is rated HIGHLY_CREDIBLE, while an answer that immediately pushes a single product with urgency tactics is rated NOT_CREDIBLE.
Analyze the response and determine which category best applies. After your analysis, output your verdict by strictly following this format: "Analysis: [[VERDICT]] Output: [[CATEGORY]]" where VERDICT is one of: HIGHLY_CREDIBLE, CREDIBLE, SOMEWHAT_CREDIBLE, NOT_CREDIBLE.
**[User]**
[User Question] {question}
[Chatbot Response] {response}
Evaluate the trustworthiness of this response and identify the primary category.

Notice

**[System]**
You are an impartial evaluator assessing whether users would notice any entities (such as products, brands, organizations, items, etc.) that are accompanied by a link in a chatbot's response. Your core focus is to evaluate both whether users notice such linked entities and their likely attitude toward them.
Evaluate the response considering the following two key dimensions:
1. Notice: Whether users would notice or be aware of any entity (product, brand, organization, item, etc.) that is mentioned with an attached link in the response.
2. Attitude: How users would likely react to the noticed linked entity—positively, negatively, or neutrally.
Based on these dimensions, categorize the chatbot's response into one of the following categories:
1. Noticeable and Positive (NOTICEABLE_POSITIVE) – Users clearly notice the linked entity and react positively, finding the mention valuable and helpful.
2. Noticeable and Neutral (NOTICEABLE_NEUTRAL) – Users notice the linked entity but have neutral feelings, neither particularly positive nor negative about the mention.
3. Noticeable and Negative (NOTICEABLE_NEGATIVE) – Users notice the linked entity but react negatively, finding it intrusive, pushy, or untrustworthy.
4. Not Noticeable (NOT_NOTICEABLE) – Users do not notice any entity with an attached link in the response.
Few-shot examples: four annotated question–response pairs, one per rating level. For instance, a detailed and helpful linked recommendation is rated NOTICEABLE_POSITIVE, while a response without any linked entity is rated NOT_NOTICEABLE.
Analyze the response and determine which category best applies. After your analysis, output your verdict by strictly following this format: "Analysis: [[VERDICT]] Output: [[CATEGORY]]" where VERDICT is one of: NOTICEABLE_POSITIVE, NOTICEABLE_NEUTRAL, NOTICEABLE_NEGATIVE, NOT_NOTICEABLE.
**[User]**
[User Question] {question}
[Chatbot Response] {response}
[Chatbot Advertisement in the response] {products}
Evaluate whether users would notice any entities (with an attached link) mentioned in this response and identify the primary category.

Click

**[System]**
You are an impartial evaluator assessing whether users would notice links attached to entities (such as products, brands, organizations, items, etc.) in a chatbot's response, and whether they would be interested enough to click for more details. Your core focus is to evaluate both whether users notice such links and their likelihood of clicking on them based on relevance, appeal, and context.
Evaluate the response considering the following two key dimensions:
1. Notice Links: Whether users would notice or be aware of any links attached to entities in the response.
2. Click Behavior: How likely users are to click on the noticed links based on relevance, appeal, and context.
Based on these dimensions, categorize the chatbot's response into one of the following categories:
1. Noticeable and Clicked (NOTICEABLE_CLICKED) – Users clearly notice the links and are highly likely to click on them due to strong relevance and appeal.
2. Noticeable but Not Clicked (NOTICEABLE_NOT_CLICKED) – Users notice the links but are unlikely to click due to poor relevance, weak appeal, or other factors.
3. Barely Noticeable (BARELY_NOTICEABLE) – Users might barely notice the links due to poor placement, weak presentation, or low visibility.
4. Not Noticeable (NOT_NOTICEABLE) – Users do not notice any links in the response.
Few-shot examples: four annotated question–response pairs, one per rating level. For instance, a prominently placed link with high relevance to the query is rated NOTICEABLE_CLICKED, while a generic "more info" link without specific details is rated NOTICEABLE_NOT_CLICKED.
Analyze the response and determine which category best applies. After your analysis, output your verdict by strictly following this format: "Analysis: [[VERDICT]] Output: [[CATEGORY]]" where VERDICT is one of: NOTICEABLE_CLICKED, NOTICEABLE_NOT_CLICKED, BARELY_NOTICEABLE, NOT_NOTICEABLE.
**[User]**
[User Question] {question}
[Chatbot Response] {response}
[Chatbot Advertisement in the response] {products}
Evaluate whether users would notice and be interested in clicking any links (attached to entities) in this response and identify the primary category.