Decoding-Free Sampling Strategies for LLM Marginalization

David Pohl[1*†], **Marco Cognetta**[1*], **Junyoung Lee**[2], **Naoaki Okazaki**[1]

[1]Institute of Science Tokyo, Japan
[2]Independent Researcher

Abstract

Modern language models operate on subword-tokenized text in order to make a trade-off between model size, inference speed, and vocabulary coverage. A side effect of this is that, during inference, models are evaluated by measuring the probability of only the specific tokenization produced as the output, despite there being many possible ways to represent the same text with a subword vocabulary. Recent studies have argued instead for evaluating LLMs by *marginalization* — the probability mass of all tokenizations of a given text.

Marginalization is difficult due to the number of possible tokenizations of a text, so often approximate marginalization is done via sampling. However, a downside of sampling is that an expensive generation step must be performed by the LLM for each sample, which limits the number of samples that can be acquired given a runtime budget, and therefore also the accuracy of the approximation. Since computing the probability of a sequence given the tokenization is relatively cheap compared to actually generating it, we investigate sampling strategies that are *decoding-free* — they require no generation from the LLM, instead relying entirely on extremely cheap sampling strategies that are model and tokenizer agnostic.

We investigate the approximation quality and speed of decoding-free sampling strategies for a number of open models to find that they provide sufficiently accurate marginal estimates at a small fraction of the runtime cost and demonstrate its use on a set of downstream inference tasks.

1 Introduction

Language models consume and generate text at a subword granularity. In particular, when generating text, they output a subword sequence that is detokenized to give the raw output text. However, this subword-sequence-to-text mapping is many-to-one, as there are an exponential number of possible ways to tokenize an input sequence under a given subword vocabulary. A common practice is to consider the *canonical* tokenization of a text (that is, the subword sequence produced by the model's tokenizer when given the input) or the single sequence that is generated by the model (under some decoding policy that is designed to find the maximum probability sequence given the model and context) as the representative tokenization of an input to be used a proxy for the probability of its detokenized surface form. Formally, given a (text) sequence $\mathbf{s}$, we often approximate the probability

$$p(\mathbf{s}) \approx p(t, \mathbf{s}) = \prod_{i}^{|t|} P_\theta(t_i \mid t_{<i}),$$

where t is some valid tokenization of $\mathbf{s}$, and $P_\theta(t_i \mid t_{<i})$ is a model's auto-regressive next token probability according to the parameters of the model, θ. However, an argument can be made that we should instead use the *marginal* probability

$$p(\mathbf{s}) = \sum_{\mathbf{s} \rightsquigarrow t} p(t, \mathbf{s}),$$

where $\mathbf{s} \rightsquigarrow t$ is the set of all possible tokenizations of $\mathbf{s}$.

The marginal is difficult to compute, since $\mathbf{s} \rightsquigarrow t$ can be very large (exponential in the length of $\mathbf{s}$ and the size of the subword vocabulary). Instead, techniques like importance sampling are often used to approximate the marginal (Cao and Rimell 2021; Geh et al. 2024). However, importance sampling requires *generating* many sequences from

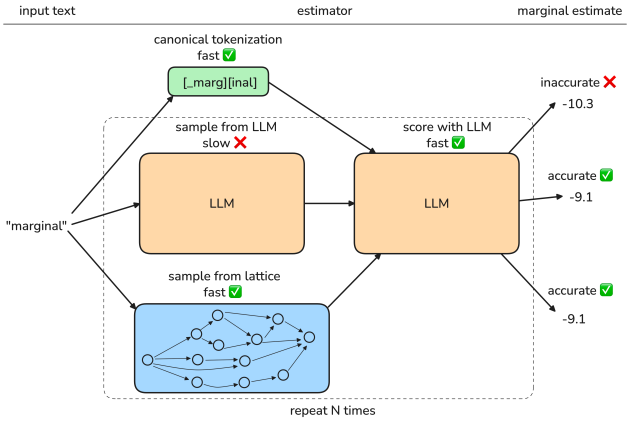

Figure 1: Three methods for approximating the marginal of an input text. The typical approach is to compute the probability of the canonical tokenization, which is fast but inaccurate. Importance sampling allows for more accurate marginalization, but sampling from an LLM is very slow. Our lattice approach is fast and produces good estimates.

[*]These authors contributed equally.
[†]Correspondence to david.pohl@nlp.c.titech.ac.jp

the underlying LLM, which is computationally expensive. We additionally find that importance sampling often significantly *underestimates* the marginal across many models and tasks.

We investigate alternative marginalization strategies that are *decoding-free*, in that they never require *generation* by an LLM, but only *scoring*, which is relatively cheap. These strategies are language model agnostic and rely only on manipulations of a *subword-lattice* that succinctly encodes all possible tokenizations of an output while still remaining easy to construct and sample from.

Several works have found that the marginal, and specifically the marginal excluding the canonical tokenization, carries meaningful signal in that it can match or outperform canonical-only inference on downstream tasks (Chirkova et al. 2023; Geh et al. 2024; Zheng et al. 2025). Our experimental results also support this and further motivate our contributions since we provide more accurate marginal estimates at a fraction of the inference-time cost.

2 Related Work

Tokenization is the first step of the modern neural language modeling pipeline, where text is converted into a subword sequence that is understandable by the model. Many algorithms exist for subword tokenization, but Byte-Pair Encoding (BPE) (Sennrich, Haddow, and Birch 2016) has found the most use in modern LLMs (Brown et al. 2020; Touvron et al. 2023; Gemma Team et al. 2025; Qwen et al. 2025).

Tokenization is generally a deterministic process, but, given a subword vocabulary, there are often many ways to represent a given input text and recent research has considered the problem of computing the *marginal* of an input — the total probability mass assigned to all possible tokenizations of a given input sequence, not just the canonical tokenization from the tokenizer (Cao and Rimell 2021; Chirkova et al. 2023; Geh et al. 2024).

In particular, Cao and Rimell (2021) find that marginalization helps improve LM accuracy, especially on out-of-domain data, and Geh et al. (2024) find using the marginal of *non-canonical* tokenizations can even out-perform traditional decoding methods on Q&A tasks. Both of these use importance sampling, a method for estimating difficult-to-sample-from distributions, for estimating the marginal. This is necessary as computing the marginal is known to be NP-Hard in general (Geh et al. 2024).

Other work has analyzed and improved language models' ability to handle non-canonical tokenization by replacing static tokenizers with stochastic tokenizers so that models are exposed to many different tokenizations of the same text (Kudo 2018; Provilkov, Emelianenko, and Voita 2020; Cognetta, Zouhar, and Okazaki 2024; Bauwens, Kaczér, and De Lhoneux 2025). Additionally, Zheng et al. (2025) found that, even LLMs that are trained with deterministic tokenizers are robust to major perturbations of the input tokenization. Further, several works have found that LLMs have implicit knowledge of their subword vocabulary surface forms, even without being exposed to them directly (Kaushal and Mahowald 2022; Hiraoka and Okazaki 2024; Edman, Schmid, and Fraser 2024), again implying that non-canonical tokenizations carry meaningful information.

