Tokenization and the Noiseless Channel

Vilém Zouhar[E] Clara Meister[E] Juan Luis Gastaldi[E] Li Du[J]
Mrinmaya Sachan[E] Ryan Cotterell[E]

ETH Zürich[E] Johns Hopkins University[J]
{vzouhar,cmeister,gjuan,msachan,rcotterell}@ethz.ch leodu@cs.jhu.edu

Abstract

Subword tokenization is a key part of many NLP pipelines. However, little is known about why some tokenizer and hyperparameter combinations lead to better downstream model performance than others. We propose that good tokenizers lead to *efficient* channel usage, where the channel is the means by which some input is conveyed to the model and efficiency can be quantified in information-theoretic terms as the ratio of the Shannon entropy to the maximum possible entropy of the token distribution. Yet, an optimal encoding according to Shannon entropy assigns extremely long codes to low-frequency tokens and very short codes to high-frequency tokens. Defining efficiency in terms of Rényi entropy, on the other hand, penalizes distributions with either very high or very low-frequency tokens. In machine translation, we find that across multiple tokenizers, the Rényi entropy with $\alpha = 2.5$ has a very strong correlation with BLEU: 0.78 in comparison to just -0.32 for compressed length.

1 Introduction

Tokenization, the practice of breaking up text into words or subword pieces, or more generally, *tokens*,[1] is often the first step in an NLP pipeline. A wide variety of tokenization functions have been proposed in the NLP literature (Mermer, 2010; Sennrich et al., 2016; Kudo, 2018). And, indeed, research on developing a good tokenization function continues because *how* one tokenizes may have a large impact on model performance in the downstream task. For instance, Gowda and May (2020) note BLEU ranges from 28 to 37 just by changing the size of the vocabulary in their machine translation (MT) pipeline. A direct *extrinsic* evaluation of a tokenization function, however, is computationally intensive: One first has to retokenize the corpora (generally quick), but then retrain the NLP model (often computationally intensive) to evaluate the effect. For this reason, characterizing the *intrinsic* properties of a good tokenization function has practical benefits (Gallé, 2019; Gowda and May, 2020).

Our paper takes an information-theoretic approach to characterizing a good tokenization function. Following Gallé (2019), we contend that tokenization may be fruitfully viewed as determining a good dictionary code for a language. Fortunately, dictionary codes are equipped with a natural intrinsic metric of utility (expected code length) whereas there are many ways to extrinsically measure the tokenization quality. For simplicity and in line with previous research, we choose a specific downstream task metric: BLEU (Papineni et al., 2002) in the domain of MT.[2]

We hypothesize that, *ceteris paribus*, downstream task metrics should correlate with the expected code length of the unigram token distribution. While not immediately intuitive, the motivation is that there is a theoretical connection between the expected code length (under an optimal encoder) of a token distribution and that distribution's Shannon entropy: The latter gives us a lower bound on the former. And given a fixed vocabulary size, higher entropy token distributions are more desirable because they are more balanced, i.e., there are fewer tokens that occur too rarely or too frequently. This characteristic should in turn balance a model's ability to learn representations for the entire vocabulary, which requires exposure to enough instances of each token while also penalizing the use of very frequent character sequences as tokens, which is often inefficient due to their lack of distinct meaning.

Yet when using Shannon entropy as our metric of a distribution's balance, the optimal token distribution may still include a large number of in-

📦 We release the `tokenization-scorer` package (App. B).

[1]To avoid ambiguity, we eschew the common expressions *word* and *subword* and, instead, adopt the term *token* to mean an element of the vocabulary *after* tokenization. We formally define token in §2.

[2]In Tab. 2 (Appendix) we replicate the findings with CHRF (Popović, 2015), BLEURT (Sellam et al., 2020) and COMET (Rei et al., 2020).

frequent tokens. This behavior may be undesirable for a number of reasons that we subsequently discuss. Accordingly, we formulate the **compression principle**, which states that downstream task metrics, e.g., BLEU, should correlate with the expected code length subject to a penalty for long codewords (which correspond to infrequent tokens). Consequently, we introduce a more nuanced formulation of efficiency that employs Rényi entropy (Rényi, 1961),[3] whose hyperparameter α allows us to penalize the use of long codes to varying degrees.

In the experimental portion of our paper, we predict the performance of MT models. We find that the channel efficiency with Rényi entropy with $\alpha = 2.5$ yields a Pearson correlation of 0.78 with BLEU on German $\rightarrow$ English MT (1M parallel sentences from CommonCrawl). This stands in contrast to Shannon entropy or expected sequence length, which yield Pearson correlations of only 0.22 and -0.30, respectively.

We also provide an easy-to-use package to score tokenizations. See App. B for usage instructions.

2 Tokenization

Tokenization is generally defined informally as the breaking up of text into a sequence of tokens which are then encoded into a machine-interpretable format. However, to proceed with our analysis, we require a more formal treatment. First, we assume that there exists an alphabet, a finite, non-empty set of **characters** Σ. We call a string of characters $\boldsymbol{\sigma} = \langle \sigma_1 \sigma_2 \cdots \sigma_N \rangle \in \Sigma^*$ a **text**. In this formulation, we assume that the alphabet Σ includes all characters, including punctuation and a distinguished white space character. Finally, an unordered multiset of texts $\{\boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_M\} \subset \Sigma^*$ is termed a **corpus** of size M. We denote the true distribution over all texts as p_{Σ^*}. Every p_{Σ^*} induces a marginal distribution over Σ, which we call the Σ-**unigram distribution**:

$$p_\Sigma(\sigma) \overset{\text{def}}{=} \sum_{\boldsymbol{\sigma} \in \Sigma^*} p_{\Sigma^*}(\boldsymbol{\sigma}) \frac{\text{count}(\sigma, \boldsymbol{\sigma})}{|\boldsymbol{\sigma}|} \qquad (1)$$

where $\text{count}(\sigma, \boldsymbol{\sigma})$ returns the number of times the character σ appears in text $\boldsymbol{\sigma}$. In general, we do not have access to p_{Σ^*} but rather only to samples from p_{Σ^*} with which we can represent an empirical distribution $\widehat{p_{\Sigma^*}}$. Our formal analysis, however, will consider p_{Σ^*}.

Let Δ be a second alphabet, which we call the **tokenization alphabet**. We define a **tokenization function** $t : \Sigma^* \rightarrow D \subseteq \Delta^*$ as a function mapping texts in alphabet Σ to sequences of **tokens** in $D = t(\Sigma^*)$, One popular choice in NLP is to have Σ be a set of Unicode characters and Δ be a set of *strings* of Unicode characters. In this case, the tokenization function t segments the text $\boldsymbol{\sigma}$ into tokens corresponding to smaller chunks of text. There are many approaches for devising different t's; a brief overview of some of them is offered in App. E.

Furthermore, for our purposes, it is useful to restrict tokenization functions to those that are invertible (bijections), i.e., rules where we can undo the tokenization. This way, the original text can be reconstructed and no information is lost during tokenization.

Example 2.1. *Any injective tokenization function, i.e., mapping different inputs* $\boldsymbol{\delta}', \boldsymbol{\delta}''$ *to different outputs* $\boldsymbol{\sigma}', \boldsymbol{\sigma}''$*, satisfies our requirements. As a counter example, consider a tokenization function* t_1 *for which* $t_1(\texttt{two\_cows}) = \langle \texttt{two\_}, \texttt{[UNK]} \rangle$ *and* $t_1(\texttt{two\_birds}) = \langle \texttt{two\_}, \texttt{[UNK]} \rangle$*. The* t_1*'s lack of injectivity prevents us from recovering the original text from the token sequence* $\langle \texttt{two\_}, \texttt{[UNK]} \rangle$.

Because of our restriction to invertible tokenization functions, with a change of variable we can convert the distribution over texts in Σ^* into one over token sequences $\boldsymbol{\delta}$ in D in a straightforward manner: $p_{\Delta^*}(\boldsymbol{\delta}) = p_{\Sigma^*}(t^{-1}(\boldsymbol{\delta}))$. Note that the pushforward $p_{\Delta^*}(\boldsymbol{\delta})$ induces a distribution over Δ^* but with support limited to D.

In applied NLP, there is currently no widely accepted notion of the *intrinsic* quality of a tokenization function. Rather, practitioners are generally interested in its *extrinsic* performance, i.e., the performance of a model trained on a corpus tokenized using a certain tokenization function. Under such an evaluation, given two tokenization functions, the one that enables better performance on the downstream task is taken to be better. However, gauging the quality of a tokenizer function in this manner is computationally expensive. Thus, we develop an information-theoretic intrinsic evaluation.

3 Communication in a Noiseless Channel

Our analysis of tokenization schemes relies on the following framing: Our ultimate goal when tokenizing a text $\boldsymbol{\sigma} \sim p_{\Sigma^*}$ is the transmission of this text

[3] Campbell (1965) shows that Rényi gives a lower-bound on the expected length of an optimal code subject to a length penalty, i.e., overly long or short codewords are penalized.

across a hypothetical channel. To perform this feat, we first tokenize $\boldsymbol{\sigma}$ into a sequence in $D \subseteq \Delta^*$. We then encode each token in Δ as a sequence of **symbols** from the set $\{1, \ldots, b\}$, where b is determined by the channel. Our goal is to analyze the properties of tokenization schemes that lead to models with good downstream performance.

In the case of a noisy channel, we seek an encoding scheme that will help ensure that $\boldsymbol{\sigma}$ is resilient to noise in addition to efficiently encoding $\boldsymbol{\sigma}$. However, in the noiseless case, we *only* care about efficiency. We can assume that we are working with a noiseless channel because, in the process of encoding data, no information is ever altered by a stochastic process. In this case, one can equivalently think of noiseless channel encoding as compression. Thus, our analysis proceeds by considering the efficiency of different tokenization functions as if our goal is to use them to communicate over a noiseless channel. To this end, we first discuss the conditions for building such an encoding and then discuss the concept of efficient channel usage.

Definition 3.1. *A **token-level encoder*** enc_Δ *is a function* $\mathsf{enc}_\Delta : \Delta \to \{1, \ldots, b\}^*$ *that maps every token* $\delta \in \Delta$ *to a **string** of symbols in base b, which we call a codeword. We can naturally lift the token-level encoder to a **sequence-level encoder** using concatenation as* $\mathsf{enc}_\Delta(\boldsymbol{\delta}) = \bigoplus_{n=1}^{|\boldsymbol{\delta}|} \mathsf{enc}_\Delta(\delta_n)$.[4]

In order to be able to uniquely decode a string $\boldsymbol{\delta}$, we further require that enc_Δ produces prefix-free[5] codes for all tokens in Δ. As an example, Huffman encoding provides a fast and nearly optimal (in a sense to be discussed in the subsequent section) method to construct prefix-free codes (Huffman, 1952).

Example 3.2 (One-hot encoding)**.** *Consider a tokenization alphabet* Δ*. In NLP, when* $b = 2$*, the most straightforward way of encoding the* n^{th} *element of* Δ *is a vector of zeroes with length* $|\Delta|$ *with* 1 *on position* n*.*

Example 3.3 (Transmission)**.** *We consider an arbitrary encoder* enc_Δ *over a given alphabet* Δ *and a channel with* $b = 2$*. Now given a text and some tokenization function* $t(\texttt{two\_cows}) = \langle \texttt{two\_}, \texttt{cow}, \texttt{s} \rangle$ *we apply the encoder:* $\mathsf{enc}_\Delta(\texttt{two\_}) = \texttt{1010101}$ $\mathsf{enc}_\Delta(\texttt{cow}) = \texttt{101111101}$ *and* $\mathsf{enc}_\Delta(\texttt{s}) = \texttt{01010}$ *and for the whole sequence* $\mathsf{enc}_\Delta(t(\texttt{two\_cows})) = \texttt{101010110111110101010}$.

For the remainder of the paper, we will not be interested in any specific enc_Δ, but rather in an *optimal* token-level encoder that we can achieve, as measured by expected code length measures.

Definition 3.4. *The **expected code length*** $\mathcal{L}_{\mathsf{enc}_\Delta}$ *of a token-level encoder* enc_Δ *is defined as*

$$\mathcal{L}_{\mathsf{enc}_\Delta}(p_\Delta) = \sum_{\delta \in \Delta} p_\Delta(\delta) |\mathsf{enc}_\Delta(\delta)| \tag{2}$$

A well-known result from information theory tells us that Eq. (2) is bounded by the Shannon entropy of W_Δ, a Δ-valued random variable with law p_Δ. To introduce the theorem, we first define Shannon entropy.

