Where is the signal in tokenization space?

Renato Lui Geh, Honghua Zhang,
Kareem Ahmed, Benjie Wang, Guy Van den Broeck
University of California, Los Angeles
{renatolg,hzhang19,ahmedk,benjiewang,guyvdb}@cs.ucla.edu

Abstract

Large Language Models (LLMs) are typically shipped with tokenizers that *deterministically* encode text into so-called *canonical* token sequences, to which the LLMs assign probability values. One common assumption is that the probability of a piece of text is the probability of its canonical token sequence. However, the tokenization of a string is not unique: e.g., the `Llama2` tokenizer encodes `Tokens` as `[Tok,ens]`, but `[Tok,en,s]` also represents the same text. In this paper, we study non-canonical tokenizations. We prove that, given a string, it is computationally hard to find the most likely tokenization for an autoregressive LLM, as well as to compute the marginal probability over all possible tokenizations. We then show how the marginal is, in most cases, indistinguishable from the canonical probability. Surprisingly, we then empirically demonstrate the existence of a significant amount of signal hidden within tokenization space. Notably, by simply aggregating the probabilities of non-canonical tokenizations, we achieve improvements across a range of LLM evaluation benchmarks for a variety of architectures, including transformers and state space models.

1 Introduction

Autoregressive large language models (LLMs) generate text by predicting the next word sequentially. A crucial yet often overlooked step in this process is *tokenization*, whereby each word is broken down into subwords. It allows the model to generate text beyond what it was trained on, enabling open-vocabulary generation. However, this approach also introduces a significant challenge: a given string can be tokenized in exponentially many ways (Figure 1). For example, this paper's abstract can be tokenized in more than 10^{267} ways under the `Llama2` vocabulary (Touvron et al., 2023).

While a given string can be tokenized in multiple ways, at inference time, almost all successful modern LLMs utilize a fixed, or *canonical*, tokenization: a deterministic, rule-based mapping from text to token sequences (Gage, 1994). Consequently, it has become commonplace to use this token sequence as a proxy for the underlying text. In particular, the probability of the token sequence is often used in place of the probability of the text (e.g. for evaluation metrics like perplexity), even though these quantities are not necessarily equal. To complicate matters, some language models are pretrained with *stochastic tokenizations* (Kudo, 2018; Provilkov et al., 2020), exposing them to multiple ways of tokenizing the same string, with the hope of obtaining models with a more developed understanding of the compositionality of words.

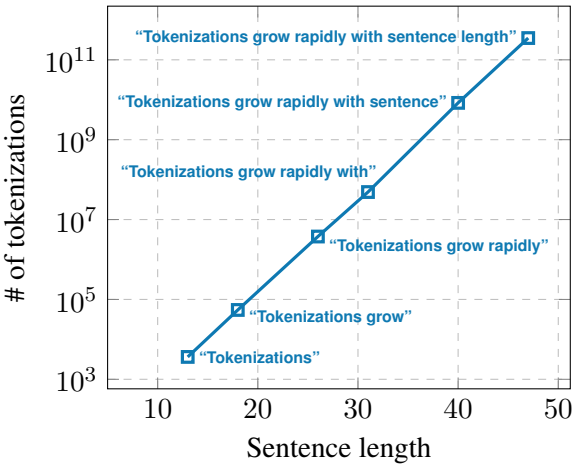


Figure 1: **Exponential growth of the number of tokenizations.** The (log-scale) y-axis shows the number of tokenizations as a function of the sentence length.

In this paper, in the context of modern LLMs, we ask whether non-canonical tokenizations of a string can provide additional signal *at inference time*, which would be lost by considering just the canonical tokenization. To this end, we investigate two natural alternatives: finding the *most likely tokenization* and *marginalizing over tokeniza-*

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 3966–3979
November 12-16, 2024

tions. For example, one natural way to answer a multiple-choice question is to choose the answer with the highest probability conditioned on the question (Zellers et al., 2019); instead of always assuming the canonical tokenizations of the answers, one could compare answers based on the probability of their most likely tokenizations, or alternatively the marginal probability of all possible tokenizations.

We first study the problem of finding the most likely tokenization and show that it is NP-hard under some mild assumptions. As such, we propose an anytime branch-and-bound algorithm to approximate the most likely tokenization. We find that, for text lengths where the branch-and-bound strategy is practical, the canonical tokenization is usually the most likely one.

Then we ask the question of whether there is a significant amount of probability mass concentrated on tokenizations *other* than the canonical. We first observe that as we sample token sequences of varying length from the LLM distribution *unconditionally*, the proportion of canonical tokenizations decreases significantly as the sequence length increases. To further investigate this phenomenon, ideally one would need to compute the marginal probability of all tokenizations for a given string, which we show to be #P-hard. Hence, we implement an importance sampling estimator for the marginal probability. Surprisingly, despite the extremely large number of non-canonical tokenizations, we empirically find that the estimated marginal probability is usually very close to the canonical tokenization's probability.

This raises our last question: does the complete tokenization space add any meaningful signal at all, in addition to the canonical tokenization alone? Remarkably, we show that, even for the cases where there is little probability mass on non-canonical tokenizations, they seem to carry *some* meaningful signal. Specifically, we show that for `Gemma-2B` (Gemma Team et al., 2024), `Llama2-7B` (Touvron et al., 2023) and `Mamba-130M` (Gu and Dao, 2024), by employing ensemble strategies for weighting different tokenizations at inference time, we achieve significant performance improvements on challenging LLM evaluation benchmarks.

Contributions. In summary, we show that: (i) while it is tempting to consider computing the marginal probability of a string, this quantity is #P-hard to compute; (ii) in fact, even computing the probability of the most likely tokenization is NP-hard; and (iii) while in most cases the marginal probability of a string is practically the same as the canonical probability, non-canonical tokenizations seem to provide some signal to downstream tasks, to the point that we achieve consistent improvement across a range of open source models on Q&A datasets.

2 Related Work

Many previous works have explored tokenization strategies within the LLM pipeline, and the (often undesirable) inductive biases they may introduce: for example, in introducing unfairness between languages (Petrov et al., 2023), gender bias (Ovalle et al., 2024), and in performing arithmetic (Singh and Strouse, 2024). Some recent works have avoided the many downsides of tokenization by employing byte-level models, but either suffer from slow decoding due to longer sequences (Yu et al., 2023), or rely on token-level models for more efficient generation (Wang et al., 2024). To overcome the limitations of tokenization, prior works have examined (approximately) marginalizing over the distribution of possible token sequences (Buckman and Neubig, 2018; Cao and Rimell, 2021; Chirkova et al., 2023). In this work, we analyze modern LLMs and consider multiple strategies for extracting information from tokenization space; finding that, contrary to prior belief, the signal is present not in the most-likely tokenization or (approximated) marginals, but rather in a mixture of canonical and non-canonical tokenizations.

3 An LLM Induces a Distribution over Tokenizations

Let $\boldsymbol{x} = (x_1, x_2, \ldots)$ denote a string (a sequence of characters). A vocabulary $\mathcal{V}$ is a set of strings that represent subwords, or *tokens*. A *token sequence* w.r.t. a vocabulary $\mathcal{V}$ is a sequence $\boldsymbol{v} = (v_1, v_2, \ldots)$ where each $v_i \in \mathcal{V}$. A *tokenization* of string $\boldsymbol{x}$ w.r.t. a vocabulary $\mathcal{V}$ is a token sequence w.r.t. $\mathcal{V}$ such that the concatenation of all tokens is equal to $\boldsymbol{x}$. Simply put, a tokenization breaks down a string into substrings, each recognized by the vocabulary. The substrings are ordered by their position in the original string. We write $\boldsymbol{v} \models_{\mathcal{V}} \boldsymbol{x}$ to denote that token sequence $\boldsymbol{v}$ is a tokenization of string $\boldsymbol{x}$ w.r.t. the vocabulary $\mathcal{V}$, sometimes omitting $\mathcal{V}$ when meaning is clear.

An autoregressive LLM p defines a conditional

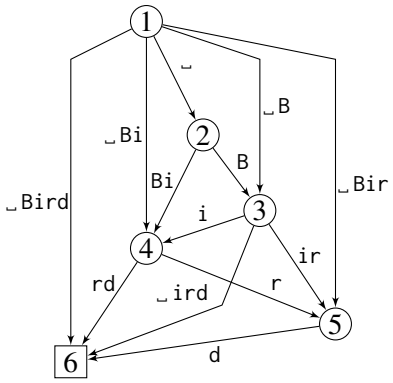


Figure 2: **Multi-valued decision diagram for the tokenization of `Bird`**. The square is a terminal node.

probability distribution $p(v_i|v_1, \ldots v_{i-1})$ over tokens from its vocabulary $\mathcal{V}$. Thus, an LLM *induces a distribution over tokenizations of a given string*.

