Is Your LLM Overcharging You? Tokenization, Transparency, and Incentives

Ander Artola Velasco
Max Planck Institute for Software Systems
Kaiserslautern, Germany
avelasco@mpi-sws.org

Stratis Tsirtsis
Hasso Plattner Institute
Potsdam, Germany
stratis.tsirtsis@hpi.de

Nastaran Okati
Max Planck Institute for Software Systems
Kaiserslautern, Germany
nastaran@mpi-sws.org

Manuel Gomez-Rodriguez
Max Planck Institute for Software Systems
Kaiserslautern, Germany
manuelgr@mpi-sws.org

Abstract

State-of-the-art large language models require specialized hardware and substantial energy to operate. As a consequence, cloud-based services that provide access to large language models have become very popular. In these services, the price users pay for an output provided by a model depends on the number of tokens the model uses to generate it—they pay a fixed price per token. In this work, we show that this pricing mechanism creates a financial incentive for providers to strategize and misreport the (number of) tokens a model used to generate an output, and users cannot prove, or even know, whether a provider is overcharging them. However, we also show that, if an unfaithful provider is obliged to be transparent about the generative process used by the model, misreporting optimally without raising suspicion is hard. Nevertheless, as a proof-of-concept, we develop an efficient heuristic algorithm that allows providers to significantly overcharge users without raising suspicion. Crucially, we demonstrate that the cost of running the algorithm is lower than the additional revenue from overcharging users, highlighting the vulnerability of users under the current pay-per-token pricing mechanism. Further, we show that, to eliminate the financial incentive to strategize, a pricing mechanism must price tokens linearly on their character count. While this makes a provider's profit margin vary across tokens, we introduce a simple prescription under which the provider who adopts such an incentive-compatible pricing mechanism can maintain the average profit margin they had under the pay-per-token pricing mechanism. Along the way, to illustrate and complement our theoretical results, we conduct experiments with several large language models from the `Llama`, `Gemma` and `Ministral` families, and input prompts from LMSYS Chatbot Arena, a popular platform for crowdsourced benchmarking.

1 Introduction

Large language models (LLMs) are becoming ubiquitous across multiple industries—from powering chatbots and virtual assistants to driving innovation in research, healthcare, and finance [1–4]. However, since the computational resources required to run these models are significant, most (enterprise) users are unable to host them locally. As a result, users rely on a few cloud-based providers that offer LLMs-as-a-service to obtain access [5].

1st EurIPS Conference (EurIPS 2025) Workshop: Beyond Regulation – Private Governance & Oversight Mechanisms for AI (PAIG Workshop).

In a typical LLM-as-a-service, a user submits a prompt to the provider via an application programming interface (API). Then, the provider feeds the prompt into an LLM running on their own hardware, which (stochastically) generates a sequence of tokens as an output using a generative process.[1] Finally, the provider shares the output with the user and charges them based on a simple pricing mechanism: a fixed price per token.[2] In this paper, we focus on the following fundamental question:

What incentives does the pay-per-token pricing mechanism create for providers?

Our key observation is that, in the interaction between a user and a provider, there is an asymmetry of information [6–8]. The provider observes the entire generative process used by the model to generate an output, including its intermediate steps and the final output tokens, whereas the user only observes and pays for the (output) tokens shared with them by the provider. This asymmetry sets the stage for a situation known in economics as moral hazard [9], where one party (the provider) has the opportunity to take actions that are not observable by the other party (the user) to maximize their own utility at the expense of the other party.

The core of the problem lies in the fact that the tokenization of a string is not unique. For example, consider that the user submits the prompt "`What is the oldest city in the world?`" to the provider, the provider feeds it into an LLM, and the model generates the output "|`Dam`|`ascus`|" consisting of two tokens. Since the user is oblivious to the generative process, a self-serving provider has the capacity to misreport the tokenization of the output to the user without even changing the underlying string. For instance, the provider could simply claim that the LLM generated the tokenization "|`Da`|`ma`|`s`|`cus`|" and overcharge the user for four tokens instead of two!

A simple remedy to build trust between the two parties would be to require providers to share with the user more information about the generative process used by the model, such as the next-token distribution in each step of the process. This would grant the user a form of (partial) auditability, since tokenizations, such as the one mentioned above, may have negligible probability in practice. Importantly, if the provider implements procedures to prevent the generation of low-probability tokens (*e.g.*, top-p sampling [10], top-k sampling), as commonly done in practice, such tokenizations would not only be unlikely, but rather implausible, giving grounds to the user to contest the specific tokenization of the output shared with them by the provider. In this case, a provider would have to invest additional effort (and resources) to misreport the tokenization of an output while preserving its plausibility, making such a strategic behavior significantly less worthy from a financial point of view.

However, some providers may be highly reluctant to share information that could potentially expose the internal workings of their LLMs, especially if the LLMs are proprietary and such information can be used by competitors [11]. In the absence of any additional means for the users to verify the truthfulness of the providers, the only remaining option is to regulate the transactions between users and providers in a way that eliminates the incentive for providers to engage in misreporting in the first place. To this end, we introduce and argue for a pay-per-character pricing mechanism that serves exactly this purpose.

Our contributions. We start by characterizing the problem of tokenization (mis-)reporting in LLMs as a principal-agent problem [12, 13]. Building upon this characterization, we make the following contributions:

1. We show that, under the pay-per-token pricing mechanism, a provider's utility is tightly linked to the length of the reported output token sequence, and this creates an incentive for the provider to misreport an output's tokenization.
2. We show that, if the providers are transparent about the next-token distribution used by the LLMs they serve, they cannot expect to find the longest tokenization of an output that is plausible in polynomial time.
3. We show that, even under this form of transparency, the financial incentive for providers to (mis-)report tokenizations persists. As a proof-of-concept, we develop an efficient heuristic algorithm that allows providers to find plausible token sequences that are longer or equal to a generated output token sequence, and is financially profitable—the additional revenue it yields is higher than the cost of running it.

[1] Tokens are units that make up sentences and paragraphs, such as (sub-)words, symbols, and numbers.

[2] `https://ai.google.dev/gemini-api/docs/pricing`, `https://openai.com/api/pricing/`.

4. We show that *any* incentive-compatible pricing mechanism must price tokens linearly on their character count. As a consequence, if each character is priced equally, there is only one incentive-compatible pricing mechanism, which we refer to as the pay-per-character pricing mechanism.
5. We show that, under an incentive-compatible pricing mechanism, a provider's profit margin varies across tokens. However, we introduce a simple prescription under which a provider who transitions from pay-per-token to pay-per-character can maintain their average profit margin.

Along the way, to illustrate and complement the above contributions, we conduct a series of experiments using LLMs from the `Llama`, `Gemma` and `Ministral` families and user input prompts from the LMSYS Chatbot Arena platform [14].[3] Under the pay-per-token pricing mechanism, we empirically demonstrate that an unfaithful provider who is transparent about the generative process used by the LLM they serve can use our heuristic algorithm to overcharge users and, in many cases, obtain a significant financial gain.

Further related work. Our work builds upon further related work on tokenization, the economics of LLMs-as-a-service and, more broadly, the economics of digital and information services.

Multiple lines of empirical evidence have shown that tokenization plays a central role in developing and analyzing LLMs [15–21]. Consequently, there have been a variety of efforts focusing on better understanding and improving byte-pair encoding (BPE), the tokenization algorithm most commonly used in LLMs [22–26]. However, this line of work has overlooked the economic implications of tokenization (in the context of LLMs-as-a-service), which is the main focus of our work.

Within the rapidly growing literature on the economics of LLMs-as-a-service [27–38], the work most closely related to ours is by Sun et al. [38], who study the problem of providers reporting artificially injected tokens during the hidden reasoning steps of recent LLMs. However, they limit their analysis to injecting additional reasoning tokens and do not consider alternative tokenization of the output string, which is the main focus of our work. In a related line of work, Cai et al. [34] and Saig et al. [35] also study a setting in which the provider has a financial incentive to be unfaithful to the users. However, in their setting, the provider has an incentive to be unfaithful about the LLM they use to generate outputs, rather than the tokenization of the outputs—it may use a cheaper-to-run LLM than the one it charges the users for. To reduce the financial incentive to strategize, Cai et al. [34] argue for solutions based on increased transparency as well as trusted execution environments, and Saig et al. [35] argue for a pay-for-performance pricing mechanism using a contract theory formulation. However, to the best of our knowledge, our work is the first to explore incentive-compatible token pricing mechanisms in LLMs.

More broadly, our work also connects to the vast literature on the economics of digital and information services [39–41]. Within this literature, the work most relevant to ours focuses on incentive-compatible pricing in service systems with quality-of-service guarantees [42], in forecasting competitions [43], as well as in sponsored products in digital marketplaces [44]. In particular, our work extends these ideas to algorithmic services within the rapidly growing, multi-billion-dollar market of generative artificial intelligence.

2 A Principal-Agent Model of Delegated Autoregressive Generation

We characterize the interaction between a user and an LLM provider as a principal-agent problem [12, 13], where the principal (the user) delegates a task (a generation) to the agent (the provider), who performs the task on behalf of the principal and gets paid based on a commonly agreed-upon contract.

In a typical interaction between a user and a provider, the user first submits a prompt $q \in \Sigma^*$ to the provider, where Σ^* denotes the set of all finite-length strings over an alphabet Σ. Then, the provider uses their own hardware to query an LLM with the prompt q,[4] and the LLM (stochastically) generates an output token sequence $\mathbf{t} = (t_1, t_2, \ldots, t_k) \in \mathcal{V}^*$ in an autoregressive manner, one token at a time.

[3] The code we used in our experiments is available at `https://github.com/Networks-Learning/token-pricing`.

[4] In practice, the provider turns the prompt q into a sequence of tokens using a tokenizer before passing it as input to the model, but modeling this explicitly is not relevant in our work.

Here, $t_i \in \mathcal{V}$ is the i-th token in a sequence of k tokens, $\mathcal{V} \subset \Sigma^*$ is the (token) vocabulary used by the LLM,[5] and $\mathcal{V}^*$ denotes the set of all finite-length sequences over the vocabulary. Finally, the provider observes the generated sequence of tokens $\mathbf{t}$, and reports a token sequence $\tilde{\mathbf{t}} \sim \pi(\mathbf{t})$ to the user using a (non-deterministic) reporting policy π.

Before the interaction between a user and an LLM provider begins, both parties agree on a contract that specifies how the provider should be compensated for the output token sequence they report to the user. More specifically, the user and the provider agree on a *pricing mechanism* that determines the price $r(\tilde{\mathbf{t}})$ the user pays (*i.e.*, the revenue the provider earns) for the reported output token sequence $\tilde{\mathbf{t}}$:

Definition 1 (Pricing mechanism)**.** *Given a vocabulary of tokens $\mathcal{V}$, a pricing mechanism is a function $r\colon \mathcal{V}^* \to \mathbb{R}_{\geq 0}$ that assigns a price to each reported output token sequence $\tilde{\mathbf{t}} \in \mathcal{V}^*$.*

Throughout the paper, we focus on additive pricing mechanisms, which include the widely used pay-per-token pricing mechanism. An additive pricing mechanism independently assigns a price $r(\tilde{t}_i)$ to each token $\tilde{t}_i$ in a reported output token sequence $\tilde{\mathbf{t}}$, and calculates the price $r(\tilde{\mathbf{t}})$ of a reported output token sequence by adding up the price of each individual token.

Given a generated output token sequence $\mathbf{t}$ and a reported output token sequence $\tilde{\mathbf{t}} \sim \pi(\mathbf{t})$, the provider's utility $U_\pi(\tilde{\mathbf{t}}, \mathbf{t})$ is given by the difference between the revenue $r(\tilde{\mathbf{t}})$ they receive from the user for the reported sequence $\tilde{\mathbf{t}}$ and the energy costs $c_{\text{gen}}(\mathbf{t})$ and $c_\pi(\mathbf{t})$ of GPU computations[6] to generate the output sequence $\mathbf{t}$ and run the reporting policy π, respectively, *i.e.*,

$$U_\pi(\tilde{\mathbf{t}}, \mathbf{t}) = r(\tilde{\mathbf{t}}) - c_{\text{gen}}(\mathbf{t}) - c_\pi(\mathbf{t}). \tag{1}$$

In what follows, we refer to the reporting policy π_0 that always returns $\mathbf{t} = \pi_0(\mathbf{t})$ as the *truthful* policy, where note that $c_{\pi_0}(\mathbf{t}) = 0$ for all $\mathbf{t}$, and we denote the profit margin the provider obtains from generating and (truthfully) reporting the output token sequence $\mathbf{t}$ as $\rho(\mathbf{t}) = 1 - c_{\text{gen}}(\mathbf{t})/r(\mathbf{t}) = U_{\pi_0}(\mathbf{t}, \mathbf{t})/r(\mathbf{t})$.

Further, motivated by recent empirical studies showing that the energy an LLM consumes to generate an output $\mathbf{t}$ scales linearly with its length [45, 46], as well as our own experiments (refer to Figure 11 in Appendix C), we assume that $c_{\text{gen}}(\mathbf{t}) \approx c_o \cdot \texttt{len}(\mathbf{t})$, where $c_o \in \mathbb{R}_{>0}$ is a constant that represents the (energy) cost of generating a single token. Here, it is also worth clarifying that there are effective reporting policies that do not require GPU computations, that is, policies π such that $c_\pi(\mathbf{t}) = 0$ for all $\mathbf{t}$, as those implemented by Algorithm 1. However, in Section 3, we will demonstrate that, in order not to raise suspicion, the reporting policies implemented by Algorithm 2 do require GPU computations.