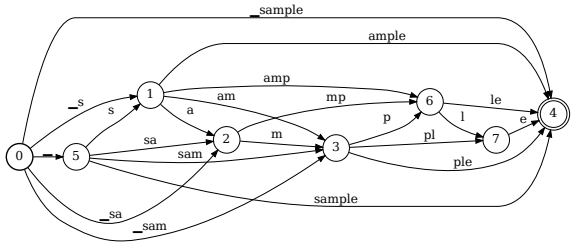

Figure 2: An example subword tokenization lattice for the input $\mathbf{s} = \texttt{sample}$ using the LLAMA2 vocabulary.

3 Background and Notation

Subword Tokenization

Subword tokenization represents raw text at an intermediate granularity between characters and words (Mielke et al. 2021). The basis of subword tokenization is a *subword vocabulary*, the set of subword tokens that can be used for representation. Given a character alphabet Σ, a subword vocabulary Γ is a finite set of strings made up of characters from Σ, with the additional requirement that $\Sigma \subseteq \Gamma$.

Subword tokenizers convert character sequences over Σ^* to subword sequences over Γ^*. Typically, subword tokenizers are deterministic (each possible input is mapped to a single, canonical output). We write $T : \Sigma^* \to \Gamma^*$ for the tokenization process and $T^{-1} : \Gamma^* \to \Sigma^*$ for the inverse detokenization process. Unlike the tokenization process, the detokenization process is not one-to-one. For example, a tokenizer might always produce $T(\texttt{sampler}) = \texttt{[\_sam][pler]}$ but the detokenization process is many-to-one: $T^{-1}(\texttt{[\_sa][mp][ler]}) = T^{-1}(\texttt{[\_s][ample][r]}) = \cdots = T^{-1}(\texttt{[\_s][a][m][p][l][e][r]}) = \texttt{sampler}$.

Subword tokenizers can also be *stochastic* by injecting randomness into the tokenization process to produce a probability distribution of sequences (Kudo 2018; Provilkov, Emelianenko, and Voita 2020).

Building a Tokenization Lattice

We utilize finite automata and transducers to build a *lattice* of subword tokenizations for an input — an automaton that encodes all subword sequences that, when detokenized, would match the input (Willard and Louf 2023; Cognetta and Okazaki 2024; Koo, Liu, and He 2024). An example is shown in Figure 2. We only briefly introduce the necessary concepts here, and point the reader to Allauzen and Mohri (2009) for a more detailed overview of automata and transducer algorithms.

Suppose we have a sequence $\mathbf{s}$ and a subword vocabulary Γ (irrespective of the underlying tokenization algorithm). We wish to find the full set of tokenizations t which corresponds to $\mathbf{s}$. Let $\mathcal{T}_\Gamma$ be a character-to-subword

transducer $\mathcal{T}_\Gamma : \Sigma^* \to \Gamma$ which recognizes the relation $\bigcup_{t\in\Gamma}(T^{-1}(t), t)$. In words, this maps sequences of characters in Σ to a subword token $t \in \Gamma$. After closure, we have a transducer $\mathcal{T}_\Gamma^* : \Sigma^* \to \Gamma^*$ which maps sequences of characters to *sequences* of subwords in Γ. Then, given $\mathbf{s} \in \Sigma^*$, we represent it as an automaton (which only accepts $\mathbf{s}$) $\mathcal{A}_\mathbf{s}$.

Let $\mathcal{A}\circ\mathcal{T}$ be the *composition* operator between an automaton and a transducer. This operator computes the relation

$$\mathcal{A} \circ \mathcal{T} = \{(\mathbf{x}, \mathbf{y}) \mid \mathbf{x} \in \mathcal{L}(\mathcal{A}) \wedge (\mathbf{x}, \mathbf{y}) \in \mathcal{L}(\mathcal{T})\},$$

the subset of transductions encoded by $\mathcal{T}$ where the input is in the language of $\mathcal{A}$.

Additionally, let $\textsc{Proj} : \Sigma^* \times \Gamma^* \to \Gamma^*$ be a unary *projection* operator which transforms a transducer to an automaton by computing the set $\textsc{Proj}(\mathcal{T}) = \{\mathbf{y} \mid (\mathbf{x}, \mathbf{y}) \in \mathcal{L}(\mathcal{T})\}$ of sequences in Γ^* such that there is some input sequence in Σ^* that $\mathcal{T}$ transduces to it.

Then, computing $\textsc{Proj}(\mathcal{A}_\mathbf{s} \circ \mathcal{T}_\Gamma^*)$ produces the subword lattice of $\mathbf{s}$.

Sampling from LLMs

Sampling sequences from an LLM involves repeatedly sampling tokens from the model's internal autoregressive probability distribution, where a sequence $t_1 t_2 \dots t_n \in \Gamma^*$ is sampled with probability:

$$p(t_1 t_2 \dots t_n) = \prod_{i=1}^{|t|} P_\theta(t_i \mid t_{<i}),$$

where $P_\theta(t_i \mid t_{<i})$ is the probability distribution over next tokens, given a context $t_1 t_2 \dots t_{i-1}$, produced by the softmax output layer of a language model parameterized by θ.

After each sample, the sampled token is added to the context and the process is repeated until we sample the end-of-sequence token or reach a maximum length.

Constrained Sampling

We often wish to constrain an LLM with vocabulary Γ to generate only sequences which match a given pattern. Let $\mathcal{A}^{sub}$ be a subword-level pattern constructed by promoting a character-level pattern $\mathcal{A}$ via $\textsc{Proj}(\mathcal{A} \circ \mathcal{T}_\Gamma^*)$.

We wish to modify the sequential probability distribution of the LLM so that $p'(t_1 t_2 \dots t_n) = 0$ if $t_1 t_2 \dots t_n \notin \mathcal{L}(\mathcal{A}^{sub})$. A common way to do this is to *locally-constrain* the autoregressive probability computation:

$$P'_\theta(t_i \mid t_{<i}, \mathcal{A}^{sub}) = \begin{cases} \frac{P_\theta(t_i \mid t_{<i})}{z}, & \text{if } t_1 t_2 \dots t_i \Gamma^* \subseteq \mathcal{A}^{sub} \\ 0 & \text{otherwise.} \end{cases}$$

When generating a token t_i to append to a context $t_{<i}$, P'_θ assigns probability 0 to all tokens in Γ that would prevent the sequence from matching the underlying pattern and uses the original autoregressive probability, scaled by a normalization factor z, for all valid tokens.

