

Toward a Theory of Tokenization in LLMs

Nived Rajaraman, Jiantao Jiao, Kannan Ramchandran ^{*†}

April 11, 2025

Abstract

While there has been a large body of research attempting to circumvent tokenization for language modeling (Clark et al., 2022; Xue et al., 2022), the current consensus is that it is a necessary initial step for designing state-of-the-art performant language models. In this paper, we investigate tokenization from a theoretical point of view by studying the behavior of transformers on simple data generating processes. When trained on data drawn from certain simple k^{th} -order Markov processes for $k > 1$, transformers exhibit a surprising phenomenon - in the absence of tokenization, they empirically are incredibly slow or fail to learn the right distribution and predict characters according to a unigram model (Makkuva et al., 2024). With the addition of tokenization, however, we empirically observe that transformers break through this barrier and are able to model the probabilities of sequences drawn from the source near-optimally, achieving small cross-entropy loss. With this observation as starting point, we study the end-to-end cross-entropy loss achieved by transformers with and without tokenization. With the appropriate tokenization, we show that even the simplest unigram models (over tokens) learnt by transformers are able to model the probability of sequences drawn from k^{th} -order Markov sources near optimally. Our analysis provides a justification for the use of tokenization in practice through studying the behavior of transformers on Markovian data.

1 Introduction

The training of language models is typically not an end-to-end process. Language models are often composed of a “tokenizer”, which encodes a sequence of characters into a sequence of token ids, which map to substrings. The subsequent language modeling task is carried out by a neural network or transformer, which is pre-trained and fine-tuned on large datasets. The ideal goal is to jointly train the tokenizer and transformer with end-to-end accuracy as the objective. This is a challenging problem to solve efficiently, and thus, the tokenizer is generally adapted on a portion of the training dataset and frozen before the transformer is trained.

The simplest tokenizer encodes at the character or at the byte level (after encoding into UTF-8), such as is done in ByT5 (Xue et al., 2022) and CANINE (Clark et al., 2022). The maximum sequence

^{*}Nived Rajaraman is with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley. Jiantao Jiao is with the Department of Electrical Engineering and Computer Sciences and the Department of Statistics, University of California, Berkeley. Kannan Ramchandran is with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley. Email: {nived.rajaraman, jiantao, kannanr}@eecs.berkeley.edu

[†]This work was accepted to NeurIPS 2025 with a different title, “An Analysis of Tokenization: Transformers under Markov data” (cf. Rajaraman et al. (2024))

length that such models can process is also smaller for the same amount of compute. In practice, byte-level/character-level models perform worse for the reason that semantic relationships can be harder to capture at the character level (Libovicky et al., 2021; Itzhak and Levy, 2021) and for this reason, tokenizing at the subword level is used more commonly.

Though used most commonly, tokenization at the subword level often has sharp edges. Test sequences may contain rare tokens which were never seen in the training dataset. The presence of such tokens may induce undesirable behavior in the outputs of models (Rumbelow and Watkins, 2023; Kharitonov et al., 2021; Yu et al., 2021) and present an attack surface for bad actors. Moreover, tokenized models struggle on tasks that involve manipulation at the character level, such as spelling out words or reversing sentences. For similar reasons, LLMs with standard tokenizers also struggle to carry out basic arithmetic (Golkar et al., 2023). Despite this brittleness, tokenization is used in nearly all state-of-the-art LLM architectures.

Since tokenizers are usually trained in isolation, they do not directly optimize for extrinsic loss metrics such as the end-to-end perplexity or precision. A number of domain and task specific *intrinsic* objectives and evaluation metrics have been proposed for tokenization. The most commonly used intrinsic metric to compare tokenizers with the same dictionary size is the average compressed sequence length. Zouhar et al. (2023a) show that the Renyi efficiency of the tokenizer correlates well with the end-to-end BLEU for machine translation. Petrov et al. (2023) propose parity, which associates a higher score if the tokenizer parses sentences across different languages having the same meaning, into a similar number of tokens. Similarly, Gallé (2019) investigate various tokenizers for machine translation from the view of compression capacity and conclude that for two token vocabularies of the same size, the one that typically requires fewer tokens to cover a sentence typically achieves a better translation, a commonly used metric known as *fertility* (Rust et al., 2021; Scao et al., 2022; Ali et al., 2023). The validity of these proxy objectives in general, is established by experimental verification.

In this paper, we introduce a statistical formulation for tokenization for next-word-prediction. Taking a step back, rather than focusing on proxy evaluation metrics, which lead to an ever-changing goalpost, we focus on understanding the behavior of the end-to-end cross-entropy loss, $\mathcal{L}(\cdot)$. We consider a simplification of real world data generating processes and study the case where data sources are k^{th} -order Markov processes. Within this framework we can compare tokenizers against each other, and in the process capture several interesting phenomena. Our main results are as follows,

1. There are very simple k^{th} -order Markov processes such that in the absence of any tokenization, transformers trained on data drawn from this source are empirically observed to predict characters according to a unigram model. This phenomenon is observed under a wide variety of hyperparameter choices. This is problematic because unigram models such as that induced by the stationary distribution are poor at modeling Markovian data and suffer from a high cross-entropy loss. This phenomenon was also recently observed in Makkula et al. (2024).
2. When trained with tokenization, transformers are empirically observed to break through this barrier and are able to capture the probability of sequences under the Markov distribution near-optimally. In other words, in the presence of tokenization, transformers appear to achieve near-optimal cross-entropy loss. This phenomenon is observed with a multitude of tokenizers used commonly in practice. Even though the end-to-end model is near optimal, we observe that the behavior of the transformer is surprisingly the same in a qualitative sense - the model largely still predicts tokens according to a unigram distribution.
3. We analyze a toy tokenizer which adds all length- k sequences into the dictionary and show

that as dictionary size grows, unigram models trained on the tokens get better at modeling the probabilities of sequences drawn from Markov sources. We then theoretically prove that tokenizers used in practice, such as the LZW tokenizer (Zouhar et al., 2023a) and a variant of the BPE tokenizer (Gage, 1994; Sennrich et al., 2016) which are learnt from data also satisfy this property but require much smaller dictionaries to achieve any target cross-entropy loss.

In our framework, the most challenging hurdle and the biggest departure from previous work such as (Zouhar et al., 2023b) is the element of generalization - understanding how a tokenizer performs on new sequences that it was not trained on. This generalization turns out to be a delicate phenomenon - we show in Appendices D and E that there exist tokenizers which generalize poorly in the sense that they may compress the dataset they are trained on into a short sequence of tokens, but completely fail to generalize to new sequences. We also show that there exist dictionaries which generalize well (in the sense of having low cross-entropy loss) to new sequences under one encoding algorithm, but fail to generalize under another.

1.1 Related Work

Tokenization has a long history of empirical study in natural language processing. In the literature, a number of tokenizers have been developed for various domains such as math (Singh and Strouse, 2024), code (Zheng et al., 2023; Parr, 2013) and morphology-aware tokenizers for different languages like Japanese (Tolmachev et al., 2018; Den et al., 2007) and Arabic (Alyafeai et al., 2023) among many others. In modern LLMs, the most commonly used tokenizers are variants of BPE (Gage, 1994), Wordpiece (Schuster and Nakajima, 2012) and the Unigram tokenizer (Kudo, 2018) which learn a dictionary from data, rather than hard-coding language dependent rules. There has been a long line of work interpreting tokenization from various lenses (Grefenstette and Tapanainen, 1994; Palmer, 2000). Notably, Zouhar et al. (2023b) take the view that the popular BPE tokenizer (Wolff, 1975; Gage, 1994) can be viewed as a greedy algorithm for the problem of finding the sequence of “token merges” which minimizes the length of the resulting string and establish approximation guarantees for the algorithm. However, the aspect of generalization is missing here - when trained on one and evaluated on another dataset, these guarantees may no longer hold.

The theoretical study of transformers has also received much attention recently and we discuss the closest relatives to our work below. Edelman et al. (2024) study the learning trajectory of transformers trained on data drawn from 1st-order Markov chains. While the authors empirically observe that the models eventually learn to predict tokens correctly according to the Markov kernel, simplicity bias slows down the optimization - the models initially predict tokens according to a unigram model (in context unigrams), which delays learning the optimal solution. This phenomenon was also observed in Makkuvu et al. (2024) who prove that when transformers are trained on data generated from a simple 1st-order switching Markov processes, the optimization landscape contains a bad local minimum at the unigram model corresponding to the stationary distribution of the process. For k^{th} -order switching Markov processes, they observe that forcing the context window of the transformer to be small is the only hyperparameter which enables the model to learn the underlying Markov kernel. While a positive result, such a fix is unlikely to be used in practice. On the positive side, Nichani et al. (2024) study an in-context causal learning task that generalizes learning in-context bigrams for 1st-order Markov processes. The authors analyze the trajectory of gradient descent and show that in the special case of data sampled from a Markov chain, transformers learn an induction head which learns to predict according to in-context bigram counts.

Notation. All logarithms are base e , unless specified otherwise. The Shannon entropy $H(X)$ of a categorical random variable X is $-\sum_{x \in \text{supp}(X)} p(x) \log p(x)$. $H_{\text{BER}}(p)$ captures the entropy of a Bernoulli random variable with parameter p . The notation $O_{p,q,r}(f(n))$ (likewise $\Omega_{\{\cdot\}}$ and $\Theta_{\{\cdot\}}$) indicate that the underlying constant depends polynomially on the parameters p, q and r . The notation $\tilde{O}(f(n))$ (likewise, $\tilde{\Theta}$ and $\tilde{\Omega}$) ignores polylog(n) terms. For a set S , $S^* = \cup_{k=1}^{\infty} S^k$, the set of all sequences with elements drawn from S . For a sequence \mathbf{t} , $\mathbf{t}_{i:j} = (\mathbf{t}_i, \mathbf{t}_{i+1}, \dots, \mathbf{t}_j)$ returns a slice of the sequence.

2 Formulation

We consider a setting where the learner’s objective is to learn a language model which models probabilities of sequences over an input alphabet \mathcal{A} . The data to be modeled is generated according to an unknown probability model $P : \mathcal{A}^* \rightarrow [0, 1]$ over strings. A tokenizer is a tuple $\mathcal{T} = (\text{Dict}, \text{DS}, \text{enc}(\cdot), \text{dec}(\cdot))$. Here Dict is a collection of tokens and DS is a data-structure which stores additional information about tokens; for instance, in BPE, every token is constructed from a pair of shorter tokens, and this additional information is stored in DS . If not explicitly instantiated, $\text{DS} = \emptyset$. The encoding function $\text{enc}(\cdot) : \mathcal{A}^* \rightarrow \text{Dict}^*$, which is implicitly a function of DS , maps strings of characters to a sequence of tokens, and likewise, the decoding function $\text{dec}(\cdot) : \text{Dict}^* \rightarrow \mathcal{A}^*$ maps a sequence of tokens to a string of characters. We assume that the tokenizer is “consistent”, namely, $\text{dec}(\text{enc}(\cdot))$ is the identity function.

We consider a setting where the learner has access to a training dataset which is a sequence of length n sampled from a data source¹. We study the likelihood maximization problem, where the objective of the learner is to learn an end to end model such that the cross-entropy loss is minimized. In the presence of tokenization, we have a model of the form $Q_{\text{end}} = Q \circ \text{enc}(\cdot)$ where Q is a joint distribution across sequences of tokens when the tokenizer corresponding to $\text{enc}(\cdot)$ is used. The cross-entropy loss, i.e. the log-perplexity, can be written down as,

$$\mathcal{L}_m(Q_{\text{end}}) \triangleq -\mathbb{E}[\log Q(\text{enc}(\mathbf{s}))], \quad (1)$$

with the objective to minimize it. Here, the expectation is over \mathbf{s} , a fresh test sequence of length m sampled from the data generating process. Fixing a tokenizer, let \mathcal{Q} denote a family of joint distributions over tokens (i.e. likelihood models). The objective then is to jointly design a tokenizer (with encoding function $\text{enc}(\cdot)$) and likelihood model $Q \in \mathcal{Q}$ such that the test loss $\mathcal{L}_m(Q \circ \text{enc}(\cdot))$ is small. In general, jointly optimizing over the tokenizer \mathcal{T} and the likelihood model Q is a chicken-and-egg problem. Optimizing over \mathcal{T} requires knowing the corresponding optimal likelihood model over its tokens, $Q^* \in \mathcal{Q}$. However, in order to learn such a model, we must have committed to a tokenizer upfront.

In this paper, the end-to-end task we consider is next token prediction, where perplexity minimization is a natural objective. In practice, other metrics may be more commonly employed depending on the domain, such as BLEU (Papineni et al., 2002) or ROUGE (Lin, 2004) for machine translation. The theoretical study of these metrics is left as future work.

2.1 Data generating process

In this paper, we consider a simplification of real-world data generating processes by considering the case where the data generating distribution is a k^{th} -order Markov process over characters. Studying the behavior of transformers trained on Markov data was the subject of the works of Makkav

¹This can be thought of as the concatenation of all the individual sequences in the training dataset.

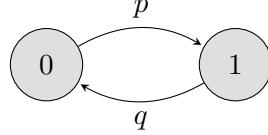


Figure 1: Switching process: The above Markov process describes the distribution of X_n conditioned on X_{n-1} . k^{th} -order extension of this Markov process: the conditional probability of X_n only depends on X_{n-k} and is given by the above process, with $\Pr(X_n = 1|X_{n-k} = 0) = p$ and $\Pr(X_n = 1|X_{n-k} = 1) = 1 - q$.

et al. (2024) and Edelman et al. (2024), where a number of interesting phenomena were unearthed. Surprisingly, when a transformer is trained on data from certain simple k^{th} -order Markov processes like the one considered in Figure 1, a very peculiar phenomenon occurs - the transformer seems to learn a unigram model over characters, and in particular one induced by the stationary distribution over characters, despite the data following a Markov process. For the switching Markov chain in Figure 1, Edelman et al. (2024) flesh out the behavior in more detail for the case $k = 1$ - the transformer has a critical point corresponding to a unigram model which estimates the probability of 0 and 1 in an in-context manner from the test sequence it sees. When trained for sufficiently long, the transformer eventually escapes this saddle point.

In the case of the switching process for $k \geq 2$, the situation is different. The transformer appears to be much slower to escape the stationary unigram model altogether regardless of the choice of a number of hyperparameters, including the number of feed-forward layers in the model, the embedding dimension, and the number of attention heads. In Figure 2a this is made clearer - the character-level transformer fails to improve its test loss beyond that of the best unigram model on the training data (dotted line) within the allotted budget. In general, given an arbitrary test sequence, the transformer learns to output the next character as 0 with probability equal to the empirical fraction of 0's observed in the test sequence. As we increase the training budget, the character-level transformer eventually also learns to represent the Markov source, but this takes significantly longer and is not plotted here. This phenomenon is not only true of the switching processes of Figure 1: we plot the validation loss of transformers trained on randomly generated higher-order Markov processes in Figures 3a and 3b and observe a similar trend.

Formally, for a dictionary Dict , the unigram family of models, $\mathcal{Q}_{1\text{-gram}}$, is defined as below: $Q \in \mathcal{Q}_{1\text{-gram}}$ associates the probability $Q(\mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{t}_j) = Q_\#(j) \prod_{i=1}^j Q_{\text{tok}}(\mathbf{t}_i)$ to the sequence of tokens $(\mathbf{t}_1, \dots, \mathbf{t}_j)$ for some measures $Q_\#$ and Q_{tok} supported on \mathbb{N} and Dict respectively. Empirically, transformers appear to learn an in-context unigram model when trained on data from the switching process in Figure 1 for $k \geq 2$. How bad can this be? It turns out that the gap between the cross-entropy of the best unigram model and that of the optimal model can be characterized precisely.

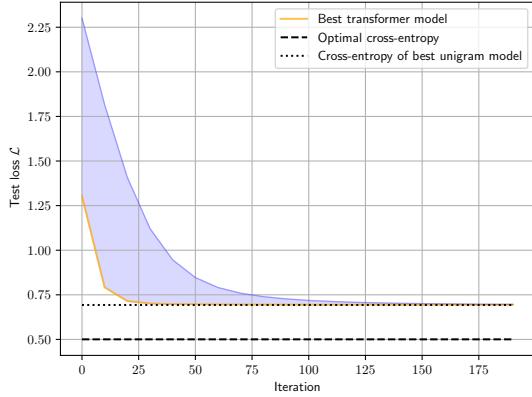
Theorem 2.1. Consider any ergodic data source with stationary distribution over characters π . The unconstrained optimal likelihood model achieves cross-entropy loss,

$$\min_Q \mathcal{L}_m(Q) = H(P)$$

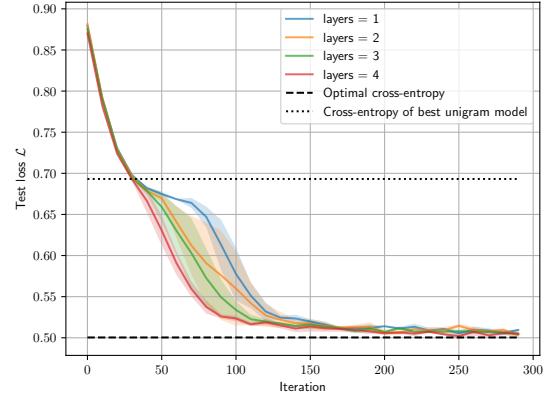
In contrast, the cross-entropy loss under any unigram model $Q \in \mathcal{Q}_{1\text{-gram}}$ is lower bounded by,

$$\mathcal{L}_m(Q) \geq mH(\pi)$$

The ratio of $H(P)$ and $mH(\pi)$ can be made arbitrarily large for the switching Markov chains in Figure 1 as the switching probabilities p and q approach 0 or 1. See Example A.1 for more details.

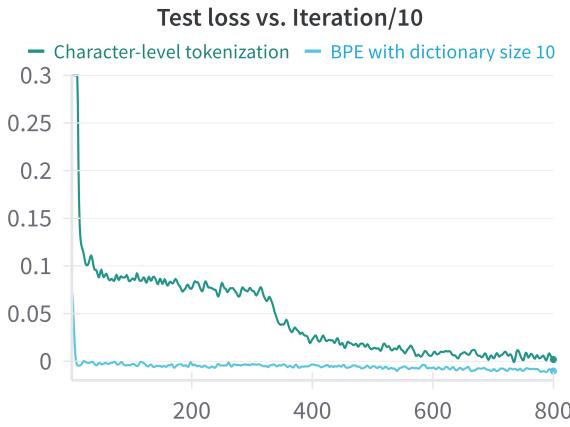


(a) The loss of the transformer fails to converge to the optimal cross-entropy loss (dashed line) within the allotted training budget, and gets stuck at the loss of the best i.i.d. model (dotted line). The shaded region captures how the test loss curves vary as hyperparameters (number of layers, embedding dimension etc.) are changed.

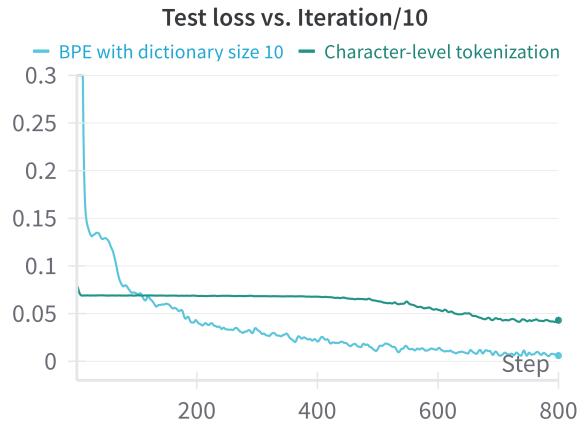


(b) In the presence of tokenization, the test loss of the model approaches the optimal bound (dashed line) well within the allotted training budget. It is worth noting that the models trained here are significantly smaller than those considered in Figure 2a, having up to $70\times$ fewer parameters and yet are able to achieve the optimal cross-entropy loss.

Figure 2: Transformers trained on the order-2 switching Markov process (Figure 1) with $p = q = 0.8$. On the left we have the model trained without tokenization and on the right the model uses BPE with a dictionary of size 10 learnt from data. The hyperparameter sweep is discussed in Appendix F (Table 3). Training for significantly longer enables the character-level model to break past this barrier (cf. Figures 3a and 3b).



(a) Order-2 Markov process



(b) Order-4 Markov process

Figure 3: One hyperparameter-tuned training run of a transformer trained on data drawn from a randomly sampled order- k Markov process P where $\Pr(X_n = \cdot | X_{n-1}, \dots, X_{n-k})$ is drawn from $\text{Dir}(\mathbf{1})$ prior. These models are trained for significantly longer ($\approx 8K$ iterations) compared to Figures 2a and 2b. Character-level tokenization results in extremely slow optimization, which worsens as the order grows.

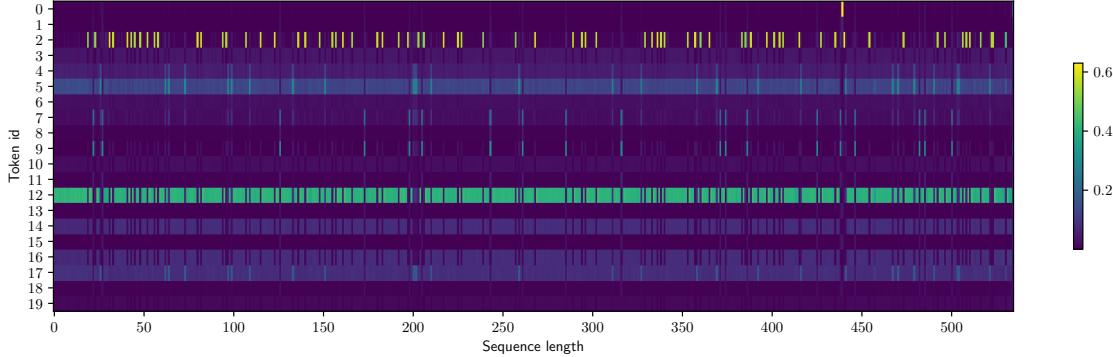


Figure 4: Token distribution returned by the transformer when fed in a sequence generated from the stochastic source, tokenized by a learnt BPE encoder with 20 tokens. A test sequence is generated from the stochastic source and encoded into a token sequence \mathbf{t} . Each narrow vertical column represents the distribution over next tokens returned by the transformer when the first x tokens of \mathbf{t} are fed into the model, where x is varied from 0 to the length of \mathbf{t} . For most values of x , the model appears to predict the same distribution over the next token.

Example 2.2. Consider the switching Markov process in Figure 1 on $\{0, 1\}$ with $p = q = 1 - \delta$. For this process, $\lim_{m \rightarrow \infty} \frac{1}{m} H(P) = H_{\text{Ber}}(\delta) = \delta \log(1/\delta) + (1 - \delta) \log(1/(1 - \delta))$, but $\pi = \{1/2, 1/2\}$ and so $H(\pi) = H_{\text{Ber}}(1/2) = \log(2)$. The ratio $\lim_{m \rightarrow \infty} \frac{mH(\pi)}{H(P)}$ goes to ∞ as $\delta \rightarrow 0$.

3 The role of tokenization

While transformers are a powerful class of models, it is concerning that they fail to learn very simple distributions such as k^{th} -order Markov processes under a wide choice of hyperparameters. Why do transformers work so well in practice if they can't learn Markovian data? It turns out that there is a simple missing ingredient in all the architectures considered so far: tokenization. The models are trained on raw character sequences drawn from the stochastic source. To understand the role of tokenization, we train the transformer on sequences generated from the stochastic source which are encoded into tokens by a learnt BPE tokenizer. The transformer operates on sequences of tokens, rather than sequences of characters. In Figure 2b we plot the results of this experiment - in the presence of tokenization, the cross-entropy loss of the end-to-end model approaches the optimal bound. Moreover, as we increase the order of the Markov process, the disparity persists: in Figures 3a and 3b, we see that the addition of tokenization enables the model to learn much more quickly

Let's peek into the model a bit more and understand its behavior. In Figure 4 we draw a test sequence from the stochastic source and tokenize it using a learnt BPE tokenizer with 20 tokens to generate a sequence \mathbf{t} with k tokens. We feed in $\mathbf{t}_{1:x}$ for each $x \in [k]$ into the model and observe the next-token distribution the model generates. For test sequences of length $k = 600$ tokens, we plot the distribution over next-tokens in Figure 4 as we vary the length of the prefix. Observe that the model learns to predict the a similar next-token distribution at almost all values of x , essentially ignoring the context it was shown. Thus the transformer learns what is essentially a unigram model.

The behavior of the transformer on the k^{th} -order switching source in Figure 1 with and without tokenization is essentially the same. In both cases, the model learns a unigram model over the tokens - in the absence of tokenization this unigram model is in fact the stationary distribution induced by

the source. If the transformer learns a unigram model in both cases, how come there is such a large gap in performance between the two? To understand this in more detail, we analyze a toy tokenizer in the next section.

4 Unigram models under tokenization

Let's consider a toy tokenizer which assigns all possible substrings of length r as tokens in the dictionary and study what happens when a unigram model is trained on the tokenized sequences. The total dictionary size $d = 2^r$. A sequence of characters is mapped to a sequence of tokens by simply chunking it into sequences of r characters which are replaced by the corresponding token index². The resulting stochastic process on the tokens is still Markovian, but over a state space of size 2^r . For any unigram model Q on the tokens, the cross-entropy loss can be written down as,

$$\mathcal{L}_m(Q \circ \text{enc}(\cdot)) = \mathbb{E} \left[\sum_{t \in \text{enc}(s)} \log(1/Q_{\text{tok}}(t)) \right],$$

where we ignore the contribution of $Q_\#(|\text{enc}(s)|)$ to the cross-entropy by simply choosing $Q_\# = \text{Unif}([m])$ and letting $\lim_{m \rightarrow \infty} \log(m)/m = 0$. For any token t , choose $Q_{\text{tok}}(t) = \pi(t_1) \prod_{i=1}^{r-1} P(t_{i+1}|t_i)$ as the stationary probability the underlying Markov process associates with t . Then,

$$\begin{aligned} \mathcal{L}_m(Q \circ \text{enc}(\cdot)) &= -\mathbb{E} \left[\log(P(s) + \sum_{i=0}^{m/k-1} \log \left(\frac{\pi(s_{ki+1})}{P(s_{ki+1}|s_{ki})} \right) \right] \\ &\stackrel{(i)}{\approx} H(P) + \frac{1}{k} (mH(\pi) - H(P)) \\ &= H(P) \left(1 - \frac{1}{\log_2(d)} \right) + \frac{mH(\pi)}{\log_2(d)}. \end{aligned} \quad (2)$$

In (i) the approximation is derived from the fact that as m grows large, $\frac{1}{m} \sum_{i=0}^{m/k} \log(P(s_{ki+\ell+1}|s_{ki+\ell}))$ approaches $\frac{H(P)}{k}$. With $d = 2$ (i.e., $r = 1$), we recover the performance of the character tokenizer in Theorem 2.1. An immediate implication of this simple calculation is that there is a unigram model which is nearly optimal as the dictionary size grows to ∞ .

While this toy tokenizer allows us to glean this intuition behind why tokenization allows unigram models to be near-optimal, there are some obvious issues. One, the tokenizer does not adapt to the distribution of the data. Indeed, for the switching Markov source in Figure 1, as $p = q = \delta \rightarrow 0$, the source contains increasingly longer sequences of contiguous 0's and 1's. In this case, it makes sense to have a dictionary containing such sequences, rather than all possible length- r sequences, many of which would be seen very few times (if at all) in a test sequence. At a more technical level, in eq. (2), to get to a cross-entropy loss of $2H(P)$, the size of the dictionary required by the toy tokenizer is $e^{mH(\pi)/H(P)}$. As discussed in Example A.1 for the switching Markov process with $p = q = \delta$, this dictionary size can be extremely large and scales exponentially (in $1/\delta$) as $e^{1/\delta \log(1/\delta)}$ when δ is small. In general, on stochastic sources on a much larger alphabet, such as English/ASCII, this toy tokenizer would result in a prohibitively large dictionary.

²The last few characters which do not add up to r in total are tokenized as characters. These boundary effects will not matter as the test sequences grow in length

A large dictionary itself is not a major problem per se, trading off the width of the model for the need for larger dimensional embeddings. However, larger dictionaries are usually correlated with the presence of rare tokens which appear infrequently at training time. This presents a problem in practice - a lot more data is often required to see enough examples of such tokens to learn good embeddings for them. More importantly, in the absence of this volume of data, rare tokens present an attack surface to elicit undesirable behavior in the model (Rumbelow and Watkins, 2023). In practice, this issue present with the toy tokenizer is, to an extent, resolved by using tokenization algorithms such as BPE or Wordpiece, which learn dictionaries from data. In the process, they are able to avoid learning extremely rare tokens, by enforcing a lower bound on the number of their occurrences in the training data to be allocated as a token. Moreover, by minimizing the number of such rare tokens, the model is able to utilize its token budget in a more efficient manner.