Given a reported output token sequence $\tilde{\mathbf{t}}$, the user's utility $U_{\text{user}}(\tilde{\mathbf{t}})$ is given by the difference between the value $v(\tilde{\mathbf{t}})$ they derive from the sequence $\tilde{\mathbf{t}}$ and the price $r(\tilde{\mathbf{t}})$ they pay to the provider for $\tilde{\mathbf{t}}$, that is, $U_{\text{user}}(\tilde{\mathbf{t}}) = v(\tilde{\mathbf{t}}) - r(\tilde{\mathbf{t}})$. However, the user typically derives value from the text that the output token sequence represents, rather than the token sequence itself. For example, in creative writing, the user may be interested in the extent to which the generated text is captivating to read, and in code generation, the user may be interested in operational aspects of the generated code, such as its correctness and efficiency. Therefore, we assume that $v(\tilde{\mathbf{t}}) = v(\texttt{str}(\tilde{\mathbf{t}}))$, where $\texttt{str}\colon \mathcal{V}^* \to \Sigma^*$ maps a sequence of tokens to the respective string, and we use $|\texttt{str}(\tilde{\mathbf{t}})|$ to denote the number of characters in the string $\texttt{str}(\tilde{\mathbf{t}})$.

Since the user does not have direct access to the generated output sequence $\mathbf{t}$, a provider can, in principle, implement any reporting policy π they prefer (*e.g.*, the one with the highest utility based on the pricing mechanism). However, arbitrary manipulations of the generated output may easily raise suspicion about the provider's practices. Therefore, motivated by the observation that LLMs can generate different token sequences corresponding to the same string [15, 47, 48], we narrow our focus to reporting policies that misreport the tokenization of the generated output sequence *while preserving its string-level representation*. More formally, given a generated output

[5] We assume $\Sigma \subset \mathcal{V}$ since this condition is satisfied by construction by most tokenization algorithms [22]. Under this assumption, Σ is the set of tokens that cannot be split further into other tokens and can include, for example, (non-)Latin characters.

[6] We focus on energy costs of GPU computations as they can be directly attributed to individual outputs. In practice, providers may have additional running costs, such as hardware maintenance, that they can amortize across outputs and incorporate into their pricing.

Algorithm 1 It returns a token sequence $\tilde{\mathbf{t}}$ longer or equal than $\mathbf{t}$

Input Generated output token sequence $\mathbf{t}$, number of iterations m
Initialize $\tilde{\mathbf{t}} \leftarrow \mathbf{t}$
for m iterations **do**
  ▷ Find tokens $\tilde{t}_i$ that can be further split into two (non-empty) tokens t'_1, t'_2
  $\texttt{valid_splits} \leftarrow \{(i, t'_1, t'_2)$ such that $i \in [\texttt{len}(\tilde{\mathbf{t}})],\ t'_1, t'_2 \in \mathcal{V},$ and $\texttt{str}(t'_1, t'_2) = \texttt{str}(\tilde{t}_i)\}$
  if |valid_splits|=0 **then**
    break ▷ If no token in $\tilde{\mathbf{t}}$ can be split further, terminate the loop
  else
    $(i, t'_1, t'_2) \leftarrow \text{Random}(\texttt{valid_splits})$ ▷ Select and implement one valid split at random
  end if
  $\tilde{\mathbf{t}} \leftarrow \left(\tilde{\mathbf{t}}_{<i}, t'_1, t'_2, \tilde{\mathbf{t}}_{>i}\right)$
end for
return $\tilde{\mathbf{t}}$

token sequence $\mathbf{t}$ with $s = \texttt{str}(\mathbf{t})$, such reporting policies return token sequences $\tilde{\mathbf{t}}$ from the set $\mathcal{V}^*_s = \left\{\tilde{\mathbf{t}} \in \mathcal{V}^* : \texttt{str}(\tilde{\mathbf{t}}) = s\right\}$. Then, it is easy to see that, as long as there exists a token sequence $\tilde{\mathbf{t}} \in \mathcal{V}^*_s$ such that $r(\tilde{\mathbf{t}}) > r(\mathbf{t})$, a reporting policy π that (mis-)reports $\mathbf{t}$ as $\tilde{\mathbf{t}}$ may achieve higher utility than the reporting policy π_0 that truthfully reports $\mathbf{t}$, *i.e.*,

$$U_\pi(\tilde{\mathbf{t}}, \mathbf{t}) > U_{\pi_0}(\mathbf{t}, \mathbf{t}), \quad \text{and} \quad v\left(\tilde{\mathbf{t}}\right) = v(\mathbf{t}).$$

In other words, the provider has an incentive not to be truthful and potentially overcharge the user, and they can do so in a way that maintains the value the user derives from the reported output sequence. In what follows, we will explore the conditions under which such strategic behavior can occur and remain undetected by the user. Later on, we will propose a pay-per-character pricing mechanism that provably eliminates the provider's incentive for this type of strategic behavior.

3 Provider Incentives under the Pay-Per-Token Pricing Mechanism

In this section, we analyze the pay-per-token-pricing mechanism using the principal-agent model introduced in Section 2. First, we show that, under this mechanism, the provider's utility is tightly linked to the length of the reported output token sequence—the longer the reported sequence, the higher the provider's utility—and this creates an incentive for the provider to (mis-)report an output's tokenization. Then, we further show that requiring the provider to be transparent about the next-token distributions used by the LLM they serve places obstacles in their ability to misreport tokenizations, as they cannot expect to find the longest plausible tokenization of a given output in polynomial time. However, we demonstrate that, in practice, this computational hardness does not preclude the provider from efficiently finding plausible tokenizations of a given output that increase their utility.

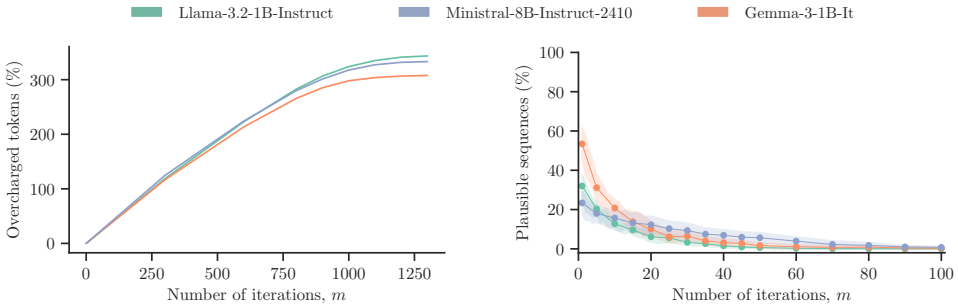
3.1 Pay-Per-Token Incentivizes (Mis-)Reporting Longer Tokenizations

To be profitable, a cloud-based LLM provider needs to at least cover their costs of generating outputs. Therefore, under the assumption that the cost of generating a single output is a linear function of its length, the widely used pay-per-token pricing mechanism is a natural choice.

Definition 2 (Pay-per-token)**.** *A pricing mechanism $r\colon \mathcal{V}^* \to \mathbb{R}_{\geq 0}$ is called pay-per-token if and only if it is additive and, for all $t \in \mathcal{V}$, it satisfies that $r(t) = r_o$, where $r_o \geq 0$ is a constant price per token.*

As an immediate consequence, under the pay-per-token pricing mechanism, the revenue that the provider earns from (mis-)reporting an output token sequence $\tilde{\mathbf{t}}$ is a linear function of the output length, *i.e.*, $r(\tilde{\mathbf{t}}) = r_o \cdot \texttt{len}(\tilde{\mathbf{t}})$. Further, since the cost to generate the output sequence $\mathbf{t}$ is independent of the reported output sequence $\tilde{\mathbf{t}}$, it is easy to see that, for any pair of reporting policies π and π' with the same running cost $c_\pi(\mathbf{t}) = c_{\pi'}(\mathbf{t})$, it holds that

$$U_\pi(\tilde{\mathbf{t}}, \mathbf{t}) > U_{\pi'}(\tilde{\mathbf{t}}', \mathbf{t}) \quad \text{for any} \quad \tilde{\mathbf{t}} \sim \pi(\mathbf{t}) \quad \text{and} \quad \tilde{\mathbf{t}}' \sim \pi'(\mathbf{t}) \quad \text{such that} \quad \texttt{len}(\tilde{\mathbf{t}}) > \texttt{len}(\tilde{\mathbf{t}}'). \tag{2}$$



(a) Overcharged tokens vs. m

(b) Plausible tokenizations vs. m

Figure 1: **Misreporting the tokenization of outputs generated by different LLMs using Algorithm 1.** Panel (a) shows the percentage of tokens overcharged by an unfaithful provider who misreports the tokenization of the outputs to 600 prompts from the LMSYS Chatbot Arena platform using Algorithm 1, for different values of m. Panel (b) shows the fraction of outputs to the same 600 prompts where Algorithm 1 returns a tokenization that is plausible under top-p sampling, for different values of m. In both panels, we set the temperature to 1.3 and use top-p sampling with $p = 0.95$. In panel (b), we repeat the experiment 5 times to obtain 90% confidence intervals.

Therefore, a rational provider has a financial incentive to use reporting policies π favoring tokenizations longer than those generated by the LLM they serve, *i.e.*, $\texttt{len}(\tilde{\mathbf{t}}) > \texttt{len}(\mathbf{t})$ for any generated output $\mathbf{t}$ and $\tilde{\mathbf{t}} \sim \pi(\mathbf{t})$. A natural choice among such reporting policies is the policy π_m, which constructs $\tilde{\mathbf{t}}$ by (randomly) splitting up to m tokens in $\mathbf{t}$ and, as shown in Algorithm 1, can be implemented with no GPU computations. Clearly, the financial incentive for (mis-)reporting tokenizations using π_m increases (linearly) with m, until all reported tokens become single characters, as shown in Figure 1a. However, the larger the value of m, the lower the chances that the reported tokenization $\tilde{\mathbf{t}}$ could have been generated by the LLM the provider serves, as shown in Figure 1b. This observation lends support to the idea that the provider should not be required to report only an output sequence, but also the next-token probability corresponding to each token in the sequence, offering the user the means to contest a reported output token sequence.[7] Next, we will show that an unfaithful provider who aims to find the reporting policy π with the highest utility among those under which $\tilde{\mathbf{t}} \sim \pi(\mathbf{t})$ is always plausible for any generated output $\mathbf{t}$ is likely to fail.

3.2 Misreporting Optimally Without Raising Suspicion Is Hard

In this section, we will focus on a setting in which the provider is transparent about next-token probabilities and implements top-p sampling [49], a widely used sampling technique that, given a (partial) token sequence $\mathbf{t}$, restricts the sampling of the next token to the smallest set $\mathcal{V}_p(\mathbf{t}) \subseteq \mathcal{V}$ whose cumulative next-token probability is at least $p \in (0, 1)$. Then, given a generated output sequence $\mathbf{t}$ with $s = \texttt{str}(\mathbf{t})$, the provider aims to find a reporting policy π with the highest utility among those under which $\tilde{\mathbf{t}} \sim \pi(\mathbf{t})$ is always plausible. In what follows, we will show that the provider cannot expect to find such a policy in polynomial time.

Our starting point is the realization that, under the pay-per-token pricing mechanism, the problem of finding a policy π with the highest utility among those under which $\tilde{\mathbf{t}} \sim \pi(\mathbf{t})$ is always plausible is at least as hard as finding the longest plausible tokenization $\tilde{\mathbf{t}}$ of s, *i.e.*,

$$\begin{aligned} \max_{\tilde{\mathbf{t}} \in \mathcal{V}_s^*} \quad & \texttt{len}\left(\tilde{\mathbf{t}}\right) \\ \text{subject to} \quad & \tilde{t}_i \in \mathcal{V}_p(\tilde{\mathbf{t}}_{\leq i-1}) \ \forall i \in [\texttt{len}\left(\tilde{\mathbf{t}}\right)], \end{aligned} \tag{3}$$

where $\tilde{\mathbf{t}}_{\leq i-1} = (\tilde{t}_1, \ldots, \tilde{t}_{i-1})$ is the prefix of the reported output sequence up to the i-th token. This is because this latter problem is a particular instance of the problem the provider aims to solve, where the cost $c_\pi(\mathbf{t})$ is constant for all output token sequences $\mathbf{t}$ and all candidate policies π available

[7] Users with access to large amounts of reported output token sequences may be able to contest them using statistical analysis techniques, as discussed in Section 5.

Algorithm 2 It returns a plausible token sequence $\tilde{\mathbf{t}}$ longer or equal than $\mathbf{t}$

Input True output token sequence $\mathbf{t}$, number of iterations m, token-to-id function $\texttt{id}(\bullet)$
Initialize $\hat{\mathbf{t}} \leftarrow \mathbf{t}$
for m iterations **do**
  $i \leftarrow \text{argmax}_{j \in [\texttt{len}(\hat{\mathbf{t}})]} \texttt{id}(\hat{t}_j)$ ▷ Pick the token with the highest index
  if $|\texttt{str}\left(\hat{t}_i\right)| = 1$ **then**
    break ▷ If it corresponds to a single character, terminate the loop
  end if
  $(t'_1, t'_2) \leftarrow \text{argmax}_{v_1, v_2 \in \mathcal{V}: \texttt{str}((v_1, v_2)) = \texttt{str}(\hat{t}_i)} \min\left(\texttt{id}(v_1), \texttt{id}(v_2)\right)$
  $\hat{\mathbf{t}} \leftarrow \left(\hat{\mathbf{t}}_{<i}, t'_1, t'_2, \hat{\mathbf{t}}_{>i}\right)$ ▷ If not, split it into a pair of tokens with the max-min index
end for
if $\texttt{plausible}\left(\hat{t}\right)$ **then**
  $\tilde{\mathbf{t}} \leftarrow \hat{\mathbf{t}}$ ▷ If the resulting token sequence is plausible, report it to the user
else
  $\tilde{\mathbf{t}} \leftarrow \mathbf{t}$ ▷ If not, report the true output token sequence
end if
return $\tilde{\mathbf{t}}$

to the provider. In this context, it is worth noting that a provider may be interested in solving this particular instance of the problem if, for example, they are considering only policies that check (once) the plausibility of a modified sequence $\tilde{\mathbf{t}} \neq \mathbf{t}$, such as the policy implemented by Algorithm 2 in Section 3.3.