Definition 3.1 (Induced Tokenization Distribution)**.** Let $\boldsymbol{x}$ be a string, $\boldsymbol{v}$ a token sequence, and p an LLM over vocabulary $\mathcal{V}$. Then, the tokenization distribution induced by p is

$$p(\boldsymbol{v}, \boldsymbol{x}) = \begin{cases} \prod_{i=1}^{|\boldsymbol{v}|} p\left(v_i|v_1, \ldots, v_{i-1}\right) & \text{if } \boldsymbol{v} \models_{\mathcal{V}} \boldsymbol{x}, \\ 0 \quad \text{otherwise.} \end{cases}$$

Most modern LLMs make use of tokenizers based on Byte-Pair Encoding (BPE) (Gage, 1994), whereby the token vocabulary is initialized with the character vocabulary, and a *merge table* is initialized to be empty. The method then iteratively counts all pairs of tokens and merges the most frequent pair into a new token. This new token is added to the vocabulary and the merge rule is added to the merge table. This process is repeated until the desired vocabulary size is reached. The resulting merge table specifies which tokens are to be merged into larger tokens, as well as the priority of these merges. In this way, it defines a *canonical tokenization* procedure as follows: first, a string is split into its constituent characters, then, the pair of adjacent tokens with the highest priority merge rule is combined. This is repeated until no further merge rules from the table are applicable.

BPE dropout (Provilkov et al., 2020) introduces an additional step during training: when tokenizing a word, merge rules are dropped with some probability. After this dropout phase, merges proceed the same way as BPE. This dropout phase acts as a regularization method that provides robustness to input noise. It also means that language models trained with BPE dropout should assign more mass to non-canonical tokenizations.

Algorithm 1 COMPILE

Input String $\boldsymbol{x}$, vocabulary $\mathcal{V}$, last token v
Output MDD of all tokenizations of $\boldsymbol{x}$

```
Initialize memoization M : ℕ × V → MDD
if x is empty then return nothing
Initialize node N
for each token t ∈ V do
    if x starts with t then
        if (|x|, t) ∈ M then C ← M(|x|, t)
        else C ← COMPILE(x_{|t|+1:|x|}, V, t)
        Add edge from N to C labelled with t
        Memoize M(|x|, t) ← C
return N
```

At inference time, for a string $\boldsymbol{x}$, the tokenizer outputs the canonical tokenization $\boldsymbol{v}^*$ (without any dropout), which is then evaluated by the LLM to get the *canonical probability* $p(\boldsymbol{v}^*, \boldsymbol{x})$. Note that this probability is one of an exponential number of tokenization probabilities for a particular string. In fact, one can compile a Multi-valued Decision Diagram (MDD) (Lee, 1959) that represents this combinatorial space for a given string tractably by decomposing and reusing subsequences. This data structure allows one to compute the total number of tokenizations in linear time in the number of edges of the diagram. Algorithm 1 shows how to compile an MDD from a string and Figure 2 shows an example of an MDD compiled from the string `Bird`. Each node in the diagram corresponds to a position in the string, and edges from node i to node j are labelled with the corresponding token $\boldsymbol{x}_{i:j} = (x_i, x_{i+1}, \ldots, x_j)$. Every path going from the root to a terminal node is a tokenization of $\boldsymbol{x}$.

Given what we know so far, a question naturally arises: since we can tractably represent all tokenizations as an MDD, and given that the number of possibilities is exponential, can we efficiently compute the most likely tokenization of a string? And perhaps more interestingly, is it the canonical one, as often is assumed in practice?

4 Computing the Most Likely Tokenization is Hard ...

We begin by noting that there exist simple distributions where finding the most likely tokenization can be done efficiently. For example, if we annotate the edges in an MDD with probabilities, that gives us a tokenization distribution where the most likely tokenization is simply the MDD path with the highest probability. By carefully modifying

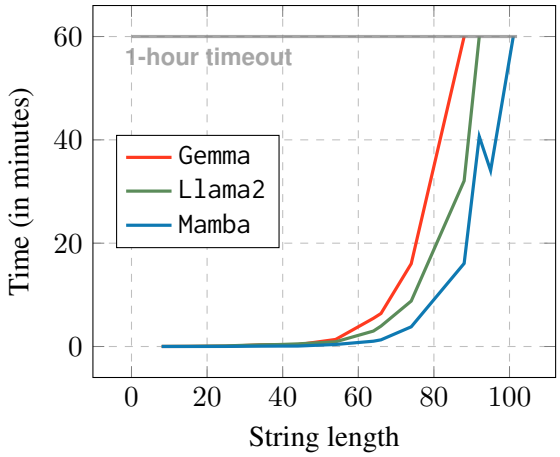


Figure 3: **Run-time for branch-and-bound over tokenizations.** The search grows exponentially with the number of characters in the string.

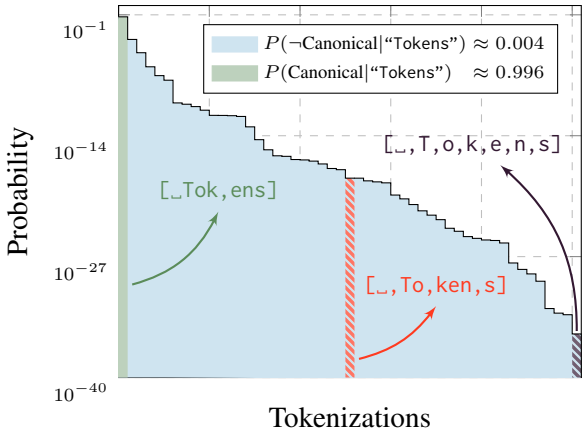


Figure 4: **Distribution of tokenizations for the word `Tokens`.** An overwhelming probability mass is on the canonical tokenization, with (an exponential number of) others sharing a miniscule portion of probability.

the MDD, it even becomes possible to efficiently compute the most likely tokenization induced by a bi-gram distribution, where each token depends only on the previous one (Dupont and Rosenfeld, 1997; Choi et al., 2020).

For more complex autoregressive language models, however, we unfortunately show that computing the most likely tokenization is computationally hard. We formalize this as follows.

Problem 4.1 (Most-Likely Tokenization)**.** Let $\boldsymbol{v}$ denote a token sequence. Given a string $\boldsymbol{x}$, an autoregressive LLM p, and a threshold $\epsilon > 0$, the most likely tokenization problem is deciding whether

$$\max_{\boldsymbol{v}} p(\boldsymbol{v}, \boldsymbol{x}) > \epsilon.$$

Theorem 4.2. *The most-likely tokenization problem is NP-complete.*

Proof. (Sketch) The proof is by reduction from the 3-SAT Boolean satisfiability problem (Karp, 2010). We encode the Boolean variables as possible tokenizations of substrings such that there is a correspondence between the probability of the most likely tokenization and the existence of a satisfiying assignment. The full proof is in Appendix B. □

Given the LLM training regime, a reasonable assumption in practice is that the canonical tokenization is (close to) the most likely tokenization. To empirically verify this claim, we devise a branch-and-bound algorithm to search through the MDD in order to find some tokenization whose probability is higher than the canonical. We do so by setting the initial lower bound as the canonical probability, and then pruning paths whose partial probability is below this bound. We set a time budget of 1-hour, after which the search returns the best tokenization at that point. As expected, we find that branch-and-bound is quickly overwhelmed by the number of tokenizations as the string length grows. Finding the most likely tokenization this way rapidly becomes intractable for longer strings.

Figure 3 shows the branch-and-bound search time across three LLM architectures for the string “Language models typically tokenize text into subwords, utilizing a learned deterministic set of merge rules to aggregate tokens.” We gradually insert new words and re-run search to visualize its scalability. Branch-and-bound always returns the canonical tokenization as the best candidate, despite the exponential number of possible candidates.

Not only does the canonical tokenization seem to be the most likely one for shorter text, but it also often is overwhelmingly so. Figure 4 shows the tokenization distribution for the word `Tokens` under the `Llama2` model; there are 52 tokenizations, with the canonical taking most of the mass.