We now introduce the main theoretical result of this paper, showing that with the appropriate tokenization algorithm with a token budget of d , a unigram model is not only asymptotically able to achieve the optimal cross-entropy loss, but also requires far smaller dictionaries to match the performance of the toy tokenizer considered earlier. In order to avoid dealing with the transient characteristics of the source, we consider the cross-entropy loss in eq. (1) under the assumption that the test sequences s are of length $m \rightarrow \infty$. Namely, define the normalized loss,

$$\mathcal{L}(\cdot) = \lim_{m \rightarrow \infty} \frac{1}{m} \mathcal{L}_m(\cdot)$$

Theorem 4.1. *Consider a Markov data generating process which satisfies Assumption 4.2. Let d denote a budget on the size of the dictionary. Then, there exists a tokenizer with at most d tokens and encoding function $\text{enc}(\cdot)$, such that,*

$$\min_{Q \in \mathcal{Q}_{\text{1-gram}}} \mathcal{L}(Q \circ \text{enc}(\cdot)) \leq \frac{1}{1 - \varepsilon} \min_{Q'} \mathcal{L}(Q') \quad (3)$$

where ε is $\log(1/\delta)/0.99 \log(d)$ ³. Furthermore, a tokenizer satisfying eq. (3) with probability $\geq 1 - d^{-\Omega_\delta(\log(d))}$ can be learnt from a dataset of $\tilde{O}_\delta(d)$ characters.

The tokenizers considered in this theorem are far more efficient with their token budget than the toy tokenizer - to achieve a cross entropy loss within a factor 2 of optimal, the dictionary size required by these tokenizers is $d \approx 1/\delta^2$ on any source satisfying Assumption 4.2. In comparison, the toy tokenizer requires a dictionary size of $e^{1/\delta \log(1/\delta)}$ to achieve the same error. We show that the LZW tokenizer proposed in (Zouhar et al., 2023a) achieves the upper bound in eq. (3) when trained on a dataset of size $\tilde{O}(d)$. Likewise, we also show that a sequential variant of BPE achieves the upper bound in eq. (3) up to a factor of 2 and with a worse dependency in the $o(1)$ term when trained on a dataset of size $\tilde{O}(d^2)$. What is interesting is that neither of these algorithms explicitly learn a unigram likelihood model, Q , while constructing the dictionary. Yet they are able to perform as well as the tokenizers which are jointly optimized with a likelihood model, such as the Unigram tokenization algorithm (Kudo, 2018).

Key insight. While the toy tokenizer provides a high level intuition as to why tokenization might enable unigram models to model Markov sources well, here we present a different explanation which captures tokenization from an operational viewpoint. Tokenizers which do a good job at learning patterns in the data and assigning these frequent patterns as tokens in the dictionary are compatible with an i.i.d. model over tokens. A hypothetical example motivating this point: consider a tokenizer

³ ε is assumed to be < 1 in this statement. The constant 0.99 can be made arbitrarily close to 1.

such that the distribution of tokens in the encoding of a fresh string sampled from the source is distributed i.i.d., except that whenever the token \mathbf{t}' appears, it is always followed by \mathbf{t}'' . An i.i.d. model on the tokens is a poor approximation since $P(\mathbf{t}'\mathbf{t}'') \gg P(\mathbf{t}')P(\mathbf{t}'')$. However, by merging \mathbf{t}' and \mathbf{t}'' into a new token \mathbf{t} and adding this to the dictionary, the new distribution over tokens is i.i.d. In general, this motivates why it is desirable for a tokenizer to allocate new tokens to substrings which appear next to each other frequently, i.e. a pattern in the data. As more tokens are added to the dictionary, one might expect the cross-entropy loss incurred by the best unigram model to improve.

Loosely, Theorem 2.1 gives an example where the converse is true as well - the character-level tokenizer does not learn patterns in the data generating process and the token budget is left unused. The cross-entropy loss incurred by this tokenizer under any unigram model is suboptimal. In the next section, we flesh out formally what it means for a tokenizer to “learn patterns” in the source process well.

4.1 Learning patterns in the source

The main result of this section is a generic reduction: dictionaries which typically encode new strings into a few long tokens (defined in a formal sense in Theorem 4.3), result in tokenizers achieving near-optimal cross-entropy loss. We prove this result for Markovian sources under a regularity assumption, which is that the associated connectivity graph of the chain is complete. The analogous assumption for k^{th} -order sources is that the transition kernel is entry-wise bounded away from 0. This assumption is satisfied by all the sources considered in the paper thus far, such as the k^{th} -order switching processes in Figure 1.

Assumption 4.2 (Data generating process). Assume that the data source is an ergodic Markov process with transition $P(\cdot|\cdot)$ and stationary distribution π . Assume that $\min_{a,a' \in \mathcal{A}} P(a'|a) \triangleq \delta > 0$.

For a substring \mathbf{s} and a character a , define $P(\mathbf{s}|a) = P(s_1|a) \prod_{i=2}^{|\mathbf{s}|} P(s_i|s_{i-1})$ denote the conditional probability of the substring \mathbf{s} . We now state the main result of this section.

Theorem 4.3 (Bound on cross-entropy loss of dictionaries under greedy encoder). *Consider a source satisfying Assumption 4.2 and any tokenizer \mathcal{T} equipped with the greedy encoder, $\text{enc}_{\text{gre}}(\cdot)$ with finitely long tokens. Define, $P(\mathbf{t}) = \mathbb{E}_{a \sim \pi}[P(\mathbf{t}|a)]$ and suppose $H(Q_{\text{MLE}}, P) \geq \frac{1}{\varepsilon} \log(1/\delta)$ for some $\varepsilon < 1$. Then,*

$$\min_{Q \in \mathcal{Q}_{1\text{-gram}}} \mathcal{L}(Q \circ \text{enc}_{\text{gre}}(\cdot)) \leq \frac{\min_Q \mathcal{L}(Q)}{1 - \varepsilon}.$$

Interpretation. $H(Q_{\text{MLE}}, P) = \mathbb{E}_{\mathbf{t} \sim Q_{\text{MLE}}} [\log(1/P(\mathbf{t}))]$ is large when the encoder places higher mass (i.e. larger values of $Q_{\text{MLE}}(\cdot)$) on tokens which have low probability under P , i.e. which correspond to longer substrings. Intuitively, this metric is higher for tokenizers which typically use long tokens (i.e. low $P(\cdot)$) to encode new strings. This is closely related to the notion of fertility (Scao et al., 2022).

4.2 LZW tokenizer

In this section we study the Lempel-Ziv-Welch (LZW) based tokenization scheme introduced by Zouhar et al. (2023a) and establish guarantees of the form of Theorem 4.1 for this tokenizer.

Definition 4.4 (LZW tokenizer). Iterating from left to right, the shortest prefix of the training dataset which does not already exist as a token is assigned as the next token in the dictionary. This substring is removed and the process is iterated on the remainder of the dataset. The tokenizer uses the greedy encoding algorithm (Definition A.3) to encode new strings into tokens.

An example of the LZW tokenizer: For the source dataset 0100111, the dictionary created is $\{0, 1, 00, 11\}$.

The LZW tokenizer is based on the LZW algorithm for compression (Ziv and Lempel, 1978; Welch, 1984). The dictionary satisfies the property that if some substring s' exists as a token in the dictionary, then all of its prefixes must also belong to the dictionary. In the next theorem, we show that the LZW tokenizer is approximately optimal.

Theorem 4.5. *Suppose the LZW tokenizer is trained on a dataset of length at most d (thereby learning a dictionary with at most d tokens). For Markov sources satisfying Assumption 4.2, with probability $\geq 1 - d^{-O_\delta(\log(d))}$, the resulting tokenizer satisfies,*

$$\min_{Q \in \mathcal{Q}_{1\text{-gram}}} \mathcal{L}(Q \cdot \mathbf{enc}_{gre}(\cdot)) \leq \frac{\min_Q \mathcal{L}(Q)}{1 - \varepsilon}.$$

where $\varepsilon = \frac{\log(1/\delta)}{0.99 \log(d)}$ ⁴.

The proof of this result considers all substrings \mathbf{t} with $P(\mathbf{t}) \geq 1/d^{0.99}$. These substrings are reasonably high probability and observed many times in a dataset of $\tilde{\Omega}(d)$ characters. We show that with high probability, the LZW tokenizer learns *all* of these substrings as tokens in the dictionary. Now, when processing a new string, since the greedy algorithm only emits the longest substring which matches a token, every token allocated must fall on the “boundary” of this set, having $P(\mathbf{t}) \leq O(1/d^{0.99})$. By definition, this means that $H(Q_{MLE}, P) = \mathbb{E}_{\mathbf{t} \sim Q_{MLE}}[\log(1/P(\mathbf{t}))] = 0.99 \log(d)$. Combining this with Theorem 4.3 completes the proof. At a high level, on the infinite tree of substrings \mathcal{A}^* we study which nodes are populated as tokens by LZW. This structure forms a Digital Search Tree (DST) and prior work analyzes the mean and variance of the profile of the DST under various source processes (Jacquet et al., 2001; Drmota and Szpankowski, 2011; Hun and Vallée, 2014; Drmota et al., 2021). A detailed proof of Theorem 4.5 is provided in Appendix A.6.

4.3 A sequential variant of BPE

In this section, we take a more practical look and try to analyze tokenizers used commonly in practice. The Byte-Pair-Encoding (BPE) algorithm (Gage, 1994; Sennrich et al., 2016), discovered in the compression literature as REPAIR (Larsson and Moffat, 2000; Navarro and Russo, 2008) was proposed as a faster alternative to LZW. It remains as one of the most commonly implemented tokenizers in natural language processing for various downstream tasks (Radford et al., 2019; Mann et al., 2020; Touvron et al., 2023). A large proportion of open source and commercial LLMs currently use BPE as the tokenization algorithm of choice, such as GPT-2/3, Llama 1/2 and Mistral-7B to name a few.

The BPE algorithm is based on constructing the dictionary iteratively by merging pairs of tokens to result in a tokens. In each iteration, the pair of tokens which appear most frequently next to each other are merged together into a single token. Subsequently, every occurrence of the pair of tokens are replaced by the newly added token, breaking ties arbitrarily. The dictionary is thus an ordered mapping of the form $\mathbf{t} \leftarrow (\mathbf{t}', \mathbf{t}'')$. To encode a new string, the BPE encoder iterates through the

⁴The constant 0.99 can be made arbitrarily close to 1.

dictionary and for each rule $\mathbf{t} \leftarrow (\mathbf{t}', \mathbf{t}'')$ replaces every consecutive occurrence of \mathbf{t}' and \mathbf{t}'' by the token \mathbf{t} breaking ties arbitrarily.

To warm up our main results, it is worth understanding the behavior of the BPE tokenizer in a bit more detail. Unlike the toy tokenizer, it is a priori unclear whether unigram models trained on sequences tokenized by BPE even asymptotically (in the dictionary size) achieve the optimal cross-entropy loss. Indeed, for $\delta > 0$, consider a training sequence of length m of the form,

$$\mathbf{s} = \underbrace{\left(\begin{array}{c} 01 \cdots 01 \\ 2/\delta \end{array} \right)}_{\times \frac{m\delta}{4}} \underbrace{\left(\begin{array}{c} 10 \cdots 10 \\ 2/\delta \end{array} \right)}_{\times \frac{m\delta}{4}} \quad (4)$$

The probability that this sequence is generated by the order-2 switching Markov source with $p = q = \delta$ is,

$$\approx (1 - \delta)^{\frac{m\delta}{4} \times \frac{4}{\delta} \times (1 - \delta)} (\delta)^{\frac{m\delta}{4} \times 4} = e^{-H(P)},$$

which uses the fact that $H(P) = m\delta \log(1/\delta) + m(1 - \delta) \log(1/(1 - \delta))$. This implies that even though the string has exponentially small probability, it is one of the typical sequences for this order-2 Markov source. Let's understand what happens when the BPE tokenizer is trained on this dataset. Assuming that ties are broken arbitrarily, consider the run of the BPE algorithm detailed in Table 1. Here, we assume that $1/\delta - 1$ is a power of 2 and denote $r = \log_2(1/\delta - 1)$. The algorithm first merges 0 and 1 into a single token \mathbf{t}_1 , which results in a long sequence of the form $\mathbf{t}_1 \dots \mathbf{t}_1 1 \mathbf{t}_1 \dots \mathbf{t}_1 0$ repeated $m\delta/4$ times. In subsequent rounds, the tokens $(\mathbf{t}_1, \mathbf{t}_1)$ is merged into \mathbf{t}_2 , then $(\mathbf{t}_2, \mathbf{t}_2)$ is merged into \mathbf{t}_3 and so on, until is no longer possible. Finally, the resulting sequence is a repeating sequence of 5 tokens where within each sequence, no pair of tokens appears more than once next to each other. Eventually these 5 tokens are merged into a single token labelled \mathbf{t}_{r+4} , and in subsequent rounds the tokens $(\mathbf{t}_{r+4}, \mathbf{t}_{r+4})$ are merged into \mathbf{t}_{r+5} , $(\mathbf{t}_{r+5}, \mathbf{t}_{r+5})$ is merged into \mathbf{t}_{r+6} and so on, until is no longer possible.

Initial	$01 \dots 01 10 \dots 10 \dots$
$\mathbf{t}_1 \leftarrow (0, 1)$	$\mathbf{t}_1 \dots \mathbf{t}_1 1 \mathbf{t}_1 \dots \mathbf{t}_1 0 \dots$
$\mathbf{t}_2 \leftarrow (\mathbf{t}_1, \mathbf{t}_1)$	$\mathbf{t}_2 \dots \mathbf{t}_2 1 \mathbf{t}_2 \dots \mathbf{t}_2 0 \dots$
\vdots	\vdots
$\mathbf{t}_r \leftarrow (\mathbf{t}_{r-1}, \mathbf{t}_{r-1})$	$\mathbf{t}_r \mathbf{t}_1 1 \mathbf{t}_r 0 \dots$
$\mathbf{t}_{r+1} \leftarrow (\mathbf{t}_r, \mathbf{t}_1)$	$\mathbf{t}_{r+1} 1 \mathbf{t}_r 0 \dots$
$\mathbf{t}_{r+2} \leftarrow (\mathbf{t}_r, 0)$	$\mathbf{t}_{r+1} 1 \mathbf{t}_{r+2} \dots$
$\mathbf{t}_{r+3} \leftarrow (\mathbf{t}_{r+1}, 1)$	$\mathbf{t}_{r+3} \mathbf{t}_{r+2} \dots$
$\mathbf{t}_{r+4} \leftarrow (\mathbf{t}_{r+3}, \mathbf{t}_{r+2})$	$\mathbf{t}_{r+4} \dots$
$\mathbf{t}_{r+5} \leftarrow (\mathbf{t}_{r+4}, \mathbf{t}_{r+4})$	$\mathbf{t}_{r+5} \dots$
$\mathbf{t}_{r+6} \leftarrow (\mathbf{t}_{r+5}, \mathbf{t}_{r+5})$	$\mathbf{t}_{r+6} \dots$
\vdots	\vdots

Table 1: A representation of the behavior of BPE when trained on the dataset in eq. (4). We assume that $1/\delta - 1$ is a power of 2 and define $r = \log_2(1/\delta - 1)$.

Observe that in the initial training dataset the substrings 0000 and 1111 never appears as a contiguous sequence. However, in a test sequence of length m sampled from the 2nd-order Markov source, with high probability these substrings disjointly occur $\Theta(m)$ times each. The learnt dictionary associates each such disjoint occurrence of these substrings with at least 1 token, for 0000, the 3rd 0 must necessarily be tokenized as the token “0”. Likewise, in 1111, the 3rd 1 must necessarily be tokenized as the token “1”. Therefore, when a new test string of length m is tokenized, with high probability the tokens “0” and “1” form a constant fraction of the total collection of tokens.

Thus on freshly sampled test sequences, the BPE tokenizer appears to behave like the character-level tokenizer on a constant fraction of the input sequence. In particular, a simple calculation shows that the cross-entropy loss of any unigram model trained on this tokenizer must be far from the optimal bound of $mH_{\text{BER}}(\delta)$ especially as δ becomes smaller,

$$\begin{aligned} & \min_{Q \in \mathcal{Q}_{1\text{-gram}}} \mathcal{L}_m(Q \circ \text{enc}(\cdot)) \\ & \geq \min_{Q \in \mathcal{Q}_{1\text{-gram}}} \mathbb{E} [n_0 \log(1/Q_{\text{tok}}(0) + n_1 \log(1/Q_{\text{tok}}(1))] \\ & \stackrel{(i)}{\geq} \Omega(m) \cdot \min_{Q \in \mathcal{Q}_{1\text{-gram}}} (\log(1/P_{\text{tok}}(0) + \log(1/Q_{\text{tok}}(1))) \\ & \geq \Omega(m). \end{aligned}$$

where (i) uses the fact that $\mathbb{E}[n_0], \mathbb{E}[n_1] \in \Omega(m)$ and the last inequality uses $P_{\text{tok}}(0)P_{\text{tok}}(1) \leq 1/4$ (AM-GM inequality) since they sum up to at most 1. The purpose of this example is to show that there exist pathological training datasets which appear to be drawn from a stochastic source, but on which BPE fails to learn a good dictionary for the source. Thus proving a result such as Theorem 4.1 for BPE would require arguing that training datasets such as that in eq. (4) are unlikely to be seen.

The analysis of the standard variant of BPE turns out to be complicated for other reasons too. After every token is added the training dataset becomes a mix of all the previously added tokens, and arguing about the statistics of which pair of tokens appears most frequently for the next iteration becomes involved. For instance, adding 00 as a token may reduce the frequency of occurrence of the substring 01, but will not affect 11. Thus, even though 01 may a priori have been seen more frequently, it may not be chosen by BPE as the next token after 00.

To avoid dealing with these issues, we consider a sequential/sample-splitting variant of BPE. At a high level, the algorithm breaks down a dataset of size $\Theta(d^2)$ into d chunks and learns at most 1 token from each chunk. The algorithm iterates over the chunks and finding the pair of tokens which appear most frequently next to each other in each chunk and adding it to the dictionary if it appears more than $\log(d)$ times. Every consecutive occurrence of the pair of tokens is replaced by the newly assigned token in the dataset. Thus, in each iteration i , at most 1 token is added, depending on the statistics of the i^{th} chunk and the tokens added so far to the dictionary. Based on the final size of the dictionary a different encoder/decoder pair is used - if the algorithm adds sufficiently many tokens to the dictionary, the greedy encoder is used, and if not, a parallel implementation of BPE’s encoding algorithm is used (Definition 4.6). A formal description of the algorithm is in Algorithm 1.

Definition 4.6 (BPE.split encoder). The BPE.split encoder parses a new string into tokens as follows. The algorithm partitions the string into contiguous chunks of length d . Then, BPE’s encoder is applied on each chunk, which iterates through DS and replaces $\mathbf{t}'\mathbf{t}''$ by \mathbf{t} for every rule $\mathbf{t} \leftarrow (\mathbf{t}', \mathbf{t}'')$ in DS, breaking ties arbitrarily. The individual token sequences are finally spliced together and returned.

The main result of this section is that up to a small additive error, Algorithm 1 approaches a 2-approximation to the optimal cross-entropy loss.

Algorithm 1 Sequential implementation of BPE

Input: $\epsilon \in (0, 1)$; a dataset of size $n = \Theta(d^2)$, split into d contiguous texts $\{\text{text}_1, \dots, \text{text}_d\}$ of length $\Theta(d)$ each.

Output: A tokenizer \mathcal{T} .

// Generate Dictionary

```

for  $i = 1, \dots, d$  do
    if  $\exists$  a pair of tokens/characters  $(t', t'')$  appearing  $\geq \log(d)$  times consecutively in  $\text{text}_i$  then
        Append the rule  $t \leftarrow (t', t'')$  to DS
        for  $j = i + 1, \dots, d$  do
             $\text{text}_j \leftarrow \text{APPLY}_{t \leftarrow (t', t'')}(text_j);$ 
    // Can be implemented in parallel
    end for
    end if
end for

// Encoder and Decoder
if  $|\text{Dict}| < d_0 \triangleq \epsilon d / 2 \log(4|\mathcal{A}|)$  then
     $\mathcal{T} \leftarrow (\text{Dict}, \text{DS}, \text{enc}_{\text{BPE}.\text{split}}(\cdot), \text{dec}_{\text{BPE}.\text{split}}(\cdot))$ 
else
     $\mathcal{T} \leftarrow (\text{Dict}, \emptyset, \text{enc}_{\text{gre}}(\cdot), \text{dec}_{\text{gre}}(\cdot))$ 
end if

def  $\text{APPLY}_{t \leftarrow (t_1, t_2)}(\text{text}):$ 
    Replace every consecutive occurrence of  $(t', t'')$  in  $\text{text}$  by  $t$ , breaking ties arbitrarily.

```

Theorem 4.7. For any $\epsilon \in (0, 1)$, run Algorithm 1 on a dataset of $n = \Theta(d^2)$ characters to learn a dictionary with at most d tokens. The resulting tokenizer \mathcal{T} satisfies with probability $\geq 1 - e^{-\Omega(de^2)}$,

$$\min_{Q \in \mathcal{Q}_{1\text{-gram}}} \mathcal{L}(Q \circ \text{enc}(\cdot)) \leq (2 + \epsilon) \min_Q \mathcal{L}(Q) + \epsilon$$

where $\epsilon = O\left(\frac{\log(1/\epsilon) \log^3(1/\delta)}{\epsilon \delta^9 \log(d)}\right)$.

While the guarantees established for the sequential BPE tokenizer are weaker than those in Theorem 4.1, the analysis turns out to be quite involved. Theorem 4.7 implies that unigram models trained on the sequential BPE tokenizer asymptotically approach the optimal cross-entropy loss up to a factor of 2.

The formal proof of this result is presented in Section 4.3. What is the intuition behind using a different encoder in Algorithm 1 depending on the number of tokens in the dictionary? When the number of tokens in the dictionary is smaller than d_0 , we know that on a $1 - d_0/d$ fraction of the iterations of Algorithm 1, a token is *not* added to the dictionary, i.e. every pair of tokens already appears at most $\log(d)$ times together. This is a datapoint of “evidence” that under the dictionary in that iteration, the BPE encoder is already good at encoding new strings (of length $\Theta(d)$) in a way where pairs of tokens do not appear consecutively with high frequency. Since future dictionaries only have more rules appended to them, dictionaries only get better at encoding new strings into tokens where pairs do not frequently appear consecutively. In other words, the BPE encoder satisfies a monotonicity property. It remains to show that dictionaries which encode new sequences in a way where no pair of tokens appear too frequently have large $H(Q_{\text{MLE}}, P)$ (to invoke Theorem 4.3). This follows from ideas introduced in (Navarro and Russo, 2008).

The case where the number of tokens is large ($\geq d_0$) turns out to present significant technical challenges for analyzing the BPE encoder. There is no longer much “evidence” that the dictionary in each iteration is good at encoding strings since in a large number of iterations a pair of tokens appear consecutively with high frequency. Analyzing the greedy encoder also presents its own challenges - although the algorithm has allocated a large number of tokens, it is possible that there are short tokens t which are maximal (i.e. they are not prefixes of other tokens). This is similar to the problem encountered by BPE when trained on the dataset in eq. (4) - although the algorithm has allocated a large number of tokens, the token 1 is maximal since every other token begins with the character 0. However, it turns out that such tokens, although present in the dictionary, are not observed frequently while encoding a fresh string drawn from the source.

5 Additional theoretical results

In this section, we discuss some additional results, which are fleshed out in more detail in Appendices C to E.

5.1 Finite sample considerations

The results in Section 4 focus on the analysis in the case where the downstream likelihood model is trained optimally. In practice, however, transformers are trained on finite datasets and not likely to be optimal. In this section, we focus on understanding the role of a finite dataset from a theoretical point of view. While the results in Theorem 4.1 indicate that a larger dictionary size is better in that the best unigram model achieves better cross-entropy loss, this statement ignores finite-sample considerations in training this model itself. In Appendix C, for the LZW dictionary, we show that the number of samples required to learn this model to a multiplicative error of $1 + \xi$, n_{lm}^* , scales as $\tilde{O}(d^{1.01}/\delta\xi^2)$. In other words, given n_{lm}^* samples, with high probability, it is possible to learn a model \hat{Q} such that,

$$\mathcal{L}(\hat{Q} \circ \text{enc}_{\text{gre}}(\cdot)) \leq (1 + \xi) \min_{Q \in \mathcal{Q}_{1\text{-gram}}} \mathcal{L}(Q \circ \text{enc}_{\text{gre}}(\cdot)).$$

where $\text{enc}(\cdot)$ is the greedy encoder on an LZW dictionary of size d learnt from data. This result indicates that it is easier to train the downstream likelihood model on smaller dictionaries. In conjunction with Theorem 4.1, this implies that there is an optimal dictionary size d^* for LZW which depends on δ and the size of the training dataset the learner has access to, which balances the limiting statistical error corresponding to a dictionary size of d^* with the finite sample error incurred in learning a likelihood model corresponding supported on d^* tokens. While our results do not establish these guarantees for transformers and are instead for a different finite-sample estimator of the optimal unigram model, it is an interesting direction for future research to characterize the learning trajectory and limiting statistical error of transformers trained with gradient descent.

5.2 The generalization ability of tokenizers

In this paper, we focus on tokenizers learnt from data, which are used to encode fresh strings which were not trained on previously. In previous work interpreting tokenizers from a theoretical point of view, this element of generalization to new strings has been missing. For instance, the work of Zouhar et al. (2023b) show that the BPE algorithm is essentially an approximation algorithm for the objective of finding the shortest encoding of the training dataset as a sequence of token merges. While this perspective is useful in formalizing the notion of compression BPE carries out,

the characterization is ultimately with respect to the training dataset, and not in BPE’s ability to compress fresh strings. In Appendix D we show that there are tokenization algorithms which are extremely good at compressing the dataset they were trained on, but on new strings, the tokenizer behaves like a character tokenizer almost everywhere. As a consequence, transformers trained on the output of this tokenizer would fail to approach the optimal cross-entropy loss and in fact get stuck close to the loss of $mH(\pi)$ loss achieved by the character level tokenizer in Theorem 2.1.

5.3 Interaction between dictionary and encoder

The choice of tokenizer in language modeling has received a large degree of study in various domains (Rust et al., 2021; Toraman et al., 2023) and is identified as a step that has been overlooked in designing performant LLMs (Mielke et al., 2021; Wu et al., 2023). While typically this has come to imply using different tokenization algorithms altogether, it is important to note that the tokenizer is composed of two parts - the dictionary and the encoding algorithm, and the interaction between the two may result in differently performing end-to-end models. In Appendix E, we show that there exist dictionaries which generalize well under one encoder, in that unigram models trained on string encoded by this dictionary perform near-optimally, but these dictionaries completely fail to generalize under a different encoder. This means that in the process of designing good tokenizers, it does not suffice to think about the dictionary in isolation. Its interaction with the encoding algorithm is pertinent.

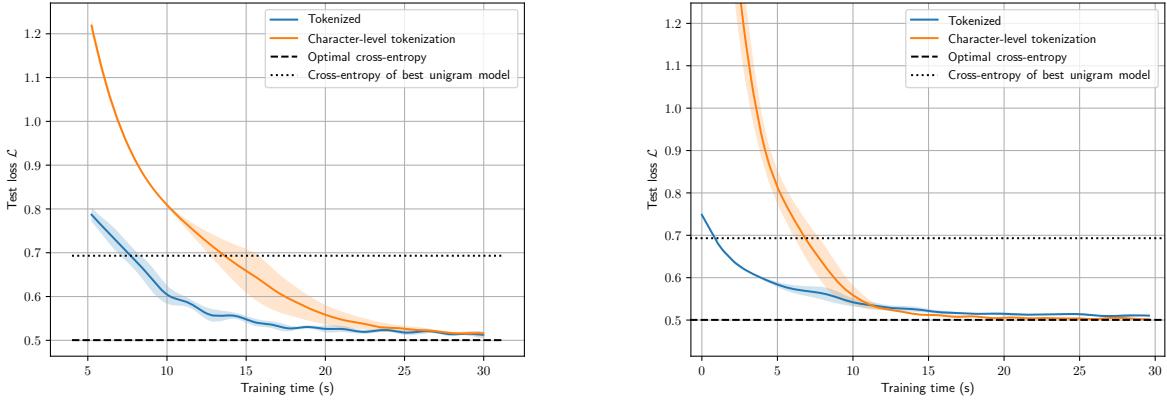
6 Experimental Results

In this section we discuss experimental results; hyperparameter choices and other details are presented in Appendix F.

Experiment 1 (Figures 5a and 5b). In this experiment we study the order-1 switching Markov chain (with $p = q = 0.8$). Specifically for the case of $k = 1$, transformers without tokenization empirically achieve a small cross-entropy on this learning task as seen in Figure 5a and earlier in Makkula et al. (2024). We vary a number of hyperparameters to find the smallest character-level model which achieves a loss within 10% of the optimal-cross entropy within the first 300 iterations. Fixing the dictionary size for BPE as 10, we also find the smallest transformer which achieves the same target loss. Although the smallest BPE tokenized model is larger than the smallest character-level tokenized model in terms of the number of parameters, the wall-clock time taken to optimize the model to any target test loss is observed to be smaller. Thus, tokenization appears to reduce the compute time required to train the model to a target test loss in the toy example we consider. In contrast, in Figure 5b we compare models with the same architecture trained with and without tokenization⁵. The model with BPE tokenization again appears to converge more quickly, although the limiting error achieved is subtly higher in comparison with the model with character-level tokenization. By making the model with BPE tokenization larger, we can tradeoff the convergence speed (in wall-clock time) with the limiting error.

Experiment 2a (Figure 6). In this experiment, we vary the dictionary size of the tokenizer to observe how the test loss curves vary, when trained on a single random Markov process drawn from

⁵The model with tokenization is trained on smaller sequences by virtue of tokenization.



(a) Convergence rate of smallest model which is within 10% of the optimal-cross entropy in 300 iterations. The smallest character-level model has 9K parameters (3 layers, embedding dimension = 10). The smallest tokenized model with a dictionary size of 10 has 18K parameters (3 layers, embedding dimension = 20). The tokenized model has more parameters but the wall-clock time taken to reach any target loss is lower.