The following theorem tells us that, in general, one cannot expect to find the longest plausible tokenization $\mathbf{t}$ of s in polynomial time and, hence, a provider aiming to find a policy π with the highest utility among those under which $\tilde{\mathbf{t}} \sim \pi(\mathbf{t})$ is always plausible is likely to fail:[8]

Theorem 3. *The problem of finding the longest tokenization of a given output that is plausible under top-p sampling, as defined in Eq. 3, is NP-Hard.*

The proof of the above theorem relies on a reduction from the Hamiltonian path problem [50]. More specifically, given a graph, it creates an instance of our problem that establishes a one-to-one correspondence between a path that does not visit any node twice and a token sequence that is plausible only if it does not include any token twice. In Appendix B.1.1, we show that the above hardness result can be extended to a setting in which the provider implements top-k sampling and, in Appendix B.1.2, we show that it can also be extended to a setting in which the provider does not implement any procedure to prevent the generation of low-probability tokens but aims to report sequences whose generation probability is greater than a given threshold.

Further, the above hardness result readily implies that there exists a computational barrier that precludes an unfaithful transparent provider from optimally benefiting from misreporting without raising suspicion. However, we will next demonstrate that, in practice, it does not rule out the possibility that a provider efficiently finds a reporting policy π under which $\tilde{\mathbf{t}} \sim \pi(\mathbf{t})$ is always plausible for any generated output $\mathbf{t}$ offering higher utility than the (truthful) reporting policy π_0.

3.3 Can a Provider Overcharge a User Without Raising Suspicion?

We answer this question affirmatively. As a proof-of-concept, we introduce a simple heuristic algorithm that, given a generated output sequence $\mathbf{t}$ with $s = \texttt{str}(\mathbf{t})$, finds a plausible tokenization $\tilde{\mathbf{t}}$ of s longer than or equal to $\mathbf{t}$, and show that this algorithm can be used to construct a reporting policy π_m offering higher utility than the (truthful) reporting policy π_0. Here, our goal is to demonstrate that, under the pay-per-token pricing mechanism predominantly used by cloud providers of LLM-as-a-service, users are indeed vulnerable to self-serving providers who may overcharge them without raising suspicion.

Our heuristic algorithm, summarized in Algorithm 2, is based on the key empirical observation that, given the most likely tokenization $\mathbf{t}$ of a string $s = \texttt{str}(\mathbf{t})$, alternative tokenizations of s that are not *too different* from $\mathbf{t}$ are very likely to be plausible, as shown in Figure 1b. In a nutshell, our

[8] All proofs of theorems and propositions can be found in Appendix B.

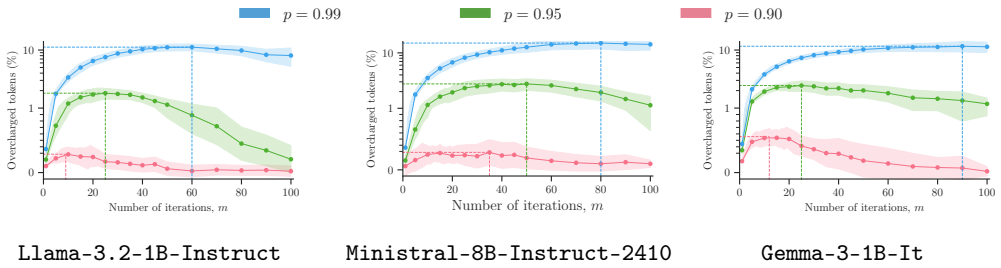


Figure 2: **Additional revenue from misreporting the tokenization of outputs using Algorithm 2.** The panels show the percentage of tokens overcharged by an unfaithful provider who misreports the tokenization of the outputs generated by an LLM to 600 prompts from the LMSYS Chatbot Arena platform using the reporting policies π_m implemented by Algorithm 2, for different values of m and p. The dashed vertical lines correspond to the optimal value of m. Here, we set the temperature of the model to 1.3 and repeat each experiment 5 times to obtain 90% confidence intervals. Refer to Appendix C.1 for additional results using alternative temperature values and other LLMs.

algorithm starts from a given output sequence $\mathbf{t}$ and iteratively splits tokens in it for a number of iterations m specified by the provider. In each iteration, the algorithm selects the token with the highest index in the vocabulary and, if it is longer than one character, it splits it into a pair of new tokens with the highest minimum index in the vocabulary whose concatenation maps to the same string.[9] The algorithm continues either until it has performed m splits or all tokens in the sequence are single characters, in which case it terminates the loop. Finally, it checks whether the resulting token sequence $\hat{\mathbf{t}}$ is plausible and, if it is indeed plausible, it reports it to the user. For example, under top-p sampling, evaluating plausibility reduces to checking whether $\hat{t}_i \in \mathcal{V}_p(\hat{\mathbf{t}}_{\leq i-1})$ for all $i \in [\texttt{len}(\hat{\mathbf{t}})]$. However, our algorithm is agnostic to the choice of plausibility criteria (refer to Appendices B.1.1 and B.1.2 for alternatives). If $\hat{\mathbf{t}}$ is not plausible, the algorithm reports the true output token sequence $\mathbf{t}$.

Using prompts from the LMSYS Chatbot Arena platform, we find empirical evidence that, despite its simplicity, the reporting policy π_m implemented by Algorithm 2 succeeds at helping a provider overcharge users whenever they serve LLMs with temperature values >1.0, as those commonly used in creative writing tasks. Figure 2 summarizes the results, which show that, for example, for `Llama-3.2-1B-Instruct`, a provider who uses Algorithm 2 can overcharge users by up to 11.2%, 1.8% and 0.28%, respectively for $p = 0.99, 0.95, 0.9$. Moreover, the results also reveal that the additional revenue is unimodal with respect to the number of iterations m, and the optimal value of m decreases as p decreases and achieving plausibility becomes harder. This is because, for large values of m, the token sequence $\hat{\mathbf{t}}$ resulting from iteratively splitting tokens becomes less likely to be plausible, as shown in Figure 5 in Appendix C.1. However, if plausible, it does provide a strictly larger additional revenue.

The above experimental results show that Algorithm 2 allows a provider to obtain additional revenue when exclusively misreporting plausible tokenizations. However, they do not directly imply that the provider's utility would increase. This is because, in order to verify the plausibility of $\hat{\mathbf{t}}$, the reporting policy π_m implemented by Algorithm 2 requires the evaluation of the next-token probabilities that a model assigns to $\hat{\mathbf{t}}$ using a forward pass, *i.e.*, $c_\pi(\mathbf{t}) > 0$ in Eq. 1. In what follows, we will demonstrate that the additional revenue from misreporting can largely surpass the cost of verifying plausibility, making misreporting a financially profitable strategy.

The key insight is that, as we demonstrate empirically in Figure 11 and Table 1 of Appendix C, and in agreement with recent works [45], the energy costs of GPU computations required to evaluate the probability a model assigns to a tokenization does not depend on the specific sequence (or its length). Therefore, we can readily conclude that $c_{\pi_m}(\mathbf{t}) = c_{\mathrm{v}}$ is constant and, for a given number of iterations m, the reporting policy π_m implemented by Algorithm 2 offers a gain in average utility with respect to the (truthful) policy π_0 if and only if

[9] We focus on splitting tokens based on their index, motivated by the BPE algorithm, where tokens with higher indices are (generally) longer, and hence are more likely to result in a plausible tokenization. Refer to Appendix C.2 for concrete examples of how our heuristic modifies token sequences.

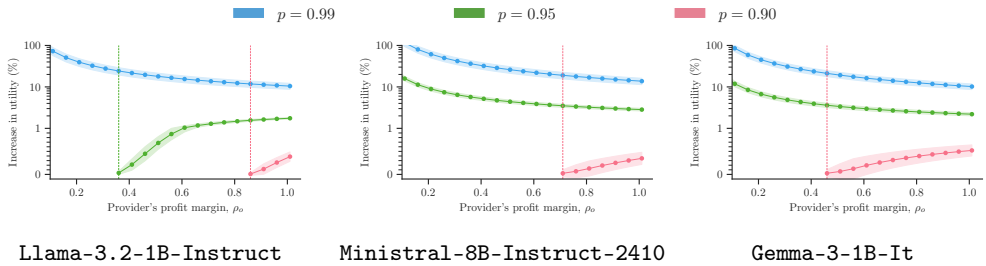


Figure 3: **Financial gain from misreporting the tokenization of outputs using Algorithm 2.** The panels show the utility gain that an unfaithful provider who misreports the tokenization of the outputs generated by an LLM to 600 prompts from the LMSYS Chatbot Arena platform using Algorithm 2 can achieve, for different values of p and the provider's profit margin ρ_0. The dashed vertical lines represent the minimum margin above which misreporting is financially viable according to Eq. 4. Here, for each value of p, we run Algorithm 2 using the optimal number of iterations m shown in Figure 2, set the temperature of the model to 1.3 and repeat each experiment 5 times to obtain 90% confidence intervals. Refer to Figure 7 of Appendix C.1 for additional results using alternative temperature values and other LLMs.

$$\mathbb{E}\left[\texttt{plausible}\left(\hat{\mathbf{t}}\right)\right] \cdot m \cdot r_o > c_{\mathrm{v}} \iff \rho(\mathbf{t}) > 1 - \mathbb{E}\left[\texttt{plausible}\left(\hat{\mathbf{t}}\right)\right] \cdot m \cdot \frac{c_o}{c_{\mathrm{v}}}, \tag{4}$$

where the expectation over $\hat{\mathbf{t}}$ implicitly depends on the choice of m and $\rho(\mathbf{t}) = 1 - c_{\text{gen}}(\mathbf{t})/r(\mathbf{t}) = 1 - c_o/r_o = \rho_o$ is the provider's profit margin under a pay-per-token pricing mechanism, which is constant since both $c_{\text{gen}}(\mathbf{t})$ and $r(\mathbf{t})$ are directly proportional to $\texttt{len}(\mathbf{t})$.

Using the same prompts from the LMSYS Chatbot Arena platform as above and the optimal number of iterations m shown in Figure 2, we find that, for a wide range of the provider's profit margin values ρ_o, the reporting policy π_m implemented by Algorithm 2 offers higher average utility than the (truthful) policy π_0. Figure 3 summarizes the results, which show that, for example, for `Llama-3.2-1B-Instruct`, the utility gain surpasses 10.5% for $p = 0.99$ regardless of the margin, and reaches 1.7% and 0.4% for $p = 0.95$ and $p = 0.90$, respectively. Interestingly, the results also reveal that, for high values of p, as the provider's margin decreases, the relative increase in utility from misreporting increases, making misreporting particularly advantageous in a competitive market.

The above empirical results demonstrate that, even if a provider is transparent about next-token probabilities and (mis-)reports only plausible tokenizations, users remain vulnerable to the (potentially) unfaithful behavior from the provider. To address this vulnerability, in the next section, we study incentive-compatible pricing mechanisms that provably eliminate the provider's incentive to misreport an output token sequence.

4 An Incentive-Compatible Pricing Mechanism

To eliminate the provider's incentive to misreport an output token sequence, in this section, we look into the design of incentive-compatible pricing mechanisms. Incentive-compatibility is a (desirable) property studied in mechanism design [51] that, in the context of our work, ensures that the pricing mechanism creates no economic incentive for the provider to misreport an output token sequence—they cannot benefit from not telling the truth.

Definition 4. *A pricing mechanism r is incentive-compatible if and only if, for any generated output token sequence $\mathbf{t} \in \mathcal{V}^*$, any reporting policy π and any $\tilde{\mathbf{t}} \sim \pi(\mathbf{t})$, it holds that $U_{\pi_0}(\mathbf{t}, \mathbf{t}) \geq U_{\pi}(\tilde{\mathbf{t}}, \mathbf{t})$, where π_0 is the policy that faithfully reports the generated sequence,* i.e., $\pi_0(\mathbf{t}) = \mathbf{t}$.

Importantly, if a pricing mechanism satisfies incentive-compatibility, the revenue a provider earns for reporting an output token sequence $\tilde{\mathbf{t}}$ depends only on the string $s = \texttt{str}\left(\tilde{\mathbf{t}}\right)$ and not on the token sequence itself, as shown by the following proposition:

Proposition 5. *If a pricing mechanism r is incentive-compatible, then, for all $\tilde{\mathbf{t}}, \tilde{\mathbf{t}}' \in \mathcal{V}^*$ such that $\texttt{str}\left(\tilde{\mathbf{t}}\right) = \texttt{str}\left(\tilde{\mathbf{t}}'\right)$, it holds that $r\left(\tilde{\mathbf{t}}\right) = r\left(\tilde{\mathbf{t}}'\right)$.*

Perhaps surprisingly, the above proposition readily allows us to provide a simple characterization of the family of incentive-compatible pricing mechanisms. In particular, the following theorem tells us that it consists of all mechanisms that charge for an output sequence $\mathbf{t}$ linearly on its character counts:

Theorem 6. *A pricing mechanism r is additive and incentive-compatible if and only if*

$$r(\mathbf{t}) = \sum_{\sigma \in \Sigma} \texttt{count}_\sigma(\mathbf{t}) \cdot r(\sigma) \text{ for all } \mathbf{t} \in \mathcal{V}^*, \tag{5}$$

where $\texttt{count}_\sigma(\mathbf{t})$ counts the number of occurrences of the character σ in $\texttt{str}(\mathbf{t})$.

This characterization directly implies that, if the provider decides to assign the same price r_c to each character $\sigma \in \Sigma$, there exists only one incentive-compatible pricing mechanism, *i.e.*, $r(\mathbf{t}) = |\texttt{str}(\mathbf{t})| \cdot r_c$, which we refer to as the *pay-per-character* pricing mechanism. Moreover, the above theorem also implies that, as long as the vocabulary $\mathcal{V}$ contains tokens composed of more than one character, which is typically the case, the pay-per-token pricing mechanism is provably not incentive-compatible:

Corollary 7. *If a pricing mechanism r is additive and incentive-compatible, there exists no constant $r_0 > 0$ such that $r(\mathbf{t}) = r_0 \cdot \texttt{len}(\mathbf{t})$ for all $\mathbf{t} \in \mathcal{V}^*$.*

Further, it is easy to see that the revenue $r(\mathbf{t})$ earned by a provider who adopts an incentive-compatible pricing mechanism cannot be directly proportional to the energy costs $c_{\text{gen}}(\mathbf{t}) \approx c_0 \cdot \texttt{len}(\mathbf{t})$ of GPU computations required to generate the output sequence $\mathbf{t}$. Consequently, the provider's profit margin $\rho(\mathbf{t}) = 1 - c_{\text{gen}}(\mathbf{t})/r(\mathbf{t})$ fluctuates with the number of characters per token in the output sequence $\mathbf{t}$: higher values yield higher profit margins, while lower values yield lower profit margins. This is because, given a fixed string s, the provider's revenue is independent of the output's tokenization due to Proposition 5 and their profit margin increases as the length of $\mathbf{t}$ (and the energy costs associated with generating it) decreases.

