The Foundations of Tokenization: Statistical and Computational Concerns

Juan Luis Gastaldi[1] **John Terilla**[2]
Luca Malagutti[1] **Brian DuSell**[1]
Tim Vieira[1] **Ryan Cotterell**[1]
[1]ETH Zürich [2]City University of New York
juan.luis.gastaldi@inf.ethz.ch jterilla@gc.cuny.edu
lmalagutti@inf.ethz.ch brian.dusell@inf.ethz.ch
tim.f.vieira@gmail.com ryan.cotterell@inf.ethz.ch

Abstract

Tokenization—the practice of converting strings of characters over an alphabet into sequences of tokens over a vocabulary—is a critical yet under-theorized step in the NLP pipeline. Notably, it remains the only major step not fully integrated into widely used end-to-end neural models. This paper aims to address this theoretical gap by laying the foundations of tokenization from a formal perspective. By articulating and extending basic properties about the category of *stochastic maps*, we propose a unified framework for representing and analyzing tokenizer models. This framework allows us to establish general conditions for the use of tokenizers. In particular, we formally establish the necessary and sufficient conditions for a tokenizer model to preserve the *consistency* of statistical estimators. Additionally, we discuss statistical and computational concerns crucial for the design and implementation of tokenizer models. The framework and results advanced in this paper represent a step toward a robust theoretical foundation for neural language modeling.

La linguistique aurait pour tâche de déterminer quelles sont réellement ces unités valables en tout genre. On ne peut pas dire qu'elle s'en soit rendu compte, car elle n'a guère fait que discuter sur des unités mal définies. Non seulement cette détermination des unités quelle manie sera la tâche la plus pressante de la linguistique, mais ce faisant, elle aura rempli toute sa tâche.

—Ferdinand de Saussure, *Deuxième cours de linguistique generale*

1 Introduction

The search for the relevant units for the analysis of language has riddled philosophers and linguists alike for millennia. Plato, for instance, judged the "correctness of a name" by whether it was composed of "letters and syllables" which were themselves names or "an inappropriate letter" was employed (Plato, 1926, 427c-d,432d-e). Even before Plato, Pāṇini's *Aṣṭādhyāyī* already proposed a complete system for the derivational analysis of Sanskrit based on an impressive account of phonological, morphological, and lexical units together with detailed rules for their composition (Sharma, 2002; Kiparsky, 2009). However, it was arguably the Swiss linguist Ferdinand de Saussure who recognized in this problem a fundamental key to modern linguistic thought. Saussure's critical insight was that in linguistics, unlike other empirical sciences, the concrete units of analysis are *not immediately given in nature*. Beasts, plants, spatial units, or chemical compounds, Saussure argues,

Preprint.

are supposed to be units given from the outset to zoology, botany, astronomy, or chemistry, which are therefore primarily concerned with their comparison, not with their delimitation (Saussure, 1997, p. 18a). In linguistics, in contrast, the elementary building blocks of language cannot be found by the sole analysis of the physical properties of a language's material support (e.g., sound, ink, electricity, light), and must, therefore, be identified through a *segmentation* procedure involving various *structural* features. For Saussure, establishing the relevant units of language not only constitutes "the most pressing task of linguistics", but once such a task has been accomplished, little else, if anything, would remain to be done from a linguistic standpoint: "in doing this [linguistics] will have fulfilled all of its task" (Saussure, 1997, p. 21a).

Halfway through the 20th century, many attempts to provide a formal account of linguistic units had been proposed (e.g., Hjelmslev, 1975; Harris, 1960). However, with Chomsky's introduction of the context-free grammar as a formal model of language (Chomsky, 1957), the formalization of linguistics took a turn toward *symbolic models of computation*. Chomsky's generativist program shifted the focus to the syntactic structure of language, and the problem of delimiting linguistic units, practically absent from Chomsky's earlier work, gradually turned into that of the choice of a *parameter* left to the linguist. In the current definition of a context-free grammar, the parameter is an alphabet Σ of symbols, and a context-free language L is a subset of Σ^* (i.e., the set of all strings over that alphabet) generated by the rewrite system described by the grammar. Within the generativist program, some linguists sought to have Σ be the set of morphemes (e.g., Marantz, 2001), railing against the implicit privilege given to words in Chomsky (1965).

By the end of the past century, thanks to the resurgence of *empiricist* approaches in linguistics (e.g., Elman, 1996; Bybee & Hopper, 2001; McEnery & Wilson, 2001; MacWhinney, 1999; Chater et al., 2015), the question of linguistic units has progressively gained a renewed significance. In line with this empiricist perspective, modern natural language processing (NLP) has shifted the focus from searching for a grammar that characterizes a language $L \subseteq \Sigma^*$ to obtaining a probability distribution over Σ^* that places high probability on common utterances of a language. This probabilistic perspective makes the modeling task in linguistics particularly sensitive to the choice of the alphabet Σ. However, just as in context-free grammars, Σ is still a parameter that the modelers must choose themselves.

In this new context, the problem of linguistic units has come to be seen through the specific lens of *tokenization*. Within NLP, tokenization generally refers to the process of breaking up sequences of symbols into subsequences that can be represented as units or "tokens". The tokenization of linguistic data has long been a common practice in the processing of natural language (cf. Palmer, 2000; Jurafsky & Martin, 2024). However, with the emergence of deep neural models for NLP, the meaning of tokenization has progressively changed. In the framework of current language models, the problem of tokenization arises from the fact that, in practice, starting from an alphabet Σ of *elementary* units, one seeks to estimate a probability distribution over Σ^* *indirectly*, that is, by estimating a probability distribution over sequences of tokens in Δ^*, where the set of tokens Δ is, in general, *different from* Σ. Therefore, the problem of tokenization is determined by the forward and backward *mappings* between Σ^* and Δ^*.

The use of tokenizers for neural language models was popularized with the development and widespread adoption of the *Byte Pair Encoding* (BPE) algorithm for subword tokenization, adapting an existing compression algorithm (Gage, 1994) to the processing of natural language in the context of neural machine translation (Sennrich et al., 2016). BPE quickly replaced previous rule-based tokenizer models such as Morfessor (Creutz & Lagus, 2002) and Moses (Koehn et al., 2007), and was followed by other data-driven models, including *WordPiece* (Wu et al., 2016, following Schuster & Nakajima, 2012) and *Unigram* (Kudo, 2018) among the most widely adopted (cf. Mielke et al., 2021, for a survey). All these recent subword models, now built into standard language modeling toolkits, consist of a set of algorithms to build an alphabet Σ and a vocabulary Δ from existing data, and to map between their respective strings.

The significance of subword tokenization for language models has grown. Among their recognized benefits, two are often advanced in the literature. Tokenizers offer the capacity of training language models over an *open vocabulary*, circumventing the difficulties associated with out-of-vocabulary terms (Sennrich et al., 2016). Also, tokenization is often described as an efficient, lossless *encoding* of the original data (Zouhar et al., 2023a). Moreover, based on empirical evidence of different kinds, tokenization has been hypothesized to introduce a helpful inductive bias in language modeling (Nawrot et al., 2023; Schmidt et al., 2024; Uzan et al., 2024), although in the current state of the art, this hypothesis remains an open question. At the same time, tokenizers can exhibit undesirable behaviors that can have a negative impact on language models. To name just a few, tokenization can introduce spurious ambiguity in LMs (Kudo, 2018; Cao & Rimell, 2021), generate alignment issues (Poesia et al., 2022; Athiwaratkun et al., 2024), or result in inconsistent scoring in the use of LMs in other scientific fields, like psycholinguistics (Salazar et al., 2020; Kauf & Ivanova, 2023).