(b) Convergence rate of models with the same embedding dimension (20), number of heads (1) and layers (3) with and without tokenization. The model with tokenization (dictionary size of 20) appears to converge more quickly. The character-level model is trained on input sequences of length 512; the tokenized model is effectively trained on smaller sequences (of length ≈ 145).

Figure 5: Test loss vs. wall-clock time for the BPE tokenized and character-level models when trained on the order-1 switching Markov chain (Figure 1) with $p = q = 0.8$.

the Dirichlet prior described in Figures 3a and 3b. We observe that increasing the dictionary size of the tokenizer helps the model to achieve smaller test losses more quickly.

Next, we present the same evaluation of tokenizers, but on real world datasets. Since evaluating the end-to-end pipeline requires pretraining a transformer which is computationally intensive even at small and medium scales, we resort to evaluating tokenizers on k -gram models at these scales.

Experiment 2b (Figure 7). Since pretraining is an expensive operation, we resort to evaluating the effect of tokenization at scale by proxy techniques. We first train tokenizers on the Wikitext-103-raw-v1 dataset (Merity et al., 2016) and compare the performance of unigram models trained on the GLUE dataset as the model size scales. In this experiment, we do not evaluate the cross entropy loss by pretraining a transformer. Rather, we estimate the cross-entropy of the best unigram model by using the approximation,

$$-\mathbb{E} \left[\sum_{t \in \text{Dict}} n_t \log Q_{\text{MLE}}(t) \right] \approx - \sum_{t \in \text{Dict}} \hat{n}_t \log(\hat{Q}(t)) \quad (5)$$

where $\hat{Q}(t) = \frac{\hat{n}_t}{\sum_t \hat{n}_t}$ is the MLE unigram model learnt from a finite dataset, which we choose here as GLUE (Wang et al., 2019), and \hat{n}_t is the number of times the token t is observed in the encoding of the dataset. This approximation allows us to separate the error stemming from learning a suboptimal likelihood model which tends to have higher sample complexity requirements and focus on the asymptotic error of the tokenizer. Since the character-level tokenizer operates on a fixed vocabulary, in order to compare with the other tokenizers, we plot the number of unique k -grams observed in the training dataset along the x -axis. While this is not an apples-to-apples comparison, we use

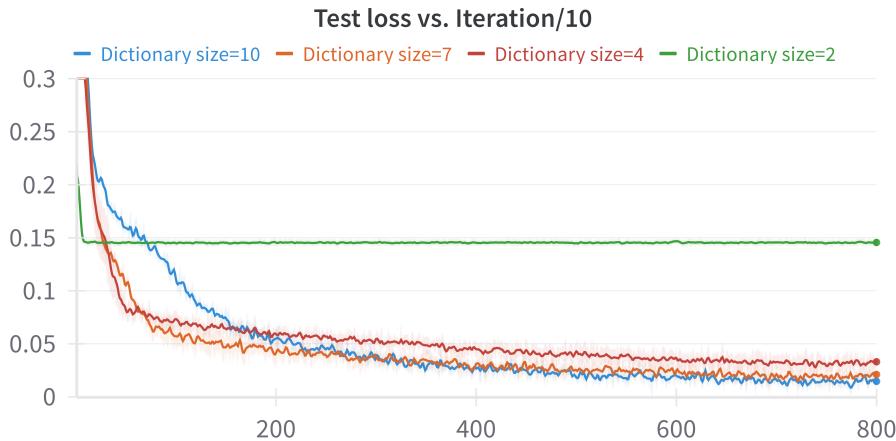


Figure 6: *Performance vs. dictionary size.* Each curve corresponds to using learnt BPE tokenizers with different dictionary sizes. Since the underlying data process is on a binary space, dictionary size = 2 corresponds to character-level tokenization. The underlying order-4 Markov chain is generated randomly, but fixed across all training runs, and the transformer’s sequence length is adapted across different dictionary sizes (with the base character-level sequences of length 512 in all cases). For each choice of the dictionary size, we average over 5 training runs to average over the randomness in learning the tokenizer and transformer. Increasing the dictionary size improves performance, albeit subtly slowing down optimization.

the number of unique k -grams in the dataset as a proxy for the complexity of the likelihood model trained. One may also use the total number of possible k -grams as a proxy; however a large fraction of these k -grams would likely never be observed in a real dataset (especially as k grows).

Experiment 3 (Table 2). In this experiment, we compare the cross entropy loss of the best unigram model trained on pretrained tokenizers on an array of datasets. All the considered tokenizers have dictionary sizes in the range 31K-51K. The best bigram model under the character-level tokenizer is consistently outperformed by the best unigram likelihood model trained under a number of pretrained tokenizers on a variety of datasets: Rotten Tomatoes (8.5K sequences), GLUE (105K), Yelp review (650K), Wikitext-103-v1 (1.8M), OpenOrca (2.9M).

7 Open Questions

In this section, we discuss some limitations of our work and open questions stemming from them. We show that when transformers are trained with or without tokenization, they learn to approximately represent k -gram models for different values of k . Transformers are capable of representing far more complex behavior, which are elicited under more complex data generating processes. Extending our formulation to these settings presents an avenue to develop an even better understanding of tokenization, and would allow finer-grained comparisons between tokenizers. The behavior and role of tokenizers may be very different in these contexts. Below we discuss some concrete questions.

Our theory assumes that the underlying Markov chain has every transition occurring with non-zero probability, which is a limitation. However, the analysis for the toy tokenizer in eq. (2) shows that when the dictionary size scales as $\exp(mH(\pi)/H(P))$, even in the absence of Assumption 4.2,

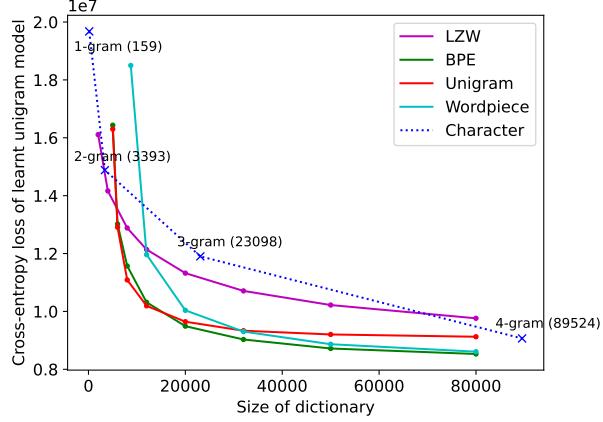


Figure 7: *Performance vs. dictionary size.* All tokenizers are trained on the Wikitext-103 dataset with early stopping when the desired vocabulary size is met. For all other tokenizers we train unigram models while for the character-level tokenizer, we train k -gram models for $k \in \{1, 2, 3, 4\}$. In all cases, the GLUE dataset is used to learn the likelihood model. The number in the parentheses denotes the number of distinct observed k -grams, which lower bounds the k -gram model complexity.

the tokenizer achieves the optimal cross-entropy to within a factor of 2. This leads to the following conjecture.

Conjecture 1. *In the spirit of eliminating Assumption 4.2, is it possible to establish a version of Theorem 4.1 applicable to data drawn from any Markov chain, where $\varepsilon = \log(1/\delta)/0.99 \log(d)$ is replaced by $\varepsilon = \log(mH(\pi)/H(P))/0.99 \log(d)$.*

References

- Mehdi Ali, Michael Fromm, Klaudia Thellmann, Richard Rutmann, Max Lübbing, Johannes Leveling, Katrin Klug, Jan Ebert, Niclas Doll, Jasper Schulze Buschhoff, et al. Tokenizer choice for llm training: Negligible or crucial? *arXiv preprint arXiv:2310.08754*, 2023.
- Zaid Alyafeai, Maged S Al-shaibani, Mustafa Ghaleb, and Irfan Ahmad. Evaluating various tokenizers for arabic text classification. *Neural Processing Letters*, 55(3):2911–2933, 2023.
- Dietrich Braess and Thomas Sauer. Bernstein polynomials and learning theory. *Journal of Approximation Theory*, 128(2):187–206, 2004.
- Yen-Chi Chen. Stochastic modeling of scientific data, Autumn 2018.
- Jonathan H Clark, Dan Garrette, Iulia Turc, and John Wieting. Canine: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, 10:73–91, 2022.
- Yasuhiro Den, Toshinobu Ogiso, Hideki Ogura, Atsushi Yamada, Nobuaki Minematsu, Kiyotaka Uchimoto, and Hanae Koiso. The development of an electronic dictionary for morphological analysis and its application to japanese corpus linguistics, Oct 2007. URL <https://repository.ninjal.ac.jp/api/records/2201>.

	RT	Wiki	OpenOrca	Yelp	GLUE
BERT tokenizer	1.58	1.55	1.50	1.60	1.50
TinyLlama	1.75	1.84	1.75	1.82	1.70
GPT-neox-20b	1.57	1.64	1.60	1.66	1.48
Mixtral-8x7b	1.69	1.80	1.71	1.75	1.66
Phi-2 tokenizer	1.54	1.62	1.60	1.64	1.45
Char + bigram	2.40	2.45	2.49	2.46	2.38

Table 2: Cross-entropy loss estimates (using eq. (56)) of unigram models trained on pretrained tokenizers under a number of datasets. The last row (blue) is the character-level tokenizer, on which a more powerful bigram model is trained. The table indicates that tokenization allows unigram models on tokenized sequences to outperform even more powerful bigram models learnt on the underlying character-level sequences. BERT is based on Wordpiece. TinyLlama, GPT-neox-20b and Mixtral-8x7b are based on BPE. The character-level tokenizer we use is ByT5.

Michael Drmota and Wojciech Szpankowski. The expected profile of digital search trees. *Journal of Combinatorial Theory, Series A*, 118(7):1939–1965, 2011.

Michael Drmota, Michael Fuchs, Hsien-Kuei Hwang, and Ralph Neininger. Node profiles of symmetric digital search trees: Concentration properties. *Random Structures & Algorithms*, 58(3):430–467, 2021.

Benjamin L Edelman, Ezra Edelman, Surbhi Goel, Eran Malach, and Nikolaos Tsilivis. The evolution of statistical induction heads: In-context learning markov chains. *arXiv preprint arXiv:2402.11004*, 2024.

Tanja Eisner, Bálint Farkas, Markus Haase, and Rainer Nagel. *Operator theoretic aspects of ergodic theory*, volume 272. Springer, 2015.

Philip Gage. A new algorithm for data compression. *C Users Journal*, 12(2):23–38, 1994.

Matthias Gallé. Investigating the effectiveness of bpe: The power of shorter sequences. 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)*, pages 1375–1381, 2019.

Siavash Golkar, Mariel Pettee, Michael Eickenberg, Alberto Bietti, Miles Cranmer, Geraud Krawezik, Francois Lanusse, Michael McCabe, Ruben Ohana, Liam Parker, et al. xval: A continuous number encoding for large language models. *arXiv preprint arXiv:2310.02989*, 2023.

Robert M Gray and RM Gray. *Probability, random processes, and ergodic properties*, volume 1. Springer, 2009.

Gregory Grefenstette and Pasi Tapanainen. What is a word, what is a sentence?: problems of tokenisation. 1994.

Yanjun Han, Soham Jana, and Yihong Wu. Optimal prediction of markov chains with and without spectral gap. *Advances in Neural Information Processing Systems*, 34:11233–11246, 2021.

Kanal Hun and Brigitte Vallée. Typical depth of a digital search tree built on a general source. In *2014 Proceedings of the Eleventh Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*, pages 1–15. SIAM, 2014.

- Itay Itzhak and Omer Levy. Models in a spelling bee: Language models implicitly learn the character composition of tokens. *arXiv preprint arXiv:2108.11193*, 2021.
- Philippe Jacquet, Wojciech Szpankowski, and Jing Tang. Average profile of the lempel-ziv parsing scheme for a markovian source. *Algorithmica*, 31:318–360, 2001.
- Eugene Kharitonov, Marco Baroni, and Dieuwke Hupkes. How bpe affects memorization in transformers. *arXiv preprint arXiv:2110.02782*, 2021.
- Taku Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates. *arXiv preprint arXiv:1804.10959*, 2018.
- N Jesper Larsson and Alistair Moffat. Off-line dictionary-based compression. *Proceedings of the IEEE*, 88(11):1722–1732, 2000.
- Jindřich Libovický, Helmut Schmid, and Alexander Fraser. Why don't people use character-level machine translation? *arXiv preprint arXiv:2110.08191*, 2021.
- Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.
- Ashok Vardhan Makkuvu, Marco Bondaschi, Adway Girish, Alliot Nagle, Martin Jaggi, Hyeji Kim, and Michael Gastpar. Attention with markov: A framework for principled analysis of transformers via markov chains. *arXiv preprint arXiv:2402.04161*, 2024.
- Ben Mann, N Ryder, M Subbiah, J Kaplan, P Dhariwal, A Neelakantan, P Shyam, G Sastry, A Askell, S Agarwal, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.
- Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.
- Sabrina J Mielke, Zaid Alyafeai, Elizabeth Salesky, Colin Raffel, Manan Dey, Matthias Gallé, Arun Raja, Chenglei Si, Wilson Y Lee, Benoît Sagot, et al. Between words and characters: A brief history of open-vocabulary modeling and tokenization in nlp. *arXiv preprint arXiv:2112.10508*, 2021.
- Jaouad Mourtada and Stéphane Gaïffas. An improper estimator with optimal excess risk in misspecified density estimation and logistic regression. *The Journal of Machine Learning Research*, 23(1):1384–1432, 2022.
- Assaf Naor, Shravas Rao, and Oded Regev. Concentration of markov chains with bounded moments. 2020.
- Gonzalo Navarro and Luís MS Russo. Re-pair achieves high-order entropy. In *DCC*, page 537. Citeseer, 2008.
- Eshaan Nichani, Alex Damian, and Jason D Lee. How transformers learn causal structure with gradient descent. *arXiv preprint arXiv:2402.14735*, 2024.
- David D Palmer. Tokenisation and sentence segmentation. *Handbook of natural language processing*, pages 11–35, 2000.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

Terence Parr. *The Definitive ANTLR 4 Reference*. Pragmatic Bookshelf, Raleigh, NC, 2 edition, 2013. ISBN 978-1-93435-699-9. URL <https://www.safaribooksonline.com/library/view/the-definitive-antlr/9781941222621/>.

Aleksandar Petrov, Emanuele La Malfa, Philip HS Torr, and Adel Bibi. Language model tokenizers introduce unfairness between languages. *arXiv preprint arXiv:2305.15425*, 2023.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Nived Rajaraman, Jiantao Jiao, and Kannan Ramchandran. An analysis of tokenization: Transformers under markov data. *Advances in Neural Information Processing Systems*, 37:62503–62556, 2024.

Jessica Rumbelow and Matthew Watkins. Solidgoldmagikarp. <https://www.alignmentforum.org/posts/aPeJE8bSo6rAFoLqg/solidgoldmagikarp-plus-prompt-generation>, 2023.

Phillip Rust, Jonas Pfeiffer, Ivan Vulić, Sebastian Ruder, and Iryna Gurevych. How good is your tokenizer? on the monolingual performance of multilingual language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3118–3135, 2021.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, et al. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2022.

Mike Schuster and Kaisuke Nakajima. Japanese and korean voice search. In *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5149–5152. IEEE, 2012.

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

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

Arseny Tolmachev, Daisuke Kawahara, and Sadao Kurohashi. Juman++: A morphological analysis toolkit for scriptio continua. In Eduardo Blanco and Wei Lu, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 54–59, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-2010. URL <https://aclanthology.org/D18-2010>.

Cagri Toraman, Eyup Halit Yilmaz, Furkan Şahinuç, and Oguzhan Ozcelik. Impact of tokenization on language models: An analysis for turkish. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 22(4):1–21, 2023.

- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. 2019. In the Proceedings of ICLR.
- Terry A. Welch. A technique for high-performance data compression. *Computer*, 17(06):8–19, 1984.
- J Gerard Wolff. An algorithm for the segmentation of an artificial language analogue. *British journal of psychology*, 66(1):79–90, 1975.
- Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabrowski, Mark Dredze, Sebastian Gehrmann, Prabhjan Kambadur, David Rosenberg, and Gideon Mann. Bloomberggpt: A large language model for finance. *arXiv preprint arXiv:2303.17564*, 2023.
- Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. Byt5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306, 2022.
- Sangwon Yu, Jongyoon Song, Heeseung Kim, Seong-min Lee, Woo-Jong Ryu, and Sungroh Yoon. Rare tokens degenerate all tokens: Improving neural text generation via adaptive gradient gating for rare token embeddings. *arXiv preprint arXiv:2109.03127*, 2021.
- Wenqing Zheng, SP Sharan, Ajay Kumar Jaiswal, Kevin Wang, Yihan Xi, Dejia Xu, and Zhangyang Wang. Outline, then details: Syntactically guided coarse-to-fine code generation. In *International Conference on Machine Learning*, pages 42403–42419. PMLR, 2023.
- Jacob Ziv and Abraham Lempel. Compression of individual sequences via variable-rate coding. *IEEE transactions on Information Theory*, 24(5):530–536, 1978.
- Vilém Zouhar, Clara Meister, Juan Gastaldi, Li Du, Mrinmaya Sachan, and Ryan Cotterell. Tokenization and the noiseless channel. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5184–5207, Toronto, Canada, July 2023a. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.284. URL <https://aclanthology.org/2023.acl-long.284>.
- Vilém Zouhar, Clara Meister, Juan Luis Gastaldi, Li Du, Tim Vieira, Mrinmaya Sachan, and Ryan Cotterell. A formal perspective on byte-pair encoding. *arXiv preprint arXiv:2306.16837*, 2023b.

Appendix

Contents

A Analysis of LZW: Proofs of Theorems 4.3 and 4.5	24
A.1 Notation and definitions	24
A.2 A basic result about the optimal achievable cross-entropy loss	25
A.3 Proof of Theorem 2.1	25
A.4 Maximum likelihood unigram model	25
A.5 Proof of Theorem 4.3	27
A.6 Heavy-hitter dictionaries and a proof of Theorem 4.5	28
B Analysis of a sequential variant of BPE	33
B.1 Analysis for the large dictionary case: $ \text{Dict} > d_0$	34
B.2 Analysis in the small dictionary case	42
C Additional Theoretical Results I: Learning the likelihood model	45
C.1 Proof of Theorem C.1	47
D Additional Theoretical Results II: The generalization ability of tokenizers	51
E Additional Theoretical Results III: Interaction between the dictionary and encoding algorithm	52
E.1 Stochastic source and dictionary.	53
E.2 Minimal encoder achieves the optimal cross-entropy loss up to a constant.	54
E.3 Greedy-encoder achieves poor cross-entropy loss	56
F Experiment details	58

A Analysis of LZW: Proofs of Theorems 4.3 and 4.5

A.1 Notation and definitions

For each character $a \in \mathcal{A}$ let \mathcal{T}_a^* denote an infinite tree, with root vertex \emptyset , and subsequent vertices labelled by strings $t \in \mathcal{A}^*$. The edge from a parent vertex t to any child ta' is labelled with the probability $P(ta'|t)$ unless $t = \emptyset$, in which case the edge probability is $P(a'|a)$. An infinite trajectory sampled on the tree \mathcal{T}_a^* corresponds to an infinite string sampled from the stochastic source conditioned on the first character of the string being a . In this paper we only consider ergodic sources (Gray and Gray, 2009) for which we can define the ‘‘entropy rate’’. The entropy rate fundamentally captures the compressibility of the source, and can be defined as $H_\infty \triangleq \lim_{m \rightarrow \infty} \frac{1}{m} H(P)$ where s is a length m string drawn from the source. By Theorem 2.1, H_∞ , captures $\min_Q \mathcal{L}(Q)$.

A.2 A basic result about the optimal achievable cross-entropy loss

The ratio of $H(P)$ and $mH(\pi)$ can be made arbitrarily large for the switching Markov chains in Figure 1 as the switching probabilities p and q approach 0 or 1. See Example A.1 for more details.

Example A.1. Consider the switching Markov process in Figure 1 on $\{0, 1\}$ with $p = q = 1 - \delta$. For this process, $\lim_{m \rightarrow \infty} \frac{1}{m} H(P) = H_{\text{Ber}}(\delta) = \delta \log(1/\delta) + (1 - \delta) \log(1/(1 - \delta))$, but $\pi = \{1/2, 1/2\}$ and so $H(\pi) = H_{\text{Ber}}(1/2) = \log(2)$. The ratio $\lim_{m \rightarrow \infty} \frac{mH(\pi)}{H(P)}$ goes to ∞ as $\delta \rightarrow 0$.

A.3 Proof of Theorem 2.1

We first characterize the minimum achievable cross-entropy loss $\mathcal{L}_m(Q)$ without any restrictions on the likelihood model class \mathcal{Q} . Choosing $Q(\text{enc}(\mathbf{s})) = Q(\mathbf{s}) = P(\mathbf{s})$, the true probability of the sequence \mathbf{s} , we have $\mathcal{L}_m(Q \circ \text{enc}(\cdot)) = H(\mathbf{s})$ where $H(\cdot)$ is the entropy function. It is not that difficult to see that this is also the minimum cross-entropy loss that can be achieved. For any distribution Q ,

$$\begin{aligned}\mathcal{L}_m(Q) &= \mathbb{E}[\log(1/Q(\mathbf{s}))] \\ &= \mathbb{E}[\log(P(\mathbf{s})/Q(\mathbf{s}))] + \mathbb{E}[\log(1/P(\mathbf{s}))] \\ &= H(P) + D_{\text{KL}}(P\|Q).\end{aligned}$$

On the other hand, the cross-entropy loss under any unigram model $Q \in \mathcal{Q}_{1\text{-gram}}$ satisfies,

$$\begin{aligned}\frac{1}{m} \mathcal{L}_m(Q \circ \text{enc}(\cdot)) &\stackrel{(i)}{=} -\frac{1}{m} \sum_{i=1}^m \mathbb{E}[\log Q_{\text{tok}}(\mathbf{t}_i)] - \frac{1}{m} \mathbb{E}[\log Q_{\#}(m)] \\ &\stackrel{(ii)}{\geq} - \sum_{a \in \mathcal{A}} \pi(a) \log Q_{\text{tok}}(a) \\ &\geq H(\pi)\end{aligned}$$

where in (i), we use the definition of the unigram model Q , and in (ii), π is the stationary distribution over characters induced by the stochastic source, and the ergodicity of the source is used. The last equation lower bounds $H(X, Y) \geq H(X)$.

A.4 Maximum likelihood unigram model

A number of our results (Theorems 4.3 and 4.5 to name a few) are related to bounding $\min_{Q \in \mathcal{Q}_{1\text{-gram}}} \mathcal{L}(Q \circ \text{enc}(\cdot))$ for some tokenizer \mathcal{T} . In this section we introduce the maximum likelihood unigram model which captures the optimizer over Q for any given tokenizer.

For the character level tokenizer, an examination of Theorem 2.1 shows that the optimal unigram likelihood model associates probability $Q_{\text{tok}}(a) = \pi(a)$, i.e. the limiting fraction of times the character a is observed in the sequence. More generally, for a non-trivial tokenizer, the corresponding optimal unigram model $Q_{\text{tok}}^*(\mathbf{t})$ ends up being the limiting expected fraction of times \mathbf{t} is observed in an encoding of a sequence. This is the maximum likelihood unigram model, which we formally define below. The unigram MLE likelihood model associates probability,

$$Q_{\text{MLE}}(\mathbf{t}) \leftarrow \lim_{m \rightarrow \infty} \mathbb{E} \left[\frac{n_{\mathbf{t}}}{\sum_{\mathbf{t}} n_{\mathbf{t}}} \right] \quad (6)$$

to each token, where $n_{\mathbf{t}}$ is the random variable capturing the number of occurrences of the token \mathbf{t} in the encoding of the length- m string \mathbf{s} . Restricting the class of likelihood models to the unigram models, $\mathcal{Q}_{1\text{-gram}}$, Q_{MLE} captures the model which minimizes eq. (1).

The unigram MLE model cannot be computed without an infinite amount of data, but can be approximated well with a finite amount of data, which forms the basis for Theorem C.1. For certain encoding algorithms, we can show that the quantity $n_t / \sum_t n_t$ asymptotically converges to its expectation (Lemma A.4). This is the reason the unigram model in eq. (6) is referred to as a “maximum likelihood” model, since $\lim_{m \rightarrow \infty} n_t / \sum_t n_t$ is the limit as $|\mathbf{s}| = m \rightarrow \infty$ of the solution to the following likelihood maximization problem: given a sequence \mathbf{s} , find the distribution over tokens, Q , which maximizes

$$\prod_{t \in \text{enc}(\mathbf{s})} Q(t) \equiv \prod_{t \in \text{Dict}} (Q(t))^{n_t}.$$

As discussed previously, the unigram MLE model over tokens in eq. (6) induces a joint distribution over sequences of tokens by looking at the product of the marginal probabilities of the composed tokens; in particular,

$$Q_{\text{MLE}}(\mathbf{t}_1, \dots, \mathbf{t}_j) = Q_{\text{MLE}}(j) \prod_{i=1}^j Q_{\text{MLE}}(\mathbf{t}_i),$$

where $Q_{\text{MLE}}(j)$ is a distribution on the total number of tokens generated and is instantiated as $\text{Unif}([m])$.

Remark A.2. Note that the unigram MLE model specifies a distribution over tokens which is a function of the underlying encoding algorithm, $\text{enc}(\cdot)$. Different encoders result in different population level distributions over tokens, and consequently different unigram MLE models.

Definition A.3 (greedy encoder). Given a dictionary Dict , the greedy encoder $\text{enc}_{\text{gre}}(\mathbf{s})$ encodes a string \mathbf{s} into tokens by greedily matching from left to right, the largest substring that exists as a token in Dict . This substring is then removed and the process iterated on the remainder of \mathbf{s} . The greedy decoder $\text{dec}_{\text{gre}}(\cdot)$ is a lookup table - a sequence of tokens is decoded by replacing each occurrence of a token by the corresponding substring it maps to in Dict .

Lemma A.4. $\lim_{m \rightarrow \infty} \frac{n_t}{\sum_{t'} n_{t'}} \stackrel{a.s.}{=} \lim_{m \rightarrow \infty} \mathbb{E} \left[\frac{n_t}{\sum_{t'} n_{t'}} \right]$ for any tokenizer having a finite vocabulary and finitely long tokens, using the greedy encoder.

Proof. This result is essentially true because under the greedy encoder, the tokens in an encoding of a fresh string \mathbf{t} may be generated by an r^{th} -order Markov process for some r . For such processes, the Cesàro average of the state distributions converges to a stationary distribution of the process (i.e., the Krylov–Bogolyubov argument).

Tokens are generated as follows. Suppose the previous tokens generated were $\mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{t}_i$. The next token \mathbf{t}_{i+1} is sampled by drawing an infinite trajectory from \mathcal{T}_a^* for $a \sim P(\cdot | \mathbf{t}_i)$ and returning the longest prefix \mathbf{t} of this trajectory which is a token in Dict , conditional on satisfying the conditions, $\mathbf{t}_j \mathbf{t}_{j+1} \dots \mathbf{t}_i \mathbf{t}_{i+1} \notin \text{Dict}$ for all $j \in \{1, 2, \dots, i\}$. This process is repeated sequentially to generate all the tokens.

Suppose the length of the longest token in the dictionary is ℓ_{\max} . Then, the distribution from which a token is sampled depends on at most the previous ℓ_{\max} tokens. The reason for this is that the dependency of the $(i+1)^{th}$ token, \mathbf{t}_{i+1} , on the previously sampled tokens emerges in the constraint $\mathbf{t}_j \mathbf{t}_{j+1} \dots \mathbf{t}_i \mathbf{t}_{i+1} \notin \text{Dict}$, satisfied by any candidate \mathbf{t}_{i+1} . Since each token is of length at least one, this condition is vacuously satisfied if $j < i - \ell_{\max}$.