A variable profit margin establishes an economic incentive for providers to continually develop LLMs and tokenization algorithms that achieve better compression than the state-of-the-art, thereby making better use of (limited) context windows and energy resources. At the same time, it also introduces an operational challenge for providers: under incentive-compatible pricing, a provider's profit margin cannot be determined exactly in advance, as it depends on the user prompt and the number of characters per token in the respective output sequence. However, as long as the number of characters in an output sequence positively correlates with the number of tokens—which is arguably the case in most realistic scenarios—the revenue and energy costs of generation do remain proportional on expectation over the distribution of (outputs to) user prompts under a pay-per-character pricing mechanism.

Building on this observation, we propose a straightforward prescription that enables a (faithful) provider who transitions from a pay-per-token to a pay-per-character pricing mechanism to maintain their *average* profit margin across outputs to user prompts. Under this prescription, given a dataset of prompts and output sequences, the provider assigns a price per character $r_c = r_o \cdot \texttt{tpc}$, where r_o is the price of a single token under the provider's current pay-per-token pricing mechanism, and $\texttt{tpc}$ is the (empirical) average ratio of tokens to characters, computed by first calculating the number of tokens divided by the number of characters for each individual output in the dataset and then averaging these ratios across all outputs.

Using responses to prompts from the LMSYS Chatbot Arena platform, we find that a provider who transitions from pay-per-token to pay-per-character following the above prescription maintains their average profit margin, as expected. Moreover, we also find that, as shown in Figure 4, the provider's profit margin under pay-per-character is positive for the (vast) majority of output sequences, and it is only negative for a small number of output sequences. For example, for `Llama-3.2-1B-Instruct`, the provider's profit margin under pay-per-character is positive for 92.4%, 93.3%, 94.1% of the output sequences for $\rho_o = 0.2$, 0.4, and 0.6, respectively. Here, it is also worth noting that output sequences leading to a negative profit margin are those with a lower than average number of characters per token, such as the ones responding to non-English prompts, as shown in Appendix C.4.

5 Discussion and Limitations

In this section, we discuss several assumptions and limitations of our work and propose avenues for future work.

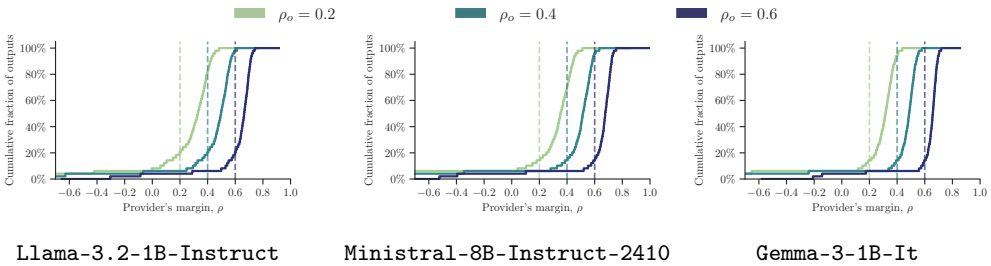


Figure 4: **Provider's profit margin under a pay-per-character pricing mechanism.** The panels show, across different models and different values ρ_o of the provider's profit margin under pay-per-token, the (empirical) cumulative distribution function of their profit margin $\rho(\mathbf{t}) = 1 - c_{\text{gen}}(\mathbf{t})/r(\mathbf{t})$ per output $\mathbf{t}$ after their transition to pay-per-character. Here, we first compute the average ratio of number of tokens to number of characters ($\texttt{tpc}$) across the responses of each model to 600 multilingual prompts sampled from the LMSYS Chatbot Arena dataset, with sampling proportional to each language's frequency. Then, we set $r_c = r_o \cdot \texttt{tpc}$ and compute the provider's profit margin $\rho(\mathbf{t})$ for outputs to 600 different multilingual prompts. In all panels, dashed vertical lines show empirical averages of the respective distributions, and we set the temperature of the models to 1.3 and use top-p sampling with $p = 0.95$.

Model assumptions. We have focused on additive pricing mechanisms, which include the widely used pay-per-token mechanism. It would be interesting to analyze provider incentives under other families of pricing mechanisms proposed in the literature, such as those based on the quality of the generated text [35]. In this context, a natural direction is to design a pricing mechanism that simultaneously incentivizes multiple desirable behaviors, such as faithful token reporting and output quality. Moreover, in the context of contract theory, a principal typically designs a contract in order to disincentivize the agent from taking hidden, unwanted actions [13]. In our case, the provider (*i.e.*, the agent) is the one who both designs the pricing mechanism (*i.e.*, the contract) and has the power to take hidden actions, leaving the user with limited leverage. Practically, a shift from pay-per-token to other pricing mechanisms, such as pay-per-character, would require external regulation (or user pressure). In this context, we note that, while the hardness result in Theorem 3 holds without additional assumptions, providers would arguably only be forced to report plausible tokenizations if the users have a mechanism to verify that a token sequence is in fact plausible. This raises the question of rigorously defining the information that both parties have access to when interacting, and how to verify that tokens are generated according to the advertised inference process. We leave this for future work and point towards trusted execution environments [52] and zero-knowledge proofs [53] as potential solutions to this problem.

Methods. To demonstrate the vulnerability of users under the pay-per-token pricing mechanism, we have introduced a heuristic algorithm that allows the provider to increase their utility by finding longer yet plausible tokenizations of the true output token sequence. However, there may exist other, more sophisticated methods for the provider to take advantage of the pay-per-token pricing mechanism, and there may also exist ways to defend users against such malicious behavior, other than a change of the pricing mechanism [54]. In this context, it would also be interesting to develop statistical methods to audit the tokenizations reported by a provider. Further, it is important to note that providers of proprietary models can also misreport other aspects of the generative process—such as the next-token distributions—or publicly release a tokenizer different from the one used internally by the model, leaving users with no means of detecting whether a provider is overcharging them. Lastly, we note that, similarly to pay-per-token, the pay-per-character pricing does not prevent the provider from artificially increasing the number of characters in the output string by increasing the model's verbosity.

Evaluation. We have conducted experiments with state-of-the-art open-weights LLMs from the `Llama`, `Gemma` and `Ministral` families, using different tokenizers and architectures. It would be interesting to evaluate the possibility of misreporting in proprietary LLMs, which are widely used in practice. Further, we have illustrated our theoretical results using prompts from the LMSYS Chatbot Arena platform. Although this platform is the most widely used for LLM evaluation based on pairwise comparisons, it is important to note that it has been recently criticized [55, 56], and the prompts submitted to it may not be representative of the real-world distribution of user prompts.

Further, we would like to emphasize that our work deliberately focuses on the micro level [57], modeling incentives in the isolated interaction between a single user and a single provider. However, altering the pricing mechanism could generate non-trivial downstream effects across the broader LLM-as-a-service ecosystem. Such changes may influence how a diverse user base selects and engages with providers based on their advertised pricing mechanisms and, in turn, reshape the competitive landscape when multiple providers are present. Extending the analysis to these broader market dynamics would require separate treatment, involving additional modeling assumptions, and remains an interesting direction for future work.

6 Conclusions

In this work, we have studied the financial incentives of cloud-based providers in LLM-as-a-service using a principal-agent model of delegated autoregressive generation. We have demonstrated that the widely used pay-per-token pricing mechanism incentivizes a provider to misreport the tokenization of the outputs generated by the LLM they serve. We have shown that, if the provider is required to be transparent about the generative process used by the LLM, it is provably hard for the provider to optimally benefit from misreporting without raising suspicion. However, we have introduced an efficient algorithm that, in practice, allows a transparent provider to benefit from misreporting, overcharging users significantly without raising suspicion. To address this vulnerability, we have introduced a simple incentive-compatible pricing mechanism, pay-per-character, which eliminates the financial incentive for misreporting tokenizations. We hope that our work will raise awareness that, under pay-per-token, users of LLM-as-a-service are vulnerable to (unfaithful) providers, and encourage a paradigm shift towards alternative pricing mechanisms, such as pay-per-character.

References

[1] Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nat.*, 625(7995):468–475, January 2024. URL `https://doi.org/10.1038/s41586-023-06924-6`.

[2] Iddo Drori, Sarah Zhang, Reece Shuttleworth, Leonard Tang, Albert Lu, Elizabeth Ke, Kevin Liu, Linda Chen, Sunny Tran, Newman Cheng, et al. A neural network solves, explains, and generates university math problems by program synthesis and few-shot learning at human level. *Proceedings of the National Academy of Sciences*, 119(32):e2123433119, 2022.

[3] Claudia E. Haupt and Mason Marks. Ai-generated medical advice—gpt and beyond. *JAMA*, 329(16): 1349–1350, 04 2023. ISSN 0098-7484. doi: 10.1001/jama.2023.5321. URL `https://doi.org/10.1001/jama.2023.5321`.

[4] Yinheng Li, Shaofei Wang, Han Ding, and Hang Chen. Large language models in finance: A survey. In *Proceedings of the fourth ACM international conference on AI in finance*, pages 374–382, 2023.

[5] Sebastião Pais, João Cordeiro, and M. Luqman Jamil. Nlp-based platform as a service: a brief review. *Journal of Big Data*, 9(1), April 2022. ISSN 2196-1115. doi: 10.1186/s40537-022-00603-5. URL `http://dx.doi.org/10.1186/s40537-022-00603-5`.

[6] Paul Milgrom and John Roberts. Informational asymmetries, strategic behavior, and industrial organization. *The American Economic Review*, 77(2):184–193, 1987.

[7] Eric Rasmusen. *Games and information*, volume 13. Basil Blackwell Oxford, 1989.

[8] Debi Prasad Mishra, Jan B Heide, and Stanton G Cort. Information asymmetry and levels of agency relationships. *Journal of marketing Research*, 35(3):277–295, 1998.

[9] Bengt Holmström. Moral hazard and observability. *The Bell journal of economics*, pages 74–91, 1979.

[10] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *8th International Conference on Learning Representations, ICLR 2020*, 2020. URL `https://openreview.net/forum?id=rygGQyrFvH`.

[11] Nicholas Carlini, Daniel Paleka, Krishnamurthy (Dj) Dvijotham, Thomas Steinke, Jonathan Hayase, A. Feder Cooper, Katherine Lee, Matthew Jagielski, Milad Nasr, Arthur Conmy, Eric Wallace, David Rolnick, and Florian Tramèr. Stealing part of a production language model. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24, 2024.

[12] Sanford J Grossman and Oliver D Hart. An analysis of the principal-agent problem. In *Foundations of insurance economics: Readings in economics and finance*, pages 302–340. Springer, 1992.

[13] Paul Dütting, Michal Feldman, Inbal Talgam-Cohen, et al. Algorithmic contract theory: A survey. *Foundations and Trends in Theoretical Computer Science*, 16(3-4):211–412, 2024.

[14] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric Xing, Joseph E. Gonzalez, Ion Stoica, and Hao Zhang. LMSYS-chat-1m: A large-scale real-world LLM conversation dataset. In *12th International Conference on Learning Representations, ICLR 2024*, 2024. URL https://openreview.net/forum?id=BOfDKxfwt0.

[15] Renato Geh, Honghua Zhang, Kareem Ahmed, Benjie Wang, and Guy Van Den Broeck. Where is the signal in tokenization space? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 3966–3979, November 2024. doi: 10.18653/v1/2024.emnlp-main.230. URL https://aclanthology.org/2024.emnlp-main.230/.

[16] Mario Giulianelli, Luca Malagutti, Juan Luis Gastaldi, Brian DuSell, Tim Vieira, and Ryan Cotterell. On the proper treatment of tokenization in psycholinguistics. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 18556–18572, November 2024. doi: 10.18653/v1/2024.emnlp-main.1032. URL https://aclanthology.org/2024.emnlp-main.1032/.

[17] Renato Geh, Zilei Shao, and Guy Van Den Broeck. Adversarial tokenization. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 20738–20765, Vienna, Austria, July 2025. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1012. URL https://aclanthology.org/2025.acl-long.1012/.

[18] Aleksandar Petrov, Emanuele La Malfa, Philip Torr, and Adel Bibi. Language model tokenizers introduce unfairness between languages. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=78yDLKi95p.

[19] Anaelia Ovalle, Ninareh Mehrabi, Palash Goyal, Jwala Dhamala, Kai-Wei Chang, Richard Zemel, Aram Galstyan, Yuval Pinter, and Rahul Gupta. Tokenization matters: Navigating data-scarce tokenization for gender inclusive language technologies. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 1739–1756, June 2024. doi: 10.18653/v1/2024.findings-naacl.113. URL https://aclanthology.org/2024.findings-naacl.113/.

[20] Ivi Chatzi, Nina L. Corvelo Benz, Eleni Straitouri, Stratis Tsirtsis, and Manuel Gomez Rodriguez. Counterfactual token generation in large language models. In *Fourth Conference on Causal Learning and Reasoning*, 2025. URL https://openreview.net/forum?id=YYztiQ2Vke.

[21] Nina Corvelo Benz, Stratis Tsirtsis, Eleni Straitouri, Ivi Chatzi, Ander Artola Velasco, Suhas Thejaswi, and Manuel Gomez-Rodriguez. Evaluation of large language models via coupled token generation. *arXiv preprint arXiv:2502.01754*, 2025.

[22] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, August 2016. doi: 10.18653/v1/P16-1162. URL https://aclanthology.org/P16-1162/.