Definition 3.5. *The **Shannon entropy** of* W_Δ *is defined as*

$$\mathrm{H}(\mathrm{W}_\Delta) \overset{\text{def}}{=} -\sum_{\delta \in \Delta} p_\Delta(\delta) \log p_\Delta(\delta) \tag{3}$$

For channels using b symbols for transmission, the logarithm is of base b. Traditionally in information theory, ones takes $b = 2$.

Theorem 3.6 (Shannon's Source Coding Theorem)**.** *Let* W_Δ *be a* Δ*-valued random variable with law* p_Δ *and let* enc_Δ *be an encoder. Then,*

$$\mathrm{H}(\mathrm{W}_\Delta) \leq \mathcal{L}_{\mathsf{enc}_\Delta}(p_\Delta) \tag{4}$$

with the optimal token-level encoder $\mathsf{enc}^\star_\Delta$ *satisfying*

$$\mathcal{L}_{\mathsf{enc}^\star_\Delta}(p_\Delta) \leq \lceil \mathrm{H}(\mathrm{W}_\Delta) \rceil \tag{5}$$

This theorem tells us that if we wish to communicate tokens from the alphabet Δ through a noiseless channel, their minimum expected length for any possible encoding is bounded by the Shannon entropy of the distribution p_Δ. An optimal token-level encoder will produce codes with the expected length within those exact bounds. We can prove a very similar result to Shannon's source coding theorem (Shannon, 1948) that tells us how well we can optimally encode a Δ^*-valued source using only the token-level encoder.

To this end, we first introduce the notions of expected sequence length and average per-token encoding length, and then offer a lower-bound on the compression achievable using only a token-level encoder. We additionally define three random

[4]Whitespace information is not lost here because it is included in the tokens (see Example 2.1).

[5]Prefix-free means that no codeword is a prefix of any other codeword.

variables that will prove useful in our analysis; all of them are pushforwards of p_{Δ^*}.

Let L be a random variable whose values range over strings' lengths, i.e., $L(\boldsymbol{\delta}) = |\boldsymbol{\delta}|$. The expected token sequence length $\mathbb{E}[L]$ for sequences sampled according to p_{Δ^*} is then

$$\mathbb{E}[L] = \sum_{\boldsymbol{\delta} \in \Delta^*} p_{\Delta^*}(\boldsymbol{\delta})|\boldsymbol{\delta}| \tag{6}$$

where for notational simplicity, we leave the dependence of this expectation on p_{Δ^*} implicit as it will always be clear from context. Let $X_\delta(\boldsymbol{\delta}) = \frac{\text{count}(\delta, \boldsymbol{\delta})}{|\boldsymbol{\delta}|}$ be the unigram random variable, i.e., a function that returns the proportion of $\boldsymbol{\delta}$ that consists of a particular δ. Finally, define the random variable $\overline{L}_{\mathsf{enc}_\Delta}(\boldsymbol{\delta}) = \sum_{\delta \in \Delta} X_\delta(\boldsymbol{\delta})|\mathsf{enc}_\Delta(\delta)|$.

We now turn to our first major theorem.

Theorem 3.7. *Let p_{Δ^*} be a distribution over Δ^*, and let p_Δ be the unigram distribution induced by p_{Δ^*} (Eq. (1)). Then, for an optimal token-level encoder $\mathsf{enc}^\star_\Delta : \Delta \to \{1, \ldots, b\}^*$ lifted to the sequence level, the following lower and upper bounds hold:*[6]

$$\begin{aligned} \mathrm{H}(\mathrm{W}_\Delta) &\leq \frac{\mathcal{L}_{\mathsf{enc}^\star_\Delta}(p_{\Delta^*}) - \mathrm{Cov}(\overline{L}_{\mathsf{enc}^\star_\Delta}, L)}{\mathbb{E}[L]} \\ &\leq \lceil \mathrm{H}(\mathrm{W}_\Delta) \rceil \end{aligned} \tag{7}$$

Proof. The proof is given in App. C. ■

In the special case of Shannon entropy, we additionally arrive at the following stronger inequality:

$$\mathcal{L}_{\mathsf{enc}^\star}(p_{\Delta^*}) \leq \mathcal{L}_{\mathsf{enc}^\star_\Delta}(p_{\Delta^*}) \tag{8}$$

This holds because $\mathsf{enc}^\star$ is *not* constrained to token-level codes and the unconstrained minimum over all codes is naturally lower than the constrained version. As a concrete example, even if two $\delta', \delta'' \in \Delta$ *always* appear together in practice, $\mathsf{enc}^\star_\Delta$ must assign both δ' and δ'' their own unique code. Such a constraint does not apply to $\mathsf{enc}^\star$. We foreshadow that this inequality does *not* generalize to Rényi entropy, as discussed in §4.

Theorem 3.7 tells us that the expected code length of a sequence-level encoder, based on a token-level encoder, is proportional to the expected code length of the unigram distribution up to an additive covariance factor. This allows us to determine both a lower-bound for the expected code length of such an encoder and an upper-bound for the expected code length of a sequence-level encoder based on an optimal token-level encoder.

We are now in the position to return to the main objective of this paper: Assessing the quality of different tokenizers. One natural way of comparing tokenizers would be to compare properties of the distributions over tokens that they each induce. At first glance, Shannon entropy looks like the most obvious candidate for such a property. However, for distributions over Δ of different sizes, it is not directly comparable. The efficiency of a tokenization function addresses this issue.

Definition 3.8. *Let p_{Σ^*} be a distribution over Σ^*, let $t : \Sigma^* \to \Delta^*$ be a tokenization function, and let p_{Δ^*} be the distribution over Δ^* induced by t. The **efficiency** of t is defined as*

$$\textit{eff}(p_{\Sigma^*}, t) \overset{\text{def}}{=} \frac{\mathcal{L}_{\mathsf{enc}^\star_\Delta}(p_{\Delta^*})}{\mathcal{L}_{\mathsf{enc}^U_\Delta}(p_{\Delta^*})} \tag{9}$$

where $\mathsf{enc}^\star_\Delta$ is an optimal token-level encoder and enc^U_Δ a uniform encoder that assigns all tokens in Δ codes of equal length: $\lceil \log |\Delta| \rceil$.

Theorem 3.9. *The efficiency of t is upper-bounded by*

$$\frac{\lceil \mathrm{H}(\mathrm{W}_\Delta) \rceil + \frac{\mathrm{Cov}\left(\overline{L}_{\mathsf{enc}^\star_\Delta}, L\right)}{\mathbb{E}[L]}}{\log |\Delta|} \geq \textit{eff}(p_{\Sigma^*}, t) \tag{10}$$

and lower-bounded by

$$\frac{\mathrm{H}(\mathrm{W}_\Delta) + \frac{\mathrm{Cov}\left(\overline{L}_{\mathsf{enc}^\star_\Delta}, L\right)}{\mathbb{E}[L]}}{\lceil \log |\Delta| \rceil} \leq \textit{eff}(p_{\Sigma^*}, t) \tag{11}$$

where W_Δ is a Δ-valued random variable with law p_Δ, the unigram distribution induced by t.

Proof. The proof is given in App. C. ■

Note that the upper bound given in Eq. (10) tells us how efficient the *best* code could be, which is the more interesting bound for our purposes. Additionally, we note that, by introducing a normalization factor, efficiency provides a better solution than directly comparing distributions' entropies. We illustrate this in the following example.

Example 3.10. *Consider a tokenization function t_1 with tokenization alphabet Δ_1 where $|\Delta_1| = 6$. We then introduce a second tokenization function t_2 with a tokenization alphabet Δ_2 defined to be*

[6]Be careful not to confuse $\mathsf{enc}^\star_\Delta$, an optimal token-level encoder (here lifted to the sequence level), with enc_{Δ^*}, an arbitrary sequence-level encoder, which we usually denote enc when clear from context.

Δ_1 *plus an additional* 6 *tokens that occur very infrequently. The difference between these two distributions is illustrated in Fig. 1. If, for example, we relied solely on the Shannon entropy, which is higher for more uniformly spread-out distributions, we would judge the second distribution to be better (*$2.50 < 3.08$*). However, the efficiency tells the opposite story (*$0.97\% > 0.86\%$*).*

As Example 3.10 suggests, the measure provided by efficiency is in line with the idea of a more balanced distribution over Δ^*. Informally, we do not want a tokenizer that induces a distribution with very low entropy, as this is indicative of an unbalanced distribution. The efficiency eff provides us with a notion of this imbalance. To relate efficiency back to our metaphor of the noiseless channel, we note that the quantity $1 - \mathsf{eff}(p_{\Delta^*}, t)$ is known as **relative redundancy** and corresponds to the maximum data compression ratio (in percentage of how much can data size be reduced) that can be achieved.

4 Rényi Efficiency

Definition 3.8, the standard definition of efficiency, is based on Shannon entropy. Upon closer inspection, we see it linearly penalizes the use of long codes. To see why, consider a case where the distribution changes such that the entropy increases by one. Then, the upper-bound for the expected code length provided by an optimal encoder also increases by one. However, in some cases, we may wish to assign a non-linear cost to code length, e.g., there may be a non-linearly higher cost for decoding longer codes. In the context of choosing the vocabulary for a model, this corresponds to our desire to avoid inducing tokens that occur very infrequently because there may not be enough examples of them in the training data for the model to learn. To add an additional degree of freedom to accommodate such preferences Campbell (1965) generalizes the measure of expected code length to **discounted expected code length** for a hyperparameter s as follows:[7]

$$\mathcal{L}^{(s)}_{\mathsf{enc}_\Delta}(p_\Delta) \overset{\text{def}}{=} \lim_{s' \to s} \frac{\log\left(\sum_{\delta \in \Delta} p_\Delta(\delta) b^{s'|\mathsf{enc}_\Delta(\delta)|}\right)}{s'} \tag{12}$$

By L'Hôpital's rule, we can show that

$$\mathcal{L}^{(0)}_{\mathsf{enc}_\Delta}(p_\Delta) = \sum_{\delta \in \Delta} p_\Delta(\delta)|\mathsf{enc}_\Delta(\delta)| \tag{13}$$

[7]Our notation differs slightly from Campbell (1965).

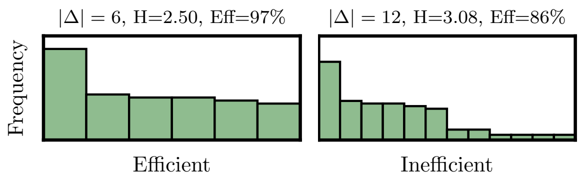

Figure 1: Examples of unigram distributions with efficient and inefficient channel usage.

and, additionally, that

$$\mathcal{L}^{(\infty)}_{\mathsf{enc}_\Delta}(p_\Delta) = \max_{\delta \in \Delta} |\mathsf{enc}_\Delta(\delta)| \tag{14}$$

Beyond the limiting cases, for $s \in (-1, \infty) \setminus \{0\}$, we further note that $\mathcal{L}^{(s)}_{\mathsf{enc}_\Delta}(p_\Delta)$ is a monotonically increasing function of s. The larger our value of s, the more disproportionately $\mathcal{L}^{(s)}_{\mathsf{enc}_\Delta}(p_\Delta)$ increases as a function of the *longest* codeword, which often corresponds to the encoding of a low-frequency character in a good code because high-frequency tokens are assigned the shorter codes in order to minimize the expected code length. For large enough s, this has the effect of encouraging all codewords to be roughly of equal length. Campbell (1965) sought an analogue of Shannon's coding theorem for $\mathcal{L}^{(s)}_{\mathsf{enc}_\Delta}(p_\Delta)$ where $s \neq 0$. As it turns out, there is a deep connection with the Rényi entropy.