Despite a growing awareness of its significance for NLP, a formal theory of tokenization is still lacking. The fact that tokenization is the one step in the classical NLP pipeline that resists being incorporated within the end-to-end neural model only makes a formal treatment of tokenization even more appealing. In the past years, some effort has been made to provide a more theoretical perspective on the principles of tokenization (Guo, 1997; Kudo, 2018; Zouhar et al., 2023b,a; Rajaraman et al., 2024). However, we see a need for a more fundamental perspective, from which necessary conditions for the consistent use of tokenizers can be established in a principled manner.

The present paper aims to make a decisive step toward a robust theoretical grounding for neural language modeling by laying the foundations of tokenization from a *formal* perspective. In §2, we present preliminary notions, including elementary aspects of formal language theory and the crucial concept of stochastic maps, followed by notational and terminological remarks. In §3, we propose a unified framework for representing and analyzing tokenization models. This framework allows us to establish general conditions for the use of tokenizers. In particular, we establish the necessary and sufficient conditions for a tokenizer model to preserve the *consistency* of statistical estimators. Finally, in §§ 4 and 5 we discuss statistical and computational concerns such as inconsistency, ambiguity, computability, and boundedness, relevant for the study, design, and implementation of tokenizer models.

2 Preliminaries

2.1 Formal Languages, Estimators, and Stochastic Maps

An **alphabet** Σ is a finite, non-empty set of **symbols**. The set $\Sigma^n \overset{\text{def}}{=} \overbrace{\Sigma \times \cdots \times \Sigma}^{n \text{ times}}$ consists of **strings** of symbols of length n. The symbol ε denotes the empty string of length 0. The union $\Sigma^* \overset{\text{def}}{=} \bigcup_{n=0}^{\infty} \Sigma^n$ consists of all finite strings (including ε) from the alphabet Σ. Similarly, we denote by $\Sigma^{\leq N}$ the set of all strings from Σ of length less or equal to N. String concatenation is an associative product $\Sigma^* \times \Sigma^* \xrightarrow{\cdot} \Sigma^*$ for which ε is an identity element. The triple $(\Sigma^*, \cdot, \varepsilon)$ defines a **monoid**, which is, in fact, a model of the free monoid on the set Σ.

A **language** L over an alphabet Σ is a set of strings $L \subseteq \Sigma^*$. A **language model** p is a probability distribution over Σ^*. That is, p is a function $p : \Sigma^* \to [0, 1]$ such that $\sum_{\boldsymbol{\sigma} \in \Sigma^*} p(\boldsymbol{\sigma}) = 1$. Language models generalize languages in the sense that the support of a language model, i.e., $\text{supp}(p) = \{\boldsymbol{\sigma} \mid p(\boldsymbol{\sigma}) \neq 0\}$, is a language. The definition of a language model as a probability distribution on Σ^* is deliberately broad. In particular, note that no compatibility between p and the monoidal structure in Σ^* is assumed.

In NLP, practitioners generally seek to **estimate** a language model p from exemplars of naturally occurring text. Formally, the modeler assumes there exists a true distribution $p^\star$ over Σ^*, and considers a multiset of naturally occurring texts $\{\boldsymbol{\sigma}_n\}_{n=1}^N \subset \Sigma^*$ to be samples from $p^\star$. In its most general form, an estimator of $p^\star$ is a sequence $\{p_n\}$ of probability distributions on Σ^* such that p_n becomes closer to $p^\star$ as n increases. We call an estimator **consistent** if the sequence $\{p_n\}$ converges *pointwise* to $p^\star$.[1] More precisely, given a probability distribution $p^\star : \Sigma^* \to [0, 1]$, and a sequence distributions $\{p_n : \Sigma^* \to [0, 1]\}$, we say that $\{p_n\}$ is a consistent estimator of $p^\star$ if and only if, for all strings $\boldsymbol{\sigma} \in \Sigma^*$, the sequence of numbers $\{p_n(\boldsymbol{\sigma})\}$ converges to the number $p^\star(\boldsymbol{\sigma})$.

This notion of consistent estimation is general enough to include many estimation methods, where the p_i can depend on various properties of the sample, such as the size N, or possibly a set of parameters θ. Likewise, we use pointwise convergence to define consistent estimation because

[1]Following common convention, we will sometimes denote convergence as $\{p_n\} \to p^\star$, which is not to be confused with the notation for functional types (e.g., $\Sigma^* \to \Delta^*$). The context of use should be enough to prevent any ambiguity.

pointwise convergence is a weak kind of convergence, and so our definition is compatible with a wide variety of convergence measures, and relatively easy to check for. For example, a common practice in NLP is to produce an estimator $\{p_n\}$ through a sequence of steps in the process of minimizing *cross entropy loss*, which amounts to minimizing the *relative entropy* also called *the Kullback–Leibler divergence*, $D_{\mathrm{KL}}(p^\star \parallel p_n)$ between $p^\star$ and p_n. This is a stronger form of convergence: if $D_{\mathrm{KL}}(p^\star \parallel p_n) \to 0$ then $p_n \to p^\star$ pointwise (a consequence of Pinsker's lemma) and so $\{p_n\}$ is a consistent estimator of $p^\star$.

Our definition of tokenizer models will require the use of a special kind of map between sets called a stochastic map. The reference (Baez & Fritz, 2014) contains a detailed introduction to the category of finite sets with stochastic maps between them. Here, we will extend some of the results in Baez & Fritz (2014) to cover the case of countably infinite sets. We assume all sets are countable, either finite or countably infinite. A **stochastic map** from a set X to a set Y is a function from X to the set of probability distributions on Y. We use

$$X \rightsquigarrow Y$$

to denote a stochastic map from X to Y and the notation $x \mapsto f(y \mid x)$ to denote the probability of $y \in Y$ in the distribution assigned to $x \in X$. In other words, a stochastic map $f\colon X \rightsquigarrow Y$ is a function

$$\begin{aligned} X \times Y &\to [0,1] \\ (x,y) &\mapsto f(y \mid x) \end{aligned}$$

satisfying $\sum_{y \in Y} f(y \mid x) = 1$. One might think of the number $f(y \mid x)$ as a kind of "conditional probability" of y given x, however this is only a metaphor since there is no assumption that the numbers $f(y \mid x)$ assemble into a joint distribution on $X \times Y$. In particular, the sum $\sum_{x \in X} f(y \mid x)$ could be infinite.

Significantly, stochastic maps can be composed. The composition

$$X \overset{f}{\rightsquigarrow} Y \overset{g}{\rightsquigarrow} Z, \qquad X \underset{gf}{\rightsquigarrow} Z$$

$gf\colon X \rightsquigarrow Z$ is defined by

$$gf(z \mid x) = \sum_{y \in Y} g(z \mid y) f(y \mid x). \tag{1}$$

Since the sum in equation (1) is infinite, it requires a check that the formula for $gf(z \mid x)$ is finite, and that $gf(-, x)$ defines a probability distribution on Z, both of which follow from the fact that:

$$\sum_{z \in Z} gf(z \mid x) = \sum_{z \in Z} \sum_{y \in Y} g(z \mid y) f(y \mid x) = \sum_{y \in Y} \sum_{z \in Z} g(z \mid y) f(y \mid x) = \sum_{y \in Y} f(y \mid x) = 1.$$

If one arranges a stochastic map into an $|X| \times |Y|$ matrix with the $f(y \mid x)$ entry in the x, y position, then every entry is nonnegative and the sum of every row is 1. The computation above shows that composition of stochastic maps is realized by matrix multiplication, and that even when the matrices are infinite, the row-column dot products are finite and the result of matrix multiplication is again a matrix with nonnegative entries whose rows sum to 1. From this perspective, it is clear that composition of stochastic maps is associative.