With this view, the evolution of the state, defined as $\text{state}_r = (\mathbf{t}_{r\ell_{\max}}, \mathbf{t}_{r\ell_{\max}-1}, \dots, \mathbf{t}_{(r-1)\ell_{\max}})$ evolves in a Markovian fashion. By the Krylov–Bogolyubov argument (cf. Proposition 4.2 in

Chen (2018)), the time averaged visitation frequencies of a Markov chain coordinate-wise asymptotically converges to its expectation, almost surely. This expectation exists by Theorems 8.5 and 8.22 of Eisner et al. (2015) which shows that for a matrix A such that $\sup_{t \in \mathbb{N}} \|A^t\|_{\text{op}} < \infty$ the limit $\lim_{t \rightarrow \infty} \frac{1}{t} \sum_{i=1}^t A^i$ exists. For the finite-state Markov transition A which captures the token generation process, condition $\sup_{t \in \mathbb{N}} \|A^t\|_{\text{op}} \leq |\text{Dict}|^{\ell_{\max}} < \infty$. This means that the limit of the time averaged state distribution exists. Moreover, for any initial distribution π_0 over tokens, $\pi = \lim_{t \rightarrow \infty} \frac{1}{t} \sum_{i=1}^t \pi_0 A^i$ satisfies the condition $\pi A = \pi$, implying that the limiting time-averaged state distribution is a stationary distribution of A . Since the limiting time-averaged measure on the state $\text{state}_r = (\mathbf{t}_{r\ell_{\max}}, \dots, \mathbf{t}_{r\ell_{\max}-1}, \dots, \mathbf{t}_{(r-1)\ell_{\max}})$ exists, this implies that the limiting time-averaged measure of $\mathbf{t}_{r\ell_{\max}-r'}$ for each $r' \in \{0, 1, \dots, \ell_{\max}\}$ exists. By taking the uniform average over r' and r , the limiting time-averaged measure of \mathbf{t}_i over $i \in \mathbb{N}$ exists. \square

A.5 Proof of Theorem 4.3

Consider a string \mathbf{s} of length $m \rightarrow \infty$ which is encoded into a sequence of tokens $(\mathbf{t}_i : i \in [|\text{enc}_{\text{gre}}(\mathbf{s})|])$. By the Asymptotic Equipartition Property (AEP) for ergodic sources, i.e. the Shannon–McMillan–Breiman theorem,

$$\Pr \left(\lim_{m \rightarrow \infty} -\frac{1}{m} \log P(\mathbf{s}) = H_{\infty} \right) = 1. \quad (7)$$

Here $\lim_{m \rightarrow \infty} \frac{H(P)}{m}$ also happens to be the entropy rate of the source. We use this property to bound the length of the greedy encoding, $|\text{enc}_{\text{gre}}(\mathbf{s})|$. Indeed, the probability of \mathbf{s} may be decomposed as,

$$P(\mathbf{s}) = P(\mathbf{t}_1) \prod_{i=2}^{|\text{enc}_{\text{gre}}(\mathbf{s})|} P(\mathbf{t}_i | \mathbf{t}_{i-1}) \leq \prod_{i=1}^{|\text{enc}_{\text{gre}}(\mathbf{s})|} \max_{a \in \mathcal{A}} P(\mathbf{t}_i | a).$$

Noting that $\delta \min_a P(\mathbf{t}|a) \geq \max_a P(\mathbf{t}|a)$, up to a δ factor we may replace the max over a by an expectation over $a \sim \pi$ where π is the stationary distribution of the stochastic source. In particular,

$$P(\mathbf{s}) \leq \prod_{i=1}^{|\text{enc}_{\text{gre}}(\mathbf{s})|} P(\mathbf{t}_i) / \delta.$$

By invoking the AEP, eq. (7),

$$\lim_{m \rightarrow \infty} \frac{1}{m} \sum_{i=1}^{|\text{enc}_{\text{gre}}(\mathbf{s})|} -\log(P(\mathbf{t}_i)) - \log(1/\delta) \stackrel{\text{a.s.}}{\leq} H_{\infty}$$

Recall that the greedy encoder satisfies Lemma A.4 and for any $\mathbf{t} \in \text{Dict}$, $\lim_{m \rightarrow \infty} \frac{n_{\mathbf{t}}}{|\text{enc}_{\text{gre}}(\mathbf{s})|} \stackrel{\text{a.s.}}{=} Q_{\text{MLE}}(\mathbf{t})$. Furthermore, note that for any token $\mathbf{t} \in \text{Dict}$, $P(\mathbf{t}) > \delta^{|\mathbf{t}|} > 0$, and $|\text{enc}_{\text{gre}}(\mathbf{s})| \leq m$ surely. By almost sure convergence,

$$\begin{aligned} & \lim_{m \rightarrow \infty} \frac{|\text{enc}_{\text{gre}}(\mathbf{s})|}{m} \sum_{\mathbf{t} \in \text{Dict}} -\frac{n_{\mathbf{t}}}{|\text{enc}_{\text{gre}}(\mathbf{s})|} (\log(P(\mathbf{t})) - \log(1/\delta)) \\ & \stackrel{\text{a.s.}}{\lim_{m \rightarrow \infty}} \frac{|\text{enc}_{\text{gre}}(\mathbf{s})|}{m} (H(Q_{\text{MLE}}, P) - \log(1/\delta)) \end{aligned}$$

Furthermore, utilizing the assumption that $\varepsilon H(Q_{\text{MLE}}, P) \geq \log(1/\delta)$ satisfied by the tokenizer,

$$\lim_{m \rightarrow \infty} \frac{(1 - \varepsilon) |\text{enc}_{\text{gre}}(\mathbf{s})| (H(Q_{\text{MLE}}, P))}{m} \stackrel{\text{a.s.}}{\leq} H_{\infty}. \quad (8)$$

Now we are ready to bound the expected cross-entropy loss of the tokenizer. Define the unigram model $P_\pi(\mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{t}_j) = P_{\text{unif}}(j) \prod_{i=1}^j P(\mathbf{t}_i)$ where P_{unif} is the uniform measure over $[m]$. Note that we have the inequality $\min_{Q \in \mathcal{Q}_{1\text{-gram}}} \lim_{m \rightarrow \infty} \frac{1}{m} \mathcal{L}_m(Q \circ \text{enc}_{\text{gre}}(\cdot)) \leq \lim_{m \rightarrow \infty} \frac{1}{m} \mathcal{L}_m(P_\pi \circ \text{enc}_{\text{gre}}(\cdot))$ and therefore, it suffices to upper bound the RHS. In particular,

$$\begin{aligned} \mathcal{L}_m(P_\pi \circ \text{enc}_{\text{gre}}(\cdot)) &= -\mathbb{E}[\log P_{\text{unif}}(|\text{enc}_{\text{gre}}(\mathbf{s})|)] - \mathbb{E}\left[\sum_{\mathbf{t} \in \text{enc}_{\text{gre}}(\mathbf{s})} \log(P(\mathbf{t}))\right] \\ &\leq \log(m) - \mathbb{E}\left[\sum_{\mathbf{t} \in \text{enc}_{\text{gre}}(\mathbf{s})} \log(P(\mathbf{t}))\right] \end{aligned} \quad (9)$$

where the last inequality uses the fact that $P_{\text{unif}}(|\text{enc}_{\text{gre}}(\mathbf{s})|) = 1/m$. Note that as $m \rightarrow \infty$, by assumption on the tokenizer, the fraction of times the token \mathbf{t} appears in the encoding of \mathbf{s} converges almost surely converges to $Q_{\text{MLE}}(\mathbf{t})$. Since $|\text{enc}_{\text{gre}}(\mathbf{s})| \leq m$ surely and $P(\mathbf{t}) > \delta^{|\mathbf{t}|} > 0$, by an application of the Dominated Convergence Theorem,

$$\begin{aligned} -\lim_{m \rightarrow \infty} \frac{1}{m} \mathbb{E}\left[\sum_{\mathbf{t} \in \text{enc}_{\text{gre}}(\mathbf{s})} \log(P(\mathbf{t}))\right] &= -\lim_{m \rightarrow \infty} \frac{1}{m} \mathbb{E}\left[|\text{enc}_{\text{gre}}(\mathbf{s})| \cdot \sum_{\mathbf{t} \in \text{Dict}} Q_{\text{MLE}}(\mathbf{t}) \log(P(\mathbf{t}))\right] \\ &= \lim_{m \rightarrow \infty} \frac{1}{m} \mathbb{E}[|\text{enc}_{\text{gre}}(\mathbf{s})| H(Q_{\text{MLE}}, P)] \end{aligned} \quad (10)$$

Combining eq. (9) with eq. (10) and setting $\lim_{m \rightarrow \infty} \log(m)/m = 0$, and invoking eq. (8),

$$\begin{aligned} \min_{Q \in \mathcal{Q}_{1\text{-gram}}} \lim_{m \rightarrow \infty} \frac{1}{m} \mathcal{L}_m(Q_{\text{MLE}} \circ \text{enc}_{\text{gre}}(\cdot)) &= \lim_{m \rightarrow \infty} \frac{1}{m} \mathbb{E}[|\text{enc}_{\text{gre}}(\mathbf{s})| H(Q_{\text{MLE}}, P)] \\ &\leq \frac{H_\infty}{1 - \varepsilon}. \end{aligned} \quad (11)$$

By Theorem 2.1, we have that $\min_Q \lim_{m \rightarrow \infty} \frac{1}{m} \mathcal{L}_m(Q \circ \text{enc}_{\text{gre}}(\cdot)) = \lim_{m \rightarrow \infty} \frac{H(P)}{m} = H_\infty$, which uses the fact that the source is ergodic. Combining with eq. (11) completes the proof.

A.6 Heavy-hitter dictionaries and a proof of Theorem 4.5

In this section we prove Theorem 4.5 and introduce the notion of a heavy-hitting dictionary. At a high level, these dictionaries contain all the substrings which have reasonably high probability of being observed many times in a dataset of size $n = \tilde{\Omega}_\delta(d)$. We first show in Lemma A.6 that heavy hitting dictionaries generalize well in the sense of having $H(Q_{\text{MLE}}, P)$ being large (in conjunction with Theorem 4.3 this implies an upper bound on the cross-entropy loss of the best unigram model). Next, we will prove that the LZW algorithm (Definition 4.4) results in a heavy hitting dictionary with high probability.

Definition A.5 (β -heavy-hitting dictionary). A token \mathbf{t} of a dictionary is said to be maximal if there exists an arbitrary substring containing \mathbf{t} as a strict prefix, and in addition, \mathbf{t} is also the largest prefix of the substring which is a token. A dictionary Dict is said to be β -heavy hitting if the set of maximal tokens is a subset of $\{\mathbf{s}' : \max_{a \in \mathcal{A}} P(\mathbf{s}'|a) \leq 1/d^\beta\}$.

A pictorial depiction of the heavy hitting property is illustrated in Figure 8.

Lemma A.6. *For a β -heavy-hitting dictionary, with the greedy encoder, $H(Q_{\text{MLE}}, P) \geq \beta \log(d)$.*

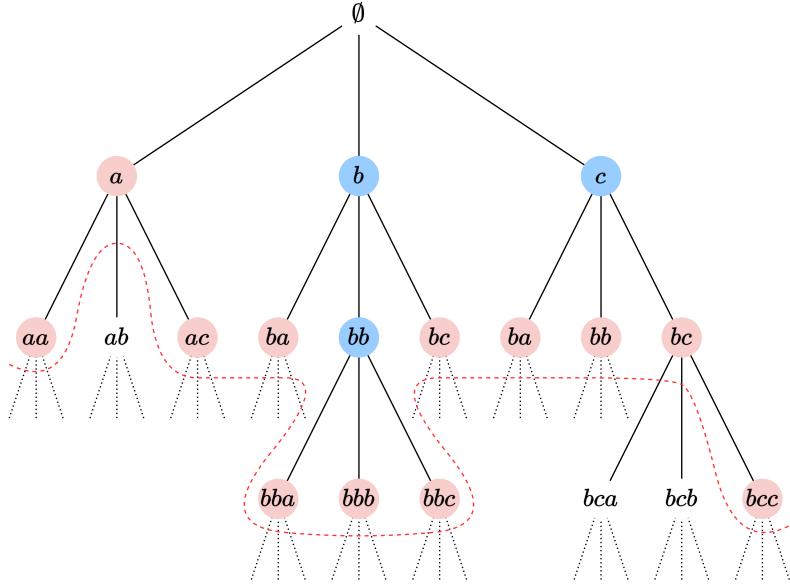


Figure 8: The circled nodes indicates substrings which are tokens in Dict. Red nodes indicate the set of “maximal tokens”, which are the set of tokens which the greedy encoder assigns, leaving out those which can only be assigned as the last token of some string. Tokens like “ b ” are never assigned by the greedy encoder (save as the last token of the encoding of a string) since any sufficiently long trajectory starting with b must have a longer prefix which is also a token, namely, one of ba , bc , bba , bbb or bbc . The vertices of the tree which are assigned by the greedy encoder as tokens (together with all their prefixes) forms a cut of the tree, which marks the dotted red line. The heavy hitting property asserts that this cut is uniformly far away from the root node \emptyset , and that every vertex s marked red has $P(s) \leq 1/d^\beta$.

Proof. Note that the greedy encoder assigns tokens only among the set of maximal substrings (save for potentially the last token). If every maximal substring has $\max_{a \in \mathcal{A}} P(s|a) \leq 1/d^\beta$, by the heavy-hitting property, for any token t ,

$$P(t) \leq \max_{a \in \mathcal{A}} P(s'|a) \leq 1/d^\beta.$$

Therefore,

$$H(Q_{\text{MLE}}, P) = \mathbb{E}_{t \sim Q_{\text{MLE}}} [\log(1/P(t))] \geq \beta \log(d).$$

□

Define $\mathcal{M}_\beta = \{t : \max_{a \in \mathcal{A}} P(t|a) \geq \delta/d^\beta\}$. These are the set of “high-probability” substrings under the stochastic source. We will show that for β bounded away from 1, with high probability, every substring in \mathcal{M}_β is added as a token to the dictionary in a run of the LZW tokenizer (Definition 4.4). Note that if every substring in \mathcal{M}_β is assigned as a token by LZW, then the algorithm must be β -heavy hitting since there always exists a maximal token on the “boundary” of the set \mathcal{M}_β which is strictly contained in $\{s' : \max_{a \in \mathcal{A}} P(s'|a) \leq 1/d^\beta\}$.

Lemma A.7. Every substring in \mathcal{M}_β has length at most $\ell_\star \triangleq \delta^{-1}(\beta \log(d) + \log(1/\delta))$.

Proof. Note that $\min_{a,a' \in \mathcal{A}} P(a|a') = \delta$, which implies that the probability of any transition must be bounded away from 1, i.e., $\max_{a,a' \in \mathcal{A}} P(a|a') \leq 1 - \delta$. This implies that,

$$\max_{a \in \mathcal{A}} P(\mathbf{t}|a) \leq (1 - \delta)^{|\mathbf{t}|} \leq e^{-\delta|\mathbf{t}|}. \quad (12)$$

By definition, for any substring $\mathbf{t} \in \mathcal{M}_\beta$, $\max_{a \in \mathcal{A}} P(\mathbf{t}|a) \geq \delta/d^\beta$. In conjunction with eq. (12), this implies the statement of the lemma. \square

In the remainder of this section, let n be the size of the dataset on which LZW is run. We show that the number of tokens added to the dictionary by LZW, d , is $\tilde{\Theta}_\delta(n)$. Rather than running the algorithm with early stopping (i.e., ceasing to add new tokens once the budget is hit), instead, we assume that the algorithm runs on a prefix of the dataset of length d . The number of tokens added this way cannot exceed d .

Lemma A.8. With probability $\geq 1 - d^{-\Omega(\log(d/\delta)/\delta)}$, in a run of the LZW algorithm, no substring \mathbf{t} added as a token to the dictionary satisfies $|\mathbf{t}| \geq \ell_{\max} \triangleq 4 \log(d|\mathcal{A}|)/\delta$.

Proof. Consider any $s \in \mathbb{N}$ and any substring \mathbf{t} of length s . In order for \mathbf{t} to be assigned as a token, each of its prefixes must disjointly appear at least once in the string. Since there are at most d tokens, we can upper bound the probability that \mathbf{t} is assigned as a token as,

$$\begin{aligned} P(\mathbf{t} \text{ is assigned as a token}) &\leq \binom{d}{s} \prod_{i=1}^s \max_{a \in \mathcal{A}} P(\mathbf{t}_{1:i}|a) \\ &\stackrel{(i)}{\leq} \binom{d}{s} (1 - \delta)^{s(s-1)/2} \\ &\leq e^{s \log(d) - \delta s(s-1)/2}, \end{aligned}$$

where (i) uses the fact that $\max_{a \in \mathcal{A}} P(\mathbf{t}_{1:i}) \leq \prod_{j=1}^i \max_{a \in \mathcal{A}} P(\mathbf{t}_j|a) \leq (1 - \delta)^i$. By union bounding across the $|\mathcal{A}|^s$ strings of length s ,

$$P(\text{any length } s \text{ string is assigned as a token}) \leq e^{s \log(|\mathcal{A}|) + s \log(d) - \delta s(s-1)/2}.$$

When $s = 4 \log(d|\mathcal{A}|)/\delta + 1 \triangleq \ell_{\max} + 1$, the RHS is upper bounded by $e^{-\delta \ell_{\max}^2/4} \leq d^{-\Omega(\log(d/\delta)/\delta)}$. With the same small probability, no substring of length $s' > s$ can become a token, since their length- s prefixes are never assigned as tokens. \square

Corollary A.9. With probability $\geq 1 - d^{-\Omega_\delta(\log(d))}$, learns a dictionary with at least $d^* = d/\ell_{\max}$ tokens when run on a training sequence of length n drawn from a stochastic source satisfying Assumption 4.2.

Lemma A.10. For any constant $\beta < 1$, with probability $\geq 1 - d^{-\Omega(\log(d/\delta)/\delta)} - \exp(-\tilde{\Omega}_\delta(d^{1-\beta}))$ over the source dataset, every substring in \mathcal{M}_β is added as a token to the dictionary in a run of the LZW algorithm. In other words, with the same probability, the LZW tokenizer results in a β -heavy hitting dictionary.

By Corollary A.9, note that with high probability the LZW tokenizer adds at least d^* tokens to the dictionary when processing a length d training sequence in entirety. In this proof, instead of generating d samples, we sequentially sample d^* tokens from their joint distribution, and generate a

dictionary from these samples. From Corollary A.9, with high probability this results in at most d samples being generated, implying that the dictionary generated by sampling d^* tokens is a subset of the dictionary generated by a full run of the LZW tokenizer. Here, we use the fact that the LZW tokenizer adds tokens to the dictionary in a left to right fashion, and therefore a subset of the dictionary learnt by the LZW tokenizer can be generated by processing a portion of the dataset.

Next we consider a joint view for generating the dataset from the stochastic source and the dictionary learnt by LZW simultaneously. The stochastic source is sampled as a sequence of tokens. Suppose the last character of the previous token was a' . Sample a character $a \sim P(\cdot|a')$ and an infinite trajectory on the tree \mathcal{T}_a^* . Consider the first node visited in this trajectory which does not already exist as a token in the dictionary. The substring corresponding to this node is added as a token in the dictionary. By repeating this process, the dictionary and the source dataset are constructed sequentially and simultaneously. As alluded to before, we truncate this token sampling process to repeat at most d^* times, which results in a subset of the dictionary output by the LZW algorithm with high probability (Corollary A.9). This is simply a variant of the “Poissonization” trick to avoid statistical dependencies across tokens. Denote the set of infinite trajectories generated on the forest $\{\mathcal{T}_a^* : a \in \mathcal{A}\}$ as $\{\text{traj}_i : i \in [d^*]\}$.

With this view of the sampling process, observe that if the substring \mathbf{t} sampled was a prefix of traj_i at least $|\mathbf{t}|$ times across different values of i , then \mathbf{t} must be assigned as a token. In particular, in each of these $|\mathbf{t}|$ trajectories, each of the prefixes of \mathbf{t} is assigned as a token. With this observation, the event that \mathbf{t} is not assigned as a token is contained in the event that \mathbf{t} is visited at most $|\mathbf{t}| - 1$ times across the d^* trajectories. Observe that,

$$P(\mathbf{t} \text{ is not assigned as a token}) \leq \sum_{i=0}^{|\mathbf{t}|-1} \binom{d^*}{i} \max_{a \in \mathcal{A}} (P(\mathbf{t}|a))^i (1 - P(\mathbf{t}|a))^{d^*-i}.$$

Since we aim to upper bound this probability across the substrings in $\mathbf{t} \in \mathcal{M}_\beta$, note that (i) $\max_{a \in \mathcal{A}} P(\mathbf{t}|a) \geq \delta/d^\beta$, and (ii) tokens in \mathcal{M}_β have length at most $\ell_\star = \delta^{-1}(\beta \log(d) + \log(1/\delta))$ (Lemma A.7), implying there are at most $2|\mathcal{A}|^{\ell_\star}$ substrings in this set. By union bounding,

$$P(\exists \mathbf{t} \in \mathcal{M}_\beta \text{ not assigned as a token}) \leq 2|\mathcal{A}|^{\ell_\star} \sum_{i=0}^{\ell_\star-1} \binom{d^*}{i} \max_{x \geq \delta/d^\beta} x^i (1-x)^{d^*-i}. \quad (13)$$

Case I. For $i \leq \ell_\star$ and $x \geq 1/2$,

$$\begin{aligned} |\mathcal{A}|^{\ell_\star} \binom{d^*}{i} x^i (1-x)^{d^*-i} &\leq |\mathcal{A}|^{\ell_\star} \frac{(d^*)^{\ell_\star}}{2^{d^*/2}} \\ &\leq 2^{\ell_\star \log(d^*) - d^*/2} \\ &\leq 2^{-\Omega_{\beta,\delta}(d^*)}, \end{aligned} \quad (14)$$

where the last inequality uses the fact that $\ell_\star = O_{\beta,\delta}(\log(d))$.

Case II. For $i \leq \ell_*$ and $\delta/d^\beta \leq x \leq 1/2$,

$$\begin{aligned}
|\mathcal{A}|^{\ell_*} \binom{d^*}{i} x^i (1-x)^{d^*-i} &\leq |\mathcal{A}|^{\ell_*} \binom{d^*}{i} (1-x)^{d^*} \\
&\leq |\mathcal{A}|^{\ell_*} (d^*)^{\ell_*} e^{-d^* x} \\
&\leq e^{\ell_* \log(|\mathcal{A}|) + \ell_* \log(d^*) - d^* x} \\
&\leq e^{-\Omega(\delta^2 n / d^\beta / \log(d/\delta))} \\
&\leq e^{-\Omega(\delta^2 d^{1-\beta} / \log(d/\delta))}, \tag{15}
\end{aligned}$$

where the last inequality uses the fact that $\ell_* = O(\log(d))$, $x \geq \delta/d^\beta$, $d^* = \Omega(d\delta / \log(d/\delta))$. By combining eq. (14) and eq. (15) with eq. (13) completes the proof, as long as β is a constant bounded away from 1.

Lemma A.11. Fix a constant $\gamma > 0$. Then, with probability $\geq 1 - d^{-\Omega_{\gamma,\delta}(\log(d))}$, none of the substrings in the set $\mathcal{N}_\gamma = \{\mathbf{s}' : \max_{a \in \mathcal{A}} P(\mathbf{s}'|a) \leq \delta/d^{1+\gamma}\}$ are assigned as tokens in a run of LZW.

Proof. Define the following set of substrings,

$$S_\gamma = \left\{ \mathbf{t} : \delta/d^{1+\gamma/2} \leq \max_{a \in \mathcal{A}} P(\mathbf{t}|a) \leq 1/d^{1+\gamma/2} \right\}$$

Since the width of this band is sufficiently large, by Assumption 4.2 every substring \mathbf{t} such that $\max_{a \in \mathcal{A}} P(\mathbf{t}|a) \leq \delta/d^{1+\gamma/2}$ has at least one prefix which falls in S_γ , and denote the longest such prefix \mathbf{t}_γ . Define $T_\gamma = \{\mathbf{t}_\gamma : \mathbf{t} \in \mathcal{N}_\gamma\}$ as the set of longest prefixes in S_γ . Intuitively, if we think of the strings in S_γ (or T_γ) as being intermediate in length, the strings in \mathcal{N}_γ can be thought of as being particularly long: the value of $\max_{a \in \mathcal{A}} P(\mathbf{t}|a)$ for any $\mathbf{t} \in T_\gamma$ and for any $\mathbf{t} \in \mathcal{N}_\gamma$ are separated by a factor of at least $1/d^{\gamma/2}$. In particular, since the probability of any character is lower bounded by δ , each substring in $\mathbf{t} \in \mathcal{N}_\gamma$ must be at least $\Delta = \frac{\gamma \log(d)}{2 \log(1/\delta)}$ symbols longer than its corresponding longest prefix in T_γ , \mathbf{t}_γ . An implication of this is that for \mathbf{t} to be assigned as a token, \mathbf{t}_γ must be observed at least $\Delta + 1$ times disjointly in \mathbf{s} . However, note that \mathbf{t}_γ already has low marginal probability to begin with ($\ll 1/d$) so the odds of seeing this substring so many times disjointly is very small. Furthermore, note that T_γ has at most $d^{1+\gamma/2}/\delta$ substrings, which allows the probability of this event occurring simultaneously across all substrings in T_γ to be controlled by union bound. Under this condition, none of the substrings in \mathcal{N}_γ are made into tokens.

In order to argue that the dictionary *does not* contain certain tokens, we may argue this property about any superset of the dictionary. In contrast, in Lemma A.10, we construct a subset of the dictionary by running LZW on the concatenation of d^* tokens sampled from their joint distribution. The superset we consider here is just to sample d tokens from their joint distribution and concatenate them together to result in a string of length $\geq d$, and running LZW on this sequence (which simply would result in these d tokens). As in Lemma A.10, let $\{\text{traj}_i : i \in [d]\}$ denote the infinite trajectories generated from the Markov chain which are truncated to result in tokens. A sufficient condition for the event that no substring $\mathbf{t} \in \mathcal{N}_\gamma$ is assigned as a token by LZW is to the event that every substring $\mathbf{t}' \in T_\gamma$ is observed as a prefix of traj_i for Δ or fewer choices of $i \in [d]$. To this end define $\mathcal{E}(\mathbf{t}')$ as the event that $|i \in [d] : \mathbf{t}' \text{ is a prefix of } \text{traj}_i| \leq \Delta$. Then,

$$\begin{aligned}
\Pr(\mathcal{E}(\mathbf{t}')) &\leq \binom{n}{\Delta} (\max_{a \in \mathcal{A}} P(\mathbf{t}'|a))^\Delta \\
&\stackrel{(i)}{\leq} e^{\Delta \log(n)} \left(\frac{1}{d^{1+\gamma/2}} \right)^\Delta \\
&\leq e^{-\frac{\gamma}{2} \Delta \log(d)}, \tag{16}
\end{aligned}$$

where (i) uses the fact that $\max_{a \in \mathcal{A}} P(\mathbf{t}'|a) \leq 1/d^{1+\gamma/2}$ since the substring \mathbf{t}' belongs to T_γ .

Note that the number of substrings in S_γ (and by extension, T_γ) is at most $O_\delta(d^{1+\gamma/2})$. Recall that these substrings satisfy the condition $\max_{a \in \mathcal{A}} P(\mathbf{t}|a) \geq \delta/d^{1+\gamma/2}$. Observe that,

$$\begin{aligned} \frac{\delta|S_\gamma|}{d^{1+\gamma/2}} &\leq \sum_{\mathbf{t} \in S_\gamma} \max_{a \in \mathcal{A}} P(\mathbf{t}|a) \\ &\leq \sum_{\mathbf{t} \in S_\gamma} \sum_{a \in \mathcal{A}} P(\mathbf{t}|a) \leq |\mathcal{A}| \leq \frac{1}{\delta}. \end{aligned}$$

This implies that there are at most $d^{1+\gamma/2}/\delta^2$ substrings in S_γ . Finally, in conjunction with eq. (16),

$$P(\exists \mathbf{t}' \in S_\gamma : \mathcal{E}(\mathbf{t}')) \leq \frac{d^{1+\gamma/2}}{\delta^2} e^{-\frac{\gamma}{2}\Delta \log(d)} = d^{-\Omega_{\gamma,\delta}(\log(d))},$$

which implies that with high probability, no token in S_γ is observed as a prefix of \mathbf{s}^i for more than Δ choices of the index $i \in [d]$. Under this event, no substring in \mathcal{N}_γ is assigned as a token. \square

A.6.1 Proof of Theorem 4.5

Choosing $\beta = 0.99$ in Lemma A.10, with probability $\geq 1 - d^{-\Omega_\delta(\log(d))}$, the LZW tokenizer results in a 0.99-heavy-hitting dictionary. As a consequence of Lemma A.6, this implies that under the same event,

$$H(Q_{\text{MLE}}, P) \geq 0.99 \log(d).$$

Finally, combining with Theorem 4.3 completes the proof.

B Analysis of a sequential variant of BPE

In this section we analyze the sequential variant of BPE introduced in Algorithm 1. We prove a rephrased version of Theorem 4.7 which implies the statement in the main paper. Define $d_0 = \frac{\epsilon d}{2 \log(4|\mathcal{A}|)}$.

Theorem B.1 (Rephrased Theorem 4.7). *Run Algorithm 1 on a dataset of $n = \Theta(d^2)$ characters to learn a dictionary with at most d tokens. The resulting tokenizer \mathcal{T} satisfies one of the following 3 conditions,*

1. Either, $|\text{Dict}| > d_0$, and,

$$\min_{Q \in \mathcal{Q}_{1\text{-gram}}} \mathcal{L}(Q \circ \text{enc}(\cdot)) \leq \frac{H_\infty}{1 - \varepsilon}.$$

Here, $\varepsilon = O\left(\frac{\log^3(1/\delta) \log(1/\epsilon)}{\epsilon \delta^9 \log(d)}\right)$.

2. $\Pr(|\text{Dict}| < d_0) = e^{-\Omega(\epsilon^2 d / \log^2(1/\delta))}$, or,

3. Conditional on $|\text{Dict}| < d_0$, with probability $\geq 1 - e^{-\Omega(\epsilon^2 d / \log^2(1/\delta))}$,

$$\min_{Q \in \mathcal{Q}_{1\text{-gram}}} \mathcal{L}(Q \circ \text{enc}(\cdot)) \leq \left(1 - \frac{2d_0}{d}\right) \left(2H_\infty + O\left(\frac{1}{\log(d)}\right)\right) + \frac{2d_0}{d} \log(4|\mathcal{A}|).$$

With the choice of $d_0 = \epsilon d / 2 \log(4|\mathcal{A}|)$ we get the statement of Theorem 4.7.