[23] Kaj Bostrom and Greg Durrett. Byte pair encoding is suboptimal for language model pretraining. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4617–4624, November 2020. doi: 10.18653/v1/2020.findings-emnlp.414. URL https://aclanthology.org/2020.findings-emnlp.414/.

[24] Vilém Zouhar, Clara Meister, Juan Gastaldi, Li Du, Tim Vieira, Mrinmaya Sachan, and Ryan Cotterell. A formal perspective on byte-pair encoding. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 598–614, July 2023. doi: 10.18653/v1/2023.findings-acl.38. URL https://aclanthology.org/2023.findings-acl.38/.

[25] Haoran Lian, Yizhe Xiong, Jianwei Niu, Shasha Mo, Zhenpeng Su, Zijia Lin, Hui Chen, Jungong Han, and Guiguang Ding. Scaffold-bpe: enhancing byte pair encoding for large language models with simple and effective scaffold token removal. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'25/IAAI'25/EAAI'25. AAAI Press, 2025. ISBN 978-1-57735-897-8. doi: 10.1609/aaai.v39i23.34633. URL https://doi.org/10.1609/aaai.v39i23.34633.

[26] Haoran Lian, Yizhe Xiong, Zijia Lin, Jianwei Niu, Shasha Mo, Hui Chen, Peng Liu, and Guiguang Ding. Lbpe: Long-token-first tokenization to improve large language models. In *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2025. doi: 10.1109/ICASSP49660.2025.10889733.

[27] Sara Fish, Paul Gölz, David C. Parkes, Ariel D. Procaccia, Gili Rusak, Itai Shapira, and Manuel Wüthrich. Generative social choice. In *Proceedings of the 25th ACM Conference on Economics and Computation*, EC '24, page 985, 2024. ISBN 9798400707049. doi: 10.1145/3670865.3673547. URL https://doi.org/10.1145/3670865.3673547.

[28] John J Horton. Large language models as simulated economic agents: What can we learn from homo silicus? Technical report, National Bureau of Economic Research, 2023.

[29] Vojtěch Kovařík, Caspar Oesterheld, and Vincent Conitzer. Game theory with simulation of other players. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pages 2800–2807, 8 2023. doi: 10.24963/ijcai.2023/312. URL https://doi.org/10.24963/ijcai.2023/312. Main Track.

[30] Narun Raman, Taylor Lundy, Samuel Joseph Amouyal, Yoav Levine, Kevin Leyton-Brown, and Moshe Tennenholtz. Steer: assessing the economic rationality of large language models. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24, 2024.

[31] Emanuele La Malfa, Aleksandar Petrov, Simon Frieder, Christoph Weinhuber, Ryan Burnell, Raza Nazar, Anthony Cohn, Nigel Shadbolt, and Michael Wooldridge. Language-models-as-a-service: Overview of a new paradigm and its challenges. *Journal of Artificial Intelligence Research*, 80:1497–1523, 2024.

[32] Rafid Mahmood. Pricing and competition for generative AI. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=8LbJfEjIrT.

[33] Benjamin Laufer, Jon Kleinberg, and Hoda Heidari. Fine-tuning games: Bargaining and adaptation for general-purpose models. In *Proceedings of the ACM Web Conference 2024*, WWW '24, page 66–76, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400701719. doi: 10.1145/3589334.3645366. URL https://doi.org/10.1145/3589334.3645366.

[34] Will Cai, Tianneng Shi, Xuandong Zhao, and Dawn Song. Are you getting what you pay for? auditing model substitution in llm apis. *arXiv preprint arXiv:2504.04715*, 2025.

[35] Eden Saig, Ohad Einav, and Inbal Talgam-Cohen. Incentivizing quality text generation via statistical contracts. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NeurIPS '24, 2025. ISBN 9798331314385.

[36] 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 2024*, WWW '24, page 144–155, 2024. ISBN 9798400701719. doi: 10.1145/3589334.3645511. URL https://doi.org/10.1145/3589334.3645511.

[37] Dirk Bergemann, Alessandro Bonatti, and Alex Smolin. The economics of large language models: Token allocation, fine-tuning, and optimal pricing. *arXiv preprint arXiv:2502.07736*, 2025.

[38] Guoheng Sun, Ziyao Wang, Bowei Tian, Meng Liu, Zheyu Shen, Shwai He, Yexiao He, Wanghao Ye, Yiting Wang, and Ang Li. Coin: Counting the invisible reasoning tokens in commercial opaque llm apis. *arXiv preprint arXiv:2505.13778*, 2025.

[39] Hal R. Varian. Buying, sharing and renting information goods. *The Journal of Industrial Economics*, 48(4): 473–488, 2000. doi: https://doi.org/10.1111/1467-6451.00133. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/1467-6451.00133.

[40] Yannis Bakos and Erik Brynjolfsson. Bundling and competition on the internet. *Marketing Science*, 19(1): 63–82, 2000. ISSN 07322399, 1526548X. URL http://www.jstor.org/stable/193259.

[41] Arun Sundararajan. Nonlinear pricing of information goods. *Management Science*, 50(12):1660–1673, 2004. doi: 10.1287/mnsc.1040.0291. URL https://doi.org/10.1287/mnsc.1040.0291.

[42] Vernon N. Hsu, Susan H. Xu, and Boris Jukic. Optimal scheduling and incentive compatible pricing for a service system with quality of service guarantees. *Manufacturing & Service Operations Management*, 11(3): 375–396, 2009. doi: 10.1287/msom.1080.0226. URL https://doi.org/10.1287/msom.1080.0226.

[43] Jens Witkowski, Rupert Freeman, Jennifer Wortman Vaughan, David M. Pennock, and Andreas Krause. Incentive-compatible forecasting competitions. *Management Science*, 69(3):1354–1374, 2023. doi: 10.1287/mnsc.2022.4410. URL https://doi.org/10.1287/mnsc.2022.4410.

[44] Santiago R. Balseiro and Antoine Désir. Incentive-compatible assortment optimization for sponsored products. *Management Science*, 69(8):4668–4684, 2023. doi: 10.1287/mnsc.2022.4603. URL https://doi.org/10.1287/mnsc.2022.4603.

[45] Jared Fernandez, Clara Na, Vashisth Tiwari, Yonatan Bisk, Sasha Luccioni, and Emma Strubell. Energy considerations of large language model inference and efficiency optimizations. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32556–32569, July 2025. ISBN 979-8-89176-251-0. URL https://aclanthology.org/2025.acl-long.1563/.

[46] Soham Poddar, Paramita Koley, Janardan Misra, Niloy Ganguly, and Saptarshi Ghosh. Towards sustainable NLP: Insights from benchmarking inference energy in large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 12688–12704, April 2025. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.632. URL https://aclanthology.org/2025.naacl-long.632/.

[47] Kris Cao and Laura Rimell. You should evaluate your language model on marginal likelihood over tokenisations. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2104–2114, November 2021. doi: 10.18653/v1/2021.emnlp-main.161. URL https://aclanthology.org/2021.emnlp-main.161/.

[48] Nadezhda Chirkova, Germán Kruszewski, Jos Rozen, and Marc Dymetman. Should you marginalize over possible tokenizations? In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–12, July 2023. doi: 10.18653/v1/2023.acl-short.1. URL https://aclanthology.org/2023.acl-short.1/.

[49] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *8th International Conference on Learning Representations, ICLR 2020*, 2020. URL https://openreview.net/forum?id=rygGQyrFvH.

[50] Richard M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA, 1972. ISBN 978-1-4684-2001-2. doi: 10.1007/978-1-4684-2001-2_9. URL https://doi.org/10.1007/978-1-4684-2001-2_9.

[51] Noam Nisan and Amir Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35 (1):166–196, 2001. ISSN 0899-8256. doi: https://doi.org/10.1006/game.1999.0790. URL https://www.sciencedirect.com/science/article/pii/S089982569990790X.

[52] Patrick Jauernig, Ahmad-Reza Sadeghi, and Emmanuel Stapf. Trusted execution environments: properties, applications, and challenges. *IEEE Security & Privacy*, 18(2):56–60, 2020.

[53] Haochen Sun, Jason Li, and Hongyang Zhang. zkllm: Zero knowledge proofs for large language models. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, CCS '24, page 4405–4419, 2024. ISBN 9798400706363. doi: 10.1145/3658644.3670334. URL https://doi.org/10.1145/3658644.3670334.

[54] Ivi Chatzi, Nina Corvelo Benz, Stratis Tsirtsis, and Manuel Gomez-Rodriguez. Canonical autoregressive generation. *arXiv preprint arXiv:2506.06446*, 2025.

[55] Shivalika Singh, Yiyang Nan, Alex Wang, Daniel D'Souza, Sayash Kapoor, Ahmet Üstün, Sanmi Koyejo, Yuntian Deng, Shayne Longpre, Noah A Smith, et al. The leaderboard illusion. *arXiv preprint arXiv:2504.20879*, 2025.

[56] Kun Zhou, Yutao Zhu, Zhipeng Chen, Wentong Chen, Wayne Xin Zhao, Xu Chen, Yankai Lin, Ji-Rong Wen, and Jiawei Han. Don't make your llm an evaluation benchmark cheater. *arXiv preprint arXiv:2311.01964*, 2023.

[57] A. Mas-Colell, M.D. Whinston, and J.R. Green. *Microeconomic Theory*. Microeconomic Theory. Oxford University Press, 1995. ISBN 9780195073409. URL https://books.google.de/books?id=KGtegVXqD8wC.

[58] Richard M. Karp. *Reducibility Among Combinatorial Problems*, pages 219–241. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. ISBN 978-3-540-68279-0. doi: 10.1007/978-3-540-68279-0_8. URL https://doi.org/10.1007/978-3-540-68279-0_8.

[59] Shiyue Zhang, Vishrav Chaudhary, Naman Goyal, James Cross, Guillaume Wenzek, Mohit Bansal, and Francisco Guzman. How robust is neural machine translation to language imbalance in multilingual tokenizer training? In Kevin Duh and Francisco Guzmán, editors, *Proceedings of the 15th biennial conference of the Association for Machine Translation in the Americas (Volume 1: Research Track)*, pages 97–116, Orlando, USA, September 2022. Association for Machine Translation in the Americas. URL `https://aclanthology.org/2022.amta-research.8/`.

[60] Orevaoghene Ahia, Sachin Kumar, Hila Gonen, Jungo Kasai, David Mortensen, Noah Smith, and Yulia Tsvetkov. Do all languages cost the same? tokenization in the era of commercial language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9904–9923, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.614. URL `https://aclanthology.org/2023.emnlp-main.614/`.

A Additional Experimental Details

Here, we provide additional details on the experimental setup, including the hardware used, the dataset and models used, as well as details on the generation process.

Hardware setup. Our experiments are executed on a compute server equipped with $2 \times$ Intel Xeon Gold 5317 CPU, $1{,}024$ GB main memory, and $2 \times$ A100 Nvidia Tesla GPU (80 GB, Ampere Architecture). In each experiment, a single Nvidia A100 GPU is used.

Datasets. For the results presented across all figures, we generated model responses to prompts obtained from the LMSYS-Chat-1M dataset [14]. We use the LMSYS-Chat-1M dataset exclusively to obtain a varied sample of potential user prompts. We filter user prompts to obtain the first 1200 questions that are in English, Spanish, Russian and Chinese languages (by using the `language` keyword) and whose length (in number of characters) is in the range $[20, 100]$, to avoid trivial or overly elaborated prompts. Non-English prompts are only used in the results of Figure 4 and Figure 12. We have repeated our experiments with a different set of 1200 randomly selected prompts from the LMSYS-Chat-1M dataset and have found indistinguishable results.

Models. In our experiments, we use the models `Llama-3.2-3B-Instruct` and `Llama-3.2-3B-Instruct` from the `Llama` family, the models `Gemma-3-1B-It` and `Gemma-3-4B-It` from the `Gemma` family, and `Ministral-8BInstruct-2410`. The models are obtained from publicly available repositories from `Hugging Face`[10].

Generation details. For all experiments involving the LMSYS dataset, we use the `transformers` library in `Python 3.11` to generate outputs of varying length between 200 and 300 tokens under various temperature and p values. We use the system prompt "`You are a helpful assistant. Be clear and concise.`", and compute standard deviations by running 5 independent generations with different seeds, and 90% symmetric confidence intervals for the mean values assuming a $t-$distribution value of 2.015. The 90% confidence intervals are shown in the plots.

For the experiments involving GPU energy measurements (Figure 11 and Table 1), we use the `pynvml` library in `Python 3.11` to record the power consumption of the GPU during the generation and verification of a token sequence. During the experiments, we isolated the GPU so that the unique (energy-intensive) process running was the model execution.

Licenses. The LMSYS-Chat-1M dataset is licensed under the LMSYS-Chat-1M Dataset License Agreement.[11] The `Llama-3.2-1B-Instruct` and `Llama-3.2-3B-Instruct` models are licensed under the LLAMA 3.2 COMMUNITY LICENSE AGREEMENT.[12]. The `Gemma-3-1B-It` and `Gemma-3-4B-It` models are licensed under the GEMMA TERMS OF USE.[13]. The `Ministral-8B-Instruct-2410` model is licensed under the MISTRAL AI RESEARCH LICENSE.[14].

[10] `https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct`
`https://huggingface.co/meta-llama/Llama-3.2-1B-Instruct`
`https://huggingface.co/google/gemma-3-1b-it`
`https://huggingface.co/google/gemma-3-4b-it`
`https://huggingface.co/mistralai/Ministral-8B-Instruct-2410`

[11] `https://huggingface.co/datasets/lmsys/lmsys-chat-1m`

[12] `https://ai.google.dev/gemma/terms`

[13] `https://www.gemma.com/gemma3_0/license/`

[14] `https://mistral.ai/static/licenses/MRL-0.1.md`

B Proofs

B.1 Proof of Theorem 3

We prove the theorem by reduction from the Hamiltonian path problem [58], which is known to be NP-complete, to the problem of finding a plausible tokenization under top-p sampling longer than a given number of tokens. Consequently, this will prove the hardness of the problem of finding a longest plausible token sequence $\tilde{\mathbf{t}}$ under top-p sampling, as stated in Eq. 3. In the Hamiltonian path problem, we are given a directed graph $\mathcal{G}$, that is, a set of nodes $\mathcal{N} = \{1, \ldots, n\}$ and a set of edges $\mathcal{E}$ between them, where $e = (\nu, \nu')$ denotes an edge from node ν to node ν'. The goal is to decide whether there exists a path that visits all nodes exactly once.