Definition 4.1. *The **Rényi entropy** of order* $\alpha > 0$ *is defined as*

$$\mathrm{H}_\alpha(p_\Delta) = \lim_{\alpha' \to \alpha} \frac{1}{1-\alpha'} \log\left(\sum_{\delta \in \Delta} p_\Delta(\delta)^{\alpha'}\right) \tag{15}$$

Prima facie, Rényi entropy bears some semblance to $\mathcal{L}^{(s)}_{\mathsf{enc}_\Delta}(p_\Delta)$. To see this, consider the limiting cases. At $\alpha = 0$, we have

$$\mathrm{H}_0(p_\Delta) = \log|\Delta| \tag{16}$$

And, at $\alpha = \infty$, we have

$$\mathrm{H}_\infty(p_\Delta) = \max_{\delta \in \Delta} -\log p_\Delta(\delta) \tag{17}$$

Finally, we have $\mathrm{H}_1(p) = \mathrm{H}(p)$, i.e., $\alpha = 1$ corresponds to Shannon entropy, another result which can be shown by L'Hôpital's rule. These examples suggest the correspondence $\alpha = (1+s)^{-1}$, which fits the three cases considered, e.g., note that $\alpha = 0$ when $s \to \infty$. Moreover, this is exactly the intuition we argued for above: When $\alpha = 0$, we encode tokens with codewords of the same length which follows from minimizing the length of the longest codeword. On the other hand, when $s = -1$, we encourage shorter codes for high-probability tokens. This case corresponds to $\alpha = \infty$. We now prove that, similarly to how $\mathrm{H}(\mathrm{W}_\Delta)$ provides bounds for $\mathcal{L}_{\mathsf{enc}^\star_\Delta}(p_{\Delta^*})$ in Theorem 3.7, $\mathrm{H}_\alpha(\mathrm{W}_\Delta)$ provides bounds for $\mathcal{L}^{(s)}_{\mathsf{enc}^s_\Delta}(p_{\Delta^*})$, where enc^s_Δ is an encoder optimal with respect to a given $s = \alpha^{-1} + 1$. We term such an encoder s-optimal.

Theorem 4.2 (Generalization of Campbell (1965))**.** *Let H_α be the Rényi entropy of order α and let $\mathcal{L}^{(s)}_{\mathsf{enc}_\Delta}(p_\Delta)$ (Eq. (12)) be the discounted expected code length for the encoder enc_Δ, where $s = \alpha^{-1} - 1$. Moreover, let W_Δ be a Δ-valued random variable with law p_Δ. Then for an s-optimal token-level encoder enc^s_Δ, the following bound holds on the discounted expected code length:*

$$\mathrm{H}_\alpha(\mathrm{W}_\Delta) \leq \mathcal{L}^{(s)}_{\mathsf{enc}^s_\Delta}(p_\Delta) \leq \lceil \mathrm{H}_\alpha(\mathrm{W}_\Delta) \rceil \tag{18}$$

Proof. Proof in App. C. ■

Note that we have further generalized Campbell's (1965) result by allowing some negative values for s, namely, $s > -1$. As a result, we can induce additional non-linear weight on too *short* codes as opposed to only *long* codes.

Now we generalize the efficiency with respect to Shannon entropy to Rényi entropy. Let enc^s_Δ be an s-optimal token-level encoder over token alphabet Δ. Note that several terms from our prior notation can now be expressed in terms of enc^s_Δ, i.e., $\mathsf{enc}^\star_\Delta = \mathsf{enc}^0_\Delta$ and $\mathsf{enc}^U_\Delta = \mathsf{enc}^\infty_\Delta$.

Theorem 4.3. *Let $\alpha = (1+s)^{-1}$ and p_{Δ^*} be a distribution over Δ^*, and let p_Δ be the unigram distribution induced by p_{Δ^*} (Eq. (1)). Then, the following inequality holds*

$$\lceil \mathrm{H}_\alpha(\mathrm{W}_\Delta) \rceil \geq \frac{\mathcal{L}_{\mathsf{enc}^s_\Delta}(p_{\Delta^*}) - \mathrm{Cov}(\overline{L}_{\mathsf{enc}^s_\Delta}, L)}{\mathbb{E}[L]} \tag{19}$$

for an s-optimal sequence-level encoder enc^s_Δ based on token-level encoder $\mathsf{enc}_\Delta : \Delta \to \{1, \ldots, b\}^*$.

Proof. Proof in App. C. ■

Definition 4.4. *Let p_{Σ^*} be a distribution over Σ^*, let $t : \Sigma^* \to \Delta^*$ be a tokenization function, and let p_{Δ^*} be the distribution over Δ^* induced by t. The **Rényi efficiency** of t at α is defined as*

$$\mathit{eff}_\alpha(p_{\Sigma^*}, t) \overset{\text{def}}{=} \frac{\mathcal{L}_{\mathsf{enc}^s_\Delta}(p_{\Delta^*})}{\mathcal{L}_{\mathsf{enc}^\infty_\Delta}(p_{\Delta^*})} \tag{20a}$$

$$= \frac{\mathcal{L}_{\mathsf{enc}^s_\Delta}(p_{\Delta^*})}{\mathcal{L}_{\mathsf{enc}^U_\Delta}(p_{\Delta^*})} \tag{20b}$$

where $s = \alpha^{-1} - 1$.

The Rényi efficiency can be easily upper-bounded in a similar manner to the Shannon efficiency.

Theorem 4.5. *Let p_{Σ^*} be a distribution over Σ^*, let $t : \Sigma^* \to \Delta^*$ be a tokenization function, and let p_{Δ^*} be the distribution over Δ^* induced by t. Then, for an s-optimal token-level encoder enc^s_Δ lifted to the sequence-level, the Rényi efficiency of t at α is upper-bounded by*

$$\frac{\lceil \mathrm{H}_\alpha(\mathrm{W}_\Delta) \rceil + \frac{\mathrm{Cov}\left(\overline{L}_{\mathsf{enc}^s_\Delta}, L\right)}{\mathbb{E}[L]}}{\log |\Delta|} \geq \mathit{eff}_\alpha(p_{\Sigma^*}, t) \tag{21}$$

where W_Δ is a Δ-valued random variable with law p_Δ, the unigram distribution induced by t.

Proof. Proof in App. C. ■

To provide more intuition of why the non-linear penalization in Rényi efficiency makes for a good measure of distribution balance, we offer a worked example in Example E.1.

5 The Compression Principle

In previous sections, we discussed how different tokenizers lead to token distributions of varying properties. Now, we add the last piece necessary to link the downstream performance of a system with the choice of a tokenizer.

Hypothesis 5.1 (Compression Principle)**.** *Let p_{Σ^*} be a distribution over texts with characters from alphabet Σ and t be a tokenization function from Σ^* to Δ^* for some token alphabet Δ. Let p_Δ be the Δ-unigram distribution induced by t. Finally, let $\textsc{Performance}_M(t)$ be some measure of performance of a system M which uses tokenization t. Then, for some α dependent on M, $\mathit{eff}_\alpha(p_{\Sigma^*}, t)$ is a good predictor of $\textsc{Performance}_M(t)$.*

In words, we hypothesize that the efficiency of the tokenization function t is highly correlated with the downstream performance. We will verify this claim experimentally in §6.

Rényi Entropy α. The choice of α for H_α determines the extent to which longer codewords are penalized. On one hand, if we observe that Rényi efficiency with low α correlates the best with performance, we can conclude that longer

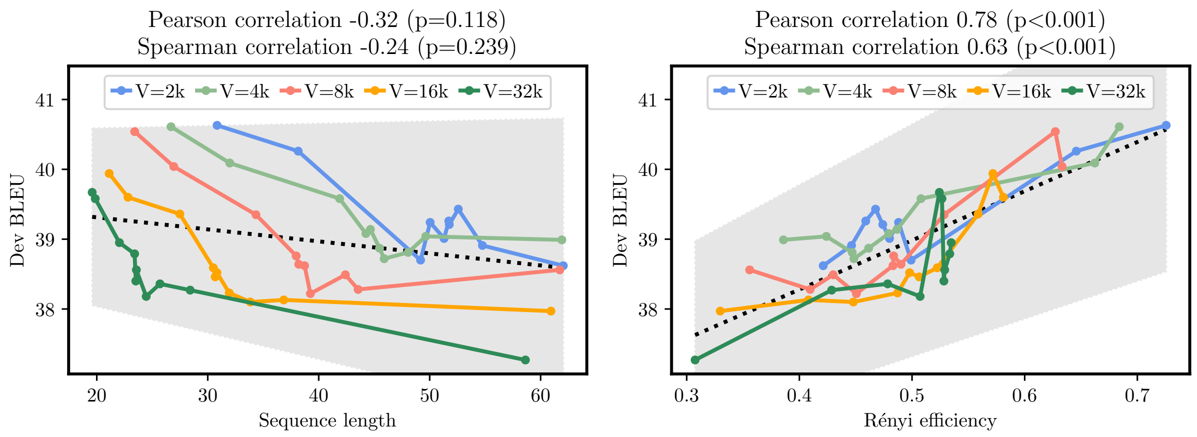

Figure 2: Efficiency of sequence length and $H_{2.5}/H_0$ as predictors of MT performance (average of 5 runs). Bands show 95% t-test confidence intervals for regression lines.

codewords and, hence, very low-frequency tokens hurt performance. On the other hand, if we observe that Rényi efficiency with high α, we can conclude that shorter codewords and, hence, very high-frequency tokens hurt performance. Note that most downstream NLP applications do not explicitly use codewords (all token representations are the same size), but long codewords are still a natural way to think about low- and high-frequency tokens.

Learnability. The most intuitive explanation for why some tokenization functions enable good downstream results and some worse is that having many low-frequent tokens will prevent the model from learning their distributional properties. This hypothesis can be related back to the sample complexity of the learning algorithm, i.e., the number of training samples needed by the model in the given setting to learn the function of interest. If we accept that part of the MT task is learning the meaning of all individual vocabulary tokens, then sample complexity could (at least partially) be expressed in terms of the number of instances of each token. This argument is made by Gowda and May (2020), who are concerned with what proportion of $\delta \in \Delta$ appears at least 100 times in the corpus for the downstream task at hand.

Nevertheless, we will see shortly that the best predictor with Rényi efficiency is for $\alpha > 1$, meaning that higher weight is given to codewords for more frequent tokens. We therefore hypothesize, that very high-frequency tokens have the most impact in downstream performance.

6 Experiments

We now seek empirical evidence for Hyp. 5.1. We focus on MT, where a standard automatic evaluation metric is BLEU (Papineni et al., 2002). We use the English→German CommonCrawl dataset in all experiments. The specifics of the MT system, data and evaluation are described in App. D. We consider two different experimental manipulations. First, we experiment with various modifications of the popular byte-pair encoding (BPE) tokenizer (Sennrich et al., 2016) to control its compression rate. The details are discussed in §6.1. Second, we experiment with a variety of tokenization schemes: Unigram (Kudo, 2018), WordPiece (Devlin et al., 2019), Lempel–Ziv–Welch (Ziv and Lempel, 1977; Welch, 1984) and Morfessor (Creutz and Lagus, 2007; Virpioja et al., 2013; Smit et al., 2014). The details are discussed in §6.2.

Note that throughout our experiments, we make the simplifying assumption of $\mathrm{Cov}\left(\overline{L}_{\mathsf{enc}^s_\Delta}, L\right) = 0$. It simplifies the upper bound of $\mathsf{eff}(p_{\Sigma^*}, t)$ (from Theorem 3.9) to $\frac{\lceil \mathrm{H}(\mathrm{W}_\Delta) \rceil}{\log|\Delta|}$ and the upper bound of $\mathsf{eff}_\alpha(p_{\Sigma^*}, t)$ (from Theorem 4.5) to $\frac{\lceil \mathrm{H}_\alpha(\mathrm{W}_\Delta) \rceil}{\log|\Delta|}$. From our preliminary results, $\mathrm{Cov}\left(\overline{L}_{\mathsf{enc}^s_\Delta}, L\right)$ is negative and small. We leave its more accurate approximation, which requires a Rényi analogue of Huffman coding as in Jelinek (1968), to future work.

6.1 Experiment 1

In our first experiment, we analyze how predictive various quantitative attributes of a tokenization scheme are of downstream model performance. We consider BPE with 5 different vocabulary sizes: 2k, 4k, 8k, 16k, and 32k. For each vocabulary size, we create multiple tokenization schemes with varying compression rates. As discussed in App. E.1, BPE produces a vocabulary through a greedy compression algorithm. However, in order to achieve a variety of different compression rates,

Predictor	**Pearson**	**Spearman**	ρ^2
Sequence len.	-0.32 (=0.118)	-0.24 (=0.239)	10%
Percentile freq.	0.76 (<0.001)	0.63 (<0.001)	58%
Entropy	0.22 (=0.281)	0.12 (=0.578)	5%
Entropy eff.	0.56 (=0.004)	0.38 (=0.006)	31%
Rényi entropy	0.49 (=0.001)	0.38 (=0.006)	24%
Rényi eff.	0.78 (<0.001)	0.66 (<0.001)	61%

Table 1: Correlations between different predictors and MT performance (BLEU). The p-values for each statistic (computed using a t-test) are in parentheses.

we inject random noise into the algorithm.[8] We achieve this by sampling from a Boltzmann distribution over the pair frequencies with temperature parameter τ; see App. E.1 for details.[9] We then treat each vocabulary size–temperature pair as a single data point in our analysis.