Even though canonical seems to take up a majority of the probability mass in these cases, if we look at generated text from these LLMs, a sizable percentage of the (unconditionally) generated tokenizations are non-canonical. Figure 5 shows canonicity as a function of the number of tokens generated by the language model. As generated text grows larger, the probability of generating non-canonical tokenizations also grows. This is surprising, as it is seemingly in contra-

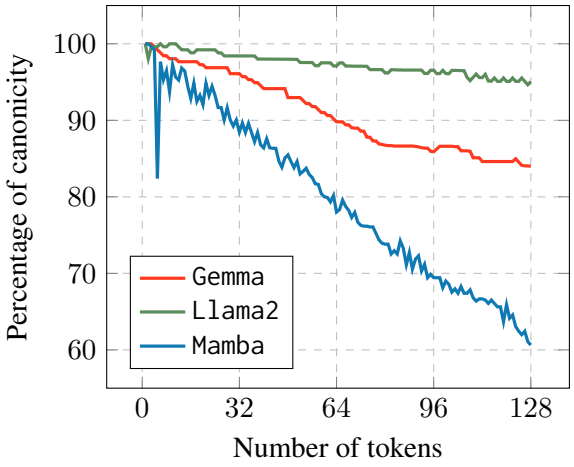


Figure 5: **Canonicity in generated text.** Percentage of canonicity drops as more tokens are generated.

diction to earlier evidence. It turns out that, if we investigate these non-canonical generated sequences in more depth, we find that a large majority of such cases are non-English, with a large portion consisting of code and languages that utilize unicode characters. It is nevertheless interesting that some non-canonical tokenizations are indeed more likely than their canonical counterparts, with some even in grammatically correct English. For example, the string $\boldsymbol{x} = \texttt{␣tongueless}$ (with ␣ denoting a whitespace) is canonically tokenized as $\boldsymbol{v}^* = \texttt{[␣tong,uel,ess]}$ by `Gemma`, with $p(\boldsymbol{v}^*|\boldsymbol{x}) \approx 0.474$; however $\boldsymbol{v} = \texttt{[␣tongue,less]}$ is a more likely tokenization according to the LLM, with $p(\boldsymbol{v}|\boldsymbol{x}) \approx 0.518$. This gap between the probability of the most likely tokenization and canonical tokenization can be even more extreme. For instance, $\boldsymbol{v} = \texttt{[Hyp,no,patu,rist]}$ is a much more likely tokenization $p(\boldsymbol{v}|\boldsymbol{x}) = 0.9948$ compared to the canonical tokenization $\boldsymbol{v}^* = \texttt{[Hyp,nop,atu,rist]}$, with the latter taking only $p(\boldsymbol{v}^*|\boldsymbol{x}) = 0.0004$ of the total mass.

This seems to suggest that there is some mass being attributed to non-canonical tokenizations, especially over longer text. We thus raise another question: instead of using a single tokenization, could we aggregate over all tokenizations, each weighted by their probability, effectively computing the marginal probability of a given string?

5 ... and Computing the Marginal Probability is Also Hard

Evaluating the probability of a string requires marginalizing over all its possible tokenizations. We now formally define this task and show it to be computationally hard.

Problem 5.1 (Marginal String Probability)**.** Let $\boldsymbol{v}$ denote a token sequence. Given a string $\boldsymbol{x}$ and an autoregressive LLM p, the marginal string probability problem is to compute

$$p(\boldsymbol{x}) = \sum_{\boldsymbol{v}} p\left(\boldsymbol{v}, \boldsymbol{x}\right).$$

Theorem 5.2. *The marginal string probability problem is #P-hard.*

Proof. (Sketch) The proof is by reduction from the counting version of the 3-SAT Boolean satisfiability problem (#3-SAT), which is known to be #P-complete. We encode the Boolean variables as possible tokens in a string, such that there is a correspondence between the number of satisfying assignments of the Boolean formula and the marginal probability of the string under the LLM. The full proof can be found in Appendix B. □

Marginal Probability Estimation

In light of the above hardness results, we now shift our attention to approximating the marginal string probability. In particular, we will focus on estimators based on *sequential importance sampling* (Kloek and van Dijk, 1978; Geweke, 1989). In this instance of importance sampling, we sample tokenizations $\boldsymbol{v}$ given a string $\boldsymbol{x}$ according to some proposal distribution $q(\boldsymbol{v}|\boldsymbol{x})$. Given a set of samples $\boldsymbol{v}^{(1)}, \ldots, \boldsymbol{v}^{(N)}$ from this distribution, an estimate of the marginal string probability $p(\boldsymbol{x})$ is

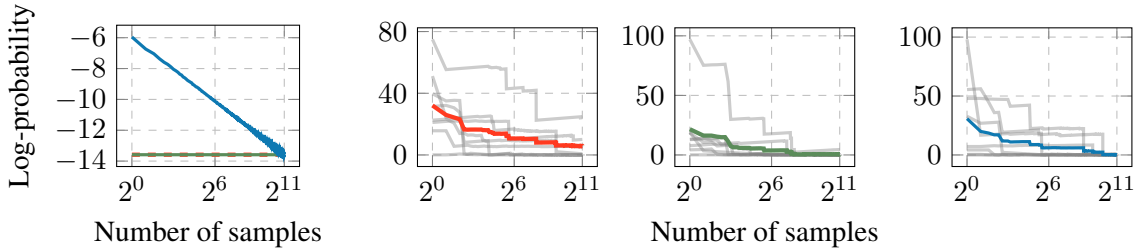
$$p(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{v}\sim q(\boldsymbol{v}|\boldsymbol{x})}\left[\frac{p(\boldsymbol{x},\boldsymbol{v})}{q(\boldsymbol{v}|\boldsymbol{x})}\right] \approx \frac{1}{N}\sum_{i=1}^{N}\frac{p(\boldsymbol{x},\boldsymbol{v}^{(i)})}{q(\boldsymbol{v}^{(i)}|\boldsymbol{x})}. \tag{1}$$

A simple proposal distribution one might consider is the prior LLM token distribution:

$$q_{\text{LLM}}(\boldsymbol{v}|\boldsymbol{x}) := \prod_{j=1}^{|\boldsymbol{v}|} p(v_j|\boldsymbol{v}_{1:j-1}),$$

where $p(v_j|\boldsymbol{v}_{1:j-1})$ is the LLM next-token distribution. However, estimating $p(\boldsymbol{x})$ this way requires rejecting all sampled token sequences where $\boldsymbol{v} \not\models \boldsymbol{x}$, making the approach infeasible in practice.

To address this issue, we use a modified proposal distribution: the *1-step look-ahead proposal distribution*, first proposed in Chirkova et al. (2023). This distribution adjusts the LLM's next-token distribution at each step by checking whether the



(a) String probability estimates (b) Log probability difference between approximate marginal and canonical probability

Figure 6: **Convergence of approximate marginal.** (a) the approximate marginal string probability as a function of the number of samples (——), compared to the canonical probability (——) and the true marginal (- - -). (b) average absolute difference in log-likelihood between the approximate marginal and the canonical probability for different strings across Gemma, Llama2 and Mamba in color, with individual examples in gray (——).

upcoming token v_j, combined with the previous tokens, forms a tokenization of a prefix of the string $\boldsymbol{x}$. Intuitively, we iteratively prune away from the support of the distribution tokenizations not consistent with x. This can be done efficiently by simply traversing the MDD compiled from the string and masking out all tokens that are not compatible with the labels of the outgoing edges at the current node. Formally, the proposal distribution is

$$q_{\text{LA}}(\boldsymbol{v}|\boldsymbol{x}) := \prod_{j=1}^{|\boldsymbol{v}|} q_{\text{LA}}(v_j|\boldsymbol{v}_{1:j-1}, \boldsymbol{x}), \quad \text{where}$$

$$q_{\text{LA}}(v_j|\boldsymbol{v}_{1:j-1}, \boldsymbol{x}) \propto p(v_j|\boldsymbol{v}_{1:j-1})[\![\boldsymbol{v}_{1:j} \models \boldsymbol{x}_{1:}]\!].$$

Here, $[\![\boldsymbol{v}_{1:j} \models \boldsymbol{x}_{1:}]\!]$ evaluates to 1 if $\boldsymbol{v}_{1:j}$ forms a tokenization of a prefix of string $\boldsymbol{x}$, and 0 otherwise. Essentially, the proposal distribution is a *greedy* and *myopic* approximation of the LLM distribution over tokenizations at every step of the sequence.