The core idea of the construction is to represent a path in the graph $\mathcal{G}$ as a sequence of tokens, where each node $j \in \mathcal{N}$ is represented by a token consisting of j times the character "a". In addition, we set the parameter $p \in (0, 1)$ of top-p sampling and the next-token distributions of the LLM such that a token sequence $\tilde{\mathbf{t}}$ with $\texttt{str}\left(\tilde{\mathbf{t}}\right) = \texttt{str}(\mathbf{t})$ and $\texttt{len}\left(\tilde{\mathbf{t}}\right) > 1$ is plausible if and only if the tokens in $\tilde{\mathbf{t}}$ correspond to a Hamiltonian path in the graph $\mathcal{G}$.

We proceed with the construction as follows. Let $\Sigma = \{\text{“a”}\}$ be the alphabet and the LLM's vocabulary be

$$\mathcal{V} = \{\text{“a”}, \text{“aa”}, \ldots, \underbrace{\text{“a...a”}}_{n \text{ times}}, \underbrace{\text{“a...a”}}_{\lambda \text{ times}}, \varnothing\},$$

where $\lambda = \sum_{j=1}^{n} j = n(n+1)/2$ and $\varnothing$ denotes the end-of-sequence token. Moreover, let the true output token sequence $\mathbf{t}$ consist of a single token—the one that contains λ times the character "a". Further, to keep the notation concise, we refer to the set of the first n tokens in $\mathcal{V}$ as $\mathcal{V}_n$. Then, we define a mapping $\Phi \colon \mathcal{V}_n \to \mathcal{N}$ from tokens to nodes as

$$\Phi(\underbrace{\text{“a...a”}}_{j \text{ times}}) = j \text{ for } j = 1, \ldots, n.$$

We fix the parameter p and a next-token distribution of the LLM such that, given a (partial) token sequence $\tilde{\mathbf{t}} = \left(\tilde{t}_1, \ldots, \tilde{t}_k\right)$, the restricted set of tokens $\mathcal{V}_p\left(\tilde{\mathbf{t}}\right)$ from which the LLM can sample the next token is given by

$$\mathcal{V}_p\left(\tilde{\mathbf{t}}\right) = \begin{cases} \{\varnothing\} & \text{if } \left|\text{str}\left(\tilde{\mathbf{t}}\right)\right| \geq \lambda \\ \mathcal{V} \setminus \varnothing & \text{if } \tilde{\mathbf{t}} = () \\ \left\{v \in \mathcal{V}_n : v \neq \tilde{t}_i \text{ for all } i \in [k] \text{ and } \left(\Phi\left(\tilde{t}_k\right), \Phi\left(v\right)\right) \in \mathcal{E}\right\} \cup \{\varnothing\} & \text{otherwise.} \end{cases} \tag{6}$$

In words, the last case states that the LLM can sample any token consisting of up to n times the character "a" as long as it is not already in the sequence $\tilde{\mathbf{t}}$, that is, the corresponding node has not been visited yet, and there is an edge in the graph $\mathcal{G}$ connecting that node to the node corresponding to the last token in $\tilde{\mathbf{t}}$. When the sequence $\tilde{\mathbf{t}}$ is empty (*i.e.*, the path has not started yet), the LLM can sample any token in $\mathcal{V}$ except for the end-of-sequence token $\varnothing$, which it is only allowed to sample when the sequence $\tilde{\mathbf{t}}$ contains at least λ characters.

We can now show that a Hamiltonian path in the graph $\mathcal{G}$ exists if and only if the solution $\tilde{\mathbf{t}}$ to the optimization problem given by Eq. 3 has $\texttt{len}\left(\tilde{\mathbf{t}}\right) > 1$.[15] Assume that the optimal solution to the problem is such that $\texttt{len}\left(\tilde{\mathbf{t}}\right) > 1$. Then, $\tilde{\mathbf{t}}$ cannot contain the token that consists of λ times the character "a" because this would imply that it consists of strictly more than λ characters and, therefore, $\texttt{str}(\tilde{\mathbf{t}}) \neq \texttt{str}(\mathbf{t})$. Additionally, $\tilde{\mathbf{t}}$ cannot contain any token twice, as that would violate its plausibility according to Eq. 6. Therefore, it has to hold that $\tilde{\mathbf{t}}$ contains all tokens in $\mathcal{V}_n$ exactly once, since this is the only way to form a sequence that contains $\lambda = \sum_{j=1}^{n} j$ characters. This implies that there exists a sequence of edges $\left(\Phi\left(\tilde{t}_1\right), \Phi\left(\tilde{t}_2\right)\right), \ldots, \left(\Phi\left(\tilde{t}_{n-1}\right), \Phi\left(\tilde{t}_n\right)\right)$ in the graph $\mathcal{G}$ that visits all nodes exactly once. Hence, a Hamiltonian path exists.

Now, assume that there exists a Hamiltonian path in the graph $\mathcal{G}$ that visits all nodes once, forming a sequence $(\nu_1, \nu_2, \ldots, \nu_n)$ with $\nu_i \in \mathcal{N}$ and $\nu_i \neq \nu_j$ for $i \neq j$. Then, the corresponding token

[15] For ease of exposition, we assume that the end-of-sequence token $\varnothing$ does not contribute to the length of the sequence $\tilde{\mathbf{t}}$.

sequence $\mathbf{t}' = (t'_1, t'_2, \ldots, t'_n)$ with $\Phi(t'_i) = \nu_i$ for $i \in [n]$ is a valid tokenization of the string $\texttt{str}(\mathbf{t})$ since $\sum_{i=1}^{n} |\texttt{str}(t'_i)| = \sum_{i=1}^{n} \nu_i = \lambda$. Moreover, the sequence $\mathbf{t}'$ is plausible by construction and satisfies $\texttt{len}(\mathbf{t}') = n > 1 = \texttt{len}(\tilde{\mathbf{t}})$. Finally, note that if $\mathcal{G}$ does not admit a Hamiltonian path, then $\texttt{str}(\mathbf{t})$ cannot be tokenized as a sequence of plausible tokens in $\mathcal{V}_n$. Hence, the only plausible tokenization is the token with λ characters, which has length 1. This concludes the proof.

In what follows, we present two extensions of the reduction to other settings where a provider may want to misreport the output token sequence without raising suspicion. Specifically, we consider the case where the provider reports a token sequence $\tilde{\mathbf{t}}$ that is plausible under top-k sampling and the case where the provider reports a token sequence $\tilde{\mathbf{t}}$ whose probability is greater than a given threshold.

B.1.1 Hardness of Finding the Longest Plausible Tokenization under Top-k Sampling

Top-k sampling is an approach of filtering out low-probability tokens during the sampling process, similar to top-p sampling. In top-k sampling, given a partial token sequence $\tilde{\mathbf{t}}$, the LLM samples the next token from the set of k most probable tokens $\mathcal{V}_k(\tilde{\mathbf{t}})$ at each step of the autoregressive process, where $k \in \{1, \ldots, |\mathcal{V}| - 1\}$. In this setting, the problem of finding the longest tokenization of a given output token sequence $\mathbf{t}$ that is plausible under top-k sampling is NP-Hard with the core idea of the reduction being similar to the one for top-p sampling.

The main difference lies in the fact that, in top-k sampling, the restricted set of tokens $\mathcal{V}_k(\tilde{\mathbf{t}})$ needs to have a fixed size k in contrast to the construction of $\mathcal{V}_p(\tilde{\mathbf{t}})$ in Eq. 6, which is a variable size set. To ensure that similar arguments for establishing a one-to-one correspondence between a Hamiltonian path in the graph $\mathcal{G}$ and a plausible token sequence $\tilde{\mathbf{t}}$ of length greater than 1 still hold, one can construct the set $\mathcal{V}_k(\tilde{\mathbf{t}})$ using a similar approach as in Eq. 6 but also including "padding" tokens that do not correspond to any node in the graph $\mathcal{G}$ to maintain a fixed size. To this end, we can maintain the same true output token sequence $\mathbf{t}$, consisting of $n(n+1)/2$ times "a" and augment the vocabulary $\mathcal{V}$ of the previous construction by adding n additional tokens

$$\mathcal{V}_b = \{\text{“b”}, \text{“bb”}, \ldots, \underbrace{\text{“b...b”}}_{n \text{ times}}\}$$

that are irrelevant for the string $s = \texttt{str}(\mathbf{t})$, do not correspond to any node in the graph $\mathcal{G}$, and do not affect the mapping Φ.

Then, note that, the set $\mathcal{V}_p(\tilde{\mathbf{t}})$ in Eq. 6 contains at most $n+1$ tokens. Here, the idea is to set $k = n+1$ and to construct the set $\mathcal{V}_k(\tilde{\mathbf{t}})$ as follows:

$$\mathcal{V}_k(\tilde{\mathbf{t}}) = \mathcal{V}_p(\tilde{\mathbf{t}}) \cup G\left(\mathcal{V}_p(\tilde{\mathbf{t}})\right), \tag{7}$$

where $G\left(\mathcal{V}_p(\tilde{\mathbf{t}})\right)$ is the set of the first $n + 1 - |\mathcal{V}_p(\tilde{\mathbf{t}})|$ tokens in $\mathcal{V}_b$. Since the additional tokens in $G\left(\mathcal{V}_p(\tilde{\mathbf{t}})\right)$ are not part of the mapping Φ and cannot be used to tokenize the string $s = \texttt{str}(\mathbf{t})$, they influence neither the plausibility of the optimal solution to the problem of Eq. 3 nor the corresponding Hamiltonian path in the graph $\mathcal{G}$. Therefore, the same arguments as in the proof of Theorem 3 hold, and we conclude that the problem of finding a longest tokenization of a given output token sequence $\mathbf{t}$ that is plausible under top-k sampling is NP-Hard.

B.1.2 Hardness of Finding the Longest Tokenization Whose Generation Probability Is Greater Than a Threshold

We now focus on a slightly different setting where the provider reports a token sequence $\tilde{\mathbf{t}}$ under the plausibility condition that the LLM does not assign a very low probability to the sequence as a whole. Formally, we require that the probability of the LLM generating the token sequence $\tilde{\mathbf{t}}$ satisfies

$$P(\tilde{\mathbf{t}}) \coloneqq P(\tilde{t}_1) \prod_{i=2}^{k} P(\tilde{t}_i \,|\, \tilde{\mathbf{t}}_{<i}) \geq \varepsilon, \tag{8}$$

where ε is a user-specified threshold and $P(\tilde{t}_i \,|\, \tilde{\mathbf{t}}_{<i})$ is the probability of the LLM generating the token $\tilde{t}_i$ given the previously generated tokens $\tilde{\mathbf{t}}_{<i} = (\tilde{t}_1, \ldots, \tilde{t}_{i-1})$. In this setting, the problem of finding a longest tokenization under Eq. 8 is also NP-hard. Similar as before, the proof is to set the next-token distributions of the LLM in a way that assigns low probability to token sequences that do

not lead to a Hamiltonian path in $\mathcal{G}$. Specifically, let δ be a constant such that $0 < \delta < 1/(n+1)$, and assume all next-token distributions are such that, given $\left(\tilde{t}_1, \ldots, \tilde{t}_k\right)$, assign probability mass $(1-\delta)/n$ to each of the tokens in

$$\mathcal{H}_i \coloneqq \left\{v \in \mathcal{V}_n : v \neq \tilde{t}_i \text{ for all } i \in [k] \text{ and } \left(\Phi\left(\tilde{t}_k\right), \Phi\left(v\right)\right) \in \mathcal{E}\right\}, \tag{9}$$

δ to each of the tokens in $\mathcal{V}_n \setminus \mathcal{H}_i$, 0 to the token with λ times the character "a", and any remaining probability mass to the end-of-sequence token $\varnothing$.[16] The high-level idea here is to set the probabilities of next tokens in such a way that the LLM assigns very low probability to the entire token sequence $\tilde{\mathbf{t}}$ if it concatenates two tokens whose corresponding nodes are not connected via an edge in the graph $\mathcal{G}$ or if the latter token has already been used in the sequence.

Given this construction, we set the user-specified threshold as $\varepsilon = \left(\frac{1-\delta}{n}\right)^n$. Now, given a Hamiltonian path in the graph $\mathcal{G}$ that visits all nodes once and forms a sequence $(\nu_1, \nu_2, \ldots, \nu_n)$ with $\nu_i \in \mathcal{N}$ and $\nu_i \neq \nu_j$ for $i \neq j$, the corresponding token sequence $\mathbf{t}' = (t'_1, t'_2, \ldots, t'_n)$ has cumulative probability exactly ε, so it is plausible and has length greater than 1. Reciprocally, given a plausible tokenization $\tilde{\mathbf{t}}$ with length greater than 1, the corresponding sequence $\left(\Phi\left(\tilde{t}_1\right), \Phi\left(\tilde{t}_2\right)\right), \ldots, \left(\Phi\left(\tilde{t}_{n-1}\right), \Phi\left(\tilde{t}_n\right)\right)$ has to be a Hamiltonian path. If this is not true, at least one of the tokens in $\tilde{\mathbf{t}}$ does not belong in its respective set $\mathcal{H}_i$ defined by Eq. 9, and hence the probability of the sequence $\tilde{\mathbf{t}}$ is at most

$$P\left(\tilde{\mathbf{t}}\right) \leq \delta \left(\frac{1-\delta}{n}\right)^{n-1} < \varepsilon, \tag{10}$$

which contradicts the assumption that $\tilde{\mathbf{t}}$ is plausible.