B.1 Analysis for the large dictionary case: $|\text{Dict}| > d_0$

In the large dictionary case, Algorithm 1 uses the greedy encoder/decoder pair in conjunction with the dictionary. The proof of Theorem 4.7 relies on establishing that the cross-entropy $H(Q_{\text{MLE}}, P)$ of the tokenizer is large. Namely, we prove that,

Lemma B.2. *In Algorithm 1, assuming at least d_0 tokens are allocated,*

$$H(Q_{\text{MLE}}, P) = \Omega \left(\frac{\epsilon \delta^9 \log(d)}{\log(1/\epsilon) \log^3(1/\delta)} \right).$$

To show this, it suffices to argue that conditioned on any previous set of tokens, with nontrivial probability over the underlying string generated from the stochastic source, the next token is long (i.e. having conditional probability at most $O(1/\sqrt{d})$).

Lemma B.3. *Suppose that in a run of Algorithm 1, at least d_0 tokens are allocated. Suppose a set of tokens $\mathbf{t}_1, \dots, \mathbf{t}_k$ have been sampled so far by the greedy encoder. Let T_{i+1} be the random variable which denotes the next token returned by the greedy encoder, where the randomness comes from the underlying string being tokenized. Then,*

$$\Pr \left(P(T_{i+1} | \mathbf{t}_i) \leq 1/\sqrt{C\delta d} \mid \mathbf{t}_1, \dots, \mathbf{t}_i \right) \geq \frac{d_0 \delta^6 (1-\delta)^2}{8Cd\Delta|\mathcal{A}| \log(2|\mathcal{A}|) n_D} = \Omega \left(\frac{\epsilon \delta^9}{\log^3(1/\delta) \log(1/\epsilon)} \right)$$

Proof sketch of Lemma B.3. The proof will proceed in 2 parts. We first show in Lemma B.7 that there is a set D_{valid} of $\Omega(d)$ tokens in the dictionary which are neither prefixes nor suffixes of any other token in Dict . The reason for considering this set of tokens is twofold,

1. Irrespective of what the previous set of tokens were, it is legal for a token D_{valid} to be sampled in the current step by the greedy encoder, since for any candidate $\mathbf{t} \in D_{\text{valid}}$, by definition, $\mathbf{t}_j \cdots \mathbf{t}_i \mathbf{t} \notin \text{Dict}$ for every $j \leq i$.
2. Suppose a sequence of tokens $\mathbf{t}_1, \dots, \mathbf{t}_i$ have already been sampled, ending with the character a . Then, we may sample the next token using rejection sampling. Sample $a' \sim P(\cdot|a)$ and an infinitely long trajectory on $\mathcal{T}_{a'}^*$. Return the last token on this trajectory which belongs to Dict , and if it so happens that $\exists j \in [i]$ such that $\mathbf{t}_j \cdots \mathbf{t}_i \mathbf{t} \in \text{Dict}$, then reject this trajectory and repeat. Since all the tokens in D_{valid} are not prefixes of another token, any trajectory which reaches a token in D_{valid} must terminate the rejection sampling process.

Next, in Lemma B.8, we show that since the number of tokens in D_{valid} is sufficiently large, $\Omega(d)$, with constant (in d) probability, a trajectory rolled out in the first round of the rejection sampling process will reach a token $\mathbf{t} \in D_{\text{valid}}$ which has small probability, i.e. $\max_{a \in \mathcal{A}} P(\mathbf{t}|a) \leq 1/\text{poly}(d)$. By the previous arguments, this must mean that the rejection sampling process terminates on this “low probability” token, resulting in the statement of the lemma. \square

Proof of Theorem B.1.1 and Lemma B.2 It is easy to see why Lemma B.3 implies a lower bound on the cross entropy $H(Q_{\text{MLE}}, P)$ of the tokenizer. By Lemma A.4 for the greedy encoder,

$$Q_{\text{MLE}}(\mathbf{t}) = \lim_{m \rightarrow \infty} \mathbb{E} \left[\frac{n_{\mathbf{t}}}{\sum_{\mathbf{t}'} n_{\mathbf{t}'} \right] \stackrel{\text{a.s.}}{\equiv} \lim_{m \rightarrow \infty} \frac{n_{\mathbf{t}}}{\sum_{\mathbf{t}'} n_{\mathbf{t}'}}. \quad (17)$$

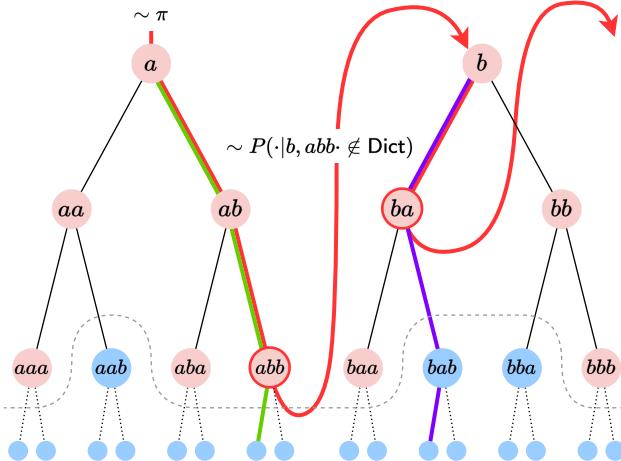


Figure 9: *Jointly generating a sequence and its greedy encoding:* In this example we use the greedy encoder under the dictionary composed of all the substrings shadowed red. The first character (a) is sampled from the stationary distribution. Then an infinite string is sampled on the tree with a as root (green path). The last substring on this path which is a token ($t_1 = abb$) is returned by the greedy encoder. Then the next character $x = b$ is sampled from the source conditioned on the previous character (b) and further conditioned on $t_1 x \notin \text{Dict}$. Finally, another infinite string is sampled on the tree with $x = b$ as root (purple path) and the last substring on this path which is a token ($t_2 = ba$) is returned by the greedy encoder. Repeating this process, we can generate a string, here, $abbba \dots$, as well as its greedy encoding, (abb, ba, \dots) .

Since the limit $m \rightarrow \infty$ of the RHS exists by Lemma A.4, we may let $m \rightarrow \infty$ in any way we like, and in particular we may simply sample i^* tokens, $\mathbf{t}_1, \dots, \mathbf{t}_{i^*}$ sequentially according to the process in Figure 9. Here, the first token sampled is returned by generating an infinitely long string on \mathcal{T}_a^* where $a \sim \pi$ and then truncating this trajectory to the longest token which belongs to Dict . Subsequently for every $i > 1$, \mathbf{t}_i is generated by sampling a fresh infinitely long string from \mathcal{T}_a^* where a is sampled from the $P(\cdot | a')$ where a' is the last character \mathbf{t}_{i-1} and then returning the largest prefix of this string which is a token in Dict , conditioned on $\mathbf{t}_j \dots \mathbf{t}_{i-1} \mathbf{t}_i \notin \text{Dict}$ for any $j < i$.

and concatenate the corresponding substrings to get an $m = \sum_{i=1}^{i^*} |\mathbf{t}_i|$ length character string. Letting $i^* \rightarrow \infty$, we must have $m \rightarrow \infty$ surely since $m \geq i^*$. In this view, eq. (17) can be rewritten as,

$$Q_{\text{MLE}}(\mathbf{t}) = \lim_{i^* \rightarrow \infty} \frac{n_{\mathbf{t}}}{i^*} = \lim_{i^* \rightarrow \infty} \frac{1}{i^*} \sum_{i=1}^{i^*} \mathbb{I}(\mathbf{t}_i = \mathbf{t}) \stackrel{\text{a.s.}}{=} \lim_{i^* \rightarrow \infty} \frac{1}{i^*} \sum_{i=1}^{i^*} \mathbb{E}[\mathbb{I}(\mathbf{t}_i = \mathbf{t}) | \mathbf{t}_1, \dots, \mathbf{t}_{i-1}] \quad (18)$$

where the last inequality follows by the sequential nature of the token sampling process and a martingale argument. Consider the set of tokens T such that $\mathbf{t} \in T$ satisfies $\max_{a \in \mathcal{A}} P(\mathbf{t}|a) \leq \sqrt{1/C\delta^3d}$. From eq. (18), summing across $\mathbf{t} \in T$, we have that,

$$\sum_{\mathbf{t} \in T} Q_{\text{MLE}}(\mathbf{t}) \stackrel{\text{a.s.}}{=} \lim_{i^* \rightarrow \infty} \frac{1}{i^*} \sum_{i=1}^{i^*} \Pr(\mathbf{t}_i \in T | \mathbf{t}_1, \dots, \mathbf{t}_{i-1}) = \Omega\left(\frac{\epsilon\delta^9}{\log^3(1/\delta)\log(1/\epsilon)}\right) \quad (19)$$

where in the last inequality, we use Lemma B.3 and the fact that $\delta \max_{a \in \mathcal{A}} P(\mathbf{t}|a) \geq \min_{a \in \mathcal{A}} P(\mathbf{t}|a)$.

Therefore,

$$H(Q_{\text{MLE}}, P) \geq \sum_{t \in T} Q_{\text{MLE}}(\mathbf{t}) \log(1/P(\mathbf{t})) \geq \sum_{t \in T} Q_{\text{MLE}}(\mathbf{t}) \log(\sqrt{C\delta^3 d})$$

where in (i) we use the fact that for $\mathbf{t} \in T$, $\max_{a \in \mathcal{A}} P(\mathbf{t}|a) \leq 1/\sqrt{C\delta^3 d}$, which implies that $P(\mathbf{t}) \leq 1/\sqrt{C\delta^3 d}$. Finally, combining with eq. (19) completes the proof of Lemma B.2. Furthermore, since the cross-entropy $H(Q_{\text{MLE}}, P)$ was established to be large, by invoking the reduction in Theorem 4.3, we complete the proof of Theorem B.1.1.

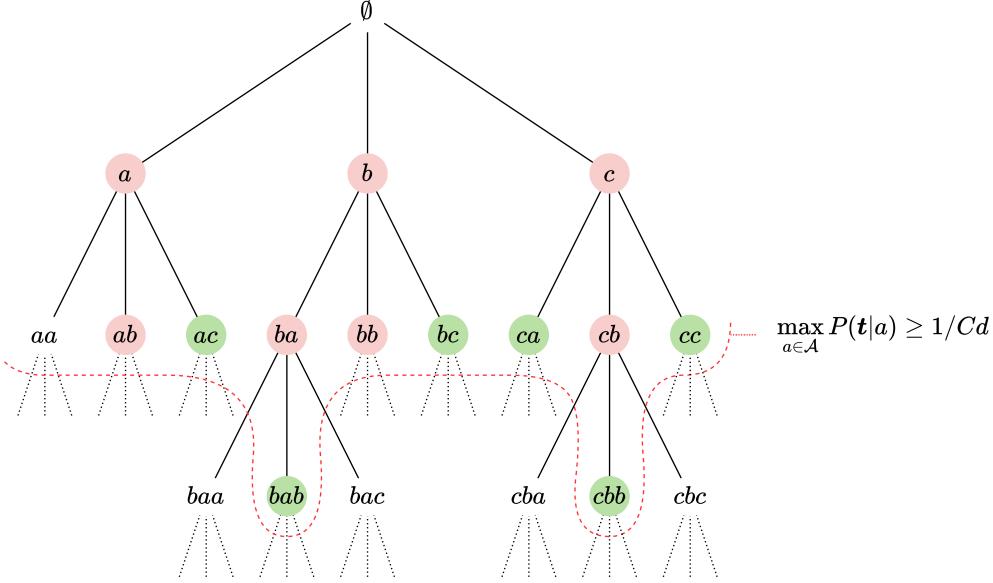


Figure 10: The circled nodes indicate substrings which are tokens in Dict . The red boundary is the set of substrings \mathbf{t} such that $\max_{a \in \mathcal{A}} P(\mathbf{t}|a) \geq 1/Cd$. By Lemma B.6, none of the nodes which fall outside this boundary are assigned as tokens in a run of Algorithm 1. The set of circled substrings are the set of tokens in Dict . Among them, the ones circled green are the tokens in D_{valid} , which are not prefixes or suffixes of any other tokens in Dict . Substrings such as cb or ba which are tokens in Dict do not belong to D_{valid} because they are prefixes of longer tokens (in this case, cbb and bab respectively). On the other hand, substrings like ab do not belong to D_{valid} since they are suffixes of tokens in Dict , in this case, bab . Lemma B.7 asserts that the number of tokens in D_{valid} are $\Omega(d)$ in number, assuming that Dict has $\Omega(d)$ tokens to begin with.

Notation. For each $a \in \mathcal{A}$ and $j \in \mathbb{N} \cup \{0\}$, define a *level set* of substrings,

$$S_j^a = \left\{ (1 - \delta)^{j+1} < P(\mathbf{t} | \mathbf{t}_1 = a) \leq (1 - \delta)^j \right\}$$

where \mathbf{t}_1 denotes the first character of \mathbf{t} . And likewise, define the sets $S_j = \cup_{a \in \mathcal{A}} S_j^a$, $S_{\leq j}^a$ and $S_{\geq j}^a$ as the union of $S_{j'}^a$ over $j' \geq j$, $j' \leq j$ and $S_{\leq j}$ and $S_{\geq j}$ as the union of $S_{\leq j}^a$ and $S_{\geq j}^a$ over $a \in \mathcal{A}$. Furthermore for a large universal constant $C > 0$, define parameters,

$$\Delta = \frac{\log(\delta)}{\log(1 - \delta)} \asymp \Theta\left(\frac{\log(1/\delta)}{\delta}\right); \quad n_D = 1 - \frac{2 \log(4Cd/\delta d_0)}{\log(1 - \delta)} \asymp \Theta\left(\frac{\log(1/\epsilon\delta)}{\delta}\right). \quad (20)$$

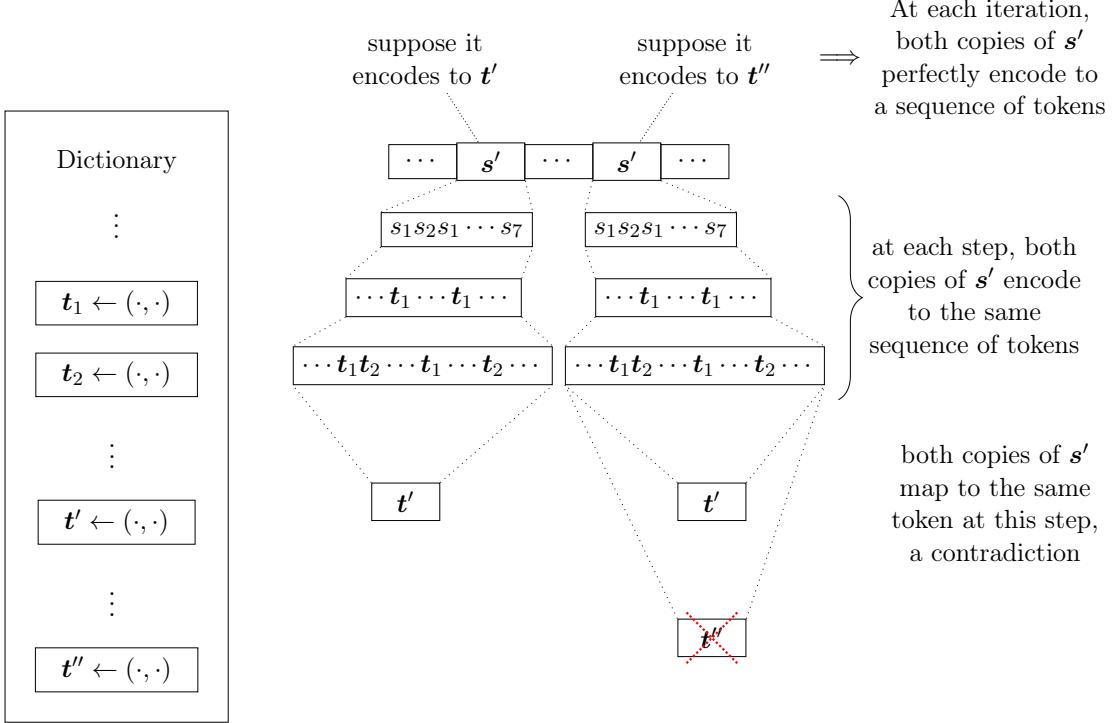


Figure 11: A pictorial representation of the proof of Lemma B.4

We first begin by stating a folklore result: every pair of tokens assigned by a merging-based dictionary generation algorithm have distinct character representations.

Lemma B.4. *If Algorithm 1 assigns a new token in some round, its character representation must be distinct from that of all previously assigned tokens.*

Proof. A pictorial proof is in Figure 11. We will prove this result by contradiction. Suppose \mathbf{t} and \mathbf{t}' are tokens which decode to the same character substring, s' . Consider all occurrences of s' in the dataset which in some iteration encode into \mathbf{t}' or \mathbf{t}'' , and denote these disjoint locations \mathcal{S} . Recall that at these locations, s' eventually is assumed to map to a singular token t' or t'' . Therefore, at every step in the merging process these occurrences of s' must perfectly map to a sequence of tokens.

Now consider the merging process at the first time before any of the rules corresponding to tokens in \mathbf{t}' or \mathbf{t}'' are implemented. Prior to this time, all the occurrences of s' corresponding to the locations in \mathcal{S} have not been tokenized yet. When the first rule corresponding to one of the tokens in $\{\mathbf{t}', \mathbf{t}''\}$ is implemented, all the strings in \mathcal{S} must be modified identically. This uses the fact that we can isolate each of these occurrences of s' while carrying out the merging process, since each location must be distinct. At every step, the encodings of these copies of s' must be the same, and therefore \mathbf{t}' and \mathbf{t}'' cannot be two distinct tokens. \square

Lemma B.5. *The size of the level set S_j^a is bounded by $(1 - \delta)^{-(j+1)}$.*

Proof. Since the probability of any transition is at most $1 - \delta$, this implies that any infinite trajectory on the tree \mathcal{T}_a^* can intersect at most one vertex in S_j^a . Therefore, $\sum_{\mathbf{t} \in S_j^a} P(\mathbf{t} | \mathbf{t}_1 = a) \leq 1$. By the lower bound on $P(\mathbf{t} | \mathbf{t}_1 = a)$ for $\mathbf{t} \in S_j^a$, this implies the statement of the lemma. \square

Next we show that with high probability none of the substrings \mathbf{t} having probability mass (under P) of at most δ/Cd conditioned on the first character, are assigned as tokens by Algorithm 1.

Lemma B.6. *In a run of Algorithm 1, for a sufficiently large constant $C > 0$, with probability $d^{-\Omega(1)} \text{poly}(1/\delta)$ all assigned tokens $\mathbf{t} \in \text{Dict}$ satisfy $\max_{a \in \mathcal{A}} P(\mathbf{t}|a) \geq 1/Cd$. In other words, none of the substrings in $S_{\geq j^*}$ are added as tokens to the dictionary in a run of Algorithm 1, where,*

$$j^* \triangleq \log(\delta/Cd) / \log(1 - \delta)$$

Proof. Consider some $j \geq j^*$ and $a \in \mathcal{A}$ and substring $\mathbf{t} \in S_j^a$. In the i^{th} stage of the algorithm where text_i is being processed, for \mathbf{t} to be assigned as a token, at the very least, \mathbf{t} must appear at least $\log(d)$ times disjointly in text_i . Therefore,

$$\begin{aligned} P(\mathbf{t} \in S_j^a \text{ is assigned as a token in } \text{text}_i) &\leq \binom{d}{\log(d)} \left(\max_{a \in \mathcal{A}} P(\mathbf{t}|a) \right)^{\log(d)} \\ &\leq d^{\log(d)} \left(\frac{1}{Cd} (1 - \delta)^{j-j^*} \right)^{\log(d)} \\ &\leq d^{-\log(C)} (1 - \delta)^{(j-j^*) \log(d)} \end{aligned}$$

Union bounding over S_j^a over $j \geq j^*$ using the bound on $|S_j^a|$ in Lemma B.5, and over $a \in \mathcal{A}$ and $i \in [d]$ results in the bound,

$$P(\mathbf{t} \in S_{\geq j^*} \text{ is assigned as a token in step } i \text{ for some } i \in [d]) \leq d^{-\Omega(1)} \sum_{j \geq j^*} \frac{(1 - \delta)^{(j-j^*) \log(d)}}{(1 - \delta)^{j+1}} \leq \frac{d^{-\Omega(1)}}{\delta(1 - \delta)}$$

□

Lemma B.7. *Consider the set of tokens D_{valid} which are not a prefix or a suffix of any other token in Dict . That is, $D_{\text{valid}} = \{\mathbf{t} \in \text{Dict} : \nexists \mathbf{s} : \mathbf{st} \in \text{Dict}\} \cap \{\mathbf{t} \in \text{Dict} : \nexists \mathbf{s} : \mathbf{ts} \in \text{Dict}\}$. If $|\text{Dict}| \geq d_0$, then,*

$$|D_{\text{valid}}| \geq \frac{d_0}{4n_D}.$$

where n_D is defined in eq. (20).

Proof. For any token $\mathbf{t} \in D_{\text{valid}}$, there may be at most $2|\mathbf{t}|$ tokens which are suffixes or prefixes of it and belong to Dict . More importantly, every token in Dict not belonging to D_{valid} must either be a prefix or a suffix of some token in D_{valid} . Split the suffixes and prefixes of the tokens in D_{valid} into four sets,

1. $S_{\text{suff},\min} = \bigcup_{\mathbf{t} \in D_{\text{valid}}} \{\mathbf{t}' \in \text{Dict} : \mathbf{t}' \in \text{suff}(\mathbf{t}), |\mathbf{t}'| \leq |\mathbf{t}| - n_D\}$,
2. $S_{\text{suff},\max} = \bigcup_{\mathbf{t} \in D_{\text{valid}}} \{\mathbf{t}' \in \text{Dict} : \mathbf{t}' \in \text{suff}(\mathbf{t}), |\mathbf{t}'| > |\mathbf{t}| - n_D\}$,
3. $S_{\text{pre},\min} = \bigcup_{\mathbf{t} \in D_{\text{valid}}} \{\mathbf{t}' \in \text{Dict} : \mathbf{t}' \in \text{pre}(\mathbf{t}), |\mathbf{t}'| \leq |\mathbf{t}| - n_D\}$,
4. $S_{\text{pre},\max} = \bigcup_{\mathbf{t} \in D_{\text{valid}}} \{\mathbf{t}' \in \text{Dict} : \mathbf{t}' \in \text{pre}(\mathbf{t}), |\mathbf{t}'| > |\mathbf{t}| - n_D\}$.

where n_D is defined in eq. (20). Note from Lemma B.6 that all the tokens $\mathbf{t} \in \text{Dict}$ all satisfy $\max_{a \in \mathcal{A}} P(\mathbf{t}|a) \geq 1/Cd$. Therefore, the tokens in $S_{\text{pre},\min}$ and $S_{\text{suff},\min}$ all satisfy, $\max_{a \in \mathcal{A}} P(\mathbf{t}|a) \geq d/C(1-\delta)^{n_D}$. By summing Lemma B.5 over appropriate j , we get that $|S_{\text{pre},\min}| + |S_{\text{suff},\min}| \leq 2Cd(1-\delta)^{n_D-1}/\delta$.

On the other hand, corresponding to any $\mathbf{t} \in D_{\text{valid}}$, there are at most n_D tokens in $S_{\text{pre},\max}$ or $S_{\text{suff},\max}$ and therefore $|S_{\text{pre},\max}|, |S_{\text{suff},\max}| \leq n_D \cdot |D_{\text{valid}}|$. Since every token in Dict either belongs to D_{valid} or is a suffix of some token in D_{valid} , $S_{\text{pre},\min} \cup S_{\text{pre},\max} \cup S_{\text{suff},\min} \cup S_{\text{suff},\max} = |\text{Dict}|$ and,

$$2n_D \cdot |D_{\text{valid}}| + \frac{2C(1-\delta)^{n_D-1}d}{\delta} \geq d_0$$

Recalling the choice of $n_D = 1 - \frac{2 \log(4Cd/\delta d_0)}{\log(1-\delta)}$, we get that,

$$|D_{\text{valid}}| \geq \frac{d_0}{4n_D}.$$

□

Lemma B.8. Suppose Algorithm 1 assigns at least d_0 tokens. For any character $a \in \mathcal{A}$, sample an $a' \sim P(\cdot|a)$ and an infinite trajectory on the tree $\mathcal{T}_{a'}^*$, denoted traj . Then,

$$\mathbb{E}_{a' \sim P(\cdot|a)} \left[\Pr_{\text{traj} \sim \mathcal{T}_{a'}^*} \left(\min_{\mathbf{t} \in \text{traj} \cap D_{\text{valid}}} P(\mathbf{t}|a) \leq \sqrt{\delta/Cd} \middle| a' \right) \right] \geq \frac{d_0 \delta^6 (1-\delta)^2}{8Cd\Delta|\mathcal{A}|n_D}.$$

where the notation $\mathcal{T}_{a'}^*$ is used to overload the distribution over infinite trajectories on $\mathcal{T}_{a'}^*$. The parameters n_D and Δ are defined in eq. (20).

Proof. By Lemma B.6, recall that the $\geq d_0$ tokens assigned in a run of Algorithm 1, with high probability, are substrings in $S_{\leq j^*}$. For any $a \in \mathcal{A}$, the total number of substrings in $S_{\leq j^*}$ can be bounded as,

$$|S_{\leq j^*}| \leq \sum_{a \in \mathcal{A}} \sum_{j=0}^{j^*} |S_j^a| \leq \sum_{a \in \mathcal{A}} \sum_{j=0}^{j^*} \frac{1}{(1-\delta)^{j+1}} \leq \frac{C|\mathcal{A}|d}{\delta(1-\delta)}. \quad (21)$$

In order to prove this result, we use a counting argument and the fact that no tokens in $S_{>j^*}$ are assigned. Consider some character a and all the leaves in the forest $S_{\leq j^*}$. Since every transition has $\geq \delta$ probability of occurring, across all leaf nodes $\mathbf{t} \in S_{\leq j^*}$, $P(\mathbf{t}|a')$ are within a $\delta^2(1-\delta)$ factor of each other across different $a' \in \mathcal{A}$. In particular, by counting the number of paths in \mathcal{T}^* (i.e. paths in \mathcal{T}_a^* from \emptyset to leaf nodes in $S_{\leq j^*}^a$ across $a \in \mathcal{A}$) along which a token in Dict exists in $S_{\geq j^*/2}$, we can also compute the probability mass across such trajectories up to a factor of $\delta^2(1-\delta)$.

Taking the union across $a \in \mathcal{A}$, consider the paths in \mathcal{T}_a^* from \emptyset to leaf nodes in $S_{\leq j^*}^a$. From Lemma B.7, $\sum_{j \leq j^*} |D_{\text{valid}} \cap S_j| \geq d_0/4n_D$, where $n_D = 1 - 2 \log(4Cd/\delta d_0)/\log(1-\delta)$. Note that for sufficiently large $d = \Omega(\log(1/\epsilon\delta)/\delta^5)$, by Lemma B.5, $\sum_{j \leq j^*/2} |S_j| = \sqrt{Cd/\delta}/\delta(1-\delta) \leq d_0/8n_D$. Therefore,

$$\sum_{j^*/2 < j \leq j^*} |D_{\text{valid}} \cap S_j| \geq \frac{d_0}{8n_D}. \quad (22)$$

Define $\Delta = \log(\delta)/\log(1-\delta)$. Combining eq. (22) with eq. (21) and applying the probabilistic method, there exists an $i^* \geq j^*/2$ such that,

$$\frac{|D_{\text{valid}} \cap (S_{i^*+1} \cup \dots \cup S_{i^*+\Delta})|}{|S_{i^*+1} \cup \dots \cup S_{i^*+\Delta}|} \geq \frac{\delta(1-\delta)d_0}{8Cd|\mathcal{A}|n_D}. \quad (23)$$

Note that Δ is chosen to be sufficiently large, so that every infinite trajectory on $\mathcal{T}_{a'}^*$ must intersect at least once with the band of vertices $S_{i^*+\Delta+1}^{a'} \cup \dots \cup S_{i^*+2\Delta}^{a'}$. Note that this band is different from the one considered in eq. (23). Define $L_{a'}$ as the set of longest prefixes across infinite trajectories in $\mathcal{T}_{a'}^*$ which belong to $S_{i^*+\Delta+1}^{a'} \cup \dots \cup S_{i^*+2\Delta}^{a'}$.