Interestingly, we observe that, for short strings where we are able to compute the true marginal, even though the proposal eventually converges to the true marginal as the number of samples increases, the probability of the canonical tokenization is just as close to the true marginal. This seems to suggest that the canonical probability is, in fact, practically the marginal probability of the string in these cases. Figure 6a shows one instance of the approximate marginal slowly converging to the true marginal as the number of samples increases for a single small example of the OpenBookQA dataset (Mihaylov et al., 2018).

For longer text, we observe that, for most cases, the approximate marginal also converges close to the canonical probability. As the number of tokenizations to be summed out is enormous, we are unable to compute the true marginal. In none of the cases we evaluated, the approximate marginal probability was meaningfully higher than the canonical. Figure 6b shows the difference in log-probability of several marginal estimates across different architectures and OpenBookQA strings. Notably, estimates that were very different from the canonical probability contained no canonical samples, further confirming that most of the probability mass is in the canonical tokenization.

So far, we have presented empirical evidence that seems to confirm that: (1) canonical is, in most cases, the most likely tokenization, and (2) it carries so much of the probability mass that it is practically the marginal itself. Curiously, in an arguably contradictory twist, we experimentally show evidence that suggests that there exists some signal in non-canonical tokenizations to the point where we are able to achieve consistently better downstream performance in Q&A tasks.

6 Non-Canonical Tokenizations in Question Answering

In multiple-choice question answering, a model is given a question (possibly with context) and is asked to choose between a number of different answers to the question. Typically, this is performed by evaluating the probability of each answer under the default canonical tokenization, and selecting the answer with the highest probability. Formally, given a question $\boldsymbol{c}$ with canonical tokenization $\boldsymbol{v}^*_{\boldsymbol{c}}$ and set of K answers $\{\boldsymbol{a}_i\}_{i=1}^K$ with canonical tokenizations $\{\boldsymbol{v}^*_{\boldsymbol{a}_i}\}_{i=1}^K$, the classification is given by

$$\arg\max_i p(\boldsymbol{v}^*_{\boldsymbol{a}_i}|\boldsymbol{v}^*_{\boldsymbol{c}}).$$

Alternatively, we can compute these probabilities over other tokenizations, for instance, by com-

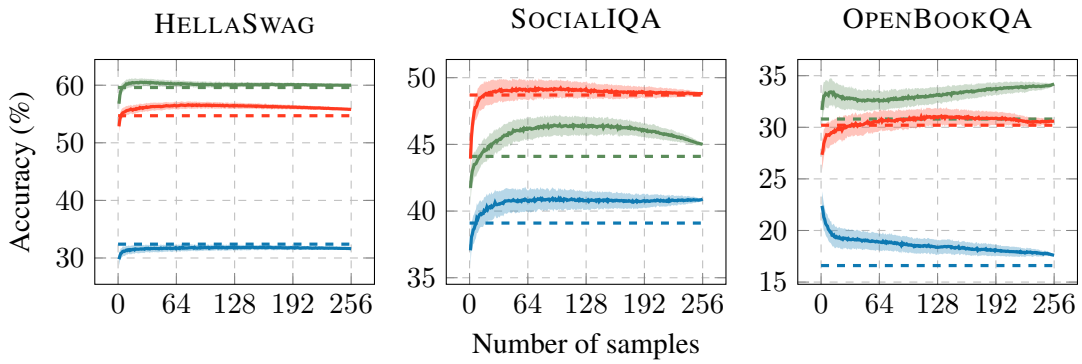


Figure 7: **Accuracy of approximated marginal string probability over number of samples.** Solid curves (——) show marginal mean accuracy, shaded areas (▭) show marginal standard deviation, dashed lines (- - -) show canonical baseline across `Gemma`, `Llama2` and `Mamba`.

	HELLASWAG			SOCIALIQA			OPENBOOKQA			
	CAN	MAR	DIFF	CAN	MAR	DIFF	CAN	MAR	DIFF	AVG DIFF
`Llama2`	59.6	**60.4**	0.81	44.1	**45.4**	1.33	30.8	**33.1**	2.33	1.49
`Gemma`	54.7	**55.8**	1.10	48.7	**48.9**	0.21	30.2	**31.0**	0.76	0.69
`Mamba`	**32.4**	31.6	-0.80	39.1	**40.9**	1.80	16.6	**22.3**	5.69	2.23

Table 1: **Accuracy after tuning the number of samples to estimate marginal string probability.** CAN stands for the canonical baseline accuracy, MAR for the tuned approximate marginal, and DIFF the difference between the last two. **Bold** entries indicate highest accuracy. The last column shows the average difference across the three datasets.

puting an approximation to the marginal:

$$\arg\max_i p(\boldsymbol{a}_i|\boldsymbol{v}_c^*) = \arg\max_i \sum_{\boldsymbol{v}_{\boldsymbol{a}_i} \models \boldsymbol{a}_i} p(\boldsymbol{v}_{\boldsymbol{a}_i}|\boldsymbol{v}_c^*).$$

From prior discussion, we expect the approximate marginal to gradually converge to canonical. However, we empirically find that, before convergence, there is a surprising increase in accuracy when weighting over non-canonical tokenizations compared to the canonical baseline. Figure 7 shows accuracy for the marginal approximation as a function of the number of samples in three different question answering datasets: HELLASWAG (Zellers et al., 2019), SOCIALIQA (Sap et al., 2019) and OPENBOOKQA (Mihaylov et al., 2018). Due to computational constraints, we only evaluate on randomly sampled subsets of 1000 examples for each dataset.

By parameter tuning the number of samples, we are able to achieve a consistent performance increase in accuracy, as Table 1 shows. Tuning the number of samples consisted of sampling 256 samples from a 1000 examples validation hold-out subset, computing the accuracy for 256 trials of 256-choose-k samples, and taking the k which maximizes average accuracy on the hold-out subset. This k is then used to sample that number of tokenizations in the test set. The precise values of k are shown in Table 3 in appendix.

To further understand how much non-canonical tokenizations play a role in this improvement, and find out where the signal in tokenization comes from, we construct a mixture of canonical and non-canonical tokenizations. We evaluate the improvement in accuracy, effectively measuring how much non-canonicity plays a role in the downstream task. More formally, we compute

$$\arg\max_i \ \alpha \cdot p\big(\boldsymbol{v}_{\boldsymbol{a}_i}^*|\boldsymbol{v}_c^*, \boldsymbol{v}_{\boldsymbol{a}_1}^* \vee \boldsymbol{v}_{\boldsymbol{a}_2}^* \vee \ldots \vee \boldsymbol{v}_{\boldsymbol{a}_k}^*\big) + (1-\alpha) \cdot p(\tilde{\boldsymbol{v}}_{\boldsymbol{a}_i}|\boldsymbol{v}_c^*, \tilde{\boldsymbol{v}}_{\boldsymbol{a}_1} \vee \tilde{\boldsymbol{v}}_{\boldsymbol{a}_2} \vee \ldots \vee \tilde{\boldsymbol{v}}_{\boldsymbol{a}_k}),$$

where $0 \leq \alpha \leq 1$ and we use $\tilde{\boldsymbol{v}}_{\boldsymbol{a}_i}$ to denote the set of all non-canonical tokenizations of $\boldsymbol{a}_i$, i.e. $\tilde{\boldsymbol{v}}_{\boldsymbol{a}_i} = \{\boldsymbol{v} : (\boldsymbol{v} \neq \boldsymbol{v}_{\boldsymbol{a}_i}^*) \wedge (\boldsymbol{v} \models \boldsymbol{a}_i)\}$. When $\alpha = 1$, the equation above reduces to the standard canonical tokenization baseline, while $\alpha = 0$ weighs only non-canonical tokenizations of the same answer. To make sure that both terms are on the same scale, we condition the distributions on the possible answers, yielding two classifiers over answers instead of tokenizations.