B.2 Proof of Proposition 5

We will prove the proposition by contradiction. Assume that, under an incentive-compatible pricing mechanism r, there exist token sequences $\tilde{\mathbf{t}}, \tilde{\mathbf{t}}' \in \mathcal{V}^*$ such that $\texttt{str}(\tilde{\mathbf{t}}) = \texttt{str}(\tilde{\mathbf{t}}')$ and $r(\tilde{\mathbf{t}}) \neq r(\tilde{\mathbf{t}}')$ where, without loss of generality, it holds that $r(\tilde{\mathbf{t}}') > r(\tilde{\mathbf{t}})$. Now, consider that the output token sequence generated by the LLM is $\tilde{\mathbf{t}}$ and the provider runs a policy π that reports $\tilde{\mathbf{t}}' \sim \pi(\tilde{\mathbf{t}})$. Moreover, let the policy π be such that the energy costs of GPU computations it requires are zero, *i.e.*, $c_\pi(\tilde{\mathbf{t}}) = 0$.[17] Then, it follows that

$$r(\tilde{\mathbf{t}}') > r(\tilde{\mathbf{t}}) \overset{(*)}{\implies} r(\tilde{\mathbf{t}}') - c_{\text{gen}}(\tilde{\mathbf{t}}) - c_\pi(\tilde{\mathbf{t}}) > r(\tilde{\mathbf{t}}) - c_{\text{gen}}(\tilde{\mathbf{t}}) - c_{\pi_0}(\tilde{\mathbf{t}}) \implies U_\pi(\tilde{\mathbf{t}}', \tilde{\mathbf{t}}) > U_{\pi_0}(\tilde{\mathbf{t}}, \tilde{\mathbf{t}}),$$

where $(*)$ holds because the truthful policy π_0 also requires no GPU computations, *i.e.*, $c_{\pi_0}(\tilde{\mathbf{t}}) = 0$. The latter inequality contradicts the fact that the pricing mechanism r is incentive-compatible and concludes the proof.

B.3 Proof of Theorem 6

Let $\mathbf{t}' = (t'_1, \ldots, t'_k)$ be the tokenization of the string $s = \texttt{str}(\mathbf{t})$ that consists only of single-character tokens, *i.e.*, $\texttt{str}\left(\mathbf{t}\right) = \texttt{str}\left(\mathbf{t}'\right)$ with $|\texttt{str}\left(\mathbf{t}'\right)| = |\texttt{str}\left(\mathbf{t}\right)| = k$. Note that such a tokenization exists, since $\Sigma \subseteq \mathcal{V}$. From Proposition 5, we get

$$\begin{aligned} r\left(\mathbf{t}\right) = r\left(\mathbf{t}'\right) &\overset{(*)}{=} \sum_{i=1}^{k} r\left(t'_i\right) = \sum_{i=1}^{k} \sum_{\sigma \in \Sigma} \mathbb{1}[t'_i = \sigma] \cdot r(\sigma) \\ &= \sum_{\sigma \in \Sigma} \texttt{count}_\sigma\left(\mathbf{t}'\right) \cdot r(\sigma) \overset{(**)}{=} \sum_{\sigma \in \Sigma} \texttt{count}_\sigma\left(\mathbf{t}\right) \cdot r(\sigma), \end{aligned}$$

where $\mathbb{1}$ denotes the indicator function, $(*)$ holds because the pricing mechanism is additive and $(**)$ holds because $\texttt{str}\left(\mathbf{t}'\right) = \texttt{str}\left(\mathbf{t}\right)$.

[16] Using the assumption that $\delta < 1/(n+1)$, it is easy to verify that the above construction leads to a valid probability distribution.

[17] Note that, such policies exist, as exemplified by Algorithm 1.

C Additional Experimental Results

C.1 Performance of Algorithm 2 under Different LLMs and Temperature Values

In this section, we evaluate Algorithm 2 across different LLMs and temperatures.

Figure 5 shows the fraction of generated outputs for which Algorithm 2 finds a longer plausible tokenization. We observe that the higher the values of p and temperature, the higher the likelihood that Algorithm 2 finds plausible longer tokenizations.

Figure 6 shows the percentage of tokens overcharged by an unfaithful provider who uses Algorithm 2. We observe that the percentage of overcharged tokens is unimodal with respect to the number of iterations m, and the higher the value of p and the temperature, the higher the percentage of overcharged tokens, as the top-p sets become larger and the likelihood that a longer tokenization is plausible increases.

Figure 7 shows the relative increase in utility that a provider using Algorithm 2 can achieve under pay-per-token compared to the faithful reporting policy π_0, as a function of their margin ρ for output tokens. We observe that providers with low margins can benefit the most from misreporting, since their utility when faithfully reporting is reduced.

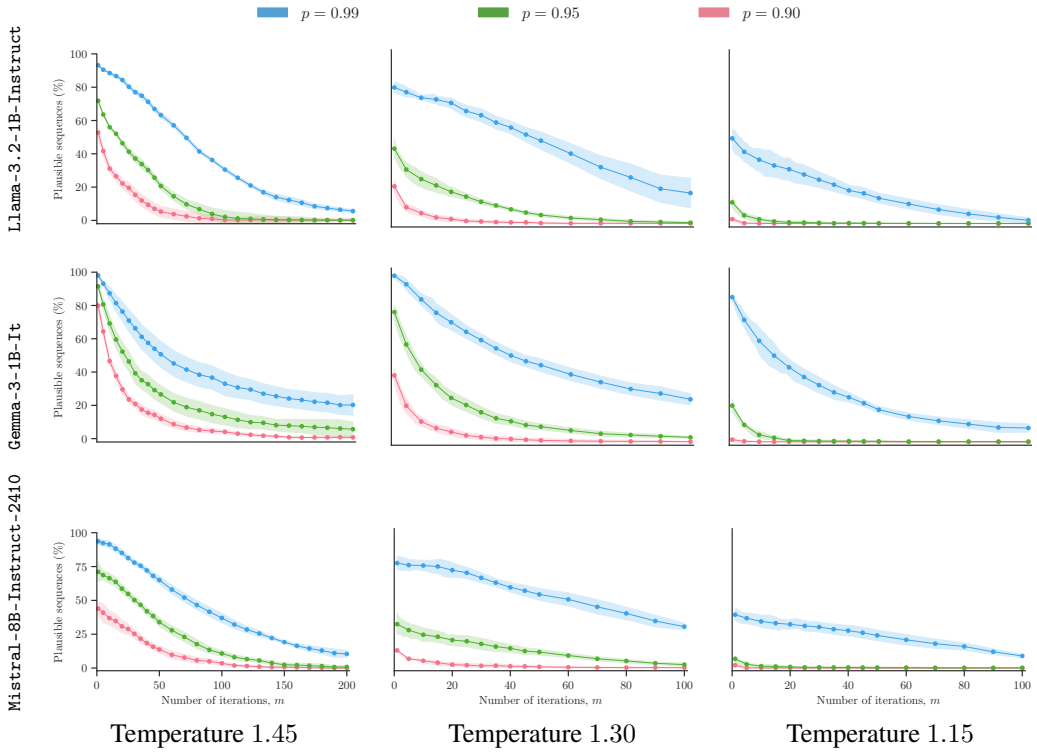


Figure 5: **Fraction of outputs for which Algorithm 2 finds a longer plausible tokenization.** The figure shows the fraction of outputs generated by different LLMs to 600 prompts from the LMSYS Chatbot Arena platform for which Algorithm 2 finds a longer plausible tokenization under top$-p$ sampling, for values of m, p and temperature. We repeat each experiment 5 times to calculate 90% confidence intervals.

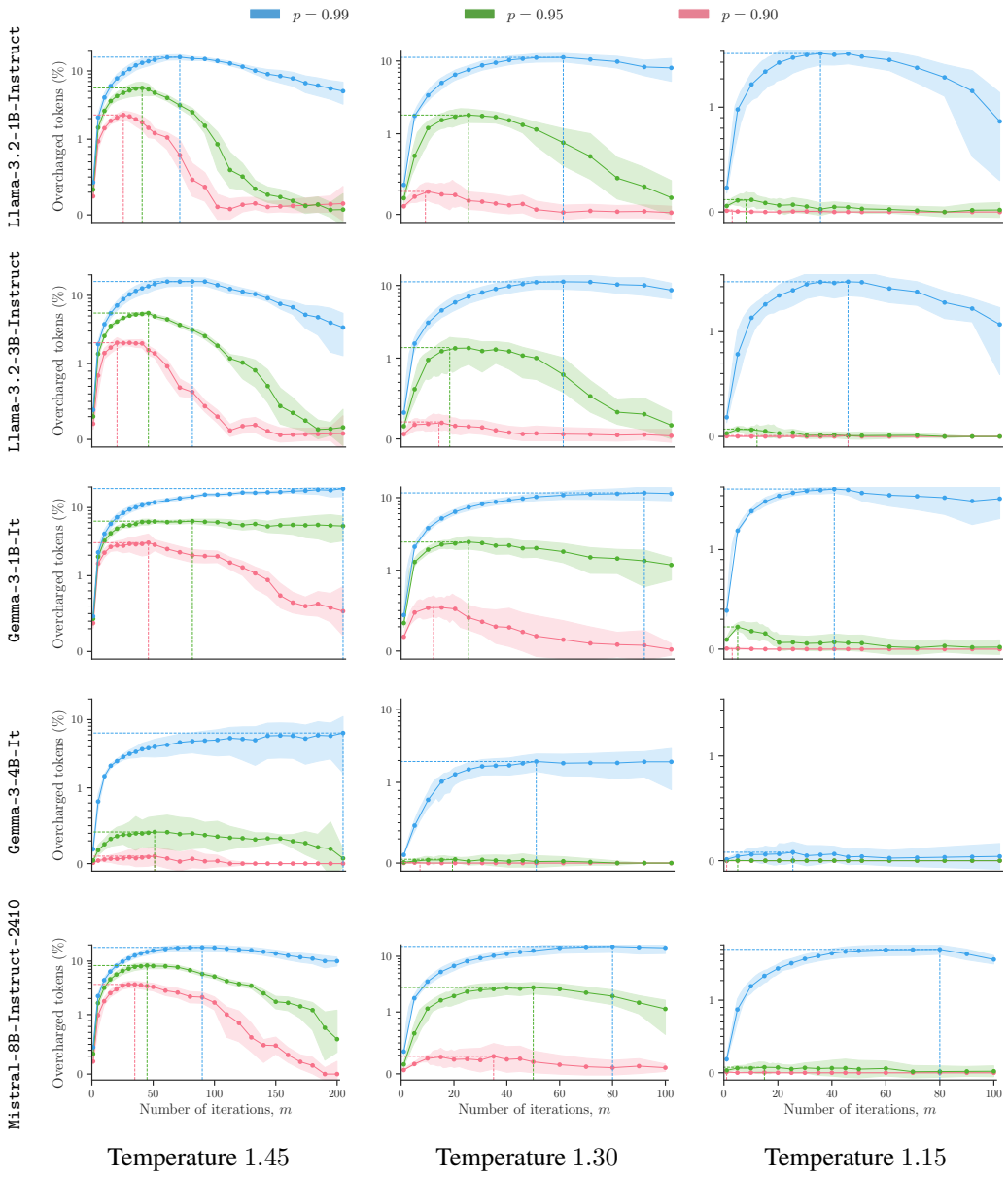


Figure 6: **Additional revenue from misreporting the tokenization of outputs using Algorithm 2.** The panels show the percentage of tokens overcharged by an unfaithful provider who misreports the tokenization of the outputs generated by several LLMs to 600 prompts from the LMSYS Chatbot Arena platform using Algorithm 2, for different values of m, p and temperature. The dashed lines highlight the optimal value of m. We repeat each experiment 5 times to obtain 90% confidence intervals.

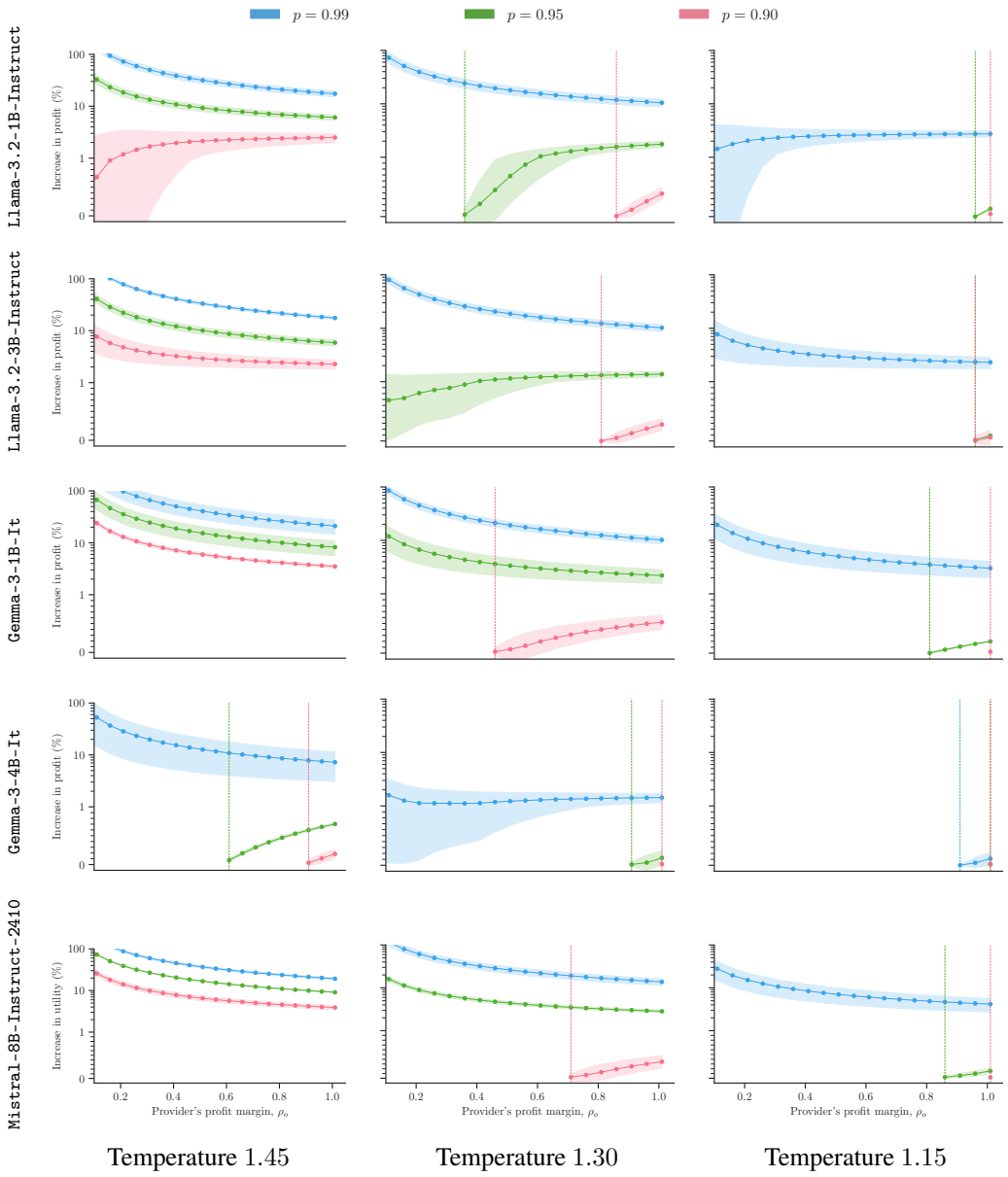


Figure 7: **Financial gain from misreporting the tokenization of outputs using Algorithm 2.** The panels show the financial gain that an unfaithful provider who misreports the tokenization of the outputs generated by several LLMs to 600 prompts from the LMSYS Chatbot Arena platform using Algorithm 2 can achieve, as a function of the profit margin on the output tokens ρ_o and p. The dashed vertical lines represent the minimum margin under which misreporting is financially viable according to Eq. 4. Here, for each value of p, we select the optimal number of iterations m in Algorithm 2 shown on Figure 2, and repeat each experiment 5 times to obtain 90% confidence intervals.