Our main quantitative attribute of interest, i.e., predictor, is Rényi efficiency. Aside from Rényi efficiency, we further consider Shannon and Rényi entropies, Shannon efficiency, and average tokenized sequence length. Further, one popular heuristics for choosing the vocabulary size is given and justified by Gowda and May (2020). It can be summarized as: "*Use the highest possible vocabulary size such that 95% of [tokens] occur at least 100 times in the data.*" While the constants seem arbitrary, this rule of thumb works well in practice (Gowda et al., 2022; Dramko et al., 2022; Kumar and Thawani, 2022). Nevertheless, it is stated in an algorithmic manner and not as a predictor of performance or learnability. We attempt to turn it into a regressive predictor so as to make it more comparable with the other quantities studied. Given $p_\triangle$, let $f_n(p_\triangle)$ symbolize the frequency of the n^{th} percentile. We then define the quantity $F_{\gamma_1,\gamma_2}(p_\triangle) = \sum_{\gamma_1 \le n \le \gamma_2} f_n(p_\triangle)$, which in words, is the sum of token frequencies from the γ_1^{th} to γ_2^{th} percentile. The original work suggests examining the frequency of the 95th percentile, i.e., $\gamma_1 = \gamma_2 = 0.95$. In contrast, we add an additional degree of freedom as we do not inspect a single percentile frequency but rather a sum across an interval. Later, we scan the whole space for γ_1 and γ_2 and show that there are better choices that lead to much higher correlations.

We use Pearson and Spearman correlations with

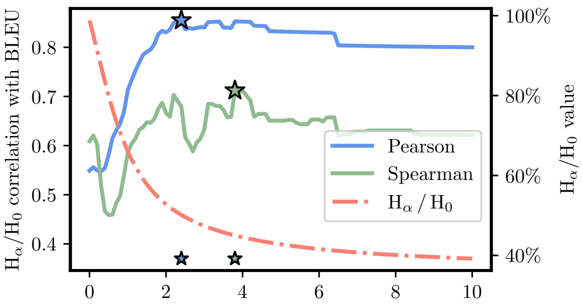

Figure 3: Correlation of Rényi efficieny (H_α/H_0) with BLEU on train data in the experiment 1. Maximums of Pearson and Spearman correlations are marked with $\star$.

downstream model performance (measured with BLEU) as our metrics of predictor quality. Recall that Pearson correlation tells us the strength of a *linear* relationship between two variables. On the other hand, Spearman correlation quantifies the strength of a linear relationship of the *ranking*.

Results. In order to select α (for eff_α) as well as γ_1 and γ_2 (for F_{γ_1,γ_2}), we use half of the data to perform a grid-search, selecting the hyperparameters that lead to the highest Pearson correlation. We show the results of this grid search for H_α/H_0 in Fig. 3 ($\alpha^* \doteq 2.5$) and for F_{γ_1,γ_2} in Fig. 4 ($\gamma_1^* \doteq 0.03, \gamma_2^* \doteq 0.83$). Unless otherwise stated, we use these values in subsequent experiments. We show the relationship between BLEU, sequence length and Rényi efficiency as approximated by the lower bound (Theorem 4.5) in Fig. 2. A comprehensive comparison for all predictors is shown in Tab. 1. The visualization of the other predictors is in Fig. 6. From these analyses, we can see that the Rényi efficiency provides a significantly better explanation for downstream model performance than any of our other predictors.

When examining which α leads to the highest absolute correlation with BLEU, we can conclude that tokenization schemes that result in fewer very high-frequency tokens are the best for downstream performance. This is evinced by both the relatively high value of α that leads to the best correlation with performance (Fig. 3, $\alpha^* \doteq 2.5$) and by Fig. 4, which shows that frequencies in the top percentile correlate *negatively* with performance. Importantly, this finding does not contradict Gowda and May's (2020) rule of thumb, which focuses on *low* frequency tokens. While very high and very low frequencies produced by a tokenization scheme are not independent, a tokenization scheme may feasibly produce both, neither or only one.

Furthermore, the Pearson correlation between

[8] The empirical effects of choosing BPE merge operations uniformly at random is studied by Sälevä and Lignos (2023).

[9] Note that the stochasticity in our case is introduced during training and not during inference, as in the popular BPE-dropout method (Provilkov et al., 2020).

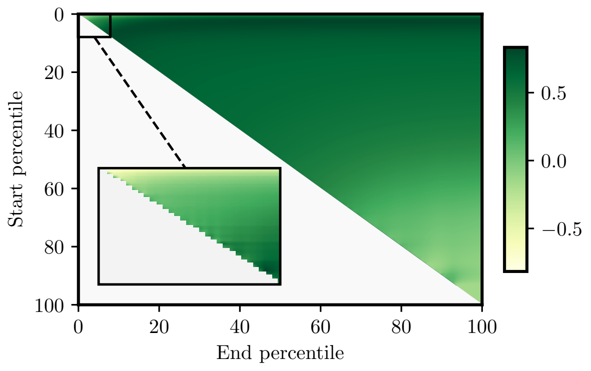

Figure 4: Results for grid search over the best hyperparameters for percentile frequency predictor to maximize the absolute Pearson correlation. The highest is for 3rd to 83th percentile with $\rho = 0.81$.

the efficiency ($H_{2.5}/H_0$) and percentile frequency ($F_{0.03,0.83}$) is 0.96, which suggests that both predictors are capturing the same underlying effect.

6.2 Experiment 2

In this experiment, we evaluate whether there exist aspects of a tokenization scheme that influence BLEU *beyond* the Rényi efficiency. Following results in Experiment 1, we focus on Rényi efficiency at $\alpha = 2.5$. In contrast to the first experiment, we consider different tokenization schemes (BPE, Unigram, WordPiece, LZW, Morfessor). We manipulate their efficiency by lowering the amount of tokenizer training data (2k, 8k, 100k parallel lines) together with varying vocabulary sizes of 4k, 8k, and 16k tokens. We then treat each tokenization-scheme–training-data-size–vocabulary-size triple as a single data point in this analysis. We compare three different linear models (Gelman and Hill, 2006), where BLEU is always the dependent variable: (i) with the tokenization scheme as a random effect, (ii) with Rényi efficiency as a fixed effect, and (iii) with both. Importantly, we treat tokenization scheme as a random effect because the set of tokenization algorithms that we consider does not encompass all possible methods, i.e., only a sample of all possible algorithms are observed.

To compare the ability of these different models to predict BLEU, we look at the average change in log-likelihood of held-out data points under a given model with respect to a baseline model: A model trained with only an intercept term. A larger value of Δ log-likelihood indicates that the data point is more probable under the comparison model, i.e., the comparison model more closely fits the observed data. We use 10-fold cross-validation to estimate these differences: Our data is split randomly into 10 folds, where 9 of the folds are used to learn model coefficients and the 10th fold is held back for evaluation. The same process is performed until we have a Δ log-likelihood value for each data point.

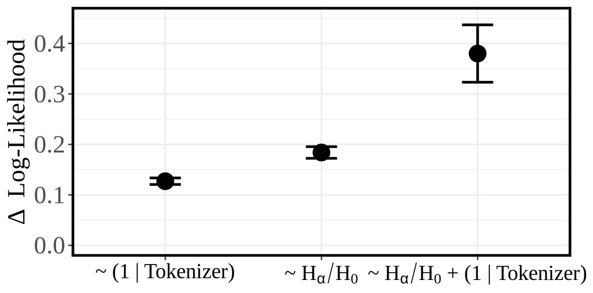

Figure 5: Mean change in log-likelihood on held-out data under linear models using different predictors. Bars indicate 95% confidence interval around the mean.

Results. In Fig. 5, we see that Rényi efficiency is a stronger predictor of MT performance than the tokenization scheme alone. Interestingly though, the predictive power of these two predictors seems to be orthogonal, as evinced by the mean Δ log-likelihood of a model with both predictors. This finding suggests that there are additional qualities of a good tokenization scheme that Rényi efficiency alone cannot capture. We leave the investigation of such qualities to future work.

7 Conclusion

Our paper presents a new information-theoretic approach to characterizing a good tokenization scheme. We contend that the Rényi efficiency of the unigram distribution that a tokenization scheme produces is a principled measure of the tokenization quality. To test this claim, we evaluate a large set of tokenization schemes, with varying vocabulary sizes and produced by different tokenization schemes. We observe how the Rényi efficiency of these different tokenizations relates to the performance of a downstream MT model. We find that, for an appropriate choice of the parameter α, this new metric has a very strong Pearson correlation with BLEU: 0.78 in comparison to just -0.32 for baseline sequence length. From a theoretical perspective, this property can be connected to a penalization of token distributions that are too unbalanced, having, in particular, very high-frequency tokens. This finding is in line with the more general principle that compression is connected with learnability. Our framework also has practical benefits as it allows for an intrinsic evaluation of tokenization functions.

Limitations

It is possible that there is a hidden effect caused by the language pair direction, model selection, or training data and its size. However, our results bear high statistical significance for cases where we desire high correlation and low statistical significance where we expect low correlation. Assured by this and concerned by the large cost of training a large number of MT systems, we did not experiment with larger data or other language directions apart from limited additional experiments in Tab. 2.

Acknowledgements

We would like to acknowledge Eleanor Chodroff and Shayne Sloggett for extensive discussion about the experimental design in Experiment 2. Clara Meister was supported by the Google PhD Fellowship. Juan Luis Gastaldi has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 839730.

References

J. Aczél and E. F. Beckenbach. 1980. *On Hölder's Inequality*, pages 145–150. Birkhäuser Basel, Basel.

Duygu Ataman and Marcello Federico. 2018. An evaluation of two vocabulary reduction methods for neural machine translation. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Track)*, pages 97–110.

L. L. Campbell. 1965. A coding theorem and Rényi's entropy. *Information and Control*, 8(4):423–429.

Rohan Chitnis and John DeNero. 2015. Variable-length word encodings for neural translation models. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2088–2093.

Marta R. Costa-jussà and José A. R. Fonollosa. 2016. Character-based neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 357–361.

Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing (TSLP)*, 4(1):1–34.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Luke Dramko, Jeremy Lacomis, Pengcheng Yin, Edward J. Schwartz, Miltiadis Allamanis, Graham Neubig, Bogdan Vasilescu, and Claire Le Goues. 2022. DIRE and its data: Neural decompiled variable renamings with respect to software class. *ACM Transactions on Software Engineering and Methodology*.

Ahmed El-Kishky, Vishrav Chaudhary, Francisco Guzmán, and Philipp Koehn. 2020. CCAligned: A massive collection of cross-lingual web-document pairs. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5960–5969.

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

Matthias Gallé. 2019. 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, Hong Kong, China.

Andrew Gelman and Jennifer Hill. 2006. *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Analytical Methods for Social Research. Cambridge University Press, Cambridge.

Thamme Gowda, Mozhdeh Gheini, and Jonathan May. 2022. Checks and strategies for enabling code-switched machine translation. *arXiv preprint arXiv:2210.05096*.

Thamme Gowda and Jonathan May. 2020. Finding the optimal vocabulary size for neural machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3955–3964.

G. H. Hardy, John E. Littlewood, and George Pólya. 1934. *Inequalities*. Cambridge University Press, Cambridge.

David A. Huffman. 1952. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10.

F. Jelinek. 1968. Buffer overflow in variable length coding of fixed rate sources. *IEEE Transactions on Information Theory*, 14(3):490–501.

Leon Gordon Kraft. 1949. *A device for quantizing, grouping, and coding amplitude-modulated pulses*. Ph.D. thesis, Massachusetts Institute of Technology.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75.

Dipesh Kumar and Avijit Thawani. 2022. BPE beyond word boundary: How NOT to use multi word expressions in neural machine translation. In *Proceedings of the Third Workshop on Insights from Negative Results in NLP*, pages 172–179.

Gurvan L'Hostis, David Grangier, and Michael Auli. 2016. Vocabulary selection strategies for neural machine translation. *arXiv preprint arXiv:1610.00072*.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. 2015. Character-based neural machine translation. *arXiv preprint arXiv:1511.04586*.

Coşkun Mermer. 2010. Unsupervised search for the optimal segmentation for statistical machine translation. In *Proceedings of the ACL 2010 Student Research Workshop*, pages 31–36.

Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. 2016. Vocabulary manipulation for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 124–129.