We approximate $p(\tilde{\boldsymbol{v}}_{\boldsymbol{a}_i}|\boldsymbol{v}_c^*, \tilde{\boldsymbol{v}}_{\boldsymbol{a}_1} \vee \ldots \vee \tilde{\boldsymbol{v}}_{\boldsymbol{a}_k})$ by computing the (unbiased) marginal estimate over

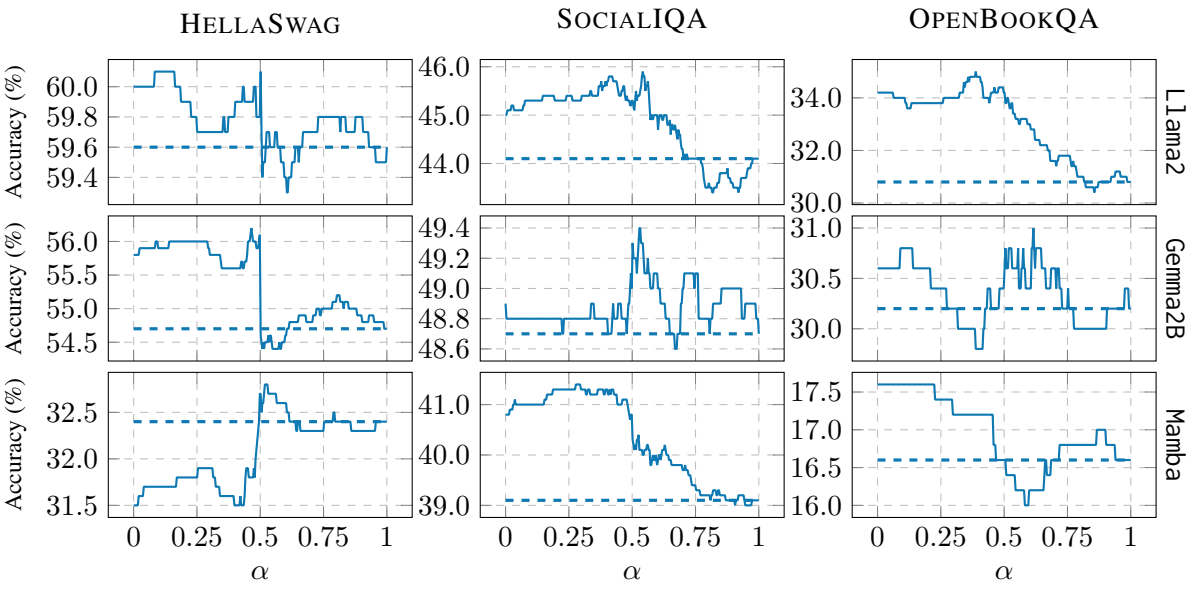


Figure 8: **Accuracy for mixture of canonical and non-canonical tokenizations.** Solid curves (——) show accuracy for the mixture of non-canonical and canonical tokenizations across models and datasets. Dashed lines (- - -) show the canonical baseline.

	HELLASWAG			SOCIALIQA			OPENBOOKQA		
	Llama3	Gemma	Mamba	Llama2	Gemma	Mamba	Llama2	Gemma	Mamba
MIXTURE	**59.7**	**55.8**	31.6	**44.8**	**48.8**	**39.8**	**34.0**	**30.6**	**17.6**
CANONICAL	59.6	54.7	**32.4**	44.1	48.7	39.1	30.8	30.2	16.6

Table 2: **Mixture accuracy after α tuning.** The first row shows accuracy values for the mixture, while the second row shows the canonical baseline. Entries in **bold** indicate highest accuracy.

tokenizations that are *not* canonical, i.e.,

$$
\begin{aligned}
&p(\tilde{\boldsymbol{v}}_{\boldsymbol{a}_i} | \boldsymbol{v}_{\boldsymbol{c}}^*, \tilde{\boldsymbol{v}}_{\boldsymbol{a}_1} \vee \ldots \vee \tilde{\boldsymbol{v}}_{\boldsymbol{a}_k}) \propto p(\tilde{\boldsymbol{v}}_{\boldsymbol{a}_i} | \boldsymbol{v}_{\boldsymbol{c}}^*) \\
&\quad = \mathop{\mathbb{E}}_{\boldsymbol{v} \sim q(\boldsymbol{v} | \boldsymbol{a}_i, \boldsymbol{v}_{\boldsymbol{c}}^*)} \left[\frac{p(\boldsymbol{v}, \boldsymbol{a}_i | \boldsymbol{v}_{\boldsymbol{c}}^*)}{q(\boldsymbol{v} | \boldsymbol{a}_i, \boldsymbol{v}_{\boldsymbol{c}}^*)} \cdot [\![\boldsymbol{v} \neq \boldsymbol{v}_{\boldsymbol{a}_i}^*]\!] \right] \\
&\quad \approx \frac{1}{N} \sum_{j=1}^{N} \frac{p(\boldsymbol{v}^{(j)} | \boldsymbol{v}_{\boldsymbol{c}}^*)}{q(\boldsymbol{v}^{(j)} | \boldsymbol{a}_i, \boldsymbol{v}_{\boldsymbol{c}}^*)} \cdot \left[\!\!\left[\boldsymbol{v}^{(j)} \neq \boldsymbol{v}_{\boldsymbol{a}_i}^* \right]\!\!\right],
\end{aligned}
$$

In practice, this amounts to zeroing-out all importance weights that *are* canonical. The first term of the mixture is computed by simply normalizing the standard canonical probability over the canonical tokenizations of possible answers

$$
p(\boldsymbol{v}_{\boldsymbol{a}_i}^* | \boldsymbol{v}_{\boldsymbol{c}}^*, \boldsymbol{v}_{\boldsymbol{a}_1}^* \vee \ldots \vee \boldsymbol{v}_{\boldsymbol{a}_k}^*) \propto p(\boldsymbol{v}_{\boldsymbol{a}_i}^* | \boldsymbol{v}_{\boldsymbol{c}}^*).
$$

The resulting mixture is then a weighted version of the marginal $p(\boldsymbol{a}_i | \boldsymbol{v}_c^*, \boldsymbol{a}_1 \vee \ldots \vee \boldsymbol{a}_k)$ where we adjust the mass attributed to canonical according to parameter α. This allows us to inspect how the model behaves when mass is "shoveled" around from non-canonical to canonical and vice versa.

Figure 8 shows how this mixture behaves for different values of α downstream, while Table 2 shows the performance change when tuning α in the validation set and applying it on the test set. Precise tuned α values are shown in Table 4.

These experiments show that there is clear and significant signal in non-canonical tokenizations to the point that we are able to achieve a consistent increase in accuracy, suggesting that non-canonical tokenizations do indeed retain meaningful information in LLMs, and hopefully motivating further research in this direction.

7 Conclusion

Modern language models make the assumption that text is represented by a unique canonical tokenization equivalent to the text itself. We showed that not only is the space of possible tokenizations exponential, but prove that reasoning probabilistically about this space is hard. We then showed empirical evidence that suggests that, in practice, not only is the canonical tokenization the most likely tok-

enization, but it is also very close to the probability of the text itself (i.e. the marginal probability of the text summed over tokenizations). Despite this, we present surprising evidence of significant signal in non-canonical tokenizations, thus motivating further research on non-canonical tokenizations.

8 Limitations

Having to evaluate several tokenizations instead of just the canonical one is an obvious limitation to this work, as it requires many more (possibly costly) calls to the LLM. Due to computational constraints, we were also unable to provide evaluation results with regards to larger LLMs.

Acknowledgements

This work was funded in part by the DARPA ANSR program under award FA8750-23-2-0004, the DARPA PTG Program under award HR00112220005, and NSF grant #IIS-1943641. This work was done in part while BW and GVdB were visiting the Simons Institute for the Theory of Computing.

References

Jacob Buckman and Graham Neubig. 2018. Neural lattice language models. *Transactions of the Association for Computational Linguistics*, 6:529–541.

Kris Cao and Laura Rimell. 2021. 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.

Nadezhda Chirkova, Germán Kruszewski, Jos Rozen, and Marc Dymetman. 2023. 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.

YooJung Choi, Antonio Vergari, and Guy Van den Broeck. 2020. Probabilistic circuits: A unifying framework for tractable probabilistic models. Technical report.

Pierre Dupont and Ronald Rosenfeld. 1997. Lattice based language models. Technical report, DTIC Document.

Philip Gage. 1994. A new algorithm for data compression. *The C Users Journal archive*, 12:23–38.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Pier Giuseppe Sessa, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Christian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, Justin Mao-Jones, Katherine Lee, Kathy Yu, Katie Millican, Lars Lowe Sjoesund, Lisa Lee, Lucas Dixon, Machel Reid, Maciej Mikuła, Mateo Wirth, Michael Sharman, Nikolai Chinaev, Nithum Thain, Olivier Bachem, Oscar Chang, Oscar Wahltinez, Paige Bailey, Paul Michel, Petko Yotov, Rahma Chaabouni, Ramona Comanescu, Reena Jana, Rohan Anil, Ross McIlroy, Ruibo Liu, Ryan Mullins, Samuel L Smith, Sebastian Borgeaud, Sertan Girgin, Sholto Douglas, Shree Pandya, Siamak Shakeri, Soham De, Ted Klimenko, Tom Hennigan, Vlad Feinberg, Wojciech Stokowiec, Yu hui Chen, Zafarali Ahmed, Zhitao Gong, Tris Warkentin, Ludovic Peran, Minh Giang, Clément Farabet, Oriol Vinyals, Jeff Dean, Koray Kavukcuoglu, Demis Hassabis, Zoubin Ghahramani, Douglas Eck, Joelle Barral, Fernando Pereira, Eli Collins, Armand Joulin, Noah Fiedel, Evan Senter, Alek Andreev, and Kathleen Kenealy. 2024. Gemma: Open models based on gemini research and technology. *Preprint*, arXiv:2403.08295.