Then, our constrained distribution is

$$p'(t_1 t_2 \dots t_n, \mathcal{A}^{sub}) = \prod_{i=1}^{|t|} P'_\theta(t_i \mid t_{<i}, \mathcal{A}^{sub}),$$

which, by the product chain rule, assigns probability 0 to all subword sequences not in $\mathcal{A}^{sub}$, as desired.

We can use this to sample only tokenizations of an input sequence $\mathbf{s}$ by using $\mathcal{A}_\mathbf{s}^{sub} = \textsc{Proj}(\mathcal{A}_\mathbf{s} \circ \mathcal{T}_\Gamma^*)$, which will be useful in the remainder of this work.

4 Marginalization

Marginalization computes the probability of the *surface form* of $\mathbf{s}$ instead of just a single tokenization of $\mathbf{s}$. In general, marginalization is intractable, since there are an exponential number of possible tokenizations of an input $\mathbf{s}$, but approximate marginalization algorithms exist. For language models, these algorithms typically take samples from the model's distribution (or a related one) and use the sample probabilities to estimate the total probability mass (Cao and Rimell 2021; Geh et al. 2024; Vieira et al. 2025).

The simplest is *rejection sampling*, where we sample directly from p, but "reject" samples that do not match our target surface form. But, an issue with rejection sampling is that if $p(\mathbf{s})$ is small, we may only rarely sample valid tokenizations, and so this sum will take a long time to converge.

Instead, we can use importance sampling, where we define a *proxy* distribution q (as opposed to the *primary* distribution, p) which only produces samples that match $\mathbf{s}$:

$$p(\mathbf{s}) = \mathbb{E}_{t\sim q}\left[\frac{p(t)}{q(t)}\right] \approx \frac{1}{N}\sum_i^N \frac{p(t^{(i)})}{q(t^{(i)})}.$$

In line with prior work, we use the locally-masked distribution for constrained LLM next-token prediction based on the subword lattice (Geh et al. 2024). Let $\mathbf{s}$ be our intended surface form and $\mathcal{A}_\mathbf{s}^{sub}$ be the lattice of all tokenizations of $\mathbf{s}$. Then, we set our proxy distribution to be $p'(\cdot, \mathcal{A}_\mathbf{s}^{sub})$, which allows us to only generate sequences which match $\mathbf{s}$.

Full and Partial Enumeration

In principle, we can exactly compute the marginal by enumerating all sequences in $\mathcal{A}_\mathbf{s}^{sub}$. If $\mathbf{s}$ is short, this may be possible, but as $\mathbf{s}$ grows it becomes infeasible due to the number of tokenizations of a sequence being exponential in the length of $\mathbf{s}$. Additionally, Geh et al. (2024) showed that computing the marginal exactly is NP-Hard for LLMs.

However, note that *any* partial enumeration (i.e., a set of sequences with no duplicates) acts a lower-bound on the true marginal which monotonically increases towards the true bound as you increase the number of samples.

5 The Distribution of Probability Mass

Language models are trained on the canonical tokenizations of text, which causes the bulk of the probability mass for a given surface form to be held by its canonical tokenization. In practice, this means that the canonical tokenization is a reasonable approximation for the marginal. However, the non-canonical marginal has been shown to carry significant signal as well (Geh et al. 2024). Therefore, we investigate approximating the non-canonical marginal.

Excluding the canonical tokenization from marginal computation offers the following two key benefits. First, for sampling schemes, a large number of samples will correspond

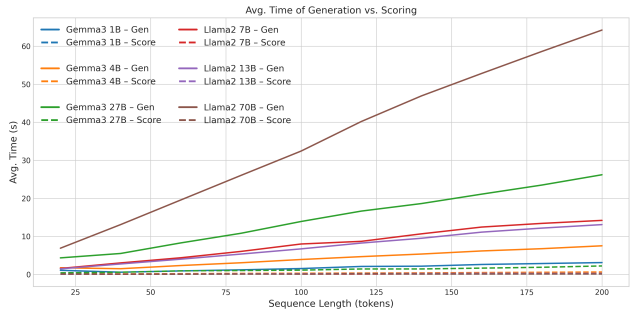

Figure 3: The runtimes of *generating* a sequence from an LLM and *scoring* a sequence with an LLM for LLAMA and GEMMA. Note that, as the sequence length grows, generation becomes significantly slower than scoring.

to the canonical tokenization, meaning we are not able to explore much of the tokenization space and we see a less diverse set of samples. And second, an estimate of the non-canonical marginal can be turned into an estimate of the full marginal by simply adding the probability of the canonical sequence to it.

A Lot of Mass is "Near" the Canonical Sequence Empirically, we observed that most of the marginal is contained in the canonical sequence like in (Geh et al. 2024). But, if we consider only the non-canonical marginal, we observed in our experiments that a huge portion of the probability mass is contained in sequences that are *almost* the canonical sequence. For example, sequences which are identical to the canonical except one token is split into two.

Since many tokenization algorithms have their roots in compression, it is unsurprising that canonical tokenizations tend to have nearly the optimal sequence length for an input sequence and vocabulary. It has been shown that language models have a bias towards shorter sequences due to the autoregressive decoding strategy (Murray and Chiang 2018; Welleck et al. 2024). Intuitively then, it is not surprising that other short sequences would have relatively high probability, especially if they are very similar to the canonical sequence except for a few tokens (Chirkova et al. 2023).

We will exploit this in Section 7 to produce a fast estimate of the marginal by lower-bounding it with the probability of tokenizations that are close to the canonical.

Downsides of a Constrained LLM as a Proxy

There are two primary downsides to using a proxy distribution based on the locally-masked distribution P'_θ: speed and sample quality.

Sampling sequences in language models (constrained or unconstrained) is computationally expensive, especially compared to scoring an existing subword sequence. Figure 3 shows the wall-clock runtime penalty of sampling from the LLM due to the quadratic runtime of self attention.

The slowdown is largely caused by generation requiring $O(n)$ calls to the forward pass[1] of a model to generate a sequence of length n, which must be performed serially. On the other hand, scoring a sequence requires only one forward pass which can be parallelized across the sequence length.

[1]We focus on forward passes rather than asymptotic runtime complexity to abstract away details of the architecture and the use of techniques like KV-caching.