Note that our objective is to show that an infinite trajectory sampled on $\mathcal{T}_{a'}^*$ where $a' \sim P(\cdot|a)$, has a long prefix in Dict. We can truncate this trajectory to lower bound this probability, and therefore, we assume that the infinite trajectories on $\mathcal{T}_{a'}^*$ terminate once they reach a substring in $L_{a'}$. Furthermore, note that although Δ is large, it is still a constant depending on δ . Therefore, the band of states $S_{i^*+\Delta+1}^{a'} \cup \dots \cup S_{i^*+2\Delta}^{a'}$ is not too wide, and all the substrings in $L_{a'}$ have approximately similar probabilities to each other. In particular, for any character $a \in \mathcal{A}$, and for any $a' \in \mathcal{A}$ and $\mathbf{t} \in L_{a'}$, decomposing $P(\mathbf{t}|a)$ as $P(\mathbf{t}|\mathbf{t}_1 = a')P(a'|a)$,

$$\delta^2(1-\delta) \cdot (1-\delta)^{i+\Delta} \stackrel{(i)}{\leq} P(\mathbf{t}|a) \stackrel{(ii)}{\leq} (1-\delta)^{i+\Delta}. \quad (24)$$

Inequality (i) follows from the fact that all transition probabilities are at least δ , so every leaf node in $L_{a'}$ must have $P(\mathbf{t}|\mathbf{t}_1 = a') \geq (1-\delta)^{i+2\Delta+1}$, and the fact that $P(a'|a) \geq \delta$. Inequality (ii) follows similarly from the fact that \mathbf{t} is a leaf node of $L_{a'}$ and therefore $P(\mathbf{t}|\mathbf{t}_1 = a') \leq (1-\delta)^{i+\Delta}$. Therefore, instead of bounding the probability of any event under the distribution over substrings in $L_{a'}$ induced by truncating the infinite strings sampled on $\mathcal{T}_{a'}^*$, it suffices to count the fraction of substrings in $L_{a'}$ satisfying the event (which are equivalent up to a $\delta(1-\delta)$ factor). Define,

$$\text{pre}(\mathbf{t}) = (\mathbf{t}_1, \mathbf{t}_{1:2}, \mathbf{t}_{1:3}, \dots, \mathbf{t}_{1:|\mathbf{t}|})$$

As the set of prefixes of \mathbf{t} (including \mathbf{t}). Note that at most Δ of the prefixes of any substring \mathbf{t} can intersect with $S_{i^*+1}^a \cup \dots \cup S_{i^*+\Delta}^a$. Therefore,

$$\begin{aligned} & \sum_{a' \in \mathcal{A}} \sum_{\mathbf{t} \in L_{a'}} \mathbf{1}(\text{pre}(\mathbf{t}) \cap D_{\text{valid}} \cap (S_{i^*+1}^{a'} \cup \dots \cup S_{i^*+\Delta}^{a'}) \neq \emptyset) \\ & \geq \sum_{a' \in \mathcal{A}} \sum_{\mathbf{t} \in L_{a'}} \frac{|\text{pre}(\mathbf{t}) \cap D_{\text{valid}} \cap (S_{i^*+1}^{a'} \cup \dots \cup S_{i^*+\Delta}^{a'})|}{\Delta} \\ & \stackrel{(i)}{\geq} \sum_{a' \in \mathcal{A}} \frac{|D_{\text{valid}} \cap (S_{i^*+1}^{a'} \cup \dots \cup S_{i^*+\Delta}^{a'})|}{\Delta} \\ & \stackrel{(ii)}{\geq} \frac{\delta d_0(1-\delta)}{8Cd\Delta|\mathcal{A}|n_D} \sum_{a' \in \mathcal{A}} |S_{i^*+1}^{a'} \cup \dots \cup S_{i^*+\Delta}^{a'}| \\ & \stackrel{(iii)}{\geq} \frac{\delta^3 d_0(1-\delta)}{8Cd\Delta|\mathcal{A}|n_D} \sum_{a' \in \mathcal{A}} |L_{a'}|, \end{aligned}$$

where (i) uses the fact that the prefixes of $\mathbf{t} \in L_{a'}$ cover all the substrings in $S_{i^*+\Delta}^{a'}$, and therefore $\cup_{\mathbf{t} \in L_{a'}} \text{pre}(\mathbf{t}) \supset S_{i^*+1}^{a'} \cup \dots \cup S_{i^*+\Delta}^{a'}$, and (ii) uses eq. (23). Finally, (iii) uses the fact that Δ is not too large, and therefore, for any substring $\mathbf{t}' \in S_{i^*+1}^{a'} \cup \dots \cup S_{i^*+\Delta}^{a'}$, there are at most $1/(1-\delta)^{2\Delta} = 1/\delta^2$

substrings $\mathbf{t} \in L_{a'}$ which contain it as a prefix. This means, $|L_{a'}| \leq |S_{i^*+1}^{a'} \cup \dots \cup S_{i^*+\Delta}^{a'}|/\delta^2$. After dividing by $\sum_{a' \in \mathcal{A}} |L_{a'}|$ on both sides, this implies,

$$\mathbb{E}_{a' \sim \text{Unif}(\mathcal{A})} \left[\Pr_{\mathbf{t} \sim \text{Unif}(L_{a'})} \left(\text{pre}(\mathbf{t}) \cap D_{\text{valid}} \cap (S_{i^*+1}^{a'} \cup \dots \cup S_{i^*+\Delta}^{a'}) \neq \emptyset \middle| a' \right) \right] \geq \frac{\delta^3 d_0 (1 - \delta)}{8Cd\Delta|\mathcal{A}|n_D}. \quad (25)$$

The event inside the inner probability term is the event that an infinitely long string (truncated at $L_{a'}$) has a prefix which lies in D_{valid} and which intersects with $S_{i^*+1}^{a'} \cup \dots \cup S_{i^*+\Delta}^{a'}$, which implies that it has probability $P(\mathbf{t}|a) \leq \sqrt{\delta/Cd}$. Therefore, we have that for any $a \in \mathcal{A}$, sampling an $a' \sim P(\cdot|a)$ and an infinite trajectory $\text{traj} \sim \mathcal{T}_{a'}^*$,

$$\begin{aligned} & \mathbb{E}_{a' \sim P(\cdot|a)} \left[\Pr_{\text{traj} \sim \mathcal{T}_{a'}^*} \left(\min_{\mathbf{t} \in \text{traj} \cap D_{\text{valid}}} P(\mathbf{t}|a) \leq \sqrt{\delta/Cd} \middle| a' \right) \right] \\ & \stackrel{(i)}{\geq} \delta^2(1 - \delta) \cdot \mathbb{E}_{a' \sim P(\cdot|a)} \left[\Pr_{\mathbf{t}' \sim \text{Unif}(L_{a'})} \left(\min_{\mathbf{t} \in \text{pre}(\mathbf{t}') \cap D_{\text{valid}}} P(\mathbf{t}|a) \leq \sqrt{\delta/Cd} \middle| a' \right) \right] \\ & \stackrel{(ii)}{\geq} \delta^2(1 - \delta) \cdot \mathbb{E}_{a' \sim P(\cdot|a)} \left[\Pr_{\mathbf{t}' \sim \text{Unif}(L_{a'})} \left(\text{pre}(\mathbf{t}') \cap D_{\text{valid}} \cap (S_{i^*+1}^{a'} \cup \dots \cup S_{i^*+\Delta}^{a'}) \neq \emptyset \middle| a' \right) \right] \\ & \stackrel{(iii)}{\geq} \delta^3(1 - \delta) \cdot \mathbb{E}_{a' \sim \text{Unif}(\mathcal{A})} \left[\Pr_{\mathbf{t}' \sim \text{Unif}(L_{a'})} \left(\text{pre}(\mathbf{t}') \cap D_{\text{valid}} \cap (S_{i^*+1}^a \cup \dots \cup S_{i^*+\Delta}^a) \neq \emptyset \middle| a' \right) \right] \\ & \geq \delta^3(1 - \delta) \cdot \frac{\delta^3 d_0 (1 - \delta)}{8Cd\Delta|\mathcal{A}|n_D}. \end{aligned}$$

Here (i) follows by truncating the trajectory traj to terminate at a node in $\cup_{a' \in \mathcal{A}} L_{a'}$ and from eq. (24), (ii) follows by arguing that $i^* \leq j^*/2$ and therefore if a prefix of \mathbf{t}' lies in $S_{i^*+1}^{a'} \cup \dots \cup S_{i^*+\Delta}^{a'}$, then it must have $P(\mathbf{t}|a) \leq \sqrt{\delta/Cd}$. Inequality (iii) follows by noting that all the transitions $P(a'|a)$ have probability $\geq \delta$, and the last inequality follows from eq. (25). \square

Proof of Lemma B.3

Lemma B.8 concludes that given any previous sequence of tokens terminating in a character a , with constant probability, an infinite trajectory sampled from $\mathcal{T}_{a'}^*$ with $a' \sim P(\cdot|a)$ has as prefix, a substring \mathbf{t} , which not only has low probability, with $P(\mathbf{t}|a) \leq \sqrt{\delta/Cd}$, but also belongs to the subset of tokens D_{valid} . Note that regardless of the previously sampled tokens, it is legal to sample any token in D_{valid} as the current token, since by definition, these tokens are not the suffixes of any other tokens in Dict . Moreover, if any trajectory on $\mathcal{T}_{a'}^*$ reaches a token in D_{valid} , then it must be largest token along that trajectory, since none of the tokens in D_{valid} are prefixes of another token in Dict .

Consider generating a new token by rejection sampling. Suppose the set of previous tokens $\mathbf{t}_1, \dots, \mathbf{t}_i$ end in some character a . Sample the next character $a' \sim P(\cdot|a)$ and an infinite trajectory on $\mathcal{T}_{a'}^*$. If it reaches an illegal token \mathbf{t} such that $\mathbf{t}_j \mathbf{t}_{j+1} \dots \mathbf{t}_i \mathbf{t}$ already exists in Dict , this token is rejected and the trajectory is resampled. By the prefix-free property of these tokens, if this trajectory visits a token in D_{valid} , it must immediately be output as the next token. Note that this probability is lower bounded by,

$$\mathbb{E}_{a' \sim P(\cdot|a)} \left[\Pr_{\text{traj} \sim \mathcal{T}_{a'}^*} \left(\min_{\mathbf{t} \in \text{traj} \cap D_{\text{valid}}} P(\mathbf{t}|a) \leq \sqrt{\delta/Cd} \middle| a' \right) \right]$$

which is lower bounded by $\text{poly}(\epsilon, \delta)$, the subject of Lemma B.8. Therefore with this probability, the process terminates in the first step with a token in D_{valid} being sampled.

B.2 Analysis in the small dictionary case

In this section, we will prove Theorem B.1.2 and Theorem B.1.3. In particular we show that, either,

1. The dictionary is small with low probability. i.e., $\Pr(|\text{Dict}| < d_0) = e^{-\Omega(\epsilon^2 d / \log^2(1/\delta))}$, or,
2. Or conditioned on the dictionary being small, $|\text{Dict}| < d_0$, with high probability $\geq 1 - e^{-\Omega(\epsilon^2 d / \log^2(1/\delta))}$,

$$\min_{Q \in \mathcal{Q}_{1\text{-gram}}} \mathcal{L}(Q \circ \text{enc}(\cdot)) \leq 4 \left(1 - \frac{2d_0}{d} + O\left(\frac{1}{\log(d)}\right) \right) H_\infty + \frac{2d_0}{d} \cdot \log(2|\mathcal{A}|).$$

For $i \in [d]$, define the indicator random variable,

$$X(s', \text{Dict}) = \mathbf{1}(\exists \text{ a pair of tokens in } \text{enc}_{\text{BPE}}(s') \text{ under Dict appears at least } \log(d) \text{ times}).$$

which captures the event that the string s' is compressed well by the dictionary Dict under the sequential encoder.

Let Dict_i denote the dictionary stored by Algorithm 1 right after text_i is processed. The key insight behind this lemma is the following statement, asserting that the sequential encoder satisfies a “monotonicity” property: for any j and string s' , if there exists a pair of tokens appearing more than $\log(d)$ times consecutively in the sequential encoding of s' under Dict_j , then there must exist a pair of tokens appearing more than $\log(d)$ times consecutively in the greedy encoding of s' under Dict_i for any $i < j$. This implies that $X(s', \text{Dict}_j) \leq X(s', \text{Dict}_i)$ if $i < j$ for any string s' . This monotonicity property implies that the last dictionary output by the learner, Dict_d sequentially encodes a $1 - \epsilon$ fraction of the previously seen texts, text_i in a way where every pair of tokens appears at most $\log(d)$ times. While Dict_d is correlated with these texts, we can circumvent this correlation by using a martingale argument to prove the statement of the lemma.

Lemma B.9. *Let Dict be the dictionary returned by Algorithm 1. Then,*

$$\min \left\{ \Pr \left(\mathbb{E}[X(s', \text{Dict}) | \text{Dict}] \geq 2d_0/d \middle| |\text{Dict}| < d_0 \right), \Pr(|\text{Dict}| < d_0) \right\} \leq e^{-\epsilon^2 d / 8 \log^2(1/\delta)}.$$

where s' is a fresh substring of length d sampled from the stochastic source.

Proof. Let Dict_i denote the state of dictionary returned by Algorithm 1 right after text_i is processed. Then, Dict_d is the final dictionary returned by Algorithm 1. Suppose $\mathbb{E}[X(s', \text{Dict}_d) | \text{Dict}_d] \geq 2d_0/d$, where s' is a fresh substring of length d sampled from the stochastic source. Using monotonicity of the sequential encoder, almost surely for any string s' , $X(s', \text{Dict}_i) \leq X(s', \text{Dict}_j)$ for any $j > i$. Therefore,

$$\mathbb{E}[X(s', \text{Dict}_d) | \text{Dict}_d] \geq 2d_0/d \implies \sum_{i=1}^{d-1} \mathbb{E}[X(s', \text{Dict}_i) | \text{Dict}_i] \geq 2d_0 \cdot \frac{d-1}{d} \quad (26)$$

Note in this expectation, s' is an independent string of length d sampled from the stochastic source. Since Dict_i and text_{i+1} are independent, we may instead write,

$$\sum_{i=1}^{d-1} \mathbb{E}[X(\text{text}_{i+1}, \text{Dict}_i) | \text{Dict}_i, \text{text}_i, \text{Dict}_{i-1}, \dots, \text{Dict}_1, \text{text}_1] \geq 2d_0 \cdot \frac{d-1}{d}.$$

For brevity, denote $X_i = X(\text{text}_{i+1}, \text{Dict}_i)$ and define the filtration $\mathcal{F}_i = \sigma(\{\text{text}_1, \text{Dict}_1, \dots, \text{text}_i, \text{Dict}_i\})$. Note that $\sum_{j=1}^i X_j - \mathbb{E}[X_j | \mathcal{F}_i]$ forms a martingale sequence under the filtration $\{\mathcal{F}_i : i \in [d]\}$. Therefore, by the Azuma-Hoeffding inequality, for any $\eta > 0$,

$$\Pr\left(\sum_{i=1}^{d-1} \mathbb{E}[X_i | \mathcal{F}_i] - X_i \leq -\eta\right) \leq e^{-\eta^2}. \quad (27)$$

Under Case I, we have that $\sum_{i=1}^d X_i \leq d_0$. Therefore, from eq. (26) and eq. (27),

$$\begin{aligned} \Pr\left(|\text{Dict}| < d_0; \mathbb{E}[X(\mathbf{s}', \text{Dict}) | \text{Dict}] \geq 2d_0/d\right) &\leq \Pr\left(\sum_{i=1}^{d-1} X_i < d_0; \sum_{i=1}^{d-1} \mathbb{E}[X_i | \mathcal{F}_i] \geq 2d_0 \cdot \frac{d-1}{d}\right) \\ &\leq \Pr\left(\sum_{i=1}^{d-1} \mathbb{E}[X_i | \mathcal{F}_i] - X_i \geq d_0 \cdot \frac{d-2}{d}\right) \\ &\leq e^{-d_0^2(1-2/d)^2} \\ &\leq e^{-d_0^2/2} = e^{-\epsilon^2 d / 8 \log^2(1/\delta)}. \end{aligned}$$

Finally, using the inequality $\Pr(A, B) = \Pr(A|B)\Pr(B) \geq (\min\{\Pr(A), \Pr(B)\})^2$ completes the proof. \square

Proofs of Theorem B.1.2 and Theorem B.1.3 If $\Pr(|\text{Dict}| < d_0) \leq e^{-\epsilon^2 d / 8 \log^2(1/\delta)}$ the proof of Theorem B.1.2 concludes. Otherwise, consider the case $\Pr(|\text{Dict}| < d_0) > e^{-\epsilon^2 d / 8 \log^2(1/\delta)}$, whereby, $\mathbb{E}[X(\mathbf{s}', \text{Dict}) | \text{Dict}] \leq 2d_0/d$ with probability $\geq 1 - e^{-\epsilon^2 d / 8 \log^2(1/\delta)}$ conditioned on $|\text{Dict}| < d_0$ by Lemma B.9. Recall that when $|\text{Dict}| < d_0$, Algorithm 1 uses a parallel implementation of the sequential encoder which chunks a new string into pieces of length d , denoted $\{\text{chunk}_i : i \in [d]\}$ and uses the sequential encoder under Dict_d to tokenize each chunk. Note that since the source is Markovian, the chunked process $\{\text{chunk}_i = (X_{id+1}, X_{id+2}, \dots, X_{(i+1)d}) : i = 1, 2, \dots\}$ is also Markovian and ergodic. Therefore, by a similar limiting argument as in Lemma A.4, using the Krylov–Bogolyubov argument (cf. Proposition 4.2 in Chen (2018)) for Markov processes, we have that,

$$\lim_{\ell \rightarrow \infty} \frac{\sum_{i=1}^{\ell} X(\text{chunk}_i, \text{Dict})}{\ell} = \mathbb{E}[X(\mathbf{s}', \text{Dict})] \leq \frac{2d_0}{d}.$$

where \mathbf{s}' is a fresh string of length d sampled with initial state distribution as the stationary measure of the stochastic source. On the remaining (limiting) $1 - 2d_0/d$ fraction of the chunks, their sequential encodings have every pair of tokens appearing at most $\log(d)$ times consecutively. Using Theorem 1 of Navarro and Russo (2008), the number of tokens in the encoding of each of these chunks cannot be too large, and satisfies,

$$\begin{aligned} |\text{enc}_{\text{BPE}}(\text{chunk}_i)| \cdot \log |\text{enc}_{\text{BPE}}(\text{chunk}_i)| &\leq 2dH_{\infty} + O(d/\log(d)) \\ \implies |\text{enc}_{\text{BPE}}(\text{chunk}_i)| \cdot \log d &\leq 2dH_{\infty} + O(d/\log(d)) \end{aligned} \quad (28)$$

For the (limiting) $2d_0/d$ fraction of the “bad” chunks, their sequential encodings may have one or more pairs of tokens which appear more than $\log(d)$ times consecutively.

Define $\mathcal{E}_i = \{X(\text{chunk}_i, \text{Dict}) = 1\}$ where $\text{Dict} = \text{Dict}_d$ is the dictionary returned by Algorithm 1 and consider the unigram model $Q_{\text{uni}}(\mathbf{t}) = \frac{1}{2}Q_1(\mathbf{t}) + \frac{1}{2}Q_2(\mathbf{t})$, which is the uniform mixture of two

models,

$$Q_1(\mathbf{t}) \propto \frac{1}{(2|\mathcal{A}|)^{|\mathbf{t}|}}, \quad \text{and} \quad Q_2(\mathbf{t}) = \mathbb{E} \left[\frac{n_{\mathbf{t}}^1}{|\text{enc}_{\text{BPE.split}}(\text{chunk}_1)|} \middle| \mathcal{E}_1^c \right],$$

and let $Q_{\text{uni}}(\mathbf{t}_1, \dots, \mathbf{t}_i) = Q_{\#}(j) \prod_{j=1}^i Q_{\text{uni}}(\mathbf{t}_i)$ for some distribution $Q_{\#}(i)$ over the number of tokens to be chosen later. We will analyze the case where the total number of chunks ℓ is finite and take the limit $m \rightarrow \infty$ later. Then, the overall loss of the algorithm is,

$$\begin{aligned} & \mathcal{L}_m(Q_{\text{uni}} \circ \text{enc}(\cdot)) \\ &= -\mathbb{E}[\log Q_{\text{uni}}(\text{enc}_{\text{BPE.split}}(\mathbf{s}))] \\ &= -\sum_{\mathbf{t} \in \text{Dict}} \mathbb{E}[n_{\mathbf{t}} \log Q_{\text{uni}}(\mathbf{t}) + \log Q_{\text{uni}}(|\text{enc}_{\text{BPE.split}}(\mathbf{s})|)] \\ &\stackrel{(i)}{=} -\sum_{i=1}^{\ell} \mathbb{E} \left[\sum_{\mathbf{t} \in \text{Dict}} n_{\mathbf{t}}^i \log Q_{\text{uni}}(\mathbf{t}) \right] + \log(m) \\ &= -\sum_{i=1}^{\ell} \mathbb{E} \left[\sum_{\mathbf{t} \in \text{Dict}} n_{\mathbf{t}}^i \log Q_{\text{uni}}(\mathbf{t}) \middle| \mathcal{E}_i \right] \Pr(\mathcal{E}_i) + \mathbb{E} \left[\sum_{\mathbf{t} \in \text{Dict}} n_{\mathbf{t}}^i \log Q_{\text{uni}}(\mathbf{t}) \middle| \mathcal{E}_i^c \right] \Pr(\mathcal{E}_i^c) + \log(m). \end{aligned} \quad (29)$$

where $n_{\mathbf{t}}^i$ is the number of times \mathbf{t} is observed in the BPE encoding of chunk_i and (i) uses the fact that $|\text{enc}_{\text{BPE.split}}(\mathbf{s})|$ follows some distribution supported on $[m]$, which implies its entropy is upper bounded by $\log(m)$. First observe that,

$$\begin{aligned} \sum_{i=1}^{\ell} \mathbb{E} \left[\sum_{\mathbf{t} \in \text{Dict}} n_{\mathbf{t}}^i \log Q_{\text{uni}}(\mathbf{t}) \middle| \mathcal{E}_i^c \right] &\leq \sum_{i=1}^{\ell} \mathbb{E} \left[|\text{enc}_{\text{BPE}}(\text{chunk}_i)| \cdot \sum_{\mathbf{t} \in \text{Dict}} \frac{n_{\mathbf{t}}^i}{|\text{enc}_{\text{BPE}}(\text{chunk}_i)|} \log Q_{\text{uni}}(\mathbf{t}) \middle| \mathcal{E}_i^c \right] \\ &\stackrel{(i)}{\leq} \ell \left(\frac{2dH_{\infty} + O(d/\log(d))}{\log(d)} \right) \sum_{\mathbf{t} \in \text{Dict}} Q_2(\mathbf{t}) \log Q_{\text{uni}}(\mathbf{t}) \end{aligned}$$

where (i) uses the upper bound on $|\text{enc}_{\text{BPE.split}}(\text{chunk}_i)|$ under the event \mathcal{E}_i^c (eq. (28)). Since $Q_{\text{uni}}(\mathbf{t}) = \frac{1}{2}Q_1(\mathbf{t}) + \frac{1}{2}Q_2(\mathbf{t}) \geq \frac{1}{2}Q_2(\mathbf{t})$ and Q_2 is a distribution supported on at most d tokens, this term results in the upper bound,

$$\sum_{i=1}^{\ell} \mathbb{E} \left[\sum_{\mathbf{t} \in \text{Dict}} n_{\mathbf{t}}^i \log Q_{\text{uni}}(\mathbf{t}) \middle| \mathcal{E}_i^c \right] \leq \ell \left(\frac{2dH_{\infty} + O(d/\log(d))}{\log(d)} \right) \log(2d). \quad (30)$$

On the other hand, since $Q_{\text{uni}}(\mathbf{t}) \geq \frac{1}{2}Q_1(\mathbf{t})$,

$$\begin{aligned} \sum_{i=1}^{\ell} \mathbb{E} \left[\sum_{\mathbf{t} \in \text{Dict}} n_{\mathbf{t}}^i \log(1/Q_{\text{uni}}(\mathbf{t})) \middle| \mathcal{E}_i \right] &\leq \sum_{i=1}^{\ell} \mathbb{E} \left[\sum_{\mathbf{t} \in \text{Dict}} n_{\mathbf{t}}^i \log(2/Q_1(\mathbf{t})) \middle| \mathcal{E}_i \right] \\ &\leq \sum_{i=1}^{\ell} \mathbb{E} \left[\sum_{\mathbf{t} \in \text{Dict}} n_{\mathbf{t}}^i (\log(2) + |\mathbf{t}| \log(2|\mathcal{A}|)) \middle| \mathcal{E}_i \right] \\ &\leq \ell d \log(2) + \ell d \log(2|\mathcal{A}|) \end{aligned} \quad (31)$$

where the last inequality uses the fact that $\sum_{t \in \text{Dict}} n_t^i \leq d$ and $\sum_{t \in \text{Dict}} |t|n_t^i = d$ computes the length of chunk_i .

Overall, since $\sum_{i=1}^{\ell} \Pr(\mathcal{E}_i) \leq 2d_0/d$ by eq. (28), combining this with eqs. (29) to (31),

$$\mathcal{L}_m(Q_{\text{uni}} \circ \text{enc}(\cdot)) \leq \left(1 - \frac{2d_0}{d}\right) \ell \left(\frac{2dH_{\infty} + O(d/\log(d))}{\log(d)} \right) \log(2d) + \frac{2d_0}{d} \ell d \log(4|\mathcal{A}|).$$

Dividing throughout by the length of the character sequence $m \in [d(\ell - 1), d\ell]$ and letting $\ell \rightarrow \infty$,

$$\min_{Q \in \mathcal{Q}_{\text{1-gram}}} \mathcal{L}(Q \circ \text{enc}(\cdot)) \leq \mathcal{L}(Q_{\text{uni}} \circ \text{enc}(\cdot)) \leq \left(1 - \frac{2d_0}{d}\right) \left(2H_{\infty} + O\left(\frac{1}{\log(d)}\right)\right) + \frac{2d_0}{d} \log(4|\mathcal{A}|).$$

C Additional Theoretical Results I: Learning the likelihood model

The guarantees we prove in Theorems 4.1, 4.5 and 4.7 on various tokenizers assume that the downstream model is trained optimally. In practice, these models are trained from a finite dataset and the sample complexity of learning this likelihood model scales with the number of tokens in the dictionary. In this section, we step away from the transformer architecture and focus on analyzing the performance of a simple estimator for the unigram model based on Laplace smoothing. We leave the problem of analyzing the finite-sample statistical error of simple transformer models trained with gradient descent as an interesting open direction for future research.

The result of Theorem 4.1 establishes that under appropriate assumptions on the Markov source, there exists a tokenizer \mathcal{T} and a unigram model over tokens $Q^* \in \mathcal{Q}_{\text{1-gram}}$ such that,

$$\begin{aligned} & \lim_{m \rightarrow \infty} \frac{1}{m} \mathbb{E} [\log(1/Q^*(\text{enc}(\mathbf{s})))] \\ & \leq (1 + \varepsilon) \cdot \lim_{m \rightarrow \infty} \frac{1}{m} \mathbb{E} [\log(1/P(\mathbf{s}))] \end{aligned}$$

Or in other words,

$$\lim_{m \rightarrow \infty} \frac{1}{m} \text{KL}(P, Q^*(\text{enc}(\cdot))) \leq \varepsilon \cdot \lim_{m \rightarrow \infty} \frac{1}{m} \mathbb{E} [\log(1/P(\mathbf{s}))].$$

This implies that with the appropriate tokenization, the measure associated to the string by the best unigram model over tokens is close to that induced by the true Markov distribution over characters in KL divergence. In this section, we establish finite-sample guarantees on learning Q^* specifically for the LZW tokenizer. The approach we consider for distribution learning is a smoothed Laplace estimator described in more detail in Algorithm 2.

For any constant $\theta \in (0, 1)$, define \mathcal{E}_{θ} as the event that every maximal token \mathbf{t} (Definition A.5) in the LZW dictionary satisfies $1/d^{1-\theta} \geq \max_a P(\mathbf{t}|a) \geq \delta/d^{1+\theta}$. By Lemmas A.10 and A.11 when the LZW tokenizer is trained on a dataset of size $\Omega_{\delta}(d)$ drawn from a stochastic source satisfying Assumption 4.2, \mathcal{E}_{θ} occurs with probability $\geq 1 - d^{-\Omega_{\theta, \delta}(\log(d))}$.

Theorem C.1. *Consider any constant $\theta \in (0, 1)$, failure probability $\eta \in (0, 1)$ and approximation error $\xi \in (0, 1)$. Assume that the learnt LZW tokenizer satisfies the event \mathcal{E}_{θ} , which occurs with probability $\geq 1 - d^{-\Omega_{\theta, \delta}(\log(d))}$. Assume that $d^{1-3\theta} \geq 1 + \delta^{-2}$ and that the stochastic source satisfies Assumption 4.2. For an absolute constant $C > 0$, assume that the size of the training dataset is at least $n_{lm}^*(\xi)$, where,*

$$n_{lm}^* \triangleq \frac{Cd^{1+\theta} \log^3(d/\eta\delta) \log \log(d/\eta)}{\delta\xi^2}$$

Then, Algorithm 2 learns a unigram model \widehat{Q} such that,

$$\mathcal{L}(\widehat{Q} \circ \text{enc}_{\text{gre}}(\cdot)) \leq (1 + \xi) \min_{Q \in \mathcal{Q}_{\text{1-gram}}} \mathcal{L}(Q \circ \text{enc}_{\text{gre}}(\cdot))$$

with probability $\geq 1 - \eta$.

In conjunction with Theorem 4.5, this gives end-to-end guarantees on the cross-entropy loss of the LZW tokenizer (with vocabulary size $\leq d$) with the Laplace estimator as the downstream unigram model. We instantiate this result choosing $\theta = 0.01$ in Theorem C.1.

Corollary C.2. *Choose any $\xi \in (0, 1)$. Suppose the data source satisfies Assumption 4.2. On a dataset of size $\widetilde{\Omega}_\delta(d)$ drawn from the source, train an LZW tokenizer having a dictionary size of d . Subsequently, using Algorithm 2, learn a unigram model \widehat{Q} using a dataset of size at least $\widetilde{\Omega}(d^{1.01}/\delta\xi^2)$ drawn from the source. Then, with probability $\geq 1 - d^{-\Omega_\delta(\log(d))}$,*

$$\mathcal{L}(\widehat{Q} \circ \text{enc}_{\text{gre}}(\cdot)) \leq \frac{1 + \xi}{1 - \varepsilon} \min_Q \mathcal{L}(Q),$$

where $\varepsilon = \frac{\log(1/\delta)}{0.99 \log(d)}$.