John Geweke. 1989. Bayesian inference in econometric models using monte carlo integration. *Econometrica*, 57(6):1317–1339.

Albert Gu and Tri Dao. 2024. Mamba: Linear-time sequence modeling with selective state spaces. *Preprint*, arXiv:2312.00752.

Richard M Karp. 2010. *Reducibility among combinatorial problems*. Springer.

T. Kloek and H. K. van Dijk. 1978. Bayesian estimates of equation system parameters: An application of integration by monte carlo. *Econometrica*, 46(1):1–19.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

C. Y. Lee. 1959. Representation of switching circuits by binary-decision programs. *The Bell System Technical Journal*, 38(4):985–999.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Conference on Empirical Methods in Natural Language Processing*.

Anaelia Ovalle, Ninareh Mehrabi, Palash Goyal, Jwala Dhamala, Kai-Wei Chang, Richard Zemel, Aram Galstyan, Yuval Pinter, and Rahul Gupta. 2024. Tokenization matters: Navigating data-scarce tokenization for gender inclusive language technologies. *Preprint*, arXiv:2312.11779.

Aleksandar Petrov, Emanuele La Malfa, Philip Torr, and Adel Bibi. 2023. Language model tokenizers introduce unfairness between languages. *Advances in Neural Information Processing Systems*, 36.

Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. 2020. BPE-dropout: Simple and effective subword regularization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social IQa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics.

Aaditya K Singh and DJ Strouse. 2024. Tokenization counts: the impact of tokenization on arithmetic in frontier llms. *arXiv preprint arXiv:2402.14903*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Junxiong Wang, Tushaar Gangavarapu, Jing Nathan Yan, and Alexander M. Rush. 2024. Mambabyte: Token-free selective state space model. *Preprint*, arXiv:2401.13660.

Lili Yu, Daniel Simig, Colin Flaherty, Armen Aghajanyan, Luke Zettlemoyer, and Mike Lewis. 2023. Megabyte: Predicting million-byte sequences with multiscale transformers. In *Advances in Neural Information Processing Systems*, volume 36, pages 78808–78823. Curran Associates, Inc.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

A Problems

For the purposes of studying the complexity of inference problems on induced tokenization distributions, we use $\mathcal{L}$ to denote a class (set) of autoregressive large language models, and make the assumption that this set covers all possible autoregressive distributions:

Assumption A.1 (Expressivity of LLMs)**.** We assume that $\mathcal{L}$ is sufficiently expressive: given any token sequence $\boldsymbol{v} = (v_1, ..., v_m)$, and sequence $\delta_1, ..., \delta_m$ with $\delta_i \in (0, 1)$ for all i, there exists $p \in \mathcal{L}$ such that $p(v_i|v_1, ..., v_{i-1}) = \delta_i$ for all $i = 1, ..., m$.

Note that we do not require that the conditional probability take the value 0 or 1, as this cannot be expressed using logits. We also need to make the (reasonable) assumption that the conditional probability distribution of LLMs can be computed in polynomial time:

Assumption A.2 (Complexity of LLMs)**.** We assume that for any $p \in \mathcal{L}$, and any sequence of tokens $\boldsymbol{v} = (v_1, \ldots, v_m)$, we can compute the distribution $p(v_i|v_1, ..., v_{i-1})$ for any $i = 1, \ldots, m$ in polynomial time in $|\boldsymbol{v}|$.

Now, we consider two inference problems related to the induced tokenization distribution; namely, computing the most likely tokenization, and marginal string probability (a more formal statement of Problems 4.1 and 5.1):

Problem A.3 (Most Likely Tokenization)**.** Given a string $\boldsymbol{x}$, vocabulary $\mathcal{V}$, and an autoregressive LLM $p \in \mathcal{L}$ over $\mathcal{V}$, and a threshold $\epsilon > 0$, we define the most likely tokenization problem $\text{MLT}(\boldsymbol{x}, \mathcal{V}, p, \epsilon)$ as deciding whether:

$$\max_{\boldsymbol{v}} p(\boldsymbol{v}, \boldsymbol{x}) > \epsilon \tag{2}$$

Problem A.4 (Marginal String Probability)**.** Given a string $\boldsymbol{x}$, vocabulary $\mathcal{V}$, and an autoregressive LLM $p \in \mathcal{L}$ over $\mathcal{V}$, the marginal string probability problem $\text{MSP}(\boldsymbol{x}, \mathcal{V}, p)$ is to compute

$$\sum_{\boldsymbol{v}} p(\boldsymbol{v}, \boldsymbol{x}) \tag{3}$$

B Hardness

In this section, we show that Problems A.3 and A.4 are both NP-hard. This will be achieved using a reduction from 3-SAT:

Definition B.1 (3-SAT)**.** Given a set of Boolean variables $a_1, ..., a_n$, a Boolean formula is in 3-CNF if it is of the form:

$$\bigwedge_{k=1}^{K} c_k \tag{4}$$

where each clause c_k is a disjunction of at most 3 literals (a literal is either a variable or its negation). The 3-SAT problem is that of determining if a given 3-CNF formula is satisfiable.

Theorem 4.2. *The most-likely tokenization problem is NP-complete.*

Proof. We begin by showing hardness. Given an instance of 3-SAT, we construct an instance of MLT such that the 3-CNF is satisfiable iff the maximal probability is above a certain threshold.

Let $\psi = \bigwedge_{k=1}^{K} c_i$ be a 3-CNF formula consisting of K clauses over n Boolean variables, where $c_k = l_{k,1} \wedge l_{k,2} \wedge l_{k,3}$, and $l_{k,j}$ is a literal. For convenience, we write $I_{k,j}$ for the index of variable $l_{k,j}$ refers to, and $P_{k,j}$ to be a Boolean variable which is true iff it is a positive literal, i.e. $l_{k,j} = a_{I_{k,j}} \wedge P_{k,j}$.

We now define a string $\boldsymbol{x}$ of length $3n + K$:

abcabcabcabc...ddd...

where $x_{3i+1} =$ "a", $x_{3i+2} =$ "b", $x_{3i+1} =$ "c" for $0 \leq i \leq n-1$, and $x_i =$ 'd' for $3n+1 \leq i \leq 3n+K$. We also define a vocabulary $\mathcal{V} = \{$"a", "bc", "ab", "c", "d"$\}$.

Finally, we define the LLM conditional probability distribution as follows.

$$p(v_i|v_0, ..., v_{i-1}) =$$

$$\begin{cases}
0.45 & \text{if } i = 0 \wedge (v_i = \text{“}a\text{”} \vee v_i = \text{“}ab\text{”}) \\
0.033 & \text{if } i = 0 \wedge \neg(v_i = \text{“}a\text{”} \vee v_i = \text{“}ab\text{”}) \\
0.9 & \text{if } i < 2n \wedge v_{i-1} = \text{“}a\text{”} \wedge v_i = \text{“}bc\text{”} \\
0.025 & \text{if } i < 2n \wedge v_{i-1} = \text{“}a\text{”} \wedge v_i \neq \text{“}bc\text{”} \\
0.9 & \text{if } i < 2n \wedge v_{i-1} = \text{“}ab\text{”} \wedge v_i = \text{“}c\text{”} \\
0.025 & \text{if } i < 2n \wedge v_{i-1} = \text{“}ab\text{”} \wedge v_i = \text{“}c\text{”} \\
0.45 & \text{if } i < 2n \wedge (v_{i-1} = \text{“}bc\text{”} \vee v_{i-1} = \text{“}c\text{”}) \\
 & \wedge(v_i = \text{“}a\text{”} \vee v_i = \text{“}ab\text{”}) \\
0.033 & \text{if } i < 2n \wedge (v_{i-1} = \text{“}bc\text{”} \vee v_{i-1} = \text{“}c\text{”}) \\
 & \wedge\neg(v_i = \text{“}a\text{”} \vee v_i = \text{“}ab\text{”}) \\
0.2 & \text{if } i < 2n \wedge (v_{i-1} = \text{“}d\text{”}) \\
0.9 & \text{if } i \geq 2n \wedge (v_i = \text{“}d\text{”}) \\
 & \wedge S(i+1-2n, \boldsymbol{v}) \\
0.025 & \text{if } i \geq 2n \wedge (v_i \neq \text{“}d\text{”}) \\
 & \wedge S(i+1-2n, \boldsymbol{v}) \\
0.1 & \text{if } i \geq 2n \wedge (v_i = \text{“}d\text{”}) \\
 & \wedge\neg S(i+1-2n, \boldsymbol{v}) \\
0.225 & \text{if } i \geq 2n \wedge (v_i \neq \text{“}d\text{”}) \\
 & \wedge\neg S(i+1-2n, \boldsymbol{v})
\end{cases}$$