Stochastic maps generalize both ordinary probability distributions and functions. A probability distribution over a set X can be represented as a stochastic map into X from a 1-element set, denoted as $\mathbf{1} := \{1\}$, i.e., $p\colon \mathbf{1} \rightsquigarrow X$. In such cases, the customary notation $p(x)$ can be used without risk of ambiguity as a shorthand of the more cumbrous $p(x \mid 1)$. An ordinary function $f\colon X \to Y$ can be regarded as a stochastic map $X \rightsquigarrow Y$ by mapping x to the probability distribution on Y concentrated on the singleton $\{f(x)\}$, in which case we say the stochastic map f is *deterministic*. For simplicity, when a stochastic map $f : X \rightsquigarrow Y$ is deterministic, we may write $f(x)$ instead of $f(y \mid x)$. Composition generalizes both composition of functions and the pushforward of a probability function via a function. If $p\colon \mathbf{1} \rightsquigarrow X$ is a probability distribution on X and $f : X \to Y$ is a deterministic function, then the composition $\mathbf{1} \overset{p}{\rightsquigarrow} X \overset{f}{\rightsquigarrow} Y$ is a stochastic map $fp : 1 \rightsquigarrow Y$, which is a probability distribution on Y whose formula is $fp(y) = \sum_{x \in X} f(y|x)p(x|1) = \sum_{x \in f^{-1}(y)} p(x)$. That is, fp is just the pushforward of the probability distribution p via the function f.

2.2 Notation and Terminology

We adopt the following notational conventions. Alphabets will be denoted by uppercase Greek letters (e.g., Σ, Δ). In the context of tokenization, we will be interested in looking at maps between strings of languages over two different alphabets, which we will denote as Σ and Δ. To avoid ambiguity, we reserve the term **alphabet** for the former and call **vocabulary** the latter instead. We denote symbols by lowercase Greek letters, e.g., $\sigma \in \Sigma, \delta \in \Delta$, calling them **characters** in the first case and **tokens** in the second. Strings will be denoted by bold lowercase Greek letters, e.g., $\boldsymbol{\sigma} \in \Sigma^*, \boldsymbol{\delta} \in \Delta^*$, reserving the name character strings or **texts** for the former and token strings or **token sequences** for the latter. When necessary, we will distinguish the empty character string $\varepsilon_\Sigma \in \Sigma^*$ from the empty token sequence $\varepsilon_\Delta \in \Delta^*$. Examples of strings and tokens will be written in monospace font (e.g., `t`, `the`). There are cases where $\Delta \cap \Sigma^* \neq \emptyset$, and it will be necessary to distinguish between concatenation in Σ^* and Δ^*. In Δ^*, concatenation will be denoted as $\shortmid$. So, for example, if $\Sigma = \{\mathtt{t}, \mathtt{h}, \mathtt{e}\}$ and $\Delta = \{\mathtt{th}, \mathtt{he}, \mathtt{e}\}$, the expression $\mathtt{t} \cdot \mathtt{h} \cdot \mathtt{e}$ denotes the concatenation in Σ^* of the three characters `t`, `h`, and `e`, while the expression $\mathtt{t}\shortmid\mathtt{he}$ represents the concatenation in Δ^* of the two tokens `t` and `he`. The cases when $\Delta \cap \Sigma^* \neq \emptyset$ are of sufficient significance that we shall generally avoid using simple juxtaposition of characters to express concatenation. Therefore, the reader should always interpret `th` as a token in Δ, and not a text in Σ^* (written $\mathtt{t} \cdot \mathtt{h}$). If further notational clarification is needed, square brackets may be used to represent the concatenation of two texts in Σ^* (and likewise for Δ^*). For instance, $[\mathtt{t} \cdot \mathtt{h}] \cdot \mathtt{e}$ denotes the concatenation of the text $\mathtt{t} \cdot \mathtt{h}$ with the character `e` in Σ^*. Should any ambiguity between specific characters and tokens arise (e.g., $\mathtt{t} \in \Sigma$ vs. $\mathtt{t} \in \Delta$), it will be explicitly disambiguated whenever there is a risk that context alone is insufficient.

3 A Formal Framework for Tokenization

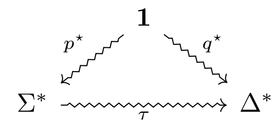
As discussed in the previous pages, in current NLP, the problem of tokenization arises from the fact that one seeks to estimate a model $p^\star$ over strings of symbols in one alphabet *indirectly*, that is, by estimating a probability distribution q over strings of symbols on a different alphabet. Therefore, from a strictly formal perspective, the problem of tokenization can be characterized as that of the respective mappings between two sets of strings, namely the set Σ^* of character strings and the set Δ^* of token sequences. In order to estimate $p^\star$ through q, Σ^* needs to be mapped into and from Δ^*. The connection between Σ^* and Δ^* is thus made through a pair of mappings (τ, κ) that constitutes the basis of our formal characterization of tokenization. Accordingly, in its most general form, a tokenizer can be defined as follows:

Definition 3.1. *A* ***tokenizer model*** *(or simply* ***tokenizer****) from Σ^* to Δ^* is a pair of stochastic maps $\mathcal{T} = (\tau, \kappa)$, respectively called the* ***encoder*** *and the* ***decoder****, where the encoder is a stochastic map $\tau \colon \Sigma^* \rightsquigarrow \Delta^*$, and the decoder is a stochastic map $\kappa \colon \Delta^* \rightsquigarrow \Sigma^*$.*

Definition 3.1 is deliberately broad, covering *any* pair of string-to-string mappings τ and κ. Other than the fact that the domain of each mapping constitutes the codomain of the other, we define the encoder and decoder as arbitrary stochastic maps. In other words, we will be regarding τ and κ primarily from the point of view of their *composition*. In particular, we do not require any specific connection between the alphabet Σ and the vocabulary Δ. However, the implicit assumption behind the established use of tokenizers in language models is that the samples $\{\boldsymbol{\sigma}_n\}_{n=1}^N \subset \Sigma^*$ of naturally occurring texts used for estimation can be mapped into Δ^* in such a way that the estimated model q can be, in turn, transformed into a model p over Σ^* through the map κ, such that $\kappa q = p$ can be considered as an estimate of the original distribution $p^\star$.

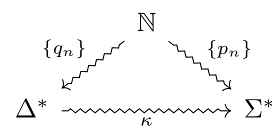
Despite the potential empirical increase in a model's predictive performance resulting for specific tokenization choices, the soundness of such a procedure is not guaranteed for arbitrary τ and κ without further conditions. On one hand, the notion of estimation in Δ^* is not well-defined unless there exists a reference distribution $q^\star$ over Δ^* to which the estimator $\{q_n\}$ can converge. On the other, assuming such an estimator is consistent, transforming it into a consistent estimator of $p^\star$ requires a way to map the sequence $\{q_n\}$ into a sequence $\{p_n\}$ that converges to $p^\star$.

Assuming a reference distribution $p^\star$ exists on Σ^*, one obtains a reference $q^\star$ on Δ^* simply as the composition (Eq. (1)) with the encoder: $q^\star = \tau p^\star$. In other words, the following diagram of stochastic maps commutes



The distribution $q^\star$ is just the **pushforward** of the measure $p^\star$ along τ, which then makes the encoder τ a measure-preserving map between $(\Sigma^*, p^\star)$ and $(\Delta^*, q^\star)$

In the same way, $\{p_n\}$ can be obtained by mapping the sequence $\{q_n\}$ through κ. By defining $p_i = \kappa q_i$, we obtain the following commutative diagram



So far, none of these requirements imposes conditions on τ and κ other than being well-defined mappings between their respective domains and codomains. Notably, the notion of estimation of $\tau p^\star$ is well defined for arbitrary τ. However, given a consistent estimator $\{q_n\}$ of $q^\star$, $\{\kappa q_n\}$ *is not guaranteed to converge to* $p^\star$ without further conditions on κ. To establish such conditions, we will need the following lemmas.