Sabrina J. Mielke, Zaid Alyafeai, Elizabeth Salesky, Colin Raffel, Manan Dey, Matthias Gallé, Arun Raja, Chenglei Si, Wilson Y. Lee, Benoît Sagot, and Samson Tan. 2021. Between words and characters: A brief history of open-vocabulary modeling and tokenization in NLP. *arXiv preprint arXiv:2112.10508*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. 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.

Maja Popović. 2015. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the tenth workshop on statistical machine translation*, pages 392–395.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. 2020. BPE-dropout: Simple and effective subword regularization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1882–1892.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. COMET: A neural framework for MT evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702.

Alfréd Rényi. 1961. On measures of entropy and information. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, volume 4, pages 547–562. University of California Press.

Jonne Sälevä and Constantine Lignos. 2023. What changes when you randomly choose BPE merge operations? Not much. In *The Fourth Workshop on Insights from Negative Results in NLP*, pages 59–66, Dubrovnik, Croatia. Association for Computational Linguistics.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725.

Claude E. Shannon. 1948. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423.

Peter Smit, Sami Virpioja, Stig-Arne Grönroos, Mikko Kurimo, et al. 2014. Morfessor 2.0: Toolkit for statistical morphological segmentation. In *The 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL), Gothenburg, Sweden, April 26-30, 2014*. Aalto University.

Sami Virpioja, Peter Smit, Stig-Arne Grönroos, and Mikko Kurimo. 2013. Morfessor 2.0: Python implementation and extensions for Morfessor baseline. Technical report, Aalto University.

Terry A. Welch. 1984. A technique for high-performance data compression. *Computer*, 17(06):8–19.

Jacob Ziv and Abraham Lempel. 1977. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343.

Vilém Zouhar, Clara Meister, Juan Gastaldi, Li Du, Tim Vieira, Mrinmaya Sachan, and Ryan Cotterell. 2023.

A formal perspective on byte-pair encoding. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 598–614, Toronto, Canada. Association for Computational Linguistics.

A Related Work

Prior to the widespread adoption of subword tokenization, large vocabulary sizes (e.g., 500k) were needed to allow for output expressivity and to avoid a high proportion of out-of-vocabulary tokens. Various tricks were devised to tackle the resulting computational issues (Jean et al., 2015; Mi et al., 2016; L'Hostis et al., 2016). On the other side of the spectrum, character-level NMT was also explored (Ling et al., 2015; Costa-jussà and Fonollosa, 2016), though issues arise with large sequence lengths. Mielke et al. (2021) provide an overview of the evolution of NLP tokenization and describe different types of tokenization approaches. They conclude that reasoning about tokenizer choices remains a vital part of modern pipeline preparation. In this context, our work quantifies and hence also automates some of this process by offering a framework to help guide the decision process and hyperparameter selection. Somewhat similar to our work, Ataman and Federico (2018) perform a comparison between BPE and Morfessor, though with only one specific vocabulary size (30k). Similarly to Gowda and May (2020), they suggest that homogeneity of token frequency is an important factor for MT model performance.

B Package Usage

The package is open-source[10] and can be downloaded via `pip` and used via the command-line:

```
$ pip3 install tokenization-scorer
$ tokenization-scorer -i en-de.tokenized_with_unigramlm.{en,de}
> 0.4826

$ tokenization-scorer -i en-de.tokenized_with_wordpiece.{en,de}
> 0.5047
```

or as a module in Python:

```
import tokenization_scorer
text1 = "pick @@ed pick @@l @@ed pick @@les"
tokenization_scorer.score(text1, metric="renyi", power=2.5)
> 0.8031528501359657

text2 = "pick @@e @@d pick @@l @@e @@d pick @@l @@e @@s"
tokenization_scorer.score(text2, metric="renyi", power=2.5)
> 0.9105681923824472
```

The supported metrics are the ones presented in Tab. 1: `renyi_efficiency` (default), `renyi_entropy`, `shannon_efficiency`, `shannon_entropy`, `percentile_freq`, `bits`, `sequence_len`. The power in Rényi can be modified using an extra parameter: `-e power=2.5`. The similar applies to the percentile frequency with `-e perc_start=0.03 perc_end=0.83`.

[10] github.com/zouharvi/tokenization-scorer

Predictor	**En→De**				**Cs→En**			
	BLEU	CHRF	BLEURT	COMET	BLEU	CHRF	BLEURT	COMET
Sequence length	0%	4%	23%	13%	4%	24%	51%	17%
Percentile freq.	9%	21%	47%	33%	25%	63%	86%	55%
Entropy	0%	5%	24%	14%	7%	31%	58%	23%
Entropy efficiency	22%	8%	0%	1%	6%	1%	3%	3%
Rényi entropy	7%	0%	6%	1%	0%	7%	24%	4%
Rényi efficiency	53%	33%	12%	19%	32%	35%	17%	39%

Table 2: Variance explained between predictors and MT performance (BLEU, CHRF, BLEURT and COMET) in Experiment 1 (only 3 MT seeds, 5 temperatures and 4 vocabulary sizes) with different language directions.

C Proofs

Lemma C.1. *Let p_{Δ^*} be a distribution over Δ^*, and let p_Δ be the unigram distribution induced by p_{Δ^*} (Eq. (1)). Then, the following equality holds*

$$\mathbb{E}[L] \cdot \sum_{\delta \in \Delta} p_\Delta(\delta)|\mathsf{enc}_\Delta(\delta)| + \mathrm{Cov}\left(\overline{L}_{\mathsf{enc}_\Delta}, L\right) = \mathcal{L}_{\mathsf{enc}_\Delta}(p_{\Delta^*}) \tag{22a}$$

Proof. Let $\boldsymbol{\delta} \in \Delta^*$. Define the **expected counts** as follows

$$\begin{aligned} \text{E-count}(\delta) &\overset{\text{def}}{=} \sum_{\boldsymbol{\delta} \in \Delta^*} p_{\Delta^*}(\boldsymbol{\delta})\, \text{count}(\delta, \boldsymbol{\delta}) && \text{(23a)} \\ &= \sum_{\boldsymbol{\delta} \in \Delta^*} p_{\Delta^*}(\boldsymbol{\delta}) \frac{\text{count}(\delta, \boldsymbol{\delta})}{|\boldsymbol{\delta}|} |\boldsymbol{\delta}| && \text{(23b)} \\ &= \mathop{\mathbb{E}}_{\boldsymbol{\delta} \sim p_{\Delta^*}} [X_\delta(\boldsymbol{\delta}) \cdot L] && \text{(23c)} \end{aligned}$$

We start by manipulating the expected code length

$$\begin{aligned} \sum_{\boldsymbol{\delta} \in \Delta^*} p_{\Delta^*}(\boldsymbol{\delta})|\mathsf{enc}_\Delta(\boldsymbol{\delta})| &= \sum_{\boldsymbol{\delta} \in \Delta^*} p_{\Delta^*}(\boldsymbol{\delta}) \sum_{n=1}^{|\boldsymbol{\delta}|} |\mathsf{enc}_\Delta(\delta_n)| && \text{(24a)} \\ &= \sum_{\boldsymbol{\delta} \in \Delta^*} \sum_{n=1}^{|\boldsymbol{\delta}|} p_{\Delta^*}(\boldsymbol{\delta}) |\mathsf{enc}_\Delta(\delta_n)| && \text{(24b)} \\ &= \sum_{\boldsymbol{\delta} \in \Delta^*} \sum_{\delta \in \Delta} p_{\Delta^*}(\boldsymbol{\delta}) \text{count}(\delta, \boldsymbol{\delta}) |\mathsf{enc}_\Delta(\delta)| && \text{(24c)} \\ &= \sum_{\delta \in \Delta} |\mathsf{enc}_\Delta(\delta)| \left(\sum_{\boldsymbol{\delta} \in \Delta^*} p_{\Delta^*}(\boldsymbol{\delta}) \text{count}(\delta, \boldsymbol{\delta}) \right) && \text{(24d)} \\ &= \sum_{\delta \in \Delta} \text{E-count}(\delta) |\mathsf{enc}_\Delta(\delta)| && \text{(24e)} \end{aligned}$$

Now, we proceed with algebraic manipulation.

$$\begin{aligned} \sum_{\delta \in \Delta} \text{E-count}(\delta)|\mathsf{enc}_\Delta(\delta)| &= \sum_{\delta \in \Delta} \mathop{\mathbb{E}}_{\boldsymbol{\delta} \sim p_{\Delta^*}} [X_\delta(\boldsymbol{\delta}) \cdot L] \, |\mathsf{enc}_\Delta(\delta)| && \text{(25a)} \\ &= \sum_{\delta \in \Delta} \left(\mathop{\mathbb{E}}_{\boldsymbol{\delta} \sim p_{\Delta^*}} [X_\delta(\boldsymbol{\delta})] \mathop{\mathbb{E}}_{\boldsymbol{\delta} \sim p_{\Delta^*}} [L] \, |\mathsf{enc}_\Delta(\delta)| + \mathrm{Cov}\left(X_\delta(\boldsymbol{\delta}), L\right) |\mathsf{enc}_\Delta(\delta)| \right) && \text{(25b)} \end{aligned}$$

$$= \sum_{\delta \in \Delta} \mathop{\mathbb{E}}_{\boldsymbol{\delta} \sim p_{\Delta^*}} [X_\delta(\boldsymbol{\delta})] \mathop{\mathbb{E}}_{\boldsymbol{\delta} \sim p_{\Delta^*}} [L] \, |\mathsf{enc}_\Delta(\delta)| + \sum_{\delta \in \Delta} \mathrm{Cov}\,(X_\delta(\boldsymbol{\delta}), L) \, |\mathsf{enc}_\Delta(\delta)| \tag{25c}$$

$$= \sum_{\delta \in \Delta} \mathop{\mathbb{E}}_{\boldsymbol{\delta} \sim p_{\Delta^*}} [X_\delta(\boldsymbol{\delta})] \mathop{\mathbb{E}}_{\boldsymbol{\delta} \sim p_{\Delta^*}} [L] \, |\mathsf{enc}_\Delta(\delta)| + \mathrm{Cov}\left(\sum_{\delta \in \Delta} |\mathsf{enc}_\Delta(\delta)| X_\delta(\boldsymbol{\delta}), L \right) \tag{25d}$$

$$= \sum_{\delta \in \Delta} \mathop{\mathbb{E}}_{\boldsymbol{\delta} \sim p_{\Delta^*}} [X_\delta(\boldsymbol{\delta})] \mathop{\mathbb{E}}_{\boldsymbol{\delta} \sim p_{\Delta^*}} [L] \, |\mathsf{enc}_\Delta(\delta)| + \mathrm{Cov}\Bigg(\underbrace{\sum_{\delta \in \Delta} |\mathsf{enc}_\Delta(\delta)| X_\delta(\boldsymbol{\delta}), L}_{\overset{\text{def}}{=} \overline{L}_{\mathsf{enc}_\Delta}} \Bigg) \tag{25e}$$

$$= \mathbb{E}[L] \cdot \underbrace{\sum_{\delta \in \Delta} p_\Delta(\delta) |\mathsf{enc}_\Delta(\delta)|}_{\text{expected unigram code length}} + \mathrm{Cov}\left(\overline{L}_{\mathsf{enc}_\Delta}, L\right) \tag{25f}$$

■

Theorem 3.7. *Let p_{Δ^*} be a distribution over Δ^*, and let p_Δ be the unigram distribution induced by p_{Δ^*} (Eq. (1)). Then, for an optimal token-level encoder $\mathsf{enc}^\star_\Delta : \Delta \to \{1, \ldots, b\}^*$ lifted to the sequence level, the following lower and upper bounds hold:*[11]

$$\begin{aligned} \mathrm{H}(\mathrm{W}_\Delta) &\leq \frac{\mathcal{L}_{\mathsf{enc}^\star_\Delta}(p_{\Delta^*}) - \mathrm{Cov}(\overline{L}_{\mathsf{enc}^\star_\Delta}, L)}{\mathbb{E}[L]} \\ &\leq \lceil \mathrm{H}(\mathrm{W}_\Delta) \rceil \end{aligned} \tag{7}$$

Proof. By Lemma C.1, we have

$$\mathbb{E}[L] \cdot \sum_{\delta \in \Delta} p_\Delta(\delta) |\mathsf{enc}^\star_\Delta(\delta)| + \mathrm{Cov}\left(\overline{L}_{\mathsf{enc}^\star_\Delta}, L\right) = \mathcal{L}_{\mathsf{enc}^\star_\Delta}(p_{\Delta^*}) \tag{26}$$