Here, $S(k, \boldsymbol{v})$ is a predicate representing the satisfaction of the k^{th} CNF clause. In particular, there is a straightforward bijection between valid tokenizations and instantiations of the CNF variables, by setting $a_i := (v_{2i} =$ "a"$)$. Then the k^{th} clause is satisfied iff the literals appearing in the clause have the appropriate sign, i.e.:

$$S(k, \boldsymbol{v}) = \begin{cases} \text{True} & \text{if } (v_{2I_{k,j}} = \text{“}a\text{”}) = P_{k,j} \\ \text{False} & \text{otherwise} \end{cases} \tag{5}$$

We define $\boldsymbol{v} \models \psi$ iff $\bigwedge_{k=1}^{K} S(k, \boldsymbol{v})$, i.e. all clauses are satisfied. Now we claim that the 3-CNF formula ψ is satisfiable iff $\max_{\boldsymbol{v}} p(\boldsymbol{v}, \boldsymbol{x}) > 0.5(0.45)^n(0.9)^{n+K}$. We begin by noting that all tokenizations of the string $\boldsymbol{x}$ are of the same length $2n + K$, since each "abc" sequenece must be split into either ("ab", "c") or ("a", "bc"), and the "ddd..." sequence must be tokenized into K "d" tokens. The probability of any valid tokenization $\boldsymbol{v}$

is thus given by:

$$
\begin{aligned}
p(\boldsymbol{v}, \boldsymbol{x}) &= \prod_{i=0}^{2n+K-1} p(v_i | v_1, \ldots, v_{i-1}) \\
&= (0.45)^n (0.9)^n \\
&\quad \prod_{i=2n}^{2n+K-1} p(v_i | v_1, \ldots, v_{i-1}) \\
&= (0.45)^n (0.9)^n \\
&\quad \prod_{i=2n}^{2n+K-1} p(\text{“}d\text{”} | v_1, \ldots, v_{i-1})
\end{aligned}
$$

Note that the remaining conditional probabilities are all either 0.9 or 0.025; thus, $p(\boldsymbol{v}, \boldsymbol{x}) > 0.5(0.45)^n(0.9)^{n+K}$ iff all of these conditional probabilities are 0.9. Since all of the tokens v_i for $i \geq 2n$ (for $\boldsymbol{x}$) are “d", this happens iff $\boldsymbol{v} \models \psi$, and the 3-CNF ψ is satisfiable. Thus MLT is NP-hard.

To show NP-completeness, we note all tokenizations have length $2n + K$ and so oracle calls to the LLM take polynomial time in n, K by Assumption A.2. If the answer to MLT is Yes, then there exists a tokenization $\boldsymbol{v}'$ with $p(\boldsymbol{v}, \boldsymbol{x}) > t$ which acts as the certificate. This certifiacte can be checked in polynomial time; thus MLT is in NP. □

We now move to the problem of computing the marginal probability over all valid tokenizations of $\boldsymbol{x}$. The proof of this result relies on a similar construction to the proof of Theorem 4.2.

Theorem 5.2. *The marginal string probability problem is #P-hard.*

Proof. Given an instance of #3-SAT, we construct an instance of MSP such that the count of the 3-CNF formula can be easily determined by the marginal string probability.

We define the 3-CNF formula ψ, string $\boldsymbol{x}$, and vocabulary $\mathcal{V}$ as in the proof of Theorem 4.2. However, we define a slightly different distribution for the LLM:

$$
p(v_i | v_0, ..., v_{i-1}) = \begin{cases}
0.45 & \text{if } i = 0 \\
& \wedge (v_i = \text{“}a\text{”} \vee v_i = \text{“}ab\text{”}) \\
0.033 & \text{if } i = 0 \\
& \wedge \neg (v_i = \text{“}a\text{”} \vee v_i = \text{“}ab\text{”}) \\
0.9 & \text{if } i < 2n \\
& \wedge (v_{i-1} = \text{“}a\text{”} \wedge v_i = \text{“}bc\text{”}) \\
0.025 & \text{if } i < 2n \\
& \wedge (v_{i-1} = \text{“}a\text{”} \wedge v_i \neq \text{“}bc\text{”}) \\
0.9 & \text{if } i < 2n \\
& \wedge (v_{i-1} = \text{“}ab\text{”} \wedge v_i = \text{“}c\text{”}) \\
0.025 & \text{if } i < 2n \\
& (\wedge v_{i-1} = \text{“}ab\text{”} \wedge v_i = \text{“}c\text{”}) \\
0.45 & \text{if } i < 2n \\
& \wedge (v_{i-1} = \text{“}bc\text{”} \vee v_{i-1} = \text{“}c\text{”}) \\
& \wedge (v_i = \text{“}a\text{”} \vee v_i = \text{“}ab\text{”}) \\
0.033 & \text{if } i < 2n \\
& \wedge (v_{i-1} = \text{“}bc\text{”} \vee v_{i-1} = \text{‘}c\text{”}) \\
& \wedge \neg (v_i = \text{“}a\text{”} \vee v_i = \text{“}ab\text{”}) \\
0.2 & \text{if } i < 2n \wedge (v_{i-1} = \text{“}d\text{”}) \\
1 - 0.5^{n+K+1} & \text{if } i \geq 2n \wedge (v_i = \text{“}d\text{”}) \\
& \wedge S(i + 1 - 2n, \boldsymbol{v}) \\
\frac{0.5^{n+K+1}}{4} & \text{if } i \geq 2n \wedge (v_i \neq \text{“}d\text{”}) \\
& \wedge S(i + 1 - 2n, \boldsymbol{v}) \\
0.5^{n+K+1} & \text{if } i \geq 2n \wedge (v_i = \text{“}d\text{”}) \\
& \wedge \neg S(i + 1 - 2n, \boldsymbol{v}) \\
\frac{1-0.5^{n+K+1}}{4} & \text{if } i \geq 2n \wedge (v_i \neq \text{“}d\text{”}) \\
& \wedge \neg S(i + 1 - 2n, \boldsymbol{v})
\end{cases}
$$

The difference between this distribution and that in Theorem 4.2 is the last 4 cases, where the probability is dependent on the number of CNF variables n. Now, we will show that the model count of the CNF formula ψ is equal to $C \in \{0, ..., 2^n\}$ iff $(C - 0.5)(0.45)^n(0.9)^n < \sum_{\boldsymbol{v}} p(\boldsymbol{v}, \boldsymbol{x}) < (C + 0.5)(0.45)^n(0.9)^n$.

As before, the probability of any valid tokenization $\boldsymbol{v}$ is given by:

$$
\begin{aligned}
p(\boldsymbol{v}, \boldsymbol{x}) =& (0.45)^n (0.9)^n \\
& \prod_{i=2n}^{2n+K-1} p(\text{“}d\text{”} | v_1, ..., v_{i-1})
\end{aligned}
$$

Now, consider the valid tokenizations $\boldsymbol{v}$ of $\boldsymbol{x}$ that correspond to a satisfying assignment of ψ. For

any such tokenization, we have that $S(k, \boldsymbol{v})$ for all $k = 1, ..., K$ and so its probability

$$\begin{aligned} p(\boldsymbol{v}, \boldsymbol{x}) &= (1 - 0.5^{n+K+1})^K (0.45)^n (0.9)^n \\ &\geq (1 - 0.5^{n+K+1})(0.45)^n (0.9)^n \\ &\geq (1 - 0.5^{n+2})(0.45)^n (0.9)^n \end{aligned}$$

On the other hand, for any valid tokenization which does not correspond to a satisfying assignment ψ, there exists a k s.t. $\neg S(k, \boldsymbol{v})$, and so all such tokenizations have probability $p(\boldsymbol{v}, \boldsymbol{x}) < \frac{(0.5)^{n+K+1}}{4}(0.45)^n (0.9)^n$.