Lemma 3.1. *Let $\{p_n\}$ be a sequence of probability distributions over a countable set X that converges pointwise to a probability distribution p. Then $\lim_{n\to\infty} \sum_{x\in X} |p_n(x) - p(x)| = 0$.*

Proof. Fatou's lemma applied to X with the counting measure implies that for any sequence of nonnegative functions $\{f_n\}$ on X,

$$\sum_{x\in X} \liminf_{n\to\infty} f_n(x) \leq \liminf_{n\to\infty} \sum_{x\in X} f_n(x). \tag{2}$$

We'll apply this to $f_n := p_n + p - |p_n - p|$. First, note that since $\lim_{n\to\infty} p_n(x) = p(x)$, we have $\liminf_{n\to\infty} f_n(x) = p(x) + p(x) - 0 = 2p(x)$ so the left hand side of (2) becomes $\sum_{x\in X} 2p(x) = 2$. Therefore,

$$\begin{aligned}
2 &\leq \liminf_{n\to\infty} \sum_{x\in X} f_n(x) \\
&= \liminf_{n\to\infty} \sum_{x\in X} p_n(x) + p(x) - |p_n(x) - p(x)| \\
&= \liminf_{n\to\infty} \sum_{x\in X} p_n(x) + \sum_{x\in X} p(x) - \sum_{x\in X} |p_n(x) - p(x)| \\
&= \liminf_{n\to\infty} 1 + 1 - \sum_{x\in X} |p_n(x) - p(x)| \\
&= 2 - \limsup_{n\to\infty} \sum_{x\in X} |p_n(x) - p(x)|.
\end{aligned}$$

It follows that $\limsup_{n\to\infty} \sum_{x\in X} |p_n(x) - p(x)| \leq 0$. So $\lim_{n\to\infty} \sum_{x\in X} |p_n(x) - p(x)| = 0$. □

Corollary 3.0.1. *Let $\{p_n\}$ be a sequence of probability distributions over a countable set X that converges pointwise to a probability distribution p. Then $\{p_n\} \to p$ uniformly.*

Proof. Since the sum of nonnegative numbers is always greater than any particular term in the sum and $\lim_{n\to\infty} \sum_{x\in X} |p_n(x) - p(x)| = 0$, we can conclude that the sequence $\{p_n\} \to p$ uniformly. □

Lemma 3.2. *Let f be a stochastic map from X to Y, and $\{p_n\}$ be an estimator for a probability distribution p on X. Then fp_n is an estimator for the probability distribution fp on Y.*

Proof. Fix $y \in Y$. We will show that $\{fp_n(y)\} \to fp(y)$. By Lemma 3.1, we have that $\lim_{n\to\infty} \sum_{x\in X} |p_n(x) - p(x)| = 0$. Therefore,

$$\lim_{n\to\infty} |fp_n(y) - fp(y)| = \lim_{n\to\infty} \left| \sum_x f(y \mid x) p_n(x) - \sum_x f(y \mid x) p(x) \right| \tag{3}$$

$$\leq \lim_{n\to\infty} \sum_x f(y \mid x) \left| p_n(x) - p(x) \right| \tag{4}$$

$$\leq \lim_{n\to\infty} \sum_x |p_n(x) - p(x)| \tag{5}$$

$$= 0. \tag{6}$$

□

In other words, Lemma 3.2 says that *stochastic maps preserve the consistency of estimators*. Armed with this lemma, it is now easy to establish a simple but **fundamental principle** for the use of tokenization models in language modeling:

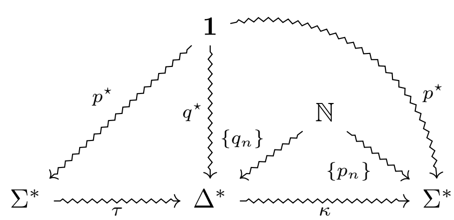
Theorem 3.1 (Fundamental Principle of Tokenization)**.** *Given a reference probability distribution $p^\star$ over Σ^*, a tokenizer $\mathcal{T} = (\tau, \kappa)$ from Σ^* to Δ^*, and a consistent estimator $\{q_n\}$ of the image reference distribution $q^\star = \tau p^\star$, the sequence $\{\kappa q_n\}$ is a consistent estimator of $p^\star$ if and only if $\kappa\tau p^\star = p^\star$.*

Proof. By hypothesis, $\{q_n\} \to q^\star$ and by definition $q^\star = \tau p^\star$. By Lemma 3.2, applying κ to both sides, we have that $\{\kappa q_n\} \to \kappa q^\star$ and so

$$\{\kappa q_n\} \to \kappa\tau p^\star.$$

Therefore, if $\kappa\tau p^\star = p^\star$ we have $\{\kappa q_n\} \to p^\star$. Conversely, if $\{\kappa q_n\} \to p^\star$ we have both $\{\kappa q_n\} \to p^\star$ and $\{\kappa q_n\} \to \kappa\tau p^\star$ and so by the uniqueness of limits, $\kappa\tau p^\star = p^\star$. □

Our whole setting can be represented with the following diagram:



This setting is quite general, and Theorem 3.1 characterizes precisely when a consistent estimator $\{q_n\}$ of $q^\star$ yields a consistent estimator $\{p_n\}$ of $p^\star$ after decoding. Based on the fundamental principle expressed in Theorem 3.1, we propose the following definitions:

Definition 3.2. *Given a probability distribution p over Σ^*, a tokenizer $\mathcal{T} = (\tau, \kappa)$ from Σ^* to Δ^* is* ***consistent with respect to*** *p if we have $\kappa\tau p = p$.*

Definition 3.3. *Let p be a probability distribution over Σ^* and $\mathcal{T} = (\tau, \kappa)$ a tokenizer from Σ^* to Δ^*. When $\kappa\tau = \mathrm{id}_{\Sigma^*}$, we say that $\mathcal{T}$ is* ***exact****.*

Notice that exact tokenizers are consistent, but a tokenizer that is consistent with respect to a distribution p is not necessarily exact. Take, for instance, a probability distribution p over some set X and $x', x'' \in X$ such that $p(x') = p(x'') = c$. Then one can fashion a tokenizer for which $\kappa\tau(x) = x$ for all x except $\kappa\tau(x') = x''$ and $\kappa\tau(x'') = x'$. Such a tokenizer is consistent with respect to p without being exact. Consistency with respect to all distributions, however, is the same as being exact.

Proposition 3.1. *A tokenizer $\mathcal{T} = (\tau, \kappa)$ from Σ^* to Δ^* is exact if and only if it is consistent with respect to every probability distribution over Σ^*.*

Proof. Exact means $\kappa\tau = \mathrm{id}_{\Sigma^*}$ hence $\kappa\tau p = p$ for every probability distribution p on Σ^*. To prove the other direction, suppose that $\kappa\tau p = p$ for every p on Σ^*. Fix an arbitrary $\boldsymbol{\sigma} \in \Sigma^*$. Let $p_{\boldsymbol{\sigma}}$ be the point mass distribution on Σ^* concentrated on $\boldsymbol{\sigma}$. So $p_{\boldsymbol{\sigma}}(\boldsymbol{\sigma}) = 1$ and $p_{\boldsymbol{\sigma}}(\boldsymbol{\sigma}') = 0$ for any $\boldsymbol{\sigma}' \neq \boldsymbol{\sigma}$. By hypothesis, $p_{\boldsymbol{\sigma}} = \kappa\tau p_{\boldsymbol{\sigma}}$. Apply to $\boldsymbol{\sigma}$ to get $1 = \kappa\tau p_{\boldsymbol{\sigma}}(\boldsymbol{\sigma})$. The right hand side, as the pushforward of $p_{\boldsymbol{\sigma}}$ via $\kappa\tau$, says $1 = p_{\boldsymbol{\sigma}}(\kappa\tau(\boldsymbol{\sigma}))$. Since $p_{\boldsymbol{\sigma}}$ takes the value 1 at only one point, it follows that the argument $\kappa\tau(\boldsymbol{\sigma}) = \boldsymbol{\sigma}$. Since $\boldsymbol{\sigma}$ was arbitrary, we conclude that $\kappa\tau = \mathrm{id}_{\Sigma^*}$, i.e., the tokenizer (τ, κ) is exact. □