MODEL	SPEARMAN'S ρ		
	$\lvert\mathbf{s}\rvert \approx 50$	$\lvert\mathbf{s}\rvert \approx 100$	$\lvert\mathbf{s}\rvert \approx 300$
LLAMA2-7B	0.70 ± 0.16	0.66 ± 0.10	0.75 ± 0.11
LLAMA2-13B	0.64 ± 0.15	0.68 ± 0.11	0.76 ± 0.09
GEMMA3-1B	0.69 ± 0.17	0.63 ± 0.14	0.65 ± 0.12
GEMMA3-4B	0.71 ± 0.19	0.58 ± 0.13	0.58 ± 0.11
GEMMA3-12B	0.80 ± 0.13	0.70 ± 0.12	0.64 ± 0.09

Table 1: Spearman's ρ for the rankings of sequences by probability under p (base language model) and q (the constrained language model) at different sequence lengths. Each sequence length test-set contains 15 sequences. For each sequence, we generate 1000 unique tokenizations by sampling from a constrained language model and then compare the sequences' relative rankings. There is only moderate positive correlation between the rankings from p and q, which can cause importance sampling from q to converge slowly.

The second issue is that the local-masking distribution P'_θ produces a warped autoregressive distribution that is not necessarily representative of the original, unconstrained language model's distribution (Vieira et al. 2025). Table 1 contains Spearman ρ values for the relative orderings of a language model and it's constrained version for a series of surface forms. The warped distribution is only moderately correlated with the original distribution (that is, not only does $p(t_1) \neq p'(t_1)$, but also $p(t_1) > p(t_2)$ does not necessarily imply $p'(t_1) > p'(t_2)$).

In Section 8, we also show that importance sampling tends to severely underestimate the marginal, possibly due to oversampling low-probability sequences due to the imperfect alignment of the proxy and primary distributions' orderings.

6 Generating Tokenizations Efficiently

Rather than sampling from the constrained LLM, we aim to generate tokenizations in a different and cheaper way by utilizing the tokenization lattice from Section 3.

Here, we discuss only the sampling methods used in our final implementation. There are a variety of stochastic tokenization methods that could be used like BPE-Dropout (Provilkov, Emelianenko, and Voita 2020), MaxMatch-Dropout (Hiraoka 2022), UnigramLM (Kudo 2018), and other variants (Hiraoka, Shindo, and Matsumoto 2019; Zheng et al. 2025). However, each of these were insufficient in some way: they do not allow for controlling tokenization output length, they cannot be sampled without replacement, they require a training step, or they are dependent on a specific underlying tokenization algorithm, so we do not cover them here.

Uniformly Sampling Tokenizations

Let $\mathbf{s} \in \Sigma^*$ be the input text and let $\mathcal{A}_{\mathbf{s}}^{sub} = \textsc{Proj}(\mathcal{A}_{\mathbf{s}} \circ \mathcal{T}_\Gamma^*)$ be the tokenization lattice, which is a DAG where every path describes a valid tokenization of $\mathbf{s}$ according to the tokeinzer's vocabulary Γ. Suppose there are N possible tokenizations, then sampling uniformly would mean sampling each individual path with probability $\frac{1}{N}$.

We briefly describe a linear time algorithm to sample uniformly from a DAG, and point the reader to (Hoens 2008; Bauwens, Kaczér, and De Lhoneux 2025) for a more thorough treatment of uniformly sampling tokenizations.

Let D be a DAG with v_s and v_f being the start and final states, respectively, and let $n(u)$ be the neighbors of a state u. Then we define a recurrence as $T_{v_f} = 1$ as the base-case (since it has no outgoing arcs) and $T_u = \sum_{v \in n(u)} T_v$ for the general case. Then, by computing T_{v_s} (the total number of paths in the DAG), we also know each T_u. Notice that, if we interpret a given u as the start state, then the set of reachable states from u also forms a DAG.

Now, we associate each of the $N = T_{v_s}$ distinct paths with an integer from 1 to N. Suppose we sample a value $z \sim \mathcal{U}(1, N)$ (the uniform distribution of integers between 1 and N, inclusive). Starting from the start state v_s, iterate over $n(v_s)$ in some predefined order and check which state must the z^{th} path pass through. We then travel to that state, and adjust the current path index (by subtracting the number of paths that lead to neighboring states that we checked before the state we ended up traveling to). We recurse on this until we reach the final state v_f and return the subword sequence corresponding to the sequence of arcs that we traversed.

Each local sample from state u can be done in constant time since the maximum out-degree at a state is equal to the length of the longest token in the vocabulary, which we consider to be a constant. The total runtime for a single sample is therefore $O(n)$.

We can use this to sample paths uniformly without replacement by instead sampling without replacement from $\mathcal{U}(1, N)$ and then extracting each path.

Length-Constrained Sampling

Uniformly from the set of all tokenizations tends to produce very long tokenizations (that is, the final tokenization tends to consist primarily of short tokens). On the other hand, most tokenizers are trained with the implicit objective of *compressing* the sequence length (Gallé 2019). Thus, we hypothesize that a perfectly uniform sampling will produce primarily low-probability sequences; a thought that has been considered before as well by Bauwens, Kaczér, and De Lhoneux (2025), who propose a similar, but biased sampling procedure designed to induce shorter tokenizations which should be more meaningful.

Instead, based on our empirical findings from Section 5, we wish to simply restrict the samples to only those with a specified maximum length. Consider an automaton that accepts the language $\Gamma^{\leq \ell}$, the set of all subword sequences of length at most ℓ made up of tokens from Γ. Then $\mathcal{A}_{\mathbf{s}}^{sub} \cap \Gamma^{\leq \ell}$ is another tokenization lattice which only accepts sequences of length at most ℓ. Since this automaton is also a DAG, we can sample uniformly from it exactly as in Section 6.

Enumerating Around the Canonical Sequence

As described in Section 5, we expect that much of the non-canonical tokenization mass is carried by tokenizations that are close to the canonical tokenization. We can easily enumerate a subset of close-to-canonical tokenizations. Let $t_1 t_2 \dots t_n$ be the canonical tokenization sequence and let $\textsc{decompose}(t)$ be the set of tokenization sequences $t'_1 t'_2 \in \Gamma^2$ such that $T^{-1}(t) = T^{-1}(t'_1 t'_2)$. Then, we can form the full set of "off-by-one" tokenizations by building the set of sequences where we select one token t_i from the canonical sequence and replace it with one of the sequences from $\textsc{decompose}(t_i)$. Let m be the length of the longest token in Γ. Then, we can fully enumerate this set of "off-by-one" tokenizations, as it contains only $O(mn)$ distinct sequences. Since m is often very small (e.g., for LLAMA2, $m = 16$), this enumeration is cheap.

7 Our Marginal Estimation Algorithm

We now have all the components for our marginal estimation algorithm. Instead of using importance sampling, we enumerate sequences that are likely to be high-probability, and then sample uniformly without replacement from the rest. This produces a lower bound on the marginal that monotonically improves with an increased sample size.