Now, by applying Theorem 3.6, we achieve

$$\mathbb{E}[L] \cdot \mathrm{H}(\mathrm{W}_\Delta) \leq \mathbb{E}[L] \sum_{\delta \in \Delta} p_\Delta(\delta) |\mathsf{enc}^\star_\Delta(\delta)| \tag{27a}$$

$$\mathbb{E}[L] \cdot \mathrm{H}(\mathrm{W}_\Delta) + \mathrm{Cov}\left(\overline{L}_{\mathsf{enc}^\star_\Delta}, L\right) \leq \mathbb{E}[L] \sum_{\delta \in \Delta} p_\Delta(\delta) |\mathsf{enc}^\star_\Delta(\delta)| + \mathrm{Cov}\left(\overline{L}_{\mathsf{enc}^\star_\Delta}, L\right) \tag{27b}$$

$$\mathbb{E}[L] \cdot \mathrm{H}(\mathrm{W}_\Delta) + \mathrm{Cov}\left(\overline{L}_{\mathsf{enc}^\star_\Delta}, L\right) \leq \mathcal{L}_{\mathsf{enc}^\star_\Delta}(p_{\Delta^*}) \tag{27c}$$

Similarly, from Theorem 3.6, we have

$$\mathbb{E}[L] \cdot \lceil \mathrm{H}(\mathrm{W}_\Delta) \rceil \geq \mathbb{E}[L] \sum_{\delta \in \Delta} p_\Delta(\delta) |\mathsf{enc}^\star_\Delta(\delta)| \tag{28a}$$

$$\mathbb{E}[L] \cdot \lceil \mathrm{H}(\mathrm{W}_\Delta) \rceil + \mathrm{Cov}\left(\overline{L}_{\mathsf{enc}^\star_\Delta}, L\right) \geq \mathbb{E}[L] \sum_{\delta \in \Delta} p_\Delta(\delta) |\mathsf{enc}^\star_\Delta(\delta)| + \mathrm{Cov}\left(\overline{L}_{\mathsf{enc}^\star_\Delta}, L\right) \tag{28b}$$

$$\mathbb{E}[L] \cdot \lceil \mathrm{H}(\mathrm{W}_\Delta) \rceil + \mathrm{Cov}\left(\overline{L}_{\mathsf{enc}^\star_\Delta}, L\right) \geq \mathcal{L}_{\mathsf{enc}^\star_\Delta}(p_{\Delta^*}) \tag{28c}$$

Putting these together with some additional algebraic manipulation we get

$$\mathrm{H}(\mathrm{W}_\Delta) \leq \frac{\mathcal{L}_{\mathsf{enc}^\star_\Delta}(p_{\Delta^*}) - \mathrm{Cov}(\overline{L}_{\mathsf{enc}^\star_\Delta}, L)}{\mathbb{E}[L]} \leq \lceil \mathrm{H}(\mathrm{W}_\Delta) \rceil \tag{29}$$

This concludes the proof. ■

[11]Be careful not to confuse $\mathsf{enc}^\star_\Delta$, an optimal token-level encoder (here lifted to the sequence level), with enc_{Δ^*}, an arbitrary sequence-level encoder, which we usually denote enc when clear from context.

Theorem 3.9. *The efficiency of t is upper-bounded by*

$$\frac{\lceil \mathrm{H}(\mathrm{W}_\Delta)\rceil + \frac{\mathrm{Cov}\left(\overline{L}_{\mathsf{enc}^\star_\Delta}, L\right)}{\mathbb{E}[L]}}{\log|\Delta|} \geq \textit{eff}(p_{\Sigma^*}, t) \tag{10}$$

and lower-bounded by

$$\frac{\mathrm{H}(\mathrm{W}_\Delta) + \frac{\mathrm{Cov}\left(\overline{L}_{\mathsf{enc}^\star_\Delta}, L\right)}{\mathbb{E}[L]}}{\lceil\log|\Delta|\rceil} \leq \textit{eff}(p_{\Sigma^*}, t) \tag{11}$$

where W_Δ *is a* Δ*-valued random variable with law* p_Δ*, the unigram distribution induced by* t*.*

Proof. Recall that $\mathsf{eff}(p_{\Sigma^*}, t) = \frac{\mathcal{L}_{\mathsf{enc}^\star_\Delta}(p_{\Delta^*})}{\mathcal{L}_{\mathsf{enc}^U_\Delta}(p_{\Delta^*})}$. To bound the denominator we use the fact, that for uniform distribution, $\mathrm{Cov}\left(\overline{L}_{\mathsf{enc}^U_\Delta}, L\right) = 0$ and obtain $\mathbb{E}[L]\log|\Delta| \leq \mathcal{L}_{\mathsf{enc}^U_\Delta}(p_{\Delta^*}) \leq \mathbb{E}[L]\lceil\log|\Delta|\rceil$. Using this fact, we can obtain an upper bound

$$\begin{aligned}
\mathsf{eff}(p_{\Sigma^*}, t) &= \frac{\mathcal{L}_{\mathsf{enc}^\star_\Delta}(p_{\Delta^*})}{\mathcal{L}_{\mathsf{enc}^U_\Delta}(p_{\Delta^*})} && \text{(30a)}\\
&\leq \frac{\mathbb{E}[L]\cdot\lceil\mathrm{H}(\mathrm{W}_\Delta)\rceil + \mathrm{Cov}\left(\overline{L}_{\mathsf{enc}^\star_\Delta}, L\right)}{\mathcal{L}_{\mathsf{enc}^U_\Delta}(p_{\Delta^*})} && \text{(numerator lower-bound from Theorem 3.7)} \quad \text{(30b)}\\
&\leq \frac{\mathbb{E}[L]\cdot\lceil\mathrm{H}(\mathrm{W}_\Delta)\rceil + \mathrm{Cov}\left(\overline{L}_{\mathsf{enc}^\star_\Delta}, L\right)}{\mathbb{E}[L]\log|\Delta|} && \text{(denominator upper-bound)} \quad \text{(30c)}\\
&= \frac{\lceil\mathrm{H}(\mathrm{W}_\Delta)\rceil + \frac{\mathrm{Cov}\left(\overline{L}_{\mathsf{enc}^\star_\Delta}, L\right)}{\mathbb{E}[L]}}{\log|\Delta|} && \text{(30d)}
\end{aligned}$$

and a corresponding lower bound

$$\begin{aligned}
\mathsf{eff}(p_{\Sigma^*}, t) &= \frac{\mathcal{L}_{\mathsf{enc}^\star_\Delta}(p_{\Delta^*})}{\mathcal{L}_{\mathsf{enc}^U_\Delta}(p_{\Delta^*})} && \text{(31a)}\\
&\geq \frac{\mathbb{E}[L]\cdot\mathrm{H}(\mathrm{W}_\Delta) + \mathrm{Cov}\left(\overline{L}_{\mathsf{enc}^\star_\Delta}, L\right)}{\mathcal{L}_{\mathsf{enc}^U_\Delta}(p_{\Delta^*})} && \text{(numerator lower-bound from Theorem 3.7)} \quad \text{(31b)}\\
&\geq \frac{\mathbb{E}[L]\cdot\mathrm{H}(\mathrm{W}_\Delta) + \mathrm{Cov}\left(\overline{L}_{\mathsf{enc}^\star_\Delta}, L\right)}{\mathbb{E}[L]\lceil\log|\Delta|\rceil} && \text{(denominator upper-bound)} \quad \text{(31c)}\\
&= \frac{\mathrm{H}(\mathrm{W}_\Delta) + \frac{\mathrm{Cov}\left(\overline{L}_{\mathsf{enc}^\star_\Delta}, L\right)}{\mathbb{E}[L]}}{\lceil\log|\Delta|\rceil} && \text{(31d)}
\end{aligned}$$

■

Theorem C.2 (Generalized Hölder's inequality; §2 in Aczél and Beckenbach (1980))**.** *Let* f*,* g *and* h *be vectors of positive values and coefficients* p*,* q *and* r *such that all but one are negative and* $\frac{1}{p} + \frac{1}{q} + \frac{1}{r} = 0$*. Further, if* $\forall i : f_i g_i h_i = 1$ *then*

$$\|f\|_p \, \|g\|_q \, \|h\|_r \leq 1. \tag{32}$$

As noted in Aczél and Beckenbach (1980), Theorem C.2 is in fact a simple special case of Theorem 12 in Hardy et al. (1934). We will use Theorem C.2 specifically with $r = -1$ and $h_i = f_i g_i$. This simplifies the requirements for exactly one of p and q to be negative and $\frac{1}{p} + \frac{1}{q} = 1$. Eq. (32) can then be restated as the following.

Corollary C.3 (Reverse Hölder's inequality)**.** *If f, g are positive vectors and p, q are such that $\frac{1}{p} + \frac{1}{q} = 1$ and exactly one is negative and the other is positive, then*

$$\|f\|_p \, \|g\|_q \leq \|fg\|_1. \tag{33}$$

Theorem 4.2 (Generalization of Campbell (1965))**.** *Let H_α be the Rényi entropy of order α and let $\mathcal{L}^{(s)}_{\mathsf{enc}_\Delta}(p_\Delta)$ (Eq. (12)) be the discounted expected code length for the encoder enc_Δ, where $s = \alpha^{-1} - 1$. Moreover, let W_Δ be a Δ-valued random variable with law p_Δ. Then for an s-optimal token-level encoder enc^s_Δ, the following bound holds on the discounted expected code length:*

$$\mathrm{H}_\alpha(\mathrm{W}_\Delta) \leq \mathcal{L}^{(s)}_{\mathsf{enc}^s_\Delta}(p_\Delta) \leq \lceil \mathrm{H}_\alpha(\mathrm{W}_\Delta) \rceil \tag{18}$$

Proof. Let $p = -s$ and $q = 1 - \alpha$. Let us consider three cases of s:

- $s = 0$, then $\alpha = 1$, then Theorem 3.6 applies;
- $s \in (-1, 0)$, then $\alpha \in (1, \infty)$, $p \in (0, 1)$ and $q \in (-\infty, 0)$.
- $s \in (0, \infty)$, then $\alpha \in (0, 1)$, $p \in (-\infty, 0)$ and $q \in (0, 1)$.

In the latter two cases, we have one of p and q is positive and one is negative. Further, our definitions of p and q imply that $p^{-1} + q^{-1} = 1$, which allows us to use the reverse Hölder's inequality (Corollary C.3). First, we note that the reverse Hölder's inequality implies that, for finite sequences (x_i) and (y_i),

$$\left(\sum x_i^p\right)^{\frac{1}{p}} \left(\sum y_i^q\right)^{\frac{1}{q}} \leq \sum x_i y_i \tag{34}$$

$$\left(\sum x_i^{-s}\right)^{-\frac{1}{s}} \left(\sum y_i^{1-\alpha}\right)^{\frac{1}{1-\alpha}} \leq \sum x_i y_i \qquad \text{(p and q substitution)} \tag{35}$$

Let $\ell(\delta) = |\mathsf{enc}_\Delta(\delta)|$ be the lengths of the codes given by our encoding. Further, let $p_\Delta(\delta)$ be the unigram probabilities and b be the base of our encoding. Now set $x_i = p_\Delta(\delta)^{-\frac{1}{s}} b^{-\ell(\delta)}$ and $y_i = p_\Delta(\delta)^{\frac{1}{s}}$. This step is valid because both quantities are positive. Then we proceed with algebraic manipulations

$$\left(\sum_{\delta\in\Delta} \left(p_\Delta(\delta)^{-\frac{1}{s}} b^{-\ell(\delta)}\right)^{-s}\right)^{-\frac{1}{s}} \left(\sum_{\delta\in\Delta} (p_\Delta(\delta)^{\frac{1}{s}})^{1-\alpha}\right)^{\frac{1}{1-\alpha}} \leq \sum_{\delta\in\Delta} (p_\Delta(\delta)^{-\frac{1}{s}} b^{-\ell(\delta)})(p_\Delta(\delta)^{\frac{1}{s}}) \tag{36a}$$

$$\left(\sum_{\delta\in\Delta} (p_\Delta(\delta)^{-\frac{1}{s}} b^{-\ell(\delta)})^{-s}\right)^{-\frac{1}{s}} \left(\sum_{\delta\in\Delta} (p_\Delta(\delta)^{\frac{1}{s}})^{1-\alpha}\right)^{\frac{1}{1-\alpha}} \leq \sum_{\delta\in\Delta} b^{-\ell(\delta)} \qquad \text{(algebra)} \tag{36b}$$

$$\underbrace{\left(\sum_{\delta\in\Delta} p_\Delta(\delta) b^{s\cdot\ell(\delta)}\right)^{-\frac{1}{s}}}_{\overset{\text{def}}{=} A} \underbrace{\left(\sum_{\delta\in\Delta} p_\Delta(\delta)^\alpha\right)^{\frac{1}{1-\alpha}}}_{\overset{\text{def}}{=} B} \leq \underbrace{\sum_{\delta\in\Delta} b^{-\ell(\delta)}}_{\overset{\text{def}}{=} C} \qquad \text{(algebra)} \tag{36c}$$