Thus, if ψ has C satisfying assignments, then we have that

$$\begin{aligned} \sum_{\boldsymbol{v}} p(\boldsymbol{v}, \boldsymbol{x}) &\geq \sum_{\boldsymbol{v} \models \psi} p(\boldsymbol{v}, \boldsymbol{x}) \\ &= C(1 - 0.5^{n+K})^K (0.45)^n (0.9)^n \\ &\geq C(1 - 0.5^{n+2})(0.45)^n (0.9)^n \\ &> (C - 0.5)(0.45)^n (0.9)^n \end{aligned}$$

where the last inequality follows because C is bounded above by 2^n. Also, since there are 2^n valid tokenizations, we have that

$$\begin{aligned} \sum_{\boldsymbol{v}} p(\boldsymbol{v}, \boldsymbol{x}) &= \sum_{\boldsymbol{v} \models \psi} p(\boldsymbol{v}, \boldsymbol{x}) + \sum_{\boldsymbol{v} \not\models \psi} p(\boldsymbol{v}, \boldsymbol{x}) \\ &< C(0.45)^n (0.9)^n \\ &\quad + 2^n \frac{(0.5)^{n+K+1}}{4}(0.45)^n (0.9)^n \\ &= C(0.45)^n (0.9)^n \\ &\quad + (0.5)^{K+3}(0.45)^n (0.9)^n \\ &< (C + 0.5)(0.45)^n (0.9)^n \end{aligned}$$

We have shown that the model count of the CNF formula ψ is equal to $C \in \{0, ..., 2^n\}$ iff $(C - 0.5)(0.45)^n (0.9)^n < \sum_{\boldsymbol{v}} p(\boldsymbol{v}, \boldsymbol{x}) < (C + 0.5)(0.45)^n (0.9)^n$. Given $\sum_{\boldsymbol{v}} p(\boldsymbol{v}, \boldsymbol{x})$, we can compute the (unique) value of $C \in \{0, ..., 2^n\}$ for which this holds by binary search, with complexity $O(n)$. Thus we have reduced #3-SAT to MSP and so MSP is #P-hard.

□

C Experiments

Figure 9 shows marginal estimates across models and datasets on the validation set as a function of the number of samples. Figure 8 shows mixture performance as a function of parameter α on the validation set.

	Llama2	Gemma	Mamba	
CAN	59.6	54.7	**32.4**	HELLASWAG
MAR	**60.4**	**55.8**	31.6	
DIFF	0.81	1.10	-0.80	
# SAMPLES	41	256	256	
CAN	44.1	48.7	39.1	SOCIALIQA
MAR	**45.4**	**48.9**	**40.9**	
DIFF	1.33	0.21	1.80	
# SAMPLES	238	140	256	
CAN	30.8	30.2	16.6	OPENBOOKQA
MAR	**33.1**	**31.0**	**22.3**	
DIFF	2.33	0.76	5.69	
# SAMPLES	7	163	1	
AVG DIFF	1.49	0.69	2.23	

Table 3: **Accuracy after parameter tuning.** See Table 1 for more details. # SAMPLES shows the number of samples after tuning on the validation set.

	MIXTURE	CANONICAL	
	Accuracy (%)		
Llama2	**59.7**	59.6	HELLASWAG
Gemma	**55.8**	54.7	
Mamba	31.6	**32.4**	
Llama2	**44.8**	44.1	SOCIALIQA
Gemma	**48.8**	48.7	
Mamba	**39.8**	39.1	
Llama2	**34.0**	30.8	OPENBOOKQA
Gemma	**30.6**	30.2	
Mamba	**17.6**	16.6	
	Fine-tuned α values		
Llama2	0.3445	—	HELLASWAG
Gemma	0.6421	—	
Mamba	0.5151	—	
Llama2	0.3244	—	SOCIALIQA
Gemma	0.4816	—	
Mamba	0.0000	—	
Llama2	0.0201	—	OPENBOOKQA
Gemma	0.5753	—	
Mamba	0.0000	—	

Table 4: **Mixture accuracy after α tuning.** See Table 2 for more details. The lower portion of the table shows α values after tuning on the validation set.

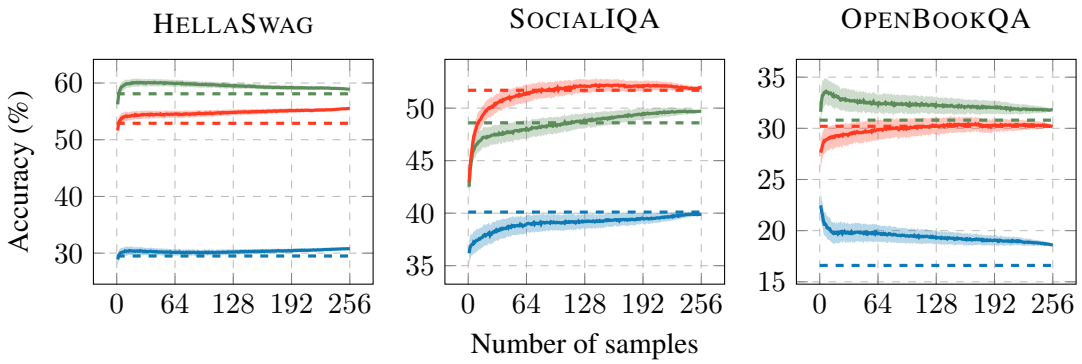


Figure 9: **Accuracy of approximated marginal over number of samples on the validation set.** Curves in solid —— show marginal mean accuracy, shaded areas ▭ show marginal standard deviation, dashed lines - - - show canonical baseline across Gemma, Llama2 and Mamba.

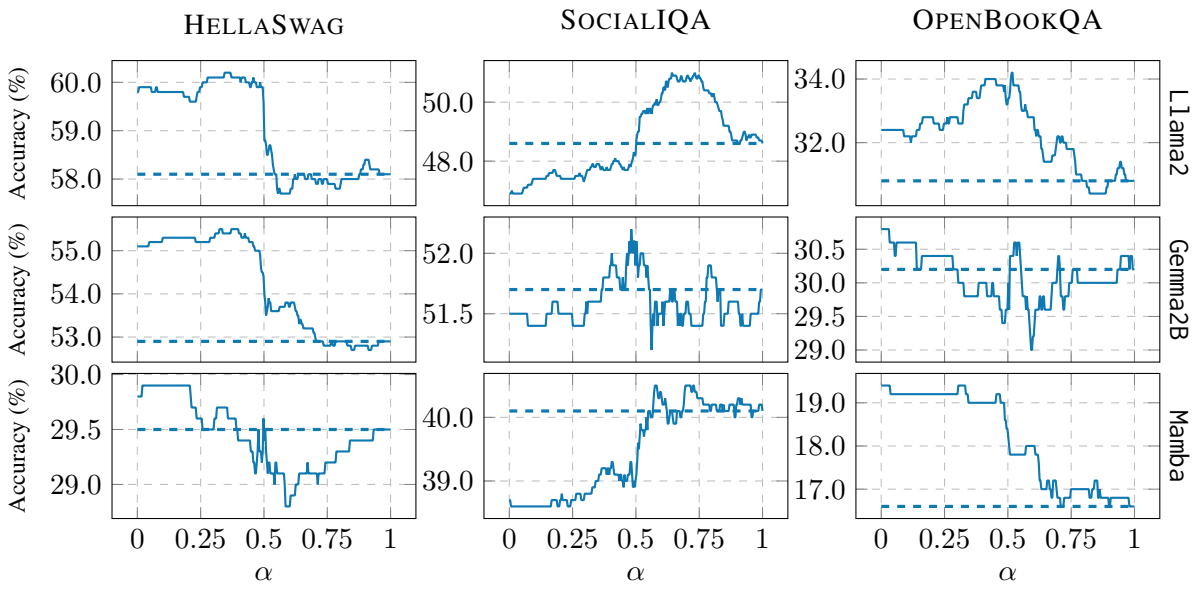


Figure 10: **Accuracy for mixture of canonical and non-canonical tokenizations.** Solid curves (——) show accuracy for the mixture of non-canonical and canonical tokenizations across models and datasets. Dashed lines (- - -) show the canonical baseline.