Significantly, exact tokenizers have special properties. If a tokenizer (τ, κ) is exact, then κ is deterministic, because id_{Σ^*} also is, i.e., they both place all the probability mass on one element. More importantly, the condition $\kappa\tau = \mathrm{id}_{\Sigma^*}$ means τ is a **section** for κ. This implies that $\tau(\boldsymbol{\delta} \mid \boldsymbol{\sigma})$ places probability zero on every $\boldsymbol{\delta} \in \Delta^*$ such that $\kappa(\boldsymbol{\delta}) \neq \boldsymbol{\sigma}$ (Baez & Fritz, 2014, p. 7). This implies, in turn, that κ is onto (surjective), otherwise we would not have $\sum_{\boldsymbol{\delta} \in \Delta^*} \tau(\boldsymbol{\delta} \mid \boldsymbol{\sigma}) = 1$.

4 Statistical Concerns: Inconsistency and Ambiguity

While in most concrete cases of neural language modeling, a tokenizer's consistency is implicitly or explicitly assumed, there are many ways in which the conditions established in the previous section can, and in practice do, fail to be satisfied. In this section, we address two main statistical concerns to be considered when implementing or using tokenizers: inconsistency and ambiguity. The following definitions will be convenient:

Definition 4.1. *Given a tokenizer $\mathcal{T} = (\tau, \kappa)$, we say $\mathcal{T}$ is **deterministic** if τ is a deterministic map. Otherwise we say $\mathcal{T}$ is **stochastic**.*

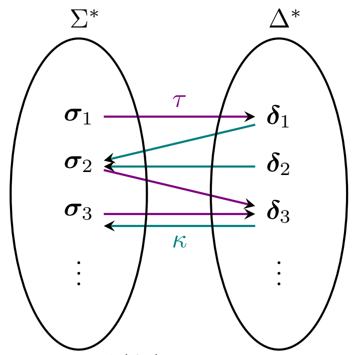
Definition 4.2. *When a tokenizer $\mathcal{T}$ is both deterministic and exact, we have that κ is a deterministic function, and $\kappa = \tau^{-1}$ over $\tau(\Sigma^*)$. Therefore, in such a case we say $\mathcal{T}$ is **bijective**.*

Most commonly used tokenizers are deterministic, including BPE (Sennrich et al., 2016) and WordPiece (Wu et al., 2016), as well as Unigram (Kudo, 2018) when used without regularization. As we have seen, deterministic functions can be understood as a particular case of stochastic maps where the probability mass is concentrated on one element. Deterministic tokenizers thus constitute a simplified form of tokenization. However, even in this simplified setting, the consistency of the tokenization process is not guaranteed. The following example offers an elementary intuition of this circumstance.

Example 4.1. *Consider the simple configuration represented in Fig. 1, where both τ and κ are deterministic maps. Let $p^\star(\boldsymbol{\sigma}_1) = 0.2$ and $p^\star(\boldsymbol{\sigma}_2) = p^\star(\boldsymbol{\sigma}_3) = 0.4$, with $p^\star(\boldsymbol{\sigma}_i) = 0$ for $i > 3$. For $q^\star = \tau p^\star$, we have, therefore, $q^\star(\boldsymbol{\delta}_1) = 0.2$, $q^\star(\boldsymbol{\delta}_2) = 0$, and $q^\star(\boldsymbol{\delta}_3) = 0.8$, with $q^\star(\boldsymbol{\delta}_i) = 0$ for $i > 3$, and hence $\kappa\tau p^\star(\boldsymbol{\sigma}_1) = 0 \neq 0.2$, $\kappa\tau p^\star(\boldsymbol{\sigma}_2) = 0.2 \neq 0.4$, and $\kappa\tau p^\star(\boldsymbol{\sigma}_1) = 0.8 \neq 0.4$. Assuming $\{q_n\}$ is a consistent estimator of $q^\star$, the pushforward of q_n through κ (i.e., κq_n) would result in an inconsistent estimation of $p^\star$. Notice that the consistency of the tokenizer is relative to the distribution. Relative to a different distribution p in Σ^*, where, for instance, $p(\boldsymbol{\sigma}_1) = p(\boldsymbol{\sigma}_2) = 0$ and $p = p^\star$ otherwise, the tokenizer specified in Fig. 1 is consistent.*

A possible cause of a tokenizer's inconsistency is the lack of injectivity of the encoder function τ. While encoder functions may superficially seem to be injective, certain decisions can result in implementations that are not injective. For instance, it can happen that τ is undefined for some elements in Σ, and is, therefore, only a partial function. If the exceptions are handled by returning a unique distinguished token in Δ (e.g., an 'unknown' token UNK), then τ becomes noninjective, incurring the risk of inconsistency. Most tokenizer models attempt to avoid this behavior by injecting Σ into Δ—in other words, by including the alphabet in the vocabulary by default. However, some implementations, for instance, limit the size of Σ to the sample's k most frequent symbols, mapping all other symbols to an UNK character in Σ (e.g., Wu et al., 2016). Understood as a preprocessing step, this operation should not affect τ's injectivity. However, some implementations keep track of the original out-of-alphabet symbols to restore them in decoding, thus violating *de facto* the tokenizer's injectivity, and with it, the model's consistency over strings including those symbols.[2]

[2]Even if all symbols in the training sample are included in Σ, it can always happen that out-of-alphabet symbols are encountered at test or inference time. The recourse to a distinguished UNK symbol both in Σ and Δ must, therefore, be carefully handled, bearing in mind that, even if the goal of subword tokenization is to



$$p^\star(\boldsymbol{\sigma}_1) = 0.2 \qquad\qquad q^\star(\boldsymbol{\delta}_1) = 0.2 \qquad\qquad \kappa\tau p^\star(\boldsymbol{\sigma}_1) = 0$$
$$p^\star(\boldsymbol{\sigma}_2) = 0.4 \quad \xrightarrow{\tau} \quad q^\star(\boldsymbol{\delta}_2) = 0 \quad \xrightarrow{\kappa} \quad \kappa\tau p^\star(\boldsymbol{\sigma}_2) = 0.2$$
$$p^\star(\boldsymbol{\sigma}_3) = 0.4 \qquad\qquad q^\star(\boldsymbol{\delta}_3) = 0.8 \qquad\qquad \kappa\tau p^\star(\boldsymbol{\sigma}_3) = 0.8$$

Figure 1: Example of an inconsistent tokenizer with τ represented in violet and κ represented in teal.

Whenever κ is noninjective, the tokenizer introduces *ambiguity* in the model because more than one token sequence is mapped to a unique text. In bijective tokenizers, decoding is injective over the encoder's image, thus preventing ambiguity in principle. However, in practice, whenever $\tau(\Sigma^*) \subset \Delta^*$, it may happen that the probability mass placed by the estimated language model outside the image of τ is nonzero, reintroducing ambiguity into the model (cf. Example 4.2 below for an elementary illustration). This ambiguity is, however, *spurious* because τ was assumed to be deterministic, and hence the ambiguity does not stem from the reference distribution $p^\star$, but is a side-effect of the estimator. An obvious source of spurious ambiguity resides in the fact that consistency is a property defined *in the limit*. As a consequence, for any $\boldsymbol{\delta} \in \Delta^*$, $q_n(\boldsymbol{\delta})$ can and will generally differ from $q^\star(\boldsymbol{\delta})$. Spurious ambiguity can also result from the fact that, due to the properties of gradient descent and certain activation functions such as softmax, neural models are incapable of assigning zero probability to elements of Δ^*. Although spurious ambiguity is usually overlooked or disregarded, it can, in principle, have a nonnegligible effect on estimation (Cao & Rimell, 2021).