Then, let $A \overset{\text{def}}{=} \left(\sum_{\delta\in\Delta} p_\Delta(\delta) b^{s\cdot\ell(\delta)}\right)^{-\frac{1}{s}}$, $B \overset{\text{def}}{=} \left(\sum_{\delta\in\Delta} p_\Delta(\delta)^\alpha\right)^{\frac{1}{1-\alpha}}$ and $C \overset{\text{def}}{=} \sum_{\delta\in\Delta} b^{-\ell(\delta)}$ in the subsequent proof descriptions. Note that by the Kraft–McMillan inequality (Kraft, 1949), because our code is prefix-free (and $b > 0$), it must be that $0 < C \leq 1$.

$$\frac{\left(\sum_{\delta\in\Delta} p_\Delta(\delta)^\alpha\right)^{\frac{1}{1-\alpha}}}{\sum_{\delta\in\Delta} b^{-\ell(\delta)}} \leq \left(\sum_{\delta\in\Delta} p_\Delta(\delta) b^{s\cdot\ell(\delta)}\right)^{\frac{1}{s}} \qquad \text{(divide by AC)} \tag{37a}$$

$$\left(\sum_{\delta\in\Delta} p_\Delta(\delta)^\alpha\right)^{\frac{1}{1-\alpha}} \leq \left(\sum_{\delta\in\Delta} p_\Delta(\delta) b^{s\cdot\ell(\delta)}\right)^{\frac{1}{s}} \qquad (0 < C \leq 1) \tag{37b}$$

$$\frac{1}{1-\alpha}\log\left(\sum_{\delta\in\Delta} p_\Delta(\delta)^\alpha\right) \le \frac{1}{s}\log\left(\sum_{\delta\in\Delta} p_\Delta(\delta) b^{s\cdot\ell(\delta)}\right) \qquad \text{(take log)} \tag{37c}$$

$$\mathrm{H}_\alpha(\mathrm{W}_\Delta) \le \mathcal{L}^{(s)}_{\text{enc}}(p_\Delta) \tag{37d}$$

Our constraint on C is met and Eq. (37d) holds with equality when the lengths of our code satisfy the following relationship

$$b^{-\ell(\delta)} = \frac{p_\Delta(\delta)^\alpha}{\sum_{\delta'\in\Delta} p_\Delta(\delta')^\alpha} \tag{38a}$$

$$\ell(\delta) = -\alpha\log_b p_\Delta(\delta) + \log_b\left(\sum_{\delta'\in\Delta} p_\Delta(\delta')^\alpha\right) \tag{38b}$$

Now we consider a sequence of M tokens $\boldsymbol{\delta} = \langle\delta_1,\dots,\delta_M\rangle$, where each δ_m is sampled according to p_Δ; note that this is *not* necessarily equivalent to a sequence $\boldsymbol{\delta} \sim p_{\Delta^*}$, as we do not assume independence between tokens in that setting. Let us first lift p_Δ to take sequences, i.e., $p_\Delta(\boldsymbol{\delta}) = \prod_{m=1}^{M} p_\Delta(\delta)$, which follows naturally due to the independence of each δ_m in this setting. Now let

$$Q = \sum_{\boldsymbol{\delta}\in\Delta^M} p_\Delta(\boldsymbol{\delta})^\alpha = \left(\sum_{\delta\in\Delta} p_\Delta(\delta)^\alpha\right)^M \tag{39}$$

where the later equality follows from the fact that there are $|\Delta|^M$ sequences in Δ^M and each δ appears an equal number of times. We will now use Eq. (38b) to reason about the length of an optimal code for $\boldsymbol{\delta}$, where we similarly denote length as $\ell(\boldsymbol{\delta})$. We will first assume that $s \in (-1, 0)$ but later show that a slight modification to the proof makes it viable also for $s > 0$. Following the result in Eq. (38b), the length $\ell(\boldsymbol{\delta})$ of an optimal integer-length code for $\boldsymbol{\delta}$ should satisfy

$$-\alpha\log_b p_\Delta(\boldsymbol{\delta}) + \log_b Q \le \ell(\boldsymbol{\delta}) < -\alpha\log_b p_\Delta(\boldsymbol{\delta}) + \log_b Q + 1 \tag{40a}$$

$$-s\alpha\log_b p_\Delta(\boldsymbol{\delta}) + s\log_b Q \ge s\cdot\ell(\boldsymbol{\delta}) > s - s\alpha\log_b p_\Delta(\boldsymbol{\delta}) + s\log_b Q \qquad \text{(multiply by } s\in(-1,0)) \tag{40b}$$

$$p_\Delta(\boldsymbol{\delta})^{-s\alpha}Q^s \ge b^{s\cdot\ell(\boldsymbol{\delta})} > b^s p_\Delta(\boldsymbol{\delta})^{-s\alpha}Q^s \qquad \text{(raise to power } b) \tag{40c}$$

$$p_\Delta(\boldsymbol{\delta})^{-s\alpha+1}Q^s \ge p_\Delta(\boldsymbol{\delta})b^{s\cdot\ell(\boldsymbol{\delta})} > b^s p_\Delta(\boldsymbol{\delta})^{-s\alpha+1}Q^s \qquad \text{(multiply by } p_\Delta(\boldsymbol{\delta})) \tag{40d}$$

$$p_\Delta(\boldsymbol{\delta})^{\alpha}Q^s \ge p_\Delta(\boldsymbol{\delta})b^{s\cdot\ell(\boldsymbol{\delta})} > b^s p_\Delta(\boldsymbol{\delta})^{\alpha}Q^s \qquad (s\alpha = 1-\alpha) \tag{40e}$$

$$\sum_{\boldsymbol{\delta}\in\Delta^M} p_\Delta(\boldsymbol{\delta})^{\alpha}Q^s \ge \sum_{\boldsymbol{\delta}\in\Delta^M} p_\Delta(\boldsymbol{\delta})b^{s\cdot\ell(\boldsymbol{\delta})} > Q^s b^s \sum_{\boldsymbol{\delta}\in\Delta^M} p_\Delta(\boldsymbol{\delta})^{\alpha} \qquad \text{(sum across all sequences of length } M) \tag{40f}$$

$$Q^{s+1} \ge \sum_{\boldsymbol{\delta}\in\Delta^M} p_\Delta(\boldsymbol{\delta})b^{s\cdot\ell(\boldsymbol{\delta})} > Q^{s+1}b^s \qquad \text{(sub. } Q) \tag{40g}$$

$$Q^{s+1} \ge \left(\sum_{\delta\in\Delta} p_\Delta(\delta)b^{s\cdot\ell(\delta)}\right)^M > Q^{s+1}b^s \qquad \text{(same logic as Eq. (39))} \tag{40h}$$

$$(s+1)\log_b Q \ge M\log_b\left(\sum_{\delta\in\Delta} p_\Delta(\delta)b^{s\cdot\ell(\delta)}\right) > (s+1)\log_b Q + s \qquad \text{(take } \log_b) \tag{40i}$$

$$\frac{s+1}{s}\log_b Q \leq \frac{M}{s}\log_b\left(\sum_{\delta\in\Delta} p_\Delta(\delta) b^{s\cdot\ell(\delta)}\right) < \frac{s+1}{s}\log_b Q + 1 \qquad \text{(divide by } s \in (-1,0)\text{)} \tag{40j}$$

$$\frac{1}{1-\alpha}\log_b Q \leq \frac{M}{s}\log_b\left(\sum_{\delta\in\Delta} p_\Delta(\delta) b^{s\cdot\ell(\delta)}\right) < \frac{1}{1-\alpha}\log_b Q + 1 \qquad \text{(substitute } \alpha\text{)} \tag{40k}$$

$$\frac{1}{1-\alpha}\log_b\left(\sum_{\delta\in\Delta} p_\Delta(\delta)^\alpha\right)^M \leq \frac{M}{s}\log_b\left(\sum_{\delta\in\Delta} p_\Delta(\delta) b^{s\cdot\ell(\delta)}\right) < \frac{1}{1-\alpha}\log_b\left(\sum_{\delta\in\Delta} p_\Delta(\delta)^\alpha\right)^M + 1 \qquad \text{(definition of } Q\text{)} \tag{40l}$$

$$M \cdot \mathrm{H}_\alpha(\mathrm{W}_\Delta) \leq M \cdot \mathcal{L}^{(s)}_{\mathsf{enc}^s_\Delta}(p_\Delta) < M \cdot \mathrm{H}_\alpha(\mathrm{W}_\Delta) + 1 \tag{40m}$$

$$\mathrm{H}_\alpha(\mathrm{W}_\Delta) \leq \mathcal{L}^{(s)}_{\mathsf{enc}^s_\Delta}(p_\Delta) < \mathrm{H}_\alpha(\mathrm{W}_\Delta) + \frac{1}{M} \qquad \text{(divide by } M\text{)} \tag{40n}$$

Note that in Eq. (40b), we multiplied by a negative value and therefore swapped the directions of the inequalities. Then, in Eq. (40j) we divided by a negative value and swapped the inequalities back to their original directions. If $s > 0$, we would not change the directions of the inequalities and the proof would proceed as before.

For large M, we can get arbitrarily close to H_α. However, if we wish for $\ell(\delta)$ to be an integer, the upper bound becomes $\lceil \mathrm{H}_\alpha(\mathrm{W}_\Delta)\rceil$. ■

Theorem 4.3. *Let $\alpha = (1+s)^{-1}$ and p_{Δ^*} be a distribution over Δ^*, and let p_Δ be the unigram distribution induced by p_{Δ^*} (Eq. (1)). Then, the following inequality holds*

$$\lceil \mathrm{H}_\alpha(\mathrm{W}_\Delta)\rceil \geq \frac{\mathcal{L}_{\mathsf{enc}^s_\Delta}(p_{\Delta^*}) - \mathrm{Cov}(\overline{L}_{\mathsf{enc}^s_\Delta}, L)}{\mathbb{E}[L]} \tag{19}$$

for an s-optimal sequence-level encoder enc^s_Δ based on token-level encoder $\mathsf{enc}_\Delta : \Delta \to \{1,\ldots,b\}^*$.

Proof. Let enc^s_Δ be an s-optimal code, i.e., a code that minimizes $\mathcal{L}^{(s)}_{\mathsf{enc}_\Delta}(p_{\Delta^*})$. By Lemma C.1, we have

$$\mathbb{E}[L]\cdot \sum_{\delta\in\Delta} p_\Delta(\delta)|\mathsf{enc}^s_\Delta(\delta)| + \mathrm{Cov}\left(\overline{L}_{\mathsf{enc}^s_\Delta}, L\right) = \mathcal{L}_{\mathsf{enc}^s_\Delta}(p_{\Delta^*}) \tag{41}$$

To prove the bound, we proceed as follows. Assume $s \neq 0$ as that case is covered by Theorem 3.7. We first start with a simple application of Jensen's inequality:

$$\begin{aligned}
\lceil \mathrm{H}_\alpha(\mathrm{W}_\Delta)\rceil &\geq \mathcal{L}^{(s)}_{\mathsf{enc}^s_\Delta}(p_\Delta) && \text{(Theorem 4.2)} && \text{(42a)}\\
&= \frac{\log_b\left(\sum_{\delta\in\Delta} p_\Delta(\delta) b^{s|\mathsf{enc}^s_\Delta(\delta)|}\right)}{s} && \text{(definition)} && \text{(42b)}\\
&\geq \frac{\sum_{\delta\in\Delta} p_\Delta(\delta) s|\mathsf{enc}^s_\Delta(\delta)|\log_b b}{s} && \text{(Jensen's inequality)} && \text{(42c)}\\
&= \sum_{\delta\in\Delta} p_\Delta(\delta)|\mathsf{enc}^s_\Delta(\delta)| && && \text{(42d)}
\end{aligned}$$

Algebraically, combining the above results

$$\lceil \mathrm{H}_\alpha(\mathrm{W}_\Delta)\rceil \geq \frac{\mathcal{L}_{\mathsf{enc}^s_\Delta}(p_{\Delta^*}) - \mathrm{Cov}(\overline{L}_{\mathsf{enc}^s_\Delta}, L)}{\mathbb{E}[L]} \tag{43}$$

which proves the theorem. ■

Theorem 4.5. *Let p_{Σ^*} be a distribution over Σ^*, let $t : \Sigma^* \to \Delta^*$ be a tokenization function, and let p_{Δ^*} be the distribution over Δ^* induced by t. Then, for an s-optimal token-level encoder enc^s_Δ lifted to the sequence-level, the Rényi efficiency of t at α is upper-bounded by*

$$\frac{\lceil \mathrm{H}_\alpha(\mathrm{W}_\Delta) \rceil + \frac{\mathrm{Cov}\left(\overline{L}_{\mathsf{enc}^s_\Delta}, L\right)}{\mathbb{E}[L]}}{\log|\Delta|} \geq \mathit{eff}_\alpha(p_{\Sigma^*}, t) \tag{21}$$

where W_Δ is a Δ-valued random variable with law p_Δ, the unigram distribution induced by t.