Algorithm 1 contains our lattice sampler pseudocode. We produce a sample of non-canonical subword sequences that are to be scored by the language model. We seed the list with the set of "off-by-one" tokenizations before uniformly sampling (without replacement) sequences of length at most ℓ from the remaining lattice. This results in a large set of unique sequences that are likely to have high probability.

We score all sequences at once using the language model and report the output as our marginal probability estimate. Each of the steps in Algorithm 1 are computationally inexpensive (especially relative to the time it takes to generate a sequence with an LLM) and scoring them is also fast (Table 3), so overall this sampling and scoring procedure has a significant runtime advantage over traditional proxy importance sampling.

As an example, in our implementation sampling $N = 10000$ paths without replacement from a lattice for a sentence with 300 letters and the GEMMA3 tokenizer takes slightly less than 1 second on a consumer CPU. Thus, this time is negligible compared to the time required to generate the same number of sequences using LLM decoding.

8 Experiments

We conducted two downstream experiments, Q&A and translation, to determine if 1) non-canonical marginalization is an effective inference strategy and 2) our method produces better marginal estimates with less time. For each experiment setting, we used a variety of language models (the LLAMA2 (Touvron et al. 2023) and GEMMA3 (Gemma Team et al. 2025) families with various parameter counts). We used the same prompts and generation hyperparameters

Algorithm 1: Our Lattice Sampler

Input: Surface Form $\mathbf{s}$, Vocabulary Γ, Samples k, Max Length ℓ
Returns: k distinct sequences from Γ^*

Let t be the canonical tokenization of $\mathbf{s}$
Let $\mathcal{O}_\mathbf{s}$ be the set of off-by-one tokenizations of t
Let $\mathcal{A}_\mathbf{s}^{sub} = \textsc{Proj}(\mathcal{A}_s \circ \mathcal{T}_\Gamma^*)$ be the lattice for $\mathbf{s}$
Compute $\mathcal{A}_{final} = \mathcal{A}_\mathbf{s}^{sub} \cap \overline{\mathcal{O}_\mathbf{s}} \cap \Gamma^{\leq \ell}$
Let N be the number of paths in $\mathcal{A}_{final}$
if $k - |\mathcal{O}_\mathbf{s}| > 0$ **then**
    Let $I \sim \mathcal{U}(1, N)$ be a set of $k - |\mathcal{O}_\mathbf{s}|$ distinct values
    for $i \in I$ **do**
        Add path π_i from $\mathcal{A}_{final}$ to $\mathcal{O}_\mathbf{s}$
    end for
end if
return $\mathcal{O}_\mathbf{s}$

(see supplementary material) and only varied the marginalization algorithm. All of our models used the base HUGGINGFACE inference engine (Wolf et al. 2020) with a custom constrained logit manipulator layer on top of it for importance sampling. We used a cluster of NVIDIA H100 GPUs for inference (we used a single GPU when possible for small models).

To isolate only the effect of the marginal and the marginalization algorithm, we did not use any decoding strategies like RAG, few-shot prompting, or chain of thought. Note that our goal is not to achieve state-of-the-art performance, but to measure the relative benefit of lattice sampling vs proxy importance sampling as alternative inference strategies to canonical tokenization scoring

Q&A Experiments

We used three standard Q&A datasets: OPENBOOKQA, a basic Q&A understanding dataset (Mihaylov et al. 2018); MEDMCQA, a medical Q&A dataset (Pal, Umapathi, and Sankarasubbu 2022); and the AI2 REASONING CHALLENGE (ARC) (Clark et al. 2018). We selected a random set of 250 questions from each. For each example, we computed the probability of the canonical sequence for each answer and estimated the non-canonical marginal for each answer using proxy importance sampling and our lattice-based estimate using 1000 samples for importance sampling and 1000 tokenizations (including the set of off-by-one tokenizations) for lattice sampling, derived from Algorithm 1. We selected the answer with the highest probability and measured the accuracy of each model.

Table 2 shows the results for each model and dataset. In nearly every setting, the canonical tokenization estimate performs the best. Curiously, both marginalization strategies beat the canonical tokenization for MEDMCQA with both LLAMA models. Lattice sampling also slightly outperforms canonical in OPENBOOK for GEMMA3-4B. Otherwise, canonical is by far the most accurate.

Another interesting finding is that in every case, LLAMA importance sampling outperforms lattice sampling, but, conversely, in every case, GEMMA lattice sampling outperforms importance sampling. The main difference is that the cases where importance sampling is better, the gap is small, but where lattice sampling is better, the gap is generally large.

Model	Dataset	Canon	Proxy	Lattice	Speedup
Llama2	Book	49.8	**48.1**	47.3	3.8×
7B	Med	29.0	**31.7**	31.3	4.8×
	Arc	51.5	**49.0**	47.1	4.5×
	Book	63.7	**61.6**	59.6	8.4×
13B	Med	36.6	**40.7**	39.0	8.6×
	Arc	68.6	**65.3**	61.9	5.4×
Gemma3	Book	38.8	32.4	**42.4**	3.3×
1B	Med	32.4	29.8	**31.9**	2.4×
	Arc	47.7	36.9	**38.2**	1.4×
	Book	67.8	32.2	**52.9**	6.2×
4B	Med	46.4	29.9	**37.3**	7.6×
	Arc	78.3	39.5	**61.5**	7.5×
	Book	84.0	46.0	**67.5**	36.5×
27B	Med	67.9	43.4	**51.9**	30.2×
	Arc	93.4	52.8	**81.1**	22.9×

Table 2: Q&A accuracy results (percentage of correctly answered questions) for each model family, size, and dataset, with marginalization estimates. Lattice sampling often outperforms proxy importance sampling while using the same number of samples and substantially less inference time. **Bolded** numbers indicate which of the marginal estimators (PROXY vs. LATTICE) had higher accuracy, since CANON was nearly always the best. SPEEDUP is the relative speed increase for marginalization when using lattice sampling instead of importance sampling.

We hypothesize that, since the answers in the dataset are very short sentences, the space of tokenizations is not that large. However, while importance sampling often resamples the same small set of tokenizations, lattice sampling is able to explore a much larger set of possible tokenizations, which produces a much more accurate estimate.

The speedup column shows that even with the relatively short sequence lengths for this dataset, lattice sampling is significantly faster (more than $30x$ for larger models) while still providing accurate marginal estimates.

These results support the claim that marginalization can carry useful signals, and strongly support our claim that our lattice sampler produces better marginal estimates at a fraction of the cost.

Translation Experiments

We additionally wanted to experiment on a more free form task which would have longer outputs compared to Q&A. We used the GEMMA3 model family to do zero-shot English→German and English→Czech translation. We used 150 sentences for each language pair from the WMT24++ translation dataset (Deutsch et al. 2025) for the datasets.