Example 4.2. *Take, for instance, a bijective tokenizer such as BPE or WordPiece, with κ performing concatenation of the token maps in the usual way. Let $\Sigma = \{\texttt{t}, \texttt{h}, \texttt{e}\}$ and $\Delta = \{\texttt{t}, \texttt{h}, \texttt{e}, \texttt{th}, \texttt{he}\}$. In this minimal configuration, it is easy to see that $\kappa(\texttt{t|h|e}) = \kappa(\texttt{t|he}) = \kappa(\texttt{th|e}) = \texttt{t} \cdot \texttt{h} \cdot \texttt{e} \in \Sigma^*$. However, BPE or WordPiece being bijective tokenizers, τ can only map the value of κ to at most one of the latter's arguments, say $\tau(\texttt{t} \cdot \texttt{h} \cdot \texttt{e}) = \texttt{th|e}$. We then have that $\tau(\kappa(\texttt{t|he})) \neq \texttt{t|he}$ (and likewise for $\texttt{t|h|e}$). If the estimator happens to place nonzero probability mass on any of the latter two token sequences, the model will exhibit spurious ambiguity.*

Spurious ambiguity is not the only kind of ambiguity that can result from the use of tokenization in language models. Whenever a tokenizer model is stochastic, a deterministic κ must be noninjective for the model to be consistent. However, the ambiguity thus introduced is not spurious in that it is deliberately designed for statistical purposes. In current tokenization practices, the main reason for the introduction of stochastic ambiguity is *regularization* (Kudo, 2018; Provilkov et al., 2020). The claim is that, by exhibiting different token sequences corresponding to the same text during training, a model increases its capability to handle text compositionality as well as its robustness to noise and tokenization errors. However, one could also conceive of a stochastic tokenizer where the possible images of a text reflect the objective probabilities of all *linguistic* ambiguities potentially affecting it (e.g., $\texttt{an|icecream}$, $\texttt{an|ice|cream}$, $\texttt{a|nice|cream}$ as three possible token sequences for the text: $\texttt{a} \cdot \texttt{n} \cdot \texttt{i} \cdot \texttt{c} \cdot \texttt{e} \cdot \texttt{c} \cdot \texttt{r} \cdot \texttt{e} \cdot \texttt{a} \cdot \texttt{m}$).

Although all these classes of ambiguity (spurious, stochastic, and linguistic) are both formally and semantically different, they all represent the same risk for the tokenizer's consistency: The probability mass indirectly assigned by the model to one text in a language is spread out over different token sequences. Notice that all these cases of ambiguity can coexist, and hence their impact is difficult to

achieve open-vocabulary models, language models are not models of arbitrary textual expressions, but of their projection onto a closed alphabet.

evaluate. Yet, from a formal perspective, the solution for all these cases is the same: The computation of κq_n for a single text $\boldsymbol{\sigma} \in \Sigma^*$ requires marginalizing over all its preimages $\boldsymbol{\delta}$ through κ, for which $q_n(\boldsymbol{\delta}) > 0$, following the composition of stochastic maps presented in the previous section (Eq. (1)). However, such operation can be computationally challenging because it can imply summing over a large or even infinite number of terms. Addressing this challenge, Cao & Rimell (2021) proposed an importance-sampling estimator to approximate the marginalization over all possible preimages of a text, using as proposal distribution the distribution over all possible tokenizations for a text provided by the Unigram model. Their results suggest that evaluating language models through marginal probabilities over tokenizations can result in significant differences compared to the usual "one-best" method, although only in the case of "strongly out-of-domain evaluation sets" (see Chirkova et al. (2023) for a critical perspective).

5 Computational Concerns: Computability, Tractability, and Boundedness

As the end of the previous section shows, even when a tokenizer model is consistent and all statistical concerns are taken into account, there are still computational aspects that can hinder the practice of tokenization. In this section, we turn to issues of computability, tractability, and boundedness.

Definitions 3.1 to 3.3 are general enough to allow for all kinds of encoding and decoding functions, including **uncomputable** ones. Consider the following example:

Example 5.1. *Let $\Sigma = \Delta = \{0, 1\}$, and define $\mathcal{T}_{unc} = (\tau_{unc}, \kappa_{unc})$ as a deterministic model in the following way:*

$$\tau_{unc}(\boldsymbol{\sigma}) = \begin{cases} \boldsymbol{\sigma} \cdot 1, & \text{if } \boldsymbol{\sigma} \text{ describes a valid Turing Machine followed by an input for which it halts.} \\ \boldsymbol{\sigma} \cdot 0, & \text{otherwise.} \end{cases} \qquad \kappa_{unc}(\boldsymbol{\delta}) = \begin{cases} \varepsilon, & \text{if } \boldsymbol{\delta} \in \Delta. \\ \boldsymbol{\sigma}, & \text{otherwise, where } \boldsymbol{\delta} = \boldsymbol{\sigma} \cdot \delta. \end{cases}$$

Significantly, $\mathcal{T}_{unc}$ is not only well-defined but also exact and therefore consistent for any language model p over Σ^*. However, τ_{unc} is famously an uncomputable function, and hence $\mathcal{T}_{unc}$ is an uncomputable tokenizer.

Even if a tokenizer model is computable, its **tractability** is important. Indeed, there are many reasons that could make the computation of tokenization intractable. Many of the operations defining tokenizer models involve sums over infinite sets. This is particularly true for the composition of stochastic maps whenever it is performed over an infinite domain, as in our case. Therefore, it is crucial to assess the tractability not only of τ and κ, but also of their composition $\kappa\tau$.

We have seen that when a tokenizer model is exact, τ is a section for κ, and, therefore, $\tau(\boldsymbol{\sigma})$ concentrates the probability mass on only a subset of Δ^*, namely the subset of all preimages of $\boldsymbol{\sigma}$ by κ. This property can help reduce the computational costs by restricting the sum only to that subset. We consider the following property:

Definition 5.1. *We say a tokenizer model $\mathcal{T} = (\tau, \kappa)$ is* ***multiplicative*** *if its decoder κ respects the concatenation products; that is, if $\kappa(\boldsymbol{\delta}' \cdot \boldsymbol{\delta}'') = \kappa(\boldsymbol{\delta}') \cdot \kappa(\boldsymbol{\delta}'')$.*

An obvious consequence of multiplicative tokenizers is that decoding preserves the prefix structure of token sequences. More precisely, let $\boldsymbol{\delta}' \preceq \boldsymbol{\delta}$ denote the fact that $\boldsymbol{\delta} = \boldsymbol{\delta}' \cdot \boldsymbol{\delta}''$ for $\boldsymbol{\delta}, \boldsymbol{\delta}', \boldsymbol{\delta}'' \in \Delta^*$ (and likewise for $\boldsymbol{\sigma}, \boldsymbol{\sigma}', \boldsymbol{\sigma}'' \in \Sigma^*$). Then we have that $\boldsymbol{\delta}' \preceq \boldsymbol{\delta}$ implies $\kappa(\boldsymbol{\delta}') \preceq \kappa(\boldsymbol{\delta})$. This property is crucial in autoregressive models, where each token in a sequence depends only on the tokens preceding it and can then be computed in a left-to-right fashion. In these cases, multiplicativity ensures that the model's output can be decoded on the fly, excluding decoding functions such as string reversal.