Proof. Recall from Definition 4.4 that $\mathsf{eff}_\alpha(p_{\Sigma^*}, t) = \frac{\mathcal{L}_{\mathsf{enc}^s_\Delta}(p_{\Delta^*})}{\mathcal{L}_{\mathsf{enc}^U_\Delta}(p_{\Delta^*})}$, with $s = \frac{1}{a} - 1$. Again, to bound the denominator, we use the fact, that for uniform distribution, $\mathrm{Cov}\left(\overline{L}_{\mathsf{enc}^U_\Delta}, L\right) = 0$ and obtain $\mathbb{E}[L] \log|\Delta| \leq \mathcal{L}_{\mathsf{enc}^U_\Delta}(p_{\Delta^*})$. Then, we proceed as follows

$$\mathsf{eff}_\alpha(p_{\Sigma^*}, t) = \frac{\mathcal{L}_{\mathsf{enc}^s_\Delta}(p_{\Delta^*})}{\mathcal{L}_{\mathsf{enc}^U_\Delta}(p_{\Delta^*})} \tag{44a}$$

$$\leq \frac{\mathbb{E}[L] \cdot \lceil \mathrm{H}_\alpha(\mathrm{W}_\Delta) \rceil + \mathrm{Cov}\left(\overline{L}_{\mathsf{enc}^s_\Delta}, L\right)}{\mathcal{L}_{\mathsf{enc}^U_\Delta}(p_{\Delta^*})} \qquad \text{(upper-bound numerator)} \tag{44b}$$

$$\leq \frac{\mathbb{E}[L] \cdot \lceil \mathrm{H}_\alpha(\mathrm{W}_\Delta) \rceil + \mathrm{Cov}\left(\overline{L}_{\mathsf{enc}^s_\Delta}, L\right)}{\mathbb{E}[L] \cdot \log|\Delta|} \qquad \text{(lower-bound denominator)} \tag{44c}$$

$$= \frac{\lceil \mathrm{H}_\alpha(\mathrm{W}_\Delta) \rceil + \frac{\mathrm{Cov}\left(\overline{L}_{\mathsf{enc}^s_\Delta}, L\right)}{\mathbb{E}[L]}}{\log|\Delta|} \tag{44d}$$

This proves the result. ■

D Model, computation and reproducibility details

For the MT model, we use the `transformer_iwslt_de_en` model architecture in Fairseq (Ott et al., 2019). For data, we use 1M training and 50k dev parallel sentences from English-German CommonCrawl (El-Kishky et al., 2020). BLEU evaluation performed using SacreBLEU (Post, 2018). Details for the model training and the code to reproduce the experiments in this paper will be made publicly available. For the first experiment we trained $5 \times 6 \times 9 = 270$ MT models. For the second experiment we trained $5 \times 5 \times 3 \times 3 = 225$ MT models. We used varying GPU models (GTX 1080, RTX 2080, RTX 3090) based on availability in shared compute cluster. Although different hardware and different tokenizations had an impact on the training time, the average per one configuration was 1.5 days. Overall, we estimate 800 GPU days.

E Tokenization Schemes

In this section, we describe the four tokenization schemes used together in the second experiment. Special attention is paid to BPE, which plays a role in the first experiment. Tab. 3 shows how different tokenizers with varying vocabulary sizes tokenize the same word. For simplicity, we do not include variable-length encoding, even though it has been previously applied to MT (Chitnis and DeNero, 2015).

E.1 Byte-Pair Encoding

BPE was first discovered by Gage (1994) as a faster compression algorithm alternative to Lempel–Ziv–Welch. It was later adapted by Sennrich et al. (2016) as a tokenizer for MT. The algorithm starts by splitting words of individual tokens of length 1 (characters). Then it repeatedly takes the *most frequent* pair of adjacent tokens and joins it into a new single token, adding this token to the vocabulary. This procedure is repeated until we fill the predefined vocabulary budget. Zouhar et al. (2023) show that this greedy approach is approximately optimal when searching for the vocabulary (merge sequence). We are however interested in intentionally suboptimal vocabularies.

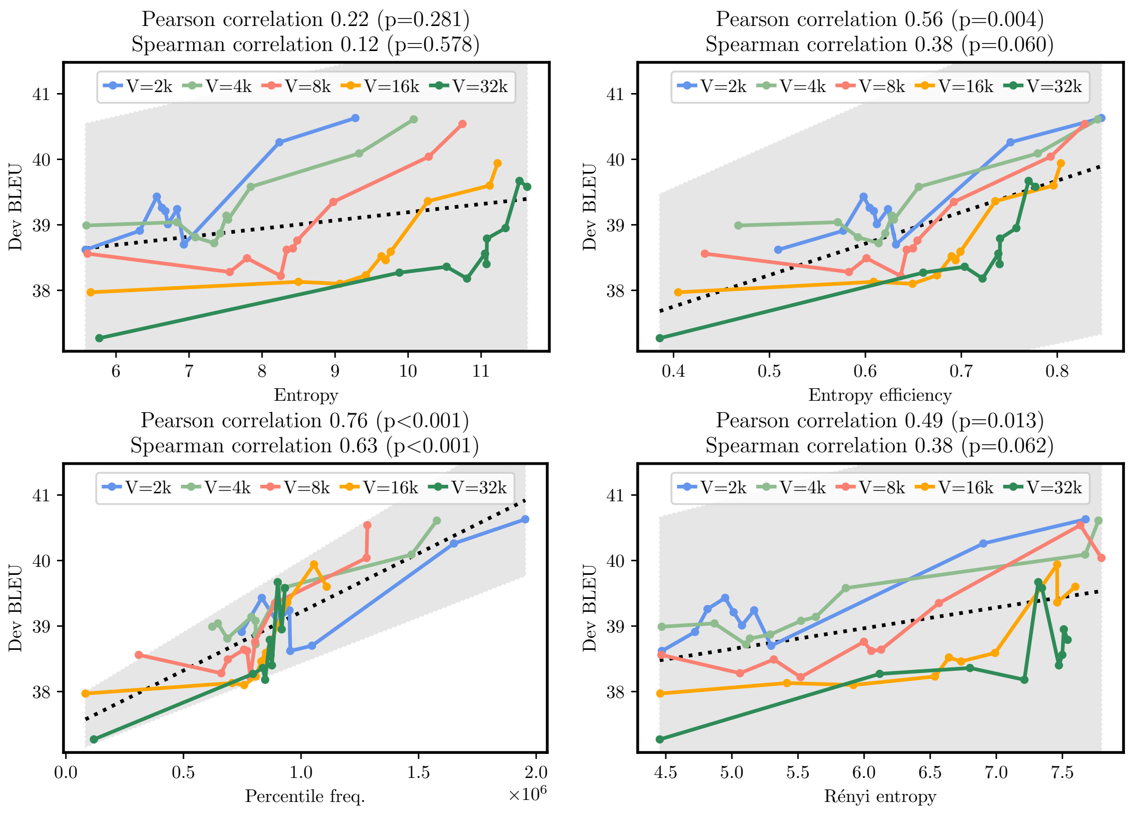

Figure 6: Predictor visualization in parallel to Fig. 2. Various features are used as predictors of MT performance (average of 5 runs). Bands show 95% t-test confidence intervals for regression lines.

Tokenizer	$V = 16k$	$V = 4k$
BPE	`Re gu lation`	`Re gu la tion`
BPE $\tau = -0.4$	`Reg ul ation`	`Reg ul a tio n`
Unigram	`Regulation`	`Re gu l ation`
WordPiece	`Regul ation`	`Re gul ation`
LZ	`Re gul ation`	`Re gu la tion`
Morfessor	`Regul ation`	`Re gul ation`

Table 3: Example tokenizations of the word `Regulation`.

Temperature. In order to introduce stochasticity into the process and intentionally alter the tokenization, instead of deterministically merging the most frequent token pair, we randomly sample a pair proportionally to their frequencies, similar to setup of Sälevä and Lignos (2023). We add an additional hyperparameter to this strategy by annealing the frequency distribution with a temperature parameter τ, where annealing is performed via a softmax. We use $\tau = 0^+$ (original greedy BPE), 0.2, 0.4, 0.9, 100, -100, -0.9, -0.4, -0.2, and 0^-. Progressively, each temperature creates less and less optimal BPE compression model, increasing the encoded length of the data. The last model, with $\tau = 0^-$, we dub antigreedy because it always chooses the least frequent pair to merge, practically leading to a character-only model.

E.2 Unigram LM

While BPE repeatedly merges the most frequent pair, Unigram LM (Kudo, 2018) tokenizer modifies this part of the algorithm with a more complex approach. The algorithm then jointly optimizes the token vocabulary and the unigram probability of the tokenized text. Because of this dual objective, the algorithm is done in Expectation-Maximization manner. We start by seeding the token vocabulary with the most frequent substrings. In one step, probability is assigned to individual tokens and also how much worse the

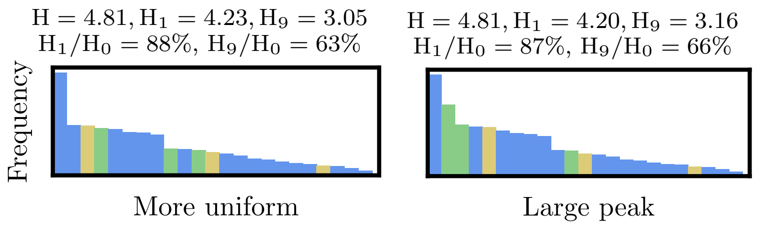

Figure 7: Examples of unigram distributions of tokens with varying imbalance of probabilities. Letters corresponding to `the` are in 🟩 and those corresponding to `cow` in 🟨.

overall probability of the whole text would be if the single token was removed. In the second step, top-$\eta\%$ of the tokens is preserved. These two steps are repeated until the vocabulary size is reduced to size V.

E.3 Linguistically Informed tokens

Morfessor (Creutz and Lagus, 2007; Virpioja et al., 2013; Smit et al., 2014) is a family of unsupervised morphological analyzers that work on the minimum description principle, which is not distant from compression. The link to tokenization is clear: Under some minimization objective, segment the input text into a list of morphemes with the constraint of at most V distinct morphemes appearing in the whole text, where V is a hyperparameter. In the case of tokenization, we would use the morphemes as tokens. For our purposes, we use vanilla Morfessor 2.0.6.

E.4 Lempel–Ziv–Welch Compression

BPE, a popular tokenization scheme used in many NLP applications, was first invented (Gage, 1994) as a faster compression algorithm alternative to Lempel–Ziv–Welch (LZW). However, there is, in fact, a family of algorithms related to LZW. LZ77 (Ziv and Lempel, 1977) and LZW (Welch, 1984) are two of the most popular variants.

Example E.1. *Consider a scenario where we want to encode a lowercased text without punctuation with the restriction of $|\Delta| = 28$. Upon adding the 26 lowercased letters of the English alphabet and space, we can add one additional token. Let us consider the tokens:* `the` *and* `cow`*. Certainly, the former token is much more frequent in natural English distribution and hence a tokenizer fully utilizing this token would result in shorter optimal expected code length. By adding the token* `the`*, we are lowering the probability mass of the most frequent English letters:* `e` *and* `t`*. This does not happen with* `cow` *and hence the first distribution is more uniform. For a depiction of this synthetic example, see Fig. 7. If we measure it using* eff_1*, we obtain only* 1% *difference but if we measure it using* eff_9*, we get 3% difference, showing that the first tokenization is much better.*