We first used the model to generate $k = 4$ unique transla-

EN→DE				
MODEL	CANON	PROXY	LATTICE	SPEEDUP
GEMMA3-1B	**73.0**	72.7	**73.0**	2.1×
GEMMA3-4B	80.8	**81.3**	81.2	2.9×
GEMMA3-27B	84.0	**84.3**	84.0	9.8×
EN→CS				
MODEL	CANON	PROXY	LATTICE	SPEEDUP
GEMMA3-1B	65.8	66.1	**66.8**	2.2×
GEMMA3-4B	**86.3**	86.2	**86.3**	3.1×
GEMMA3-27B	**88.3**	88.1	88.2	10.0×

Table 3: Translation results for our zero-shot translation pipeline. Each block is a language direction; rows are model variants, and columns are marginal approximation methods. **Bolded** numbers indicate the best of proxy vs. lattice. SPEEDUP is the relative speed increase for marginalization when using lattice sampling instead of importance sampling.

tions of the input sentence. We then computed the canonical tokenization probability and the non-canonical marginal for each translation and chose the highest one for our output. The resulting translated corpus was scored with COMET_{22}.

We find that all three sampling strategies perform well, with no clear winner among them across model sizes or language pairs. However, like Q&A, lattice sampling was significantly faster than importance sampling.

This again supports our main claims: non-canonical marginalization encodes a useful signal and our marginal approximation method provides more accurate marginal estimates than proxy sampling at a much lower runtime cost.

Intrinsic Experiment: Proxy Importance Sampling Underestimates the Marginal

In Section 4, we noted that our lattice sampler provides a strict lower-bound on the marginal, as it generates and scores sequences without replacement. We reanalyze the marginal estimates from both the Q&A and translation datasets to determine how often the importance sampler estimated the marginal to be less than what the lattice sampler produced. Since they used the same number of samples, these cases would be where the importance sampler verifiably underestimated the marginal. Note that the lattice sampler also, by definition, underestimates the marginal but if it is higher than the importance sampler estimate, then it is guaranteed to be a better estimate of the marginal.

Concretely, for each example in a dataset and for each of its possible choices (all of the multiple choice options for Q&A and the each of k candidate translations for the translation tasks; not just the highest scoring ones), we compared the marginal estimates under each sampling scheme. Table 4 gives the results for each dataset.

In every dataset and model paring, proxy importance sampling estimated a value that was lower than the lattice sampler more than 50% of the times. In the worse case, for ARC and the smaller GEMMA3 models, it was as bad as 82% of cases being underestimated. This is despite the lattice sampler and the proxy importance sampler using the same number of samples.

MODEL	Proxy Sampling Underestimation Ratio				
	BOOK	MED	ARC	En→De	En→Cs
LLAMA2-7B	72.3	76.7	74.2	-	-
LLAMA2-13B	74.8	69.1	72.2	-	-
GEMMA3-1B	76.3	73.3	82.8	52.2	62.3
GEMMA3-4B	76.0	78.8	82.8	72.4	58.3
GEMMA3-27B	76.4	78.5	78.1	72.4	74.0

Table 4: The percentage of examples (answers for Q&A and candidate translations) where proxy importance sampling underestimated the true marginal. This is measured by checking if importance sampling's estimate was less than lattice sampling's estimate, which is guaranteed to be a lower-bound on the true marginal. **NB:** We only used GEMMA3 for translation.

Thus, we see that importance sampling consistently underestimates the true marginal compared to lattice sampling, meaning that lattice sampling produces estimates that are more accurate but also substantially faster to compute.

9 Conclusion

Motivated by approximating the marginal distribution for LLMs but avoiding the expensive LLM generation step required by importance sampling, we presented a *decoding-free* approximate LLM marginalization algorithm based on the tokenization lattice — the automaton that succinctly encodes all possible tokenizations of an input text. Our method is tokenizer-agnostic, extremely fast, and provides provable lower bounds on the marginal via sampling without replacement. We entirely side-step the generation process and only require the LLM to *score* each sampled tokenization, which is comparatively cheap.

On two downstream tasks, we demonstrate that non-canonical marginalization is a viable inference strategy compared to only scoring the canonical tokenization of a sequence. Additionally, we find that our lattice sampler matches or beats the expensive LLM importance sampler using the same number of samples but with large runtime gains (up to $> 30x$ speedups). Finally, we show that the proxy importance sampler verifiably underestimates the true marginal in the majority of cases in our downstream tasks.

Thus, our method provides a cheap and powerful alternative to LLM importance sampling for marginalization.

Limitations

Our primary limitations were nearly all due to the large computational cost involved in approximating marginals with LLMs via importance sampling. For example, we used smaller subsets of datasets and did not experiment with very large (>70b parameter) LLMs or techniques like

prompt-tuning, temperature tuning, or speculative decoding all due to the prohibitively large computational costs involved. However, our goal is not to demonstrate state-of-the-art performance, but rather to show that our method is comparable with importance sampling, but faster.

We also only dealt with open source LLMs, as we required direct logit access in order to produce the constrained LLM for importance sampling.

References

Allauzen, C.; and Mohri, M. 2009. N-Way Composition of Weighted Finite-State Transducers. *International Journal of Foundations of Computer Science*, 20(04): 613–627.

Bauwens, T.; Kaczér, D.; and De Lhoneux, M. 2025. GRaMPa: Subword Regularisation by Skewing Uniform Segmentation Distributions with an Efficient Path-counting Markov Model. In Che, W.; Nabende, J.; Shutova, E.; and Pilehvar, M. T., eds., *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 24228–24257. Vienna, Austria: Association for Computational Linguistics. ISBN 979-8-89176-251-0.

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D. M.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Cao, K.; and Rimell, L. 2021. You should evaluate your language model on marginal likelihood over tokenisations. In Moens, M.-F.; Huang, X.; Specia, L.; and Yih, S. W.-t., eds., *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2104–2114. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics.

Chirkova, N.; Kruszewski, G.; Rozen, J.; and Dymetman, M. 2023. Should you marginalize over possible tokenizations? In Rogers, A.; Boyd-Graber, J.; and Okazaki, N., eds., *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 1–12. Toronto, Canada: Association for Computational Linguistics.

Clark, P.; Cowhey, I.; Etzioni, O.; Khot, T.; Sabharwal, A.; Schoenick, C.; and Tafjord, O. 2018. Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge. *arXiv:1803.05457v1*.

Cognetta, M.; and Okazaki, N. 2024. Tokenization as Finite-State Transduction. arXiv:2410.15696.

Cognetta, M.; Zouhar, V.; and Okazaki, N. 2024. Distributional Properties of Subword Regularization. In Al-Onaizan, Y.; Bansal, M.; and Chen, Y.-N., eds., *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 10753–10763. Miami, Florida, USA: Association for Computational Linguistics.