C.2 Examples of Plausible Output Token Sequences Found by Algorithm 2

To illustrate how Algorithm 2 works, here, we provide examples of output token sequences generated by the `Llama-3.2-1B-Instruct` model, where the algorithm has found plausible tokenizations that are longer than the original output token sequence. Across all examples, we use the `Llama-3.2-1B-Instruct` model and set $p = 0.95$ and the temperature of the model to 1.3. We select prompts from the LMSYS dataset. For each example, we show (i) the true output token sequence generated by the model, and (ii) the modified output token sequence returned by Algorithm 2. We use "|" to indicate separations between tokens as generated by the model, and we use "|" to indicate the split points of the tokens that result from Algorithm 2. The number above each red separator indicates the iteration of the algorithm in which the respective token was split. We show all iterations until the sequence first becomes non-plausible.

```
... The| third| film| appears| to| delve| into| the| themes| of|
societal| reaction| and|... Here| are| movies| that| offer| similar|
thematic concerns|...
```

(a) True output token sequence

```
                                        (1)
... The| third| film| appears| to| del | ve| into| the| themes| of|
     (2)
soci | etal| reaction| and|... Here| are| movies| that| offer| similar|
                    (3)
thematic conce | rns|...
```

(b) Modified output token sequence

Figure 8: Responses to the prompt "`is Dead Snow worth watching or should I watch directly Dead Snow 2?`".

```
... Here| are| a| few| options| :|
1|.| **| T|rello|**:| T|rello| is| a| visual| project| management|
tool|...
2|.| **| J|IRA|**:| As| mentioned|,| J|IRA| is| a| popular| At|lassian|
suite|...
```

(a) True output token sequence

```
... Here| are| a| few| options| :|
                        (1)
1|.| **| T|rello|** | :| T|rello| is| a| visual| project| management|
tool|...
                        (2)
2|.| **| J|IRA|** | :| As| mentioned|,| J|IRA| is| a| popular|
       (3)
At|las | sian| suite|...
```

(b) Modified output token sequence

Figure 9: Responses to the prompt "`What is a good tool to plan a complex server deployment?`".

The| easiest| way| to| invest| in| property|... Real| estate|
investment| trusts| or| RE|IT|s|,| real| estate| mutual| funds| may|
be| the| easiest|.|... There| are| many| options| for| acquiring|
income| such| as| ground| level| rental| or| owning| a| building|
through| a| partnership|.| The| highest| performing| investment| may|
remain| a| gamble| and| have| no| guarantee|.| The| next| hightest|
would| have| to| be| investing| in| stocks| and| bonds|, the| old|
main|stay|.| Div|idend| and| bonds| have| higher| reliability|...
Note|:| the| previous| responses| and| answers| have| been|
simplified|...

(a) True output token sequence

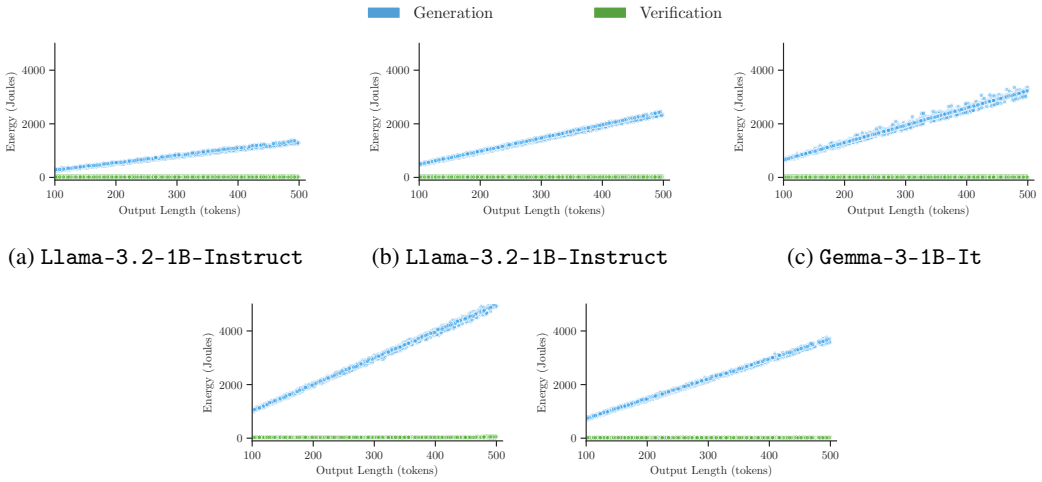
The| eas (8)| iest| way| to| invest| in| property|... Real| estate|
investment| trust (2)| s| or| RE|IT|s|,| real| estate| mutual| funds| may|
be| the| easiest|.|... There| are| many| options| for| acqu (6)| iring|
income| such| as| ground| level| rental| or| ow (7)| ning| a| building|
through| a| partnership|.| The| highest| performing| investment|
may| remain| a| gam (3)| ble| and| have| no| guarantee|.| The| next|
hightest| would| have| to| be| investing| in| stocks| and| bonds|,
the| old| main|st (4)| ay|.| Div|id (1)| end| and| bonds| have| higher|
reli (2)| ability|... Note|:| the| previous| responses| and| answers|
have| been| simpl (5)| ified|...

(b) Modified output token sequence

Figure 10: Responses to the prompt "What is currently the easiest investment opportunity with the capital and the highest game?".

C.3 Energy cost of generation and verification of plausibility

In this section, we provide empirical measurements of the (GPU) energy consumption required to (i) generate an output sequence of tokens and (ii) compute the next-token probabilities for a given sequence of tokens. Here, note that (ii) is required to verify if such a sequence is plausible under top-p sampling, as discussed in Section 3.2 and Section 3.3.



(a) `Llama-3.2-1B-Instruct` (b) `Llama-3.2-1B-Instruct` (c) `Gemma-3-1B-It`

(d) `Gemma-3-4B-It` (e) `Mistral-8B-Instruct-2410`

Figure 11: **Energy cost of autoregressive generation vs. verification.** The figure shows, across different model families and for the first 1000 LMSYS prompts, the energy cost of autoregressively generating an output and the energy cost of verifying the same generated output (*i.e.*, computing the next-token probabilities). We measure energy cost as the power consumed by the GPU, a single Nvidia H100, integrated over the execution time, as calculated by Python's `pynvml` library. All executions are with temperature 1, no top-p sampling, and with KV caching enabled.

Table 1: **Energy cost of autoregressive generation vs. verification.** The table shows, for different model families, the per-token energy cost c_{o} for autoregressively generation, the per-output energy cost of verifying the same generated output c_{v} (which, as shown in Figure 11, is approximately independent of sequence length), and the ratio between these two quantities. All values are computed over 400 prompts from the LMSYS Chatbot Arena platform. All executions are with temperature 1, no top-p sampling, and with KV caching enabled on a single Nvidia H100. We repeat each experiment 5 times to obtain 90% confidence intervals.

LLM	Per-token gen. cost c_{o} (J)	Per-output verif. cost c_{v} (J)	$c_{\mathsf{o}}/c_{\mathsf{v}}$
`Llama-3.2-1B-Instruct`	2.642 ± 0.004	15.188 ± 0.005	0.174 ± 0.009
`Llama-3.2-3B-Instruct`	4.805 ± 0.004	15.901 ± 0.007	0.302 ± 0.009
`Gemma-3-1B-In`	6.481 ± 0.006	12.730 ± 0.002	0.509 ± 0.016
`Gemma-3-4B-In`	9.993 ± 0.005	33.5 ± 0.1	0.298 ± 0.005
`Ministral-8B-Instruct-2410`	7.349 ± 0.009	17.743 ± 0.008	0.413 ± 0.016

C.4 Provider's profit margin under pay-per-character for different languages

In Figure 12 we show, stratified by language, the distribution of a provider's margin across responses to prompts from the LMSYS Chatbot Arena platform after transitioning to a pay-per-character mechanism with character price $r_c = r_o \cdot \texttt{tpc}$, where $\texttt{tpc}$ is the (empirical) average token-to-character ratio computed over multilingual prompts.

We find that languages such as Russian or Chinese, which are likely underrepresented during tokenizer training in the models we study, result in lower (average) provider margins compared to their previous pay-per-token margin ρ_o, while better-represented languages like English or Spanish yield a slight margin increase. This occurs because, under pay-per-character pricing, the provider's margin for a given output decreases as the token-to-character ratio increases, and languages that differ significantly from English tend to produce outputs with higher token-to-character ratios [18, 59]. As a consequence, by reducing their output prices, pay-per-character partially alleviates the financial burden faced by users of minority languages [60].

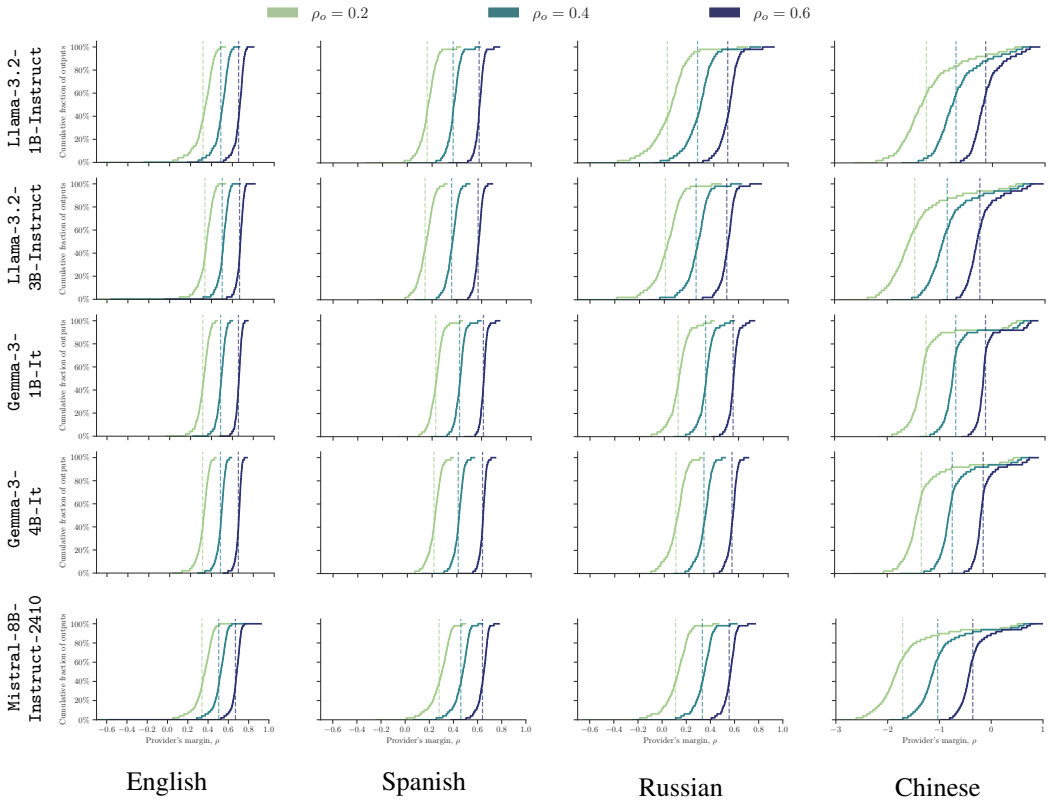


Figure 12: **Provider's profit margin under a pay-per-character pricing mechanism.** The panels show, across different languages, models and values ρ_o of the provider's profit margin under pay-per-token, the (empirical) cumulative distribution function of their profit margin $\rho(\mathbf{t}) = 1 - c_{\text{gen}}(\mathbf{t})/r(\mathbf{t})$ per output $\mathbf{t}$ after their transition to pay-per-character. Here, we first compute the average ratio of number of tokens to number of characters ($\texttt{tpc}$) across the responses of each model to 600 multilingual prompts sampled from the LMSYS Chatbot Arena dataset. Then, we set $r_c = r_o \cdot \texttt{tpc}$ and compute the provider's profit margin $\rho(\mathbf{t})$ for outputs to 600 different monolingual prompts. In all panels, dashed vertical lines show empirical averages of the respective distributions, and we set the temperature of the models to 1.3 and use top-p sampling with $p = 0.95$.