Definition 5.2. *We say the* ***kernel*** *of a multiplicative tokenizer's decoder κ is* ***trivial*** *if κ maps nonempty token sequences to nonempty token sequences (i.e., if $\boldsymbol{\delta} \neq \varepsilon_\Delta$ then $\kappa(\boldsymbol{\delta}) \neq \varepsilon_\Sigma$).*

The most commonly used tokenizers, including BPE, WordPiece, and Unigram, are multiplicative. Notice that, for a kernel of a multiplicative tokenizer's decoder to be trivial, it is enough that $\kappa(\delta) \neq$

ε_Σ for any $\delta \in \Delta$. This implies that token sequences do not include special tokens that are erased during decoding, e.g., padding or end-of-sentence tokens. When a multiplicative tokenizer's decoder has a trivial kernel, a decoded text cannot be shorter than the token sequence from which it has been decoded. This simple observation guarantees that the number of preimages of a text through κ will be finite. More precisely,

Proposition 5.1. *Let $\mathcal{T} = (\tau, \kappa)$ be a multiplicative tokenizer model whose decoder's kernel is trivial. If $\kappa(\delta_1 \shortmid \delta_2 \shortmid \ldots \shortmid \delta_m) = \sigma_1 \cdot \sigma_2 \cdot \ldots \cdot \sigma_n$ then $m \leq n$.*

Proof. We can reason by induction. The property is true when $m = 1$ since 1 is the minimun length of any possible image of κ. Assume it is true for $m = k$ and let $\boldsymbol{\delta} = \delta_1 \shortmid \ldots \shortmid \delta_k \shortmid \delta_{k+1}$. Then $\kappa(\boldsymbol{\delta}) = \kappa(\delta_1 \shortmid \ldots \shortmid \delta_k) \cdot \kappa(\delta_{k+1}) = \sigma_1 \cdot \ldots \cdot \sigma_r \cdot \sigma_{r+1} \cdot \ldots \cdot \sigma_{r+s}$, where $\kappa(\delta_{k+1}) = \sigma_{r+1} \cdot \ldots \cdot \sigma_{r+s}$. Since $r \geq k$ and $s \geq 1$, we have that $r + s \geq k + 1$. ☐

Corollary 5.0.1. *Let $\mathcal{T} = (\tau, \kappa)$ be a multiplicative tokenizer model whose decoder's kernel is trivial. Then for any text $\boldsymbol{\sigma}$, the set $\kappa^{-1}(\boldsymbol{\sigma})$ is finite.*

Proposition 5.1 guarantees that, if no token in the vocabulary is mapped to the empty string, then the length of every preimage of a text $\boldsymbol{\sigma}$ of length n has length less than or equal to n. So the number of elements in $\kappa^{-1}(\boldsymbol{\sigma})$ is bounded by $\sum_{i=1}^{n} |\Delta|^i$. Since this bound is exponential in n, indirect methods, such as Cao & Rimell's (2021) mentioned above, should be developed to approximate the correct computation of a tokenizer's encoding and decoding in such a way that the consistency of the language model is not compromised. Ideally, the complexity a tokenizer model (that is, of the composition $\kappa\tau$) should be at most linear, in the length of the input text. By placing all the probability mass on one token sequence, $\kappa\tau$ in exact deterministic tokenizers, such as BPE and WordPiece, can be computed in linear time as long as τ and κ can be computed in linear time. However, rigorously handling spurious ambiguity still represents a challenge in these cases.

Even though multiplicativity ensures that the length of all the preimages of $\kappa^{-1}(\boldsymbol{\sigma})$ is bounded by the length of $\boldsymbol{\sigma}$, texts themselves may have unbounded length. In practice, the bounded character of tokenization is secured externally, namely by fixing a hyperparameter that artificially limits the length of input texts. However, it can be desirable to address boundedness as an internal property of a tokenizer. For this reason, we introduce the following definitions:

Definition 5.3. *Given two sets A, B, and a map $f\colon A^* \to B^*$ we say f is of **finite type** if, there exists $n \in \mathbb{N}$ such that, for every $\boldsymbol{a} \in A^*$, there exists a nonempty $\boldsymbol{a}' \in A^{\leq n}$ such that $\boldsymbol{a} = \boldsymbol{a}' \cdot \boldsymbol{a}''$ and $f(\boldsymbol{a}) = f(\boldsymbol{a}') \cdot f(\boldsymbol{a}'')$.*

Definition 5.4. *A tokenizer model $\mathcal{T} = (\tau, \kappa)$ is called **bounded** if both τ and κ are of finite type.*

If a tokenizer is bounded, it means the input can be decomposed into bounded components. The "maximal munch" approach (Reps, 1998; Palmer, 2000) adopted by WordPiece, for instance, iteratively maps the successive longest prefixes of a text to tokens in the vocabulary. WordPiece's encoder is thus bounded by the maximum length of the preimages of Δ through τ, and can therefore be implemented as a finite-state transducer (Song et al., 2021). Notice, however, that, in general, if a tokenizer is bounded in the sense of Definitions 5.3 and 5.4, it does not mean a tractable algorithm exists to find the decomposition $\boldsymbol{a} = \boldsymbol{a}' \cdot \boldsymbol{a}''$, which could, in principle, depend on the entire $\boldsymbol{a}$. Significantly, Berglund & van der Merwe (2023) showed that, under specific conditions on the structure of the list of rules or "merges", BPE is also bounded and an algorithm can be found to compute $\tau(\boldsymbol{\sigma}') \shortmid \tau(\boldsymbol{\sigma}'') = \tau(\boldsymbol{\sigma})$ in an online fashion with finite lookahead. Moreover, Berglund et al. (2024) proposed an algorithm for constructing deterministic finite automata representing BPE's encoder.

6 Conclusions

In this work, we have proposed a framework to lay the formal foundations of tokenization in natural language processing. Relying on the basic properties of the category of stochastic maps, we proposed a general definition of a tokenizer as an arbitrary pair of composable maps and proved the necessary and sufficient condition for a tokenizer to preserve the consistency of estimators. Based on this framework, we addressed several statistical and computational concerns crucial for the design and implementation of tokenizers. We believe this framework will contribute to establishing and developing theoretical and practical aspects of neural language modeling on solid grounds.

7 Limitations

Statistical and computational concerns are not the only concerns relevant to a foundational approach to tokenization. In particular, this paper does not address structural concerns, i.e., structural properties of the sets Σ^* and Δ^* such as the monoidal structure, the preservation of structural features through τ and κ, or the important question of the choice of Δ. It also leaves unaddressed theoretical concerns related to interpretability and the possible relation to linguistic segmentation. Although the perspective adopted here is purely formal, and as such, self-contained, the framework proposed could benefit from the insights given by experimental results. These issues will be the object of future work.

8 Acknowledgments

The authors thank Saeed Zakeri for pointing out an interesting interplay between a general fact about the Banach spaces $L^1(X)$ for any measure space X and the fact that for the little l^p spaces, the l^∞ norm is smaller than the l^1 norm, which led to the proof of Lemma 3.1. The authors also thank Li Du for useful feedback at the early stages of this work.

References

Ben Athiwaratkun, Shiqi Wang, Mingyue Shang, Yuchen Tian, Zijian Wang, Sujan Kumar Gonugondla, Sanjay Krishna Gouda, Rob Kwiatowski, Ramesh Nallapati, and Bing Xiang. Token alignment via character matching for subword completion, 2024.

John C. Baez and Tobias Fritz. A Bayesian characterization of relative entropy. *CoRR*, abs/1402.3067, 2014. URL http://arxiv.org/abs/1402.3067.