Deutsch, D.; Briakou, E.; Caswell, I.; Finkelstein, M.; Galor, R.; Juraska, J.; Kovacs, G.; Lui, A.; Rei, R.; Riesa, J.; Rijhwani, S.; Riley, P.; Salesky, E.; Trabelsi, F.; Winkler, S.; Zhang, B.; and Freitag, M. 2025. WMT24++: Expanding the Language Coverage of WMT24 to 55 Languages & Dialects. arXiv:2502.12404.

Edman, L.; Schmid, H.; and Fraser, A. 2024. CUTE: Measuring LLMs' Understanding of Their Tokens. In Al-Onaizan, Y.; Bansal, M.; and Chen, Y.-N., eds., *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 3017–3026. Miami, Florida, USA: Association for Computational Linguistics.

Gallé, M. 2019. Investigating the Effectiveness of BPE: The Power of Shorter Sequences. In Inui, K.; Jiang, J.; Ng, V.; and Wan, X., eds., *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 1375–1381. Hong Kong, China: Association for Computational Linguistics.

Geh, R.; Zhang, H.; Ahmed, K.; Wang, B.; and Van Den Broeck, G. 2024. Where is the signal in tokenization space? In Al-Onaizan, Y.; Bansal, M.; and Chen, Y.-N., eds., *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 3966–3979. Miami, Florida, USA: Association for Computational Linguistics.

Gemma Team; Kamath, A.; Ferret, J.; Pathak, S.; Vieillard, N.; Merhej, R.; Perrin, S.; Matejovicova, T.; Ramé, A.; Rivière, M.; Rouillard, L.; Mesnard, T.; Cideron, G.; bastien Grill, J.; Ramos, S.; Yvinec, E.; Casbon, M.; Pot, E.; Penchev, I.; Liu, G.; Visin, F.; Kenealy, K.; Beyer, L.; Zhai, X.; Tsitsulin, A.; Busa-Fekete, R.; Feng, A.; Sachdeva, N.; Coleman, B.; Gao, Y.; Mustafa, B.; Barr, I.; Parisotto, E.; Tian, D.; Eyal, M.; Cherry, C.; Peter, J.-T.; Sinopalnikov, D.; Bhupatiraju, S.; Agarwal, R.; Kazemi, M.; Malkin, D.; Kumar, R.; Vilar, D.; Brusilovsky, I.; Luo, J.; Steiner, A.; Friesen, A.; Sharma, A.; Sharma, A.; Gilady, A. M.; Goedeckemeyer, A.; Saade, A.; Feng, A.; Kolesnikov, A.; Bendebury, A.; Abdagic, A.; Vadi, A.; György, A.; Pinto, A. S.; Das, A.; Bapna, A.; Miech, A.; Yang, A.; Paterson, A.; Shenoy, A.; Chakrabarti, A.; Piot, B.; Wu, B.; Shahriari, B.; Petrini, B.; Chen, C.; Lan, C. L.; Choquette-Choo, C. A.; Carey, C.; Brick, C.; Deutsch, D.; Eisenbud, D.; Cattle, D.; Cheng, D.; Paparas, D.; Sreepathihalli, D. S.; Reid, D.; Tran, D.; Zelle, D.; Noland, E.; Huizenga, E.; Kharitonov, E.; Liu, F.; Amirkhanyan, G.; Cameron, G.; Hashemi, H.; Klimczak-Plucińska, H.; Singh, H.; Mehta, H.; Lehri, H. T.; Hazimeh, H.; Ballantyne, I.; Szpektor, I.; Nardini, I.; Pouget-Abadie, J.; Chan, J.; Stanton, J.; Wieting, J.; Lai, J.; Orbay, J.; Fernandez, J.; Newlan, J.; yeong Ji, J.; Singh, J.; Black, K.; Yu, K.; Hui, K.; Vodrahalli, K.; Greff, K.; Qiu, L.; Valentine, M.; Coelho, M.; Ritter, M.; Hoffman, M.; Watson, M.; Chaturvedi, M.; Moynihan, M.; Ma, M.; Babar, N.; Noy, N.; Byrd, N.; Roy, N.; Momchev, N.; Chauhan, N.; Sachdeva, N.; Bunyan, O.; Botarda, P.; Caron, P.; Rubenstein, P. K.; Culliton, P.; Schmid, P.; Sessa, P. G.;

Xu, P.; Stanczyk, P.; Tafti, P.; Shivanna, R.; Wu, R.; Pan, R.; Rokni, R.; Willoughby, R.; Vallu, R.; Mullins, R.; Jerome, S.; Smoot, S.; Girgin, S.; Iqbal, S.; Reddy, S.; Sheth, S.; Põder, S.; Bhatnagar, S.; Panyam, S. R.; Eiger, S.; Zhang, S.; Liu, T.; Yacovone, T.; Liechty, T.; Kalra, U.; Evci, U.; Misra, V.; Roseberry, V.; Feinberg, V.; Kolesnikov, V.; Han, W.; Kwon, W.; Chen, X.; Chow, Y.; Zhu, Y.; Wei, Z.; Egyed, Z.; Cotruta, V.; Giang, M.; Kirk, P.; Rao, A.; Black, K.; Babar, N.; Lo, J.; Moreira, E.; Martins, L. G.; Sanseviero, O.; Gonzalez, L.; Gleicher, Z.; Warkentin, T.; Mirrokni, V.; Senter, E.; Collins, E.; Barral, J.; Ghahramani, Z.; Hadsell, R.; Matias, Y.; Sculley, D.; Petrov, S.; Fiedel, N.; Shazeer, N.; Vinyals, O.; Dean, J.; Hassabis, D.; Kavukcuoglu, K.; Farabet, C.; Buchatskaya, E.; Alayrac, J.-B.; Anil, R.; Dmitry; Lepikhin; Borgeaud, S.; Bachem, O.; Joulin, A.; Andreev, A.; Hardin, C.; Dadashi, R.; and Hussenot, L. 2025. Gemma 3 Technical Report. arXiv:2503.19786.

Hiraoka, T. 2022. MaxMatch-Dropout: Subword Regularization for WordPiece. In Calzolari, N.; Huang, C.-R.; Kim, H.; Pustejovsky, J.; Wanner, L.; Choi, K.-S.; Ryu, P.-M.; Chen, H.-H.; Donatelli, L.; Ji, H.; Kurohashi, S.; Paggio, P.; Xue, N.; Kim, S.; Hahm, Y.; He, Z.; Lee, T. K.; Santus, E.; Bond, F.; and Na, S.-H., eds., *Proceedings of the 29th International Conference on Computational Linguistics*, 4864–4872. Gyeongju, Republic of Korea: International Committee on Computational Linguistics.