The analysis of Theorem C.1 relies on showing that the distribution over tokens induced when a string sampled from the data source is encoded into tokens by the greedy encoder and the LZW dictionary is a Markov process. In general, given a set of previously sampled tokens t_1, \dots, t_i , the next token t_{i+1} is sampled from the distribution $P(t_{i+1}|t_i; \forall j \in [i], t_{i-j+1} \dots t_i t_{i+1} \notin \text{Dict})$. The conditioning is to simply guarantee that the previous tokens which were sampled were indeed maximal, since if $t_i t_{i+1} \in \text{Dict}$, then the previous token returned would in fact have been this longer token and not t_i (and likewise for $t_{i-1} t_i t_{i+1}$ and so on). While in general, this process is complicated and depends on all the previous tokens sampled, for the LZW dictionary, we show that the conditioning $\{\forall j \in [i], t_{i-j+1} \dots t_i t_{i+1} \notin \text{Dict}\}$ can be removed, thereby resulting in a simple Markov process over tokens.

Furthermore, we establish that this Markov process has a relatively large spectral gap. The optimal unigram model ends up being the stationary distribution over tokens induced by greedy encoder. Given the large spectral gap of the Markov process over tokens, estimating the stationary distribution of this process in KL divergence ends up being closely related to estimating a distribution from i.i.d. samples in the same metric. For this problem, the de-facto choice of estimator is the Laplace estimator, and several existing results provide finite-sample bounds on the KL divergence (Braess and Sauer, 2004; Han et al., 2021; Mourtada and Gaiffas, 2022). The Laplace estimator (Line 6 of Algorithm 2) is simply a smoothed empirical estimate to account for the degeneracy of the KL divergence in its second argument as any coordinate approaches 0. The non-i.i.d.ness of the Markov process is circumvented by using concentration inequalities which are a function of the spectral gap (Naor et al., 2020).

Algorithm 2 Training likelihood model on tokens

Input: A training dataset of size n_{lm} , likelihood model class \mathcal{Q} , likelihood model training algorithm `TrainLM`

Output: Likelihood model $Q \in \mathcal{Q}$.

- 1: Tokenize the training dataset into a sequence of tokens $\mathcal{T} = (\mathbf{t}_1, \dots, \mathbf{t}_i)$.
- 2: Train a likelihood model Q on the tokenized dataset \mathcal{T} using the `TrainLM`(\mathcal{T}, \mathcal{Q}) subroutine.

// In the case of $\mathcal{Q} = \mathcal{Q}_{1\text{-gram}}$ use a smoothed Laplace estimator, instantiated below

def `TrainLM`($\mathcal{T}, \mathcal{Q}_{1\text{-gram}}$):

- 3: Truncate the dataset to the first $n' = \lfloor n_{\text{lm}}/\ell_{\max} \rfloor$ tokens where $\ell_{\max} = 4 \log(d|\mathcal{A}|)/\delta$. Let the truncated dataset be $\mathcal{T}_{\text{trunc}}$
- 4: Construct the unigram model \widehat{Q} with $\widehat{Q}_{\#} = \text{Unif}([m])$ and $\widehat{Q}_{\text{tok}}(\mathbf{t}) = \frac{n_t+1}{n_t+|\text{Dict}|}$.

// n_t is the number of times \mathbf{t} appears in $\mathcal{T}_{\text{trunc}}$.

// Test sequences are assumed to be of length m .

C.1 Proof of Theorem C.1

Since the LZW tokenizer uses the greedy encoder, the cross-entropy loss of the unigram model learnt by Algorithm 2 is,

$$\begin{aligned} & \mathcal{L}(\widehat{Q} \circ \text{enc}_{\text{gre}}(\cdot)) - \min_{Q \in \mathcal{Q}_{1\text{-gram}}} \mathcal{L}(Q \circ \text{enc}_{\text{gre}}(\cdot)) \\ &= \max_{Q \in \mathcal{Q}_{1\text{-gram}}} \lim_{m \rightarrow \infty} \frac{1}{m} \mathbb{E}[\log(Q(\text{enc}_{\text{gre}}(\mathbf{s})) / \widehat{Q}(\text{enc}_{\text{gre}}(\mathbf{s})))] \\ &\stackrel{(i)}{=} \max_{Q \in \mathcal{Q}_{1\text{-gram}}} \lim_{m \rightarrow \infty} \frac{1}{m} \mathbb{E} \left[|\text{enc}_{\text{gre}}(\mathbf{s})| \sum_{\mathbf{t} \in \text{Dict}} \frac{n_t}{|\text{enc}_{\text{gre}}(\mathbf{s})|} \log(Q_{\text{tok}}(\mathbf{t}) / \widehat{Q}_{\text{tok}}(\mathbf{t})) \right] + \frac{\log(m)}{m} \\ &\stackrel{(ii)}{\leq} \lim_{m \rightarrow \infty} \frac{1}{m} \mathbb{E} \left[|\text{enc}_{\text{gre}}(\mathbf{s})| \sum_{\mathbf{t} \in \text{Dict}} \frac{n_t}{|\text{enc}_{\text{gre}}(\mathbf{s})|} \log \left(\frac{n_t / |\text{enc}_{\text{gre}}(\mathbf{s})|}{\widehat{Q}_{\text{tok}}(\mathbf{t})} \right) \right] + \frac{\log(m)}{m} \end{aligned}$$

where in (i) we use the fact that $\widehat{Q}_{\#} = \text{Unif}([m])$ and in (ii) we take the $\max\{\cdot\}$ inside the limit and the expectation (Fatou's lemma and Jensen's inequality) and plug in the maximizer of the negative cross-entropy, $Q_{\text{tok}}(\mathbf{t}) = \frac{n_t}{|\text{enc}_{\text{gre}}(\mathbf{s})|}$. Note that $\lim_{m \rightarrow \infty} \frac{n_t}{|\text{enc}_{\text{gre}}(\mathbf{s})|} \stackrel{\text{a.s.}}{=} Q_{\text{MLE}}(\mathbf{t})$ by Lemma A.4. Moreover, since $|\text{enc}(\mathbf{s})|/m \leq 1$ and $\widehat{Q}_{\text{tok}}(\mathbf{t}) > 0$ surely, by the Dominated Convergence Theorem,

$$\mathcal{L}(\widehat{Q} \circ \text{enc}_{\text{gre}}(\cdot)) - \min_{Q \in \mathcal{Q}_{1\text{-gram}}} \mathcal{L}(Q \circ \text{enc}_{\text{gre}}(\cdot)) \leq \lim_{m \rightarrow \infty} \frac{1}{m} \mathbb{E}[|\text{enc}_{\text{gre}}(\mathbf{s})|] \cdot \text{KL}(Q_{\text{MLE}}, \widehat{Q}_{\text{tok}}) \quad (32)$$

By eq. (8), we have that for any tokenizer using the greedy encoder,

$$\lim_{m \rightarrow \infty} \frac{|\text{enc}_{\text{gre}}(\mathbf{s})| (H(Q_{\text{MLE}}, P) - \log(1/\delta))}{m} \stackrel{\text{a.s.}}{\leq} H_{\infty}.$$

Furthermore under the event \mathcal{E}_{θ} which implies that the learnt dictionary is $(1 - \theta)$ -heavy hitting (cf. Definition A.5), which implies that,

$$H(Q_{\text{MLE}}, P) \geq (1 - \theta) \log(d).$$

Therefore, by almost sure boundedness, we have that,

$$\lim_{m \rightarrow \infty} \frac{1}{m} \mathbb{E} [|\text{enc}_{\text{gre}}(\mathbf{s})|] \leq \frac{H_\infty}{(1 - \theta) \log(d) - \log(1/\delta)} \leq \frac{\min_{Q \in \mathcal{Q}_{1\text{-gram}}} \mathcal{L}(Q \circ \text{enc}(\cdot))}{(1 - \theta) \log(d) - \log(1/\delta)}$$

Putting this together with eq. (32), we have that,

$$\mathcal{L}(\widehat{Q} \circ \text{enc}_{\text{gre}}(\cdot)) \leq \left(1 + \text{KL}(Q_{\text{MLE}}, \widehat{Q}_{\text{tok}})\right) \min_{Q \in \mathcal{Q}_{1\text{-gram}}} \mathcal{L}(Q \circ \text{enc}_{\text{gre}}(\cdot)), \quad (33)$$

which uses the assumption $(1 - \theta) \log(d) \geq 1 + \log(1/\delta)$. In the remainder of the proof we upper bound the KL term.

By the law of large numbers established in eq. (35) and the fact that $\frac{n_t}{\sum_{t'} n_{t'}} \in [0, 1]$, we have that,

$$Q_{\text{MLE}}(\mathbf{t}) = \lim_{m \rightarrow \infty} \mathbb{E} \left[\frac{n_t}{\sum_{t'} n_{t'}} \right] = \lim_{m \rightarrow \infty} \frac{\mathbb{E}[n_t]}{\mathbb{E}[\sum_{t'} n_{t'}]} = \pi(\mathbf{t}),$$

where $\pi(\mathbf{t})$ denote the stationary distribution over tokens induced by the greedy encoding process, which exists for the LZW tokenizer. This distribution is in fact an ergodic Markov process, as we discuss next.

By Lemmas A.10 and A.11, for any constant $\theta \in (0, 1)$, with probability $\geq 1 - d^{-\Omega_{\theta, \delta}(\log(d))}$, every maximal token in the LZW dictionary satisfies $1/d^{1-\theta} \geq \max_a P(\mathbf{t}|a) \geq \delta/d^{1+\theta}$. Let S_{gre} denote the set of tokens which have a non-zero probability (over a string drawn from the Markov source) of being chosen by the greedy encoder while encoding the string. More importantly, note that for any sequence of tokens $\mathbf{t}_1, \dots, \mathbf{t}_i$, the next token is necessarily in S_{gre} and can be any token in this set. The reason for this is that for any $\mathbf{t}_i, \mathbf{t} \in S_{\text{gre}}$, the concatenation $\mathbf{t}_i \mathbf{t} \notin S_{\text{gre}}$ since $\max_{a \in \mathcal{A}} P(\mathbf{t}_i \mathbf{t} | a) \leq 1/\delta d^{2(1-\theta)}$, which is smaller than the $\max_{a \in \mathcal{A}} P(\mathbf{t}' | a) \geq \delta/d^{1+\theta}$ for any token $\mathbf{t}' \in S_{\text{gre}}$ as long as $d^{1-3\theta} \geq 1/\delta^2$. This constraint implies that in the sampling procedure in Figure 9, it suffices to drop the conditioning on the event $\mathbf{t}_j \mathbf{t}_{j+1} \dots \mathbf{t}_i \mathbf{t} \notin \text{Dict}$ while sampling the next token \mathbf{t} . This condition automatically implies that the sequence of tokens conditionally follows a Markov process with $\Pr(\mathbf{t}_{i+1} = \mathbf{t} | \mathbf{t}_1, \dots, \mathbf{t}_i) = P(\mathbf{t} | \text{last}(\mathbf{t}_i))$. Since the probability of every transition is lower bounded, this means that the Markov chain is ergodic. Moreover, the pseudo-spectral gap (Naor et al., 2020), $1 - \lambda$ can be lower bounded by the Dobrushin contraction coefficient, κ ,

$$\begin{aligned} 1 - \lambda &\triangleq \max_{(\mathbf{t}, \mathbf{t}') \in \text{Dict}^2} \|\Pr(\cdot | \mathbf{t}) - \Pr(\cdot | \mathbf{t}')\|_{\text{TV}} \\ &= \max_{(\mathbf{t}, \mathbf{t}') \in \text{Dict}^2} 1 - \sum_{\mathbf{t}'' \in \text{Dict}} \min\{\Pr(\mathbf{t}'' | \mathbf{t}), \Pr(\mathbf{t}'' | \mathbf{t}')\} \\ &\leq 1 - \delta d / d^{1+\theta} \\ &= 1 - \delta d^{-\theta}. \end{aligned} \quad (34)$$

Recall that the learner is given a training dataset of n_{lm} characters to train the likelihood model. By Lemma A.8, with probability $\geq 1 - d^{-\Omega(\log(d/\delta)/\delta)}$, in the run of the LZW tokenization algorithm, every token in the dictionary has length at most $\ell_{\max} = 4 \log(d|\mathcal{A}|)/\delta$. Therefore, suppose the learner always truncates the dataset to the first $n' = \lfloor n_{\text{lm}}/\ell_{\max} \rfloor$ tokens and runs the Laplace estimator on this truncated dataset. With this, we move onto upper bounding,

$$\text{KL}(Q_{\text{MLE}}, \widehat{Q}_{\text{tok}}) = \sum_{\mathbf{t} \in \text{Dict}} \pi(\mathbf{t}) \log \left(\pi(\mathbf{t}) / \widehat{Q}_{\text{tok}}(\mathbf{t}) \right)$$

which necessitates lower bounding $\widehat{Q}_{\text{tok}}(\mathbf{t})$ for every \mathbf{t} . Recall that the learner's estimate $\widehat{Q}(\mathbf{t})$ in Algorithm 2 is the Laplace estimator, $\frac{n_{\mathbf{t}}+1}{\sum_{\mathbf{t}' \neq \mathbf{t}} n_{\mathbf{t}'} + |\text{Dict}|}$, where $\{n_{\mathbf{t}} : \mathbf{t} \in \text{Dict}\}$ is computed by truncating the dataset to the first n' tokens. Firstly, by invoking Corollary 1.3 of Naor et al. (2020) for the function $n_{\mathbf{t}} = \sum_{i=1}^{n'} \mathbb{I}(\mathbf{t}_i = \mathbf{t})$,

$$\Pr \left(|n_{\mathbf{t}} - \mathbb{E}[n_{\mathbf{t}}]| \geq c \sqrt{\frac{\mathbb{E}[n_{\mathbf{t}}]}{1-\lambda} \cdot \log(1/\eta)} \right) \leq \eta \quad (35)$$

for a universal constant $c > 0$. In particular, this implies that with probability $\geq 1-\eta$, simultaneously for all \mathbf{t} ,

$$|n_{\mathbf{t}} - \mathbb{E}[n_{\mathbf{t}}]| \leq \Delta_{\mathbf{t}} \triangleq \sqrt{\frac{d^{\theta}}{\delta} \mathbb{E}[n_{\mathbf{t}}] \cdot \log(|\text{Dict}|/\eta)}, \text{ and, } \mathbb{E}[n_{\mathbf{t}}] - n_{\mathbf{t}} \geq \mathbb{E}[n_{\mathbf{t}}].$$

Under this event, for any \mathbf{t} , the estimate is lower bounded by,

$$\begin{aligned} \widehat{Q}_{\text{tok}}(\mathbf{t}) &= \frac{n_{\mathbf{t}} + 1}{n' + |\text{Dict}|} \geq \frac{\mathbb{E}[n_{\mathbf{t}}] + 1 - \min\{\mathbb{E}[n_{\mathbf{t}}], \Delta_{\mathbf{t}}\}}{n' + |\text{Dict}|} \\ &\geq \max \left\{ \pi(\mathbf{t}) - \frac{(\Delta_{\mathbf{t}} - 1)n' + |\text{Dict}|\mathbb{E}[n_{\mathbf{t}}]}{(n')^2 + n'|\text{Dict}|}, \frac{1}{n' + |\text{Dict}|} \right\} \\ &\geq \max \left\{ \pi(\mathbf{t}) - \frac{\Delta_{\mathbf{t}}n' + |\text{Dict}|\mathbb{E}[n_{\mathbf{t}}]}{(n')^2}, \frac{1}{n' + |\text{Dict}|} \right\} \end{aligned}$$

Suppose the following condition is satisfied,

$$n' = \frac{4rd^{\theta}|\text{Dict}| \log(|\text{Dict}|/\eta)}{\delta} \quad (\text{C1})$$

for some $r \geq 4$. Under this condition, we have that $n' \geq 2\sqrt{r}\Delta$ and $n' \geq 4r|\text{Dict}|$.

Case I. $\Delta_{\mathbf{t}}n' \geq |\text{Dict}|\mathbb{E}[n_{\mathbf{t}}]$.

In this case, we have the upper bound,

$$\begin{aligned} \widehat{Q}_{\text{tok}}(\mathbf{t}) &\geq \max \left\{ \pi(\mathbf{t}) - \frac{2\Delta_{\mathbf{t}}}{n'}, \frac{1}{n' + |\text{Dict}|} \right\} \\ &= \max \left\{ \pi(\mathbf{t}) - 2 \frac{\sqrt{\frac{d^{\theta}}{\delta} \mathbb{E}[n_{\mathbf{t}}] \cdot \log(|\text{Dict}|/\eta)}}}{n'}, \frac{1}{n' + |\text{Dict}|} \right\} \\ &\geq \max \left\{ \pi(\mathbf{t}) - \sqrt{\frac{\pi(\mathbf{t})}{r|\text{Dict}|}}, \frac{1}{2n'} \right\}. \end{aligned}$$

where the last inequality uses eq. (C1).

Consider two sub-cases,

Sub-case I. $\pi(\mathbf{t}) \geq 2/r|\text{Dict}|$. Define this event \mathcal{C}_l .

Here,

$$\pi(\mathbf{t}) \log(\pi(\mathbf{t})/\widehat{Q}_{\text{tok}}(\mathbf{t})) \leq -\pi(\mathbf{t}) \log \left(1 - \sqrt{\frac{1}{\pi(\mathbf{t})r|\text{Dict}|}} \right) \leq \frac{3}{2} \sqrt{\frac{\pi(\mathbf{t})}{r|\text{Dict}|}}. \quad (36)$$

Sub-case II. $\pi(\mathbf{t}) \leq 2/r|\text{Dict}|$. Define this event \mathcal{C}_{II} .

Here,

$$\begin{aligned}\pi(\mathbf{t}) \log(\pi(\mathbf{t})/\hat{Q}_{\text{tok}}(\mathbf{t})) &\leq \pi(\mathbf{t}) \log(2n'\pi(\mathbf{t})) \leq \max \left\{ 0, \frac{2}{r|\text{Dict}|} \log \left(\frac{4n'}{r|\text{Dict}|} \right) \right\} \\ &\leq \frac{2}{r|\text{Dict}|} \log \left(16d^\theta \log(|\text{Dict}|/\eta) \right)\end{aligned}\quad (37)$$

Case III. $\Delta_t n' < |\text{Dict}| \mathbb{E}[n_t]$. Define this event \mathcal{C}_{III} .

In this case we have the upper bound,

$$\hat{Q}_{\text{tok}}(\mathbf{t}) \geq \pi(\mathbf{t}) - \frac{2|\text{Dict}| \mathbb{E}[n_t]}{(n')^2} \geq \pi(\mathbf{t}) - \frac{\pi(\mathbf{t})}{2r}$$

where the last inequality follows from eq. (C1). This implies that,

$$\pi(\mathbf{t}) \log(\pi(\mathbf{t})/\hat{Q}_{\text{tok}}(\mathbf{t})) \leq -\pi(\mathbf{t}) \log(1 - 1/2r) \leq \frac{\pi(\mathbf{t})}{r}.\quad (38)$$

By using the geometric ergodicity of this Markov process (eq. (34)), when n' tokens are sampled from an arbitrary initial distribution,

$$(1 - \kappa^{n'}) \pi(\mathbf{t}) \leq \frac{\mathbb{E}[n_t]}{n'} \leq \kappa^{n'} + (1 - \kappa^{n'}) \pi(\mathbf{t}) \implies \pi(\mathbf{t}) \leq \frac{\hat{Q}_{\text{tok}}(\mathbf{t})}{1 - e^{-4r|\text{Dict}| \log(|\text{Dict}|/\eta)}} = \frac{\hat{Q}_{\text{tok}}(\mathbf{t})}{1 - d^{-r}}$$

where in the implication, we use the condition on n' in eq. (C1) and the bound on the contraction coefficient κ in eq. (34).

$$\begin{aligned}&\mathsf{KL}(Q_{\text{MLE}}, \hat{Q}_{\text{tok}}) \\&= \sum_{\mathbf{t} \in \text{Dict}} \pi(\mathbf{t}) \log(\pi(\mathbf{t})/\hat{Q}_{\text{tok}}(\mathbf{t})) \\&\leq \sum_{\mathbf{t} \in \text{Dict}} \frac{\pi(\mathbf{t})}{1 - d^{-r}} \log(\pi(\mathbf{t})/\hat{Q}_{\text{tok}}(\mathbf{t})) - \log(1 - d^{-r}) \\&\leq \frac{1}{1 - d^{-r}} \sum_{\mathbf{t} \in \text{Dict}} \mathbb{I}(\mathcal{C}_{\text{I}}) \pi(\mathbf{t}) \log(\pi(\mathbf{t})/\hat{Q}_{\text{tok}}(\mathbf{t})) + \mathbb{I}(\mathcal{C}_{\text{II}}) \pi(\mathbf{t}) \log(\pi(\mathbf{t})/\hat{Q}_{\text{tok}}(\mathbf{t})) + \mathbb{I}(\mathcal{C}_{\text{III}}) \pi(\mathbf{t}) \log(\pi(\mathbf{t})/\hat{Q}_{\text{tok}}(\mathbf{t})) + 2d^{-r} \\&\leq \frac{1}{1 - d^{-r}} \sum_{\mathbf{t} \in \text{Dict}} \underbrace{\mathbb{I}(\mathcal{C}_{\text{I}}) \frac{3}{2} \sqrt{\frac{\pi(\mathbf{t})}{r|\text{Dict}|}}}_{\text{eq. (36)}} + \underbrace{\mathbb{I}(\mathcal{C}_{\text{II}}) \frac{2}{r|\text{Dict}|} \log \left(16d^\theta \log(|\text{Dict}|/\eta) \right)}_{\text{eq. (37)}} + \underbrace{\mathbb{I}(\mathcal{C}_{\text{III}}) \frac{\pi(\mathbf{t})}{r}}_{\text{eq. (38)}} + 2d^{-r} \\&\leq \frac{1}{1 - d^{-r}} \left(\frac{3}{2} \sqrt{\frac{|\text{Dict}|}{r|\text{Dict}|}} + \frac{2}{r} \log(16d^\theta \log(|\text{Dict}|/\eta)) + \frac{1}{r} \right) + 2d^{-r} \\&\leq \frac{5}{\sqrt{r}} \log(16d^\theta \log(|\text{Dict}|/\eta))\end{aligned}$$

Combining with eq. (33), we get the bound,

$$\begin{aligned}\mathcal{L}(\hat{Q} \circ \text{enc}_{\text{gre}}(\cdot)) &\leq \left(1 + \mathsf{KL}(Q_{\text{MLE}}, \hat{Q}) \right) \min_{Q \in \mathcal{Q}_{\text{1-gram}}} \mathcal{L}(Q \circ \text{enc}_{\text{gre}}(\cdot)) \\&\leq \left(1 + \frac{5}{\sqrt{r}} \log(16d^\theta \log(d/\eta)) \right) \min_{Q \in \mathcal{Q}_{\text{1-gram}}} \mathcal{L}(Q \circ \text{enc}_{\text{gre}}(\cdot)).\end{aligned}$$

Rescaling r to be $r(\log(16d^\theta \log(d/\eta)))^2$ completes the proof.

D Additional Theoretical Results II: The generalization ability of tokenizers

The proofs of the upper bounds in the paper (Theorems 4.5 and 4.7) relied on showing that the entropy $H(Q_{\text{MLE}}, P)$ is large, or in other words, the algorithm typically encodes new strings into long length (i.e. low probability under P) tokens. This statement about generalization to new strings is fundamentally different from having a tokenizer which compresses the training dataset well. In other words, consider the following modification: the measure Q_{MLE} is defined as the expected empirical distribution over tokens when a new string is encoded into tokens, and not on the source dataset used to construct the dictionary. Suppose the definition of Q_{MLE} is changed to the empirical distribution over tokens in the source dataset. Under this new definition of the MLE unigram model, the largeness of the $H(Q_{\text{MLE}}, P)$ metric, in a sense, captures compressing the source dataset well. However, we show that in general, this does not result in good tokenizers that minimize the population cross-entropy loss, suffering from $\min_{Q \in \mathcal{Q}_{1\text{-gram}}} \mathcal{L}(Q \circ \text{enc}(\cdot)) \approx H(\pi) \gg H_\infty$.

Theorem D.1. *Consider the stochastic source in example A.1 having entropy rate $H_\infty = \delta \log(1/\delta) + (1 - \delta) \log(1/(1 - \delta))$. Consider a training dataset of size n . For a dictionary Dict and $\mathbf{t} \in \text{Dict}$, define $\widehat{Q}_{\text{MLE}}(\mathbf{t}) = \frac{n_{\mathbf{t}}(\mathbf{s}_{\text{src}})}{\lvert \text{enc}(\mathbf{s}_{\text{src}}) \rvert}$ as the empirical distribution over tokens induced by the greedy encoder when encoding the training dataset, \mathbf{s}_{src} . There exists a dictionary Dict such that with probability $\geq 1 - e^{-\Omega(\sqrt{n})}$ over the training dataset,*

$$H(\widehat{Q}_{\text{MLE}}, P_\gamma) \geq nH_\infty(1 - O(n^{-1/4}))$$

is large. However, for this dictionary, for any encoding algorithm (including the greedy encoder), the resulting tokenizer $\mathcal{T} = (\text{Dict}, \emptyset, \text{enc}(\cdot), \text{dec}(\cdot))$ satisfies,

$$\min_{Q \in \mathcal{Q}_{1\text{-gram}}} \mathcal{L}(Q \circ \text{enc}(\cdot)) \geq (1 - \varepsilon)H(\pi)$$

where $\varepsilon = 2ne^{-nH_\infty(1 - O(n^{-1/4}))}$.

Proof. Suppose the entire training dataset was compressed into a single token, \mathbf{t}_{src} . The dictionary is $\mathcal{A} \cup \mathbf{t}_{\text{src}}$. In the following argument, we show that the number of occurrences, $n_{\mathbf{t}_{\text{src}}}$, of the entire training dataset \mathbf{t}_{src} in a new string of length m generated from the stochastic source, \mathbf{s} , converges to its expectation as $m \rightarrow \infty$. Let $\pi_n^{(i)}$ denote the stationary distribution of the Markov process induced by the stochastic source over length- n strings with a shift of i from the starting position, and let $n_{\mathbf{t}}^{(i)}$ denote the number of times \mathbf{t} appears in the training dataset starting at the position $i + rn$ for some $r > 0$. Then,

$$\lim_{m \rightarrow \infty} \frac{n_{\mathbf{t}_{\text{src}}}}{m} = \frac{1}{n} \lim_{m \rightarrow \infty} \sum_{i=0}^{n-1} \frac{n_{\mathbf{t}_{\text{src}}}^{(i)}}{m/n} \stackrel{\text{a.s.}}{=} \frac{1}{n} \sum_{i=0}^{n-1} \mathbb{E}_{\mathbf{t}' \sim \pi_n^{(i)}} [P(\mathbf{t}_{\text{src}} | \mathbf{t}')] \leq \max_{a \in \mathcal{A}} P(\mathbf{t}_{\text{src}} | a). \quad (39)$$

The second equation follows by considering the Markov process induced over length n strings and applying the Krylov–Bogolyubov argument for ergodic and homogeneous Markov processes.

In Lemma D.2, we show that with probability $\geq 1 - e^{-\Omega(\sqrt{n})}$, the token \mathbf{t}_{src} constructed from the source dataset satisfies, $\max_{a \in \mathcal{A}} P(\mathbf{t} | a) \leq e^{-nH_\infty(1 - O(n^{-1/4}))}$. In other words, the source string has exponentially small probability. Combining this with eq. (39), with probability $\geq 1 - e^{-\Omega(\sqrt{n})}$ over the source dataset, the number of occurrences of the substring \mathbf{t}_{src} in a new string \mathbf{s} is upper bounded by,

$$\lim_{m \rightarrow \infty} \frac{n_{\mathbf{t}_{\text{src}}}}{m} \stackrel{\text{a.s.}}{\leq} e^{-nH_\infty(1 - O(n^{-1/4}))} \triangleq \varepsilon/2n.$$

By the Krylov–Bogolyubov argument, for each $a \in \mathcal{A} = \{0, 1\}$, $\lim_{m \rightarrow \infty} \frac{n_a}{m} \stackrel{\text{a.s.}}{=} \pi(a)$. More importantly, the number of times a is made as a token is upper bounded by n_a and lower bounded by $n_a - nn_{\mathbf{t}_{\text{src}}}$. Therefore,

$$(1 - \varepsilon)\pi(a) = \pi(a) - \frac{\varepsilon}{2} \leq \lim_{m \rightarrow \infty} \frac{n_a}{m} \stackrel{\text{a.s.}}{\leq} \pi(a) = \frac{1}{2} \quad (40)$$

Finally, putting everything together,

$$\begin{aligned} \min_{Q \in \mathcal{Q}_{1\text{-gram}}} \lim_{m \rightarrow \infty} \frac{1}{m} \mathcal{L}_m(Q \circ \text{enc}(\cdot)) &= \min_{Q \in \mathcal{Q}_{1\text{-gram}}} \lim_{m \rightarrow \infty} -\frac{1}{m} \mathbb{E} \left[\log(Q_{\#}(|\text{enc}(\mathbf{s})|)) + \sum_{\mathbf{t} \in \text{Dict}} n_{\mathbf{t}} \log Q_{\text{tok}}(\mathbf{t}) \right] \\ &\geq \min_{Q \in \mathcal{Q}_{1\text{-gram}}} \lim_{m \rightarrow \infty} -\frac{1}{m} \mathbb{E} \left[\sum_{a \in \mathcal{A}} n_a \log Q_{\text{tok}}(a) \right] \\ &\stackrel{(i)}{\geq} \min_{Q \in \mathcal{Q}_{1\text{-gram}}} -(1 - \varepsilon) \sum_{a \in \mathcal{A}} \pi(a) \log Q_{\text{tok}}(a) \\ &\geq (1 - \varepsilon)H(\pi). \end{aligned}$$

where (i) follows from the lower bound on n_a/m in eq. (40). This completes the proof. \square

Lemma D.2. *With probability $\geq 1 - e^{-\Omega(\sqrt{n})}$ over the source dataset,*

$$\max_{a \in \mathcal{A}} P(\mathbf{t}_{\text{src}}|a) \leq e^{-nH(\delta)(1 - O(n^{-1/4}))}.$$

Proof. Let X denote the number of $i \in [n - 1]$ such that $\mathbf{s}_i \neq \mathbf{s}_{i+1}$ in \mathbf{s} , the stochastic source. Since the transition of the Markov process only depends on whether the next character is the same as the previous character, we can write down,

$$\max_{a \in \mathcal{A}} \log P(\mathbf{t}_{\text{src}}|a) = -(X + 1) \log(\delta) - (n - 1 - X) \log(1 - \delta).$$

Note that X is a sum of $n - 1$ i.i.d. random variables, since $\mathbb{I}(\mathbf{s}_i \neq \mathbf{s}_{i+1}) \sim \text{Ber}(\delta)$ does not depend on whether $\mathbf{s}_i = 0$ or $= 1$. In particular, by Hoeffding's inequality, we have that with probability $\geq 1 - e^{-\Omega(\sqrt{n})}$,

$$\left| \frac{1}{n} \max_{a \in \mathcal{A}} \log P(\mathbf{t}_{\text{src}}|a) - H(\delta) \right| \leq O\left(n^{-1/4}\right),$$

which uses the fact that $\mathbb{E}[X] = \delta(n - 1)$ and $H_\infty = \delta \log(1/\delta) + (1 - \delta) \log(1/(1 - \delta))$. Taking an exponential on both sides proves the statement of the lemma. \square

E Additional Theoretical Results III: Interaction between the dictionary and encoding algorithm

In this section, we show another kind of barrier to generalization, which brings out the relationship between the encoding algorithm and the dictionary. We show that there exist dictionaries which generalize under the minimal encoder, i.e. the encoding algorithm which encodes a string into the shortest number of possible tokens, but at the same time, completely fail to generalize under the greedy encoder. This means that in the process of constructing good tokenizers, it does not suffice to think about the dictionary in isolation. Its interaction with the encoding algorithm is pertinent.