Martin Berglund and Brink van der Merwe. Formalizing BPE tokenization. In Benedek Nagy and Rudolf Freund (eds.), *Proceedings of the 13th International Workshop on, Non-Classical Models of Automata and Applications, Famagusta, North Cyprus, 18th-19th September, 2023*, volume 388 of *Electronic Proceedings in Theoretical Computer Science*, pp. 16–27. Open Publishing Association, 2023. doi: 10.4204/EPTCS.388.4.

Martin Berglund, Willeke Martens, and Brink van der Merwe. Constructing a BPE tokenization DFA, 2024. URL https://arxiv.org/abs/2405.07671.

Joan L. Bybee and Paul J. Hopper (eds.). *Frequency and the emergence of linguistic structure*. Number 45 in Typological studies in language. Benjamins, Amsterdam, 2001. ISBN 978-90-272-2947-2 978-1-58811-027-5 978-1-58811-028-2 978-90-272-2948-9. OCLC: 216471429.

Kris Cao and Laura Rimell. You should evaluate your language model on marginal likelihood over tokenisations. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 2104–2114, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.161. URL https://aclanthology.org/2021.emnlp-main.161.

Nick Chater, Alexander Clark, John A. Goldsmith, and Amy Perfors. *Empiricism and language learnability*. Oxford University Press, Oxford, United Kingdom, first edition edition, 2015. ISBN 978-0-19-873426-0. OCLC: ocn907131354.

Nadezhda Chirkova, Germán Kruszewski, Jos Rozen, and Marc Dymetman. Should you marginalize over possible tokenizations? In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 1–12, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-short.1. URL https://aclanthology.org/2023.acl-short.1.

Noam Chomsky. *Syntactic Structures*. Mouton and Co., The Hague, 1957.

Noam Chomsky. *Aspects of the Theory of Syntax*. The MIT Press, Cambridge, 1965.

Mathias Creutz and Krista Lagus. Unsupervised discovery of morphemes. In *Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning*, pp. 21–30. Association for Computational Linguistics, July 2002. doi: 10.3115/1118647.1118650. URL https://aclanthology.org/W02-0603.

Jeffrey L. Elman (ed.). *Rethinking innateness: A connectionist perspective on development*. Neural network modeling and connectionism. MIT Press, Cambridge, Mass, 1996. ISBN 978-0-262-05052-4.

Philip Gage. A new algorithm for data compression. *C Users J.*, 12(2):23–38, feb 1994. ISSN 0898-9788. URL https://dl.acm.org/doi/abs/10.5555/177910.177914.

Jin Guo. Critical tokenization and its properties. *Computational Linguistics*, 23(4):569–596, 1997. URL https://aclanthology.org/J97-4004.

Zellig Harris. *Structural linguistics*. University of Chicago Press, Chicago, 1960. ISBN 0226317714 0226217714.

Louis Hjelmslev. *Résumé of a Theory of Language*. Number 16 in Travaux du Cercle linguistique de Copenhague. Nordisk Sprog-og Kulturforlag, Copenhagen, 1975. ISBN 0-299-07040-9.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Third edition draft edition, 2024. URL https://web.stanford.edu/~jurafsky/slp3/.

Carina Kauf and Anna Ivanova. A better way to do masked language model scoring. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 925–935, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-short.80. URL https://aclanthology.org/2023.acl-short.80.

Paul Kiparsky. On the architecture of pāṇini's grammar. In Gérard Huet, Amba Kulkarni, and Peter Scharf (eds.), *Sanskrit Computational Linguistics*, pp. 33–94, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-00155-0.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In Sophia Ananiadou (ed.), *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pp. 177–180, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL https://aclanthology.org/P07-2045.

Taku Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates. In Iryna Gurevych and Yusuke Miyao (eds.), *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 66–75, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1007. URL https://aclanthology.org/P18-1007.

Brian MacWhinney (ed.). *The emergence of language*. Carnegie Mellon symposia on cognition. Lawrence Erlbaum Associates, Mahwah, NJ, 1999. ISBN 978-0-8058-3010-1 978-0-8058-3011-8.

Alec Marantz. Words. *WCCFL XX Handout, USC*, 2001.

Anthony M. McEnery and Anita Wilson. *Corpus Linguistics: An Introduction.* Edinburgh University Press, Edinburgh, 2001.

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. Between words and characters: A brief history of open-vocabulary modeling and tokenization in NLP. *CoRR*, abs/2112.10508, 2021. URL https://arxiv.org/abs/2112.10508.

Piotr Nawrot, Jan Chorowski, Adrian Lancucki, and Edoardo Maria Ponti. Efficient transformers with dynamic token pooling. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6403–6417, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.353. URL https://aclanthology.org/2023.acl-long.353.

David D. Palmer. Tokenisation and sentence segmentation. In Robert Dale, Hermann Moisl, and Harold Somers (eds.), *Handbook of Natural Language Processing*, chapter 2, pp. 24–25. Marcel Dekker, 2000.

Plato. Cratylus. In *Plato in Twelve Volumes*, volume 4. W. Heinemann, Harvard University Press, London, Cambridge, 1926.

Gabriel Poesia, Oleksandr Polozov, Vu Le, Ashish Tiwari, Gustavo Soares, Christopher Meek, and Sumit Gulwani. Synchromesh: Reliable code generation from pre-trained language models, 2022.

Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. BPE-dropout: Simple and effective subword regularization. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 1882–1892, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.170. URL https://aclanthology.org/2020.acl-main.170.

Nived Rajaraman, Jiantao Jiao, and Kannan Ramchandran. Toward a theory of tokenization in LLMs, 2024.

Thomas Reps. "maximal-munch" tokenization in linear time. *ACM Trans. Program. Lang. Syst.*, 20(2):259–273, mar 1998. ISSN 0164-0925. doi: 10.1145/276393.276394. URL https://doi.org/10.1145/276393.276394.

Julian Salazar, Davis Liang, Toan Q. Nguyen, and Katrin Kirchhoff. Masked language model scoring. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 2699–2712, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.240. URL https://aclanthology.org/2020.acl-main.240.

Ferdinand de Saussure. *Deuxième cours de linguistique generale (1908-1909) : d'après les cahiers d'Albert Riedlinger et Charles Patois=Saussure's second course of lectures on general linguistics (1908-1909) : From the notebooks of Albert Riedlinger and Charles Patois*. Language & communication library. Pergamon, Amsterdam, Netherlands, dec 1997.

Craig W. Schmidt, Varshini Reddy, Haoran Zhang, Alec Alameddine, Omri Uzan, Yuval Pinter, and Chris Tanner. Tokenization is more than compression, 2024.

Mike Schuster and Kaisuke Nakajima. Japanese and Korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5149–5152, 2012. doi: 10.1109/ICASSP.2012.6289079.

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

Rama Nath Sharma. *The Aṣṭādhyāyī of Pāṇini*. Munshiram Manoharlal, New Delhi, 2002. 6 vols.

Xinying Song, Alex Salcianu, Yang Song, Dave Dopson, and Denny Zhou. Fast WordPiece tokenization. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 2089–2103, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.160. URL https://aclanthology.org/2021.emnlp-main.160.

Omri Uzan, Craig W. Schmidt, Chris Tanner, and Yuval Pinter. Greed is all you need: An evaluation of tokenizer inference methods, 2024.

Yonghui Wu, Mike Schuster, Z. Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason R. Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Gregory S. Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *ArXiv*, abs/1609.08144, 2016. URL https://api.semanticscholar.org/CorpusID:3603249.

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 (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 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 Gastaldi, Li Du, Tim Vieira, Mrinmaya Sachan, and Ryan Cotterell. A formal perspective on byte-pair encoding. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 598–614, Toronto, Canada, July 2023b. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.38. URL https://aclanthology.org/2023.findings-acl.38.