Hiraoka, T.; and Okazaki, N. 2024. Knowledge of Pretrained Language Models on Surface Information of Tokens. *CoRR*, abs/2402.09808.

Hiraoka, T.; Shindo, H.; and Matsumoto, Y. 2019. Stochastic Tokenization with a Language Model for Neural Text Classification. In Korhonen, A.; Traum, D.; and Màrquez, L., eds., *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 1620–1629. Florence, Italy: Association for Computational Linguistics.

Hoens, T. R. 2008. *Counting and sampling paths in graphs*. M.s. thesis, Rochester Institute of Technology, Rochester, New York.

Kaushal, A.; and Mahowald, K. 2022. What do tokens know about their characters and how do they know it? In Carpuat, M.; de Marneffe, M.-C.; and Meza Ruiz, I. V., eds., *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2487–2507. Seattle, United States: Association for Computational Linguistics.

Koo, T.; Liu, F.; and He, L. 2024. Automata-based constraints for language model decoding. In *First Conference on Language Modeling*.

Kudo, T. 2018. Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates. In Gurevych, I.; and Miyao, Y., eds., *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 66–75. Melbourne, Australia: Association for Computational Linguistics.

Mielke, S. J.; Alyafeai, Z.; Salesky, E.; Raffel, C.; Dey, M.; Gallé, M.; Raja, A.; Si, C.; Lee, W. Y.; Sagot, B.; and Tan, S. 2021. Between words and characters: A Brief History of Open-Vocabulary Modeling and Tokenization in NLP. arXiv:2112.10508.

Mihaylov, T.; Clark, P.; Khot, T.; and Sabharwal, A. 2018. Can a Suit of Armor Conduct Electricity? A New Dataset for Open Book Question Answering. In *EMNLP*.

Murray, K.; and Chiang, D. 2018. Correcting Length Bias in Neural Machine Translation. In Bojar, O.; Chatterjee, R.; Federmann, C.; Fishel, M.; Graham, Y.; Haddow, B.; Huck, M.; Yepes, A. J.; Koehn, P.; Monz, C.; Negri, M.; Névéol, A.; Neves, M.; Post, M.; Specia, L.; Turchi, M.; and Verspoor, K., eds., *Proceedings of the Third Conference on Machine Translation: Research Papers*, 212–223. Brussels, Belgium: Association for Computational Linguistics.

Pal, A.; Umapathi, L. K.; and Sankarasubbu, M. 2022. MedMCQA: A Large-scale Multi-Subject Multi-Choice Dataset for Medical domain Question Answering. In Flores, G.; Chen, G. H.; Pollard, T.; Ho, J. C.; and Naumann, T., eds., *Proceedings of the Conference on Health, Inference, and Learning*, volume 174 of *Proceedings of Machine Learning Research*, 248–260. PMLR.

Provilkov, I.; Emelianenko, D.; and Voita, E. 2020. BPE-Dropout: Simple and Effective Subword Regularization. In Jurafsky, D.; Chai, J.; Schluter, N.; and Tetreault, J., eds., *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 1882–1892. Online: Association for Computational Linguistics.

Qwen; :; Yang, A.; Yang, B.; Zhang, B.; Hui, B.; Zheng, B.; Yu, B.; Li, C.; Liu, D.; Huang, F.; Wei, H.; Lin, H.; Yang, J.; Tu, J.; Zhang, J.; Yang, J.; Yang, J.; Zhou, J.; Lin, J.; Dang, K.; Lu, K.; Bao, K.; Yang, K.; Yu, L.; Li, M.; Xue, M.; Zhang, P.; Zhu, Q.; Men, R.; Lin, R.; Li, T.; Tang, T.; Xia, T.; Ren, X.; Ren, X.; Fan, Y.; Su, Y.; Zhang, Y.; Wan, Y.; Liu, Y.; Cui, Z.; Zhang, Z.; and Qiu, Z. 2025. Qwen2.5 Technical Report. arXiv:2412.15115.

Sennrich, R.; Haddow, B.; and Birch, A. 2016. Neural Machine Translation of Rare Words with Subword Units. In Erk, K.; and Smith, N. A., eds., *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1715–1725. Berlin, Germany: Association for Computational Linguistics.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; Bikel, D.; Blecher, L.; Ferrer, C. C.; Chen, M.; Cucurull, G.; Esiobu, D.; Fernandes, J.; Fu, J.; Fu, W.; Fuller, B.; Gao, C.; Goswami, V.; Goyal, N.; Hartshorn, A.; Hosseini, S.; Hou, R.; Inan, H.; Kardas, M.; Kerkez, V.; Khabsa, M.; Kloumann, I.; Korenev, A.; Koura, P. S.; Lachaux, M.-A.; Lavril, T.; Lee, J.; Liskovich, D.; Lu, Y.; Mao, Y.; Martinet, X.; Mihaylov, T.; Mishra, P.; Molybog, I.; Nie, Y.; Poulton, A.; Reizenstein, J.; Rungta, R.; Saladi, K.; Schelten, A.; Silva, R.; Smith, E. M.; Subramanian, R.; Tan, X. E.; Tang, B.; Taylor, R.; Williams, A.; Kuan, J. X.; Xu, P.; Yan, Z.; Zarov, I.; Zhang, Y.; Fan, A.; Kambadur, M.; Narang, S.; Rodriguez, A.; Stojnic, R.; Edunov, S.; and Scialom, T. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv:2307.09288.

Vieira, T.; Liu, T.; Pasti, C.; Emara, Y.; DuSell, B.; LeBrun, B.; Giulianelli, M.; Gastaldi, J. L.; Terilla, J.; O'Donnell, T. J.; and Cotterell, R. 2025. Language Models over Canonical Byte-Pair Encodings. In *Forty-second International Conference on Machine Learning*.

Welleck, S.; Bertsch, A.; Finlayson, M.; Schoelkopf, H.; Xie, A.; Neubig, G.; Kulikov, I.; and Harchaoui, Z. 2024. From Decoding to Meta-Generation: Inference-time Algorithms for Large Language Models. *Trans. Mach. Learn. Res.*, 2024.

Willard, B. T.; and Louf, R. 2023. Efficient Guided Generation for Large Language Models. arXiv:2307.09702.

Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; Davison, J.; Shleifer, S.; von Platen, P.; Ma, C.; Jernite, Y.; Plu, J.; Xu, C.; Scao, T. L.; Gugger, S.; Drame, M.; Lhoest, Q.; and Rush, A. M. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 38–45. Online: Association for Computational Linguistics.

Zheng, B. S.; Liu, A.; Ahia, O.; Hayase, J.; Choi, Y.; and Smith, N. A. 2025. Broken Tokens? Your Language Model can Secretly Handle Non-Canonical Tokenizations. In *Tokenization Workshop*.