Definition E.1 (minimal encoder). The minimal encoder parses a new string into the fewest possible number of tokens from the dictionary as possible. Ties are broken arbitrarily.

Theorem E.2. *There exists a stochastic source parameterized by $\delta \in (0, 0.5)$ and a dictionary Dict such that under the minimal encoder/decoder pair, the resulting tokenizer, $\mathcal{T} = (\text{Dict}, \emptyset, \text{enc}_{\min}(\cdot), \text{dec}_{\min}(\cdot))$ generalizes near-optimally,*

$$\min_{Q \in \mathcal{Q}_{1\text{-gram}}} \mathcal{L}(Q \circ \text{enc}_{\min}(\cdot)) \leq 1.273H_\infty. \quad (41)$$

Here the entropy rate of the source, H_∞ , is $\delta \log(\sqrt{2}/\delta) + (1 - \delta) \log(1/(1 - \delta))$. However, the same dictionary Dict under the greedy encoder/decoder pair, i.e. $\mathcal{T}' = (\text{Dict}, \emptyset, \text{enc}_{\text{gre}}(\cdot), \text{dec}_{\text{gre}}(\cdot))$, generalizes poorly, suffering from cross-entropy scaling as,

$$\min_{Q \in \mathcal{Q}_{1\text{-gram}}} \mathcal{L}(Q \circ \text{enc}_{\text{gre}}(\cdot)) \geq \frac{1 - o_\delta(1)}{3} H(\pi). \quad (42)$$

where the entropy of the stationary distribution of the source is $H(\pi) = \frac{1}{2} \log(8)$ and the $1 - o_\delta(1)$ term is $(1 - \delta)^2(1 + \delta)^{-1}$.

This means that the greedy encoder is not really compatible with the dictionary in the sense that the cross-entropy loss of the tokenizer is a constant multiple away from that achieved by the character-level tokenizer. The separation between eq. (41), and eq. (42) only manifests as δ becomes smaller and smaller.

In this section, we prove that generalization of a dictionary is a function of the underlying tokenization algorithm used. In particular, the greedy encoder is not universal, and there exists dictionaries under the minimum-length encoder/decoder which achieve small cross-entropy loss, which do not generalize under the greedy encoder/decoder.

We split the proof of Theorem E.2 into two parts. We first define the stochastic source and dictionary we consider. Then we show that under the minimum-length encoder, the asymptotic cross-entropy loss is upper bounded by H_∞ up to a constant. Finally, we show that under the greedy-encoder, the same dictionary suffers from high cross-entropy loss, which is a constant factor away from that of the character encoder.

E.1 Stochastic source and dictionary.

Consider an extension of the switching Markov source in example A.1 to $\mathcal{A} = \{0, 1, 2\}$. The Markov chain is described in Figure 12. The transition of the Markov chain is $P(0|0) = P(1|1) = P(2|2) = 1 - \delta$, and $P(1|0) = P(2|1) = \delta$ and $P(2|1) = P(0|1) = \delta/2$, with the remaining transitions being 0-probability. For a parameter $\ell > 0$ to be instantiated later, define S_1 (resp. S_0 , S_2) as the set of all-1 (resp. all-0, all-2) strings of length $\leq \ell - 1$, including the empty string. Consider a dictionary composed of the following set of tokens, $\{1s : s \in S_0 \cup S_1 \cup S_2\}$. Therefore, the tokens follow the template $10 \cdots 0$, $11 \cdots 1$ or $12 \cdots 2$ and are of length at most ℓ . ℓ is chosen to be $1 + 2 \log(1/\delta)/\delta$.

Although we use the minimal encoder in the statement of Theorem E.2, for the purpose of analysis, define the following encoding algorithm: if the new string is prefixed by $10 \cdots 0$ or $12 \cdots 2$, select the largest prefix which exists in dictionary and assign it as a token. If the new string starts with a sequence $11 \cdots 1$ of length x , consider the first $\max\{\ell, x - 1\}$ length prefix and assign it as a token. Finally, if the string starts with 0 or 2, assign that character as token. Once the first token has been assigned, remove it and repeat.

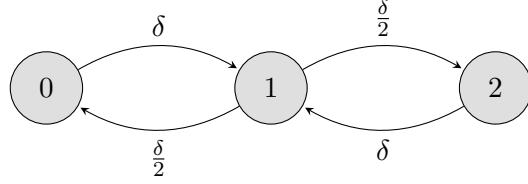


Figure 12: order-1 Markov source used in the proof of Theorem E.2

E.2 Minimal encoder achieves the optimal cross-entropy loss up to a constant.

First consider a simplification of the overall cross-entropy loss,

$$\begin{aligned} & \min_{Q \in \mathcal{Q}_{\text{1-gram}}} \lim_{m \rightarrow \infty} \frac{1}{m} \mathcal{L}_m(Q \circ \text{enc}(\cdot)) \\ &= \min_{Q \in \mathcal{Q}_{\text{1-gram}}} \lim_{m \rightarrow \infty} -\frac{1}{m} \mathbb{E} \left[\log Q_{\#}(|\text{enc}_{\min}(\mathbf{s})|) + \sum_{t \in \text{Dict}} n_t \log Q_{\text{tok}}(\mathbf{t}) \right] \end{aligned} \quad (43)$$

$$\leq \lim_{m \rightarrow \infty} \frac{1}{m} \mathbb{E} [\log(m) + |\text{enc}_{\min}(\mathbf{s})| \log |\text{Dict}|], \quad (44)$$

where in the last inequality we upper bound by choosing $Q_{\#} = \text{Unif}([m])$ and $Q_{\text{tok}}(\mathbf{t}) = 1/|\text{Dict}|$. Note that $|\text{Dict}| \leq 2\ell + 1$ and letting $\lim_{m \rightarrow \infty} \log(m)/m = 0$,

$$\begin{aligned} \min_{Q \in \mathcal{Q}_{\text{1-gram}}} \lim_{m \rightarrow \infty} \frac{1}{m} \mathcal{L}_m(Q \circ \text{enc}(\cdot)) &\leq \lim_{m \rightarrow \infty} \frac{1}{m} \mathbb{E}[|\text{enc}_{\min}(\mathbf{s})| \log(2\ell + 1)] \\ &\leq \lim_{m \rightarrow \infty} \frac{1}{m} \mathbb{E}[|\text{enc}(\mathbf{s})| \log(2\ell + 1)], \end{aligned} \quad (45)$$

where in (i), we replace $|\text{enc}_{\min}(\mathbf{s})|$ by $|\text{enc}(\mathbf{s})|$, which is the encoder we define in Appendix E.1. By definition of the minimal encoder, $|\text{enc}_{\min}(\mathbf{s})| \leq |\text{enc}(\mathbf{s})|$ surely. Recall that the encoder $\text{enc}(\cdot)$ processes strings in a sequential (left-to-right) manner. In particular, by a similar argument as Lemma A.4, we can show that under this encoder, the limit $n_t / \sum_{t'} n_{t'}$ almost surely converges to its expectation. More importantly, since, $\sum_{t \in \text{Dict}} |\mathbf{t}| n_t = m$, we have that,

$$\lim_{m \rightarrow \infty} \frac{|\text{enc}(\mathbf{s})|}{m} \stackrel{\text{a.s.}}{=} \frac{1}{\mathbb{E}_{\mathbf{t} \sim Q_{\text{MLE}}} [|\mathbf{t}|]}.$$

converges to some limit almost surely. Therefore, from eq. (45),

$$\min_{Q \in \mathcal{Q}_{\text{1-gram}}} \lim_{m \rightarrow \infty} \frac{1}{m} \mathcal{L}_m(Q \circ \text{enc}(\cdot)) \leq \text{ess limsup}_{m \rightarrow \infty} \frac{|\text{enc}(\mathbf{s})|}{m} \log(2\ell + 1). \quad (46)$$

where the essential lim-sup captures the almost sure limit $1/\mathbb{E}_{\mathbf{t} \sim Q_{\text{MLE}}} [|\mathbf{t}|]$. The almost sure convergence of $|\text{enc}(\mathbf{s})|/m$ also implies that we can let the limit m go to ∞ in any manner, and the limit will remain the same. In particular, consider a process parameterized by i^* for generating the source string, such that surely $m \geq i^*$, where the total number of characters, m , is a random variable. As $i^* \rightarrow \infty$, we will also have $m \rightarrow \infty$ surely, and so the limit of $|\text{enc}(\mathbf{s})|/m$ under this modified stochastic process should also converge to the same limit.

Rather than sampling a string of a fixed length m from the source, consider the following sampling model: for $i^* \rightarrow \infty$, sample i^* geometric random variables $X_1, \dots, X_{i^*} \stackrel{\text{i.i.d.}}{\sim} \text{Geo}(\delta)$ and construct the source string as the concatenation of i^* strings alternating between successive 1's and successive

0's or 2's (with the choice between the two made uniformly at random), with the i^{th} string of length $X_i + 1$. The overall number of characters sampled, m , is surely at least i^* .

Under this stochastic process, the size of the encoding of the string is upper bounded by,

$$|\text{enc}(\mathbf{s})| \leq |X_1 + 1| + \sum_{i=2}^{i^*} \left(1 + (X_i + 1 - \ell)_+ \right)$$

This bound follows from the fact that in any substring \mathbf{s}' of successive 1's followed by a substring \mathbf{s}'' of successive 0's or 2's, the encoder tokenizes the first $\max\{\ell, |\mathbf{s}'| - 1\}$ length prefix of \mathbf{s}' as a token, and the remaining characters in \mathbf{s}' into individual tokens except the last. Then, the last character of \mathbf{s}' and the first $\max\{\ell - 1, |\mathbf{s}''|\}$ characters of \mathbf{s}'' are assigned as token. The remainder of \mathbf{s}'' is assigned as individual tokens. Each of \mathbf{s}' or \mathbf{s}'' of length x , is allocated into at most $1 + (x + 1 - \ell)_+$ tokens.

For any i , $\Pr(X_i \geq u) = (1 - \delta)^u$, and therefore, summing over $u \geq \ell$, we get that $\mathbb{E}[(X_i + 1 - \ell)_+] = \frac{(1 - \delta)^{\ell-1}}{\delta}$. With $\ell = 1 + 2 \log(1/\delta)/\delta$, this expectation is upper bounded by δ . Therefore,

$$\lim_{i^* \rightarrow \infty} \frac{\mathbb{E}[|\text{enc}(\mathbf{s})|]}{i^*} \leq \lim_{i^* \rightarrow \infty} \frac{1}{i^*} \mathbb{E} \left[|X_1| + \sum_{i=2}^{i^*} \left(1 + (X_i + 1 - \ell)_+ \right) \right] \leq 1 + \delta$$

More importantly, by the strong law of large numbers for a sum of independent random variables, $(|X_1 + 1| + \sum_{i=2}^{i^*} (1 + (X_i + 1 - \ell)_+))/i^*$, and therefore $|\text{enc}(\mathbf{s})|/i^*$ is asymptotically almost surely upper bounded as,

$$\lim_{i^* \rightarrow \infty} \frac{|\text{enc}(\mathbf{s})|}{i^*} \stackrel{\text{a.s.}}{\leq} 1 + \delta, \quad (47)$$

On the other hand, the number of characters generated, m , equals $\sum_{i=1}^{i^*} (X_i + 1)$, and satisfies, $\lim_{i^* \rightarrow \infty} \mathbb{E}[m]/i^* = 1 + \delta^{-1}$. By another application of the strong law of large numbers for a sum of independent random variables,

$$\lim_{i^* \rightarrow \infty} \frac{m}{i^*} \stackrel{\text{a.s.}}{\equiv} 1 + \delta^{-1}. \quad (48)$$

By combining eqs. (47) and (48), we have that,

$$\lim_{i^* \rightarrow \infty} \frac{|\text{enc}(\mathbf{s})|}{m} \stackrel{\text{a.s.}}{\leq} \frac{1 + \delta}{1 + \delta^{-1}} = \frac{1}{\delta}.$$

Finally, combining with eq. (46) and the ensuing discussion, we may upper bound the limiting cross-entropy loss by,

$$\min_{Q \in \mathcal{Q}_{1\text{-gram}}} \lim_{m \rightarrow \infty} \frac{1}{m} \mathcal{L}_m(Q \circ \text{enc}(\cdot)) \leq \delta \log(2\ell + 1) = \delta \log(3 + 4 \log(1/\delta)/\delta).$$

Note for this Markovian source, it is a short calculation to see that,

$$H_\infty = \mathbb{E}_{x \sim \pi}[H(P(\cdot|x))] = \delta \log(\sqrt{2}/\delta) + (1 - \delta) \log(1/(1 - \delta))$$

Note that for any $\delta \leq 1/2$, numerical evaluation gives the inequality,

$$1 \leq \frac{\delta \log(3 + 4 \log(1/\delta)/\delta)}{H_\infty} \leq 1.273$$

with the approximation factor improving as δ becomes smaller. Therefore, this tokenizer achieves a normalized cross-entropy loss which asymptotically scales as a constant multiple of the entropy rate of the source.

E.3 Greedy-encoder achieves poor cross-entropy loss

Note that the greedy encoder picks the largest prefix of the string which is a token, assigns and removes it, and iterates on the rest of the string. The greedy encoder's behavior is easy to analyze - every string of consecutive 1's in the new string is broken into chunks of length ℓ (save potentially the last chunk) and each chunk is assigned as a token in $\{1\mathbf{s} : \mathbf{s} \in S_1\} \subset \text{Dict}$. If the length of this substring of successive 1's is not $1, \ell + 1, 2\ell + 1, \dots$, or in general, $\equiv 1 \pmod{\ell}$, every character in the next sequence, composed of 0's or 2's is tokenized into individual characters.

Similar to eq. (43) to eq. (44), consider a simplification of the overall cross-entropy loss,

$$\begin{aligned} & \min_{Q \in \mathcal{Q}_{1\text{-gram}}} \lim_{m \rightarrow \infty} \frac{1}{m} \mathcal{L}_m(Q \circ \text{enc}_{\text{gre}}(\cdot)) \\ &= \min_{Q \in \mathcal{Q}_{1\text{-gram}}} \lim_{m \rightarrow \infty} -\frac{1}{m} \mathbb{E} \left[\log Q_{\#}(|\text{enc}_{\text{gre}}(\mathbf{s})|) + |\text{enc}_{\text{gre}}(\mathbf{s})| \sum_{\mathbf{t} \in \text{Dict}} \frac{n_{\mathbf{t}}}{|\text{enc}_{\text{gre}}(\mathbf{s})|} \log Q_{\text{tok}}(\mathbf{t}) \right] \\ &\geq \min_{Q \in \mathcal{Q}_{1\text{-gram}}} \lim_{m \rightarrow \infty} -\frac{1}{m} \mathbb{E} \left[|\text{enc}_{\text{gre}}(\mathbf{s})| \sum_{\substack{\mathbf{t} \in \text{Dict} \\ Q_{\text{MLE}}(\mathbf{t}) > 0}} Q_{\text{MLE}}(\mathbf{t}) \log Q_{\text{tok}}(\mathbf{t}) \right], \end{aligned}$$

where the last equation uses the fact that by Lemma A.4, for the greedy encoder, $\lim_{m \rightarrow \infty} \frac{n_{\mathbf{t}}}{|\text{enc}_{\text{gre}}(\mathbf{s})|} \stackrel{\text{a.s.}}{=} Q_{\text{MLE}}(\mathbf{t})$. The minimizer of this objective subject to $\sum_{\mathbf{t} \in \text{Dict}: Q_{\text{MLE}}(\mathbf{t}) > 0} Q_{\text{tok}}(\mathbf{t}) \leq 1$ is $Q_{\text{tok}}(\mathbf{t}) = Q_{\text{MLE}}(\mathbf{t})$ resulting in the inequality,

$$\min_{Q \in \mathcal{Q}_{1\text{-gram}}} \lim_{m \rightarrow \infty} \frac{1}{m} \mathcal{L}_m(Q \circ \text{enc}_{\text{gre}}(\cdot)) \geq \lim_{m \rightarrow \infty} \frac{1}{m} \mathbb{E} [|\text{enc}_{\text{gre}}(\mathbf{s})| H(Q_{\text{MLE}})], \quad (49)$$

where we use the convention $0 \log(1/0) \triangleq \lim_{P \rightarrow 0} P \log(1/P) = 0$ and therefore we may sum over tokens such that $Q_{\text{MLE}}(\mathbf{t}) = 0$ for free.

Considering the same geometric sampling model as in Appendix E.2, and Lemma A.4, we may study the almost sure limit $Q_{\text{MLE}}(\mathbf{t}) = \lim_{m \rightarrow \infty} n_{\mathbf{t}} / |\text{enc}_{\text{gre}}(\mathbf{s})|$ by computing $\lim_{i^* \rightarrow \infty} n_{\mathbf{t}} / |\text{enc}_{\text{gre}}(\mathbf{s})|$ under the geometric sampling model since the almost sure limit exists. Recall that in the geometric sampling model, we generate the overall source string by concatenating i^* strings of length $X_1 + 1, \dots, X_{i^*} + 1$ where $X_i \sim \text{Geo}(\delta)$, with the strings alternating between successive 1's and successive 0's or 2's (with the choice between the two made by the flip of a fair coin). For $x \in \{0, 1, 2\}$, let $\mathcal{E}_i(x)$ denote the event that X_i is a string composed only of all x 's. The length of the greedy encoding of \mathbf{s} is lower bounded by,

$$|\text{enc}_{\text{gre}}(\mathbf{s})| \geq \sum_{i=1}^{i^*} X_i \cdot \mathbb{I}(X_{i-1} \not\equiv 1 \pmod{\ell}) \mathbb{I}(\mathcal{E}_i(0) \cup \mathcal{E}_i(2)). \quad (50)$$

Which captures for the fact that all 0's and 2's are encoded into singular tokens unless the previous string of 1's was of length $\equiv 1 \pmod{\ell}$. By the law of large numbers of the RHS of eq. (50), the following a.a.s. lower bound is satisfied,

$$\lim_{i^* \rightarrow \infty} \frac{|\text{enc}_{\text{gre}}(\mathbf{s})|}{i^*} \stackrel{\text{a.s.}}{\geq} \frac{1}{2\delta} \left(1 - \sum_{u=0}^{\infty} \delta(1-\delta)^{\ell u+1} \right) = \frac{1}{2\delta} \left(1 - \frac{\delta(1-\delta)}{1-(1-\delta)^{\ell}} \right) \geq \frac{1-\delta}{2\delta}, \quad (51)$$

where the last inequality uses the fact that $\ell = 1 + 2 \log(1/\delta)/\delta$. Likewise, observe that, $|\text{enc}_{\text{gre}}(\mathbf{s})| \leq m$ surely, and following the analysis in Appendix E.2 of eq. (48), we have that,

$$\lim_{i^* \rightarrow \infty} \frac{|\text{enc}_{\text{gre}}(\mathbf{s})|}{i^*} \leq \lim_{i^* \rightarrow \infty} \frac{m}{i^*} \stackrel{\text{a.s.}}{=} 1 + \delta^{-1}. \quad (52)$$

For $x \in \{0, 2\}$, observe that the expected number of times the token x is observed in the encoding of \mathbf{s} , n_x can be written as,

$$n_x \geq \sum_{i=1}^{i^*} ((X_i + 1) \cdot \mathbb{I}(X_{i-1} \not\equiv 1 \pmod{\ell})) \mathbb{I}(\mathcal{E}_i(x)). \quad (53)$$

In particular, taking the expectation of eq. (53),

$$\mathbb{E}[n_x | \mathcal{E}_1(0) \cup \mathcal{E}_1(2)], \mathbb{E}[n_x | \mathcal{E}_1(1)] \geq \frac{i^* - 1}{4} (1 + \delta^{-1}) \left(1 - \sum_{u=0}^{\infty} \delta(1 - \delta)^{\ell u + 1} \right) \geq \frac{i^* - 1}{4} \cdot \frac{1 - \delta^2}{\delta}. \quad (54)$$

Note that in any realization of the geometric sampling process, in eq. (53), either the odd indexed substrings are all-1's or the even indexed substrings are all-1's. Therefore, surely, all the non-zero terms in the above summation are of the same parity. Moreover, since the i^{th} term in the sum only depends on X_i and X_{i-1} , conditioned on whether the non-zero parities are even or odd, n_x can be written as a sum of $\approx i^*/2$ mutually independent terms. By the strong law of large numbers on each of the conditional processes, eqs. (53) and (54) implies that for $x \in \{0, 2\}$,

$$\lim_{i^* \rightarrow \infty} \frac{n_x}{i^*} \xrightarrow{\text{a.s.}} \frac{1 - \delta^2}{4\delta}.$$

To upper bound n_x , note that it is upper bounded by the number of times the character x appears in the source string, which by the strong law of large numbers a.a.s (after normalizing by i^*), scales as $1/4\delta$. Finally, to bound $Q_{\text{MLE}}(\mathbf{t})$ which is the sequential nature of the encoder, using a similar proof as Lemma A.4, we can show that $n_{\mathbf{t}} / \sum_{\mathbf{t}'} n_{\mathbf{t}'}$ converges to the unigram MLE model for this tokenizer. For the token $x \in \{0, 2\}$,

$$\lim_{i^* \rightarrow \infty} \frac{n_x}{|\text{enc}(\mathbf{s})|} = Q_{\text{MLE}}(x) \leq \mathbb{E} \left[\lim_{i^* \rightarrow \infty} \frac{n_x}{n_2 + n_0} \right] \quad (55)$$

Using the a.a.s. upper and lower bounds on $|\text{enc}(\mathbf{s})|$, n_0 and n_2 derived in eqs. (52) and (55), we arrive at lower and upper bounds on $Q_{\text{MLE}}(x)$ for $x \in \{0, 2\}$,

$$\frac{1}{4} \approx \frac{1 - \delta}{4} = \frac{(1 - \delta^2)}{4\delta(1 + \delta^{-1})} \leq Q_{\text{MLE}}(x) \leq \frac{1}{2(1 - \delta^2)} \approx \frac{1}{2}.$$

Since there are at least two tokens having probability bounded away from 0 and 1 by a constant under the MLE unigram model, the entropy of Q_{MLE} must also be lower bounded by a constant. Indeed,

$$H(Q_{\text{MLE}}) \geq 2 \min_{\frac{1-\delta}{4} \leq y \leq \frac{1}{2(1-\delta^2)}} y \log(1/y).$$

It is easy to verify that for $\delta \leq 0.5$, the minimizer is achieved at $y = \frac{1-\delta}{4}$, which leads to the lower bound,

$$H(Q_{\text{MLE}}) \geq \left(\frac{1 - \delta}{2} \right) \log \left(\frac{4}{1 - \delta} \right)$$

Architecture	GPT-2
Batch size	Grid-searched in {8, 16, 32}
Gradient acc. steps	1
Tokenizer dictionary size	{10, 20}
Tokenizer dataset size	10,000
Optimizer	AdamW ($\beta_1 = 0.9, \beta_2 = 0.95$)
Learning rate	0.002
Scheduler	Cosine
# Iterations	8000
Weight decay	1×10^{-3}
Dropout	0
Sequence length	512
Embedding dimension	Grid-searched in {10, 20, 30, 40}
# layers	Grid-searched in {1, 2, 4, 8}
# heads	Grid-searched in {1, 2, 4, 8, 16}
Repetitions	5

Table 3: Hyperparameter choices

Finally, combining this lower bound on $H(Q_{\text{MLE}})$ with eq. (49), we have that,

$$\begin{aligned}
\min_{Q \in \mathcal{Q}_{\text{1-gram}}} \lim_{m \rightarrow \infty} \frac{1}{m} \mathcal{L}_m(Q \circ \text{enc}(\cdot)) &= \lim_{i^* \rightarrow \infty} \mathbb{E} \left[\frac{|\text{enc}_{\text{gre}}(\mathbf{s})|}{m} H(Q_{\text{MLE}}) \right] \\
&\geq \lim_{i^* \rightarrow \infty} \mathbb{E} \left[\frac{|\text{enc}_{\text{gre}}(\mathbf{s})|}{m} \right] \cdot \left(\frac{1-\delta}{2} \right) \log \left(\frac{4}{1-\delta} \right) \\
&\stackrel{(i)}{\geq} \frac{1-\delta}{2\delta(1+\delta^{-1})} \cdot \left(\frac{1-\delta}{2} \right) \log \left(\frac{4}{1-\delta} \right) \\
&\geq \frac{(1-\delta)^2}{3(1+\delta)} H(\pi)
\end{aligned}$$

where (i) follows from the lower bound on $|\text{enc}_{\text{gre}}(\mathbf{s})|$ in eq. (51) with the almost sure limit of m in eq. (48) and noting that $|\text{enc}_{\text{gre}}(\mathbf{s})|/m \leq 1$ surely. The last inequality follows by simplifying using $\pi = (1/4, 1/2, 1/4)$ and $H(\pi) = \frac{1}{2} \log(8)$.

F Experiment details

Experiment 1 (Figures 5a and 5b). In this and previous experiments (Figures 2a, 2b and 4), we train the transformers on a single GPU on an $8 \times$ A100 node. The wall-clock time measured does not count time spent in validation loss evaluations. The hyperparameter choices are listed in Table 3.

Experiment 2 (Table 2). We evaluate pre-trained tokenizers on various datasets. In this experiment, we do not evaluate the likelihood model on test sequences, rather, we estimate the

cross-entropy of the best unigram model by using the approximation,

$$-\mathbb{E} \left[\sum_{t \in \text{Dict}} n_t \log Q_{\text{MLE}}(\mathbf{t}) \right] \approx - \sum_{t \in \text{Dict}} \hat{n}_t \log(\hat{Q}(\mathbf{t})) \quad (56)$$

where $\hat{Q}(\mathbf{t}) = \frac{\hat{n}_t}{\sum_t \hat{n}_t}$ is the MLE unigram model learnt from a finite dataset, which we choose here as GLUE (Wang et al., 2019), and \hat{n}_t is the number of times the token \mathbf{t} is observed in the encoding of the dataset. This approximation allows us to separate the error stemming from learning a suboptimal likelihood model which tends to have higher sample complexity requirements and focus on the asymptotic error of the tokenizer.

We use Monte-carlo sampling to approximate the cross-entropy loss estimator in eq. (56). These approximations tends to underestimate the true cross-entropy loss due to the concavity of $x \log(1/x)$ close to 0. In general, the gap between the approximation and the true error is expected to grow with k . Therefore, the true difference between the estimate of the best unigram model on a tokenizer and the best k -gram model for $k \geq 2$ on the character level tokenizer is likely to be larger than the reported figures.

Experiment 3 (Figure 7). We train the LZW, BPE, Unigram and Wordpiece tokenizers with dictionary sizes $\{5000, 6000, 8000, 12000, 20000, 32000, 50000, 80000\}$. The cross-entropy loss incurred by the best 1-gram model is estimated using eq. (56) while for k -gram models for $k \geq 2$, we use Monte-carlo sampling to estimate the cross-entropy of the empirical k -gram model computed using the GLUE dataset. For the k -gram models trained on the character level tokenizer, since the vocabulary size is fixed, we instead plot the number of distinct k -grams on the x -axis. While this is not a true measure of the number of parameters in the underlying k -gram model, we use this as a proxy for the same.

