Causal Estimation of Tokenisation Bias

Pietro Lesci[C][✉] Clara Meister[ETH] Thomas Hofmann[ETH] Andreas Vlachos[C][✉] Tiago Pimentel[ETH][✉]
[C]University of Cambridge [ETH]ETH Zürich
[✉]{pl487, av308}@cam.ac.uk, tiago.pimentel@inf.ethz.ch

pietrolesci/tokenisation-bias pietrolesci/Tokenisation-Bias

Abstract

Modern language models are typically trained over subword sequences, but ultimately define probabilities over character-strings. Ideally, the choice of the tokeniser—which maps character-strings to subwords—should not affect the probability assigned to the underlying character-string; in practice, it does. We define this mismatch as **tokenisation bias**. In this work, we quantify one particular type of tokenisation bias: the effect of including or not a subword (e.g., $\langle hello \rangle$) in a tokeniser's vocabulary on the probability a trained model assigns to the corresponding characters (i.e., "*hello*"). Estimating this effect is challenging because each model is trained with only one tokeniser. We address this by framing tokenisation bias as a causal effect and estimating it using the regression discontinuity design. Specifically, we exploit the fact that tokenisation algorithms rank subwords and add the first K to a tokeniser's vocabulary, where K is an arbitrary cutoff point. As such, we can estimate a causal effect by comparing similar subwords around this cutoff. Experimentally, we find that tokenisation consistently affects models' outputs across scales, vocabularies, and tokenisers. Notably, a subword's presence in a small model's vocabulary may increase its characters' probability by up to 17 times, highlighting tokenisation as a key design choice in language modelling.

1 Introduction

Language models (LMs) define probability distributions over **character-strings** (e.g., "*hello*"), i.e., finite sequences of characters[1] from an alphabet. Directly modelling character-strings, however, can be inefficient, as it might require processing long sequences. To improve computational efficiency, modern LMs (e.g., Touvron et al., 2023) typically model **subwords-strings** instead (e.g., $\langle he, llo \rangle$);

[1]We use characters as a generic term to refer to either raw bytes (e.g., in ASCII), Unicode symbols, or graphemes.

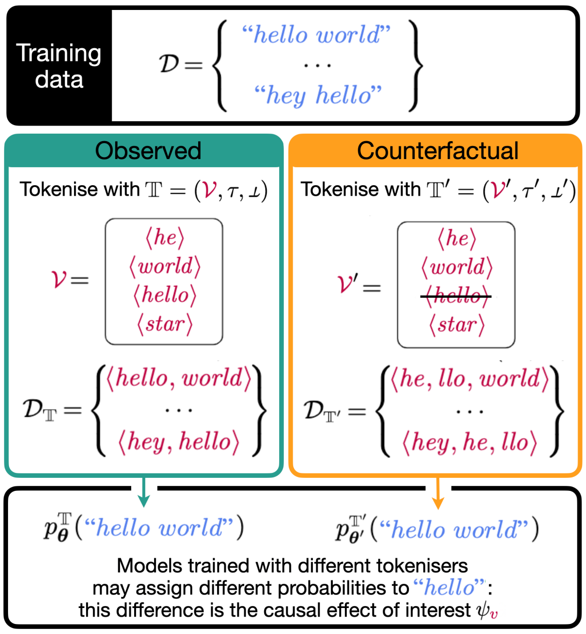
Training data
$\mathcal{D} = \{$ "hello world" ... "hey hello" $\}$
Observed
Tokenise with $\mathbb{T} = (\mathcal{V}, \tau, \lrcorner)$
$\mathcal{V} =$ ⟨he⟩ ⟨world⟩ ⟨hello⟩ ⟨star⟩
$\mathcal{D}_{\mathbb{T}} = \{ \langle hello, world \rangle \ldots \langle hey, hello \rangle \}$
Counterfactual
Tokenise with $\mathbb{T}' = (\mathcal{V}', \tau', \lrcorner')$
$\mathcal{V}' =$ ⟨he⟩ ⟨world⟩ ~~⟨hello⟩~~ ⟨star⟩
$\mathcal{D}_{\mathbb{T}'} = \{ \langle he, llo, world \rangle \ldots \langle hey, he, llo \rangle \}$
$p^{\mathbb{T}}_{\theta}$("hello world")
$p^{\mathbb{T}'}_{\theta'}$("hello world")
Models trained with different tokenisers may assign different probabilities to "hello": this difference is the causal effect of interest ψ_v

Figure 1: **Tokenisation bias.** Consider subword $\langle hello \rangle$. If it is included in the vocabulary, then each occurrence of "*hello*" in the training data is represented as a single subword; otherwise, it is split into two subwords, i.e., $\langle he, llo \rangle$. If two models are trained under these two settings, the difference in the probability they assign to "*hello*" is the tokenisation bias we aim to estimate.

these subword-strings are produced by a **tokeniser**, where each subword represents a sequence of characters. While in practice LMs thus output distributions over subword-strings, we can still map them back to distributions over character-strings (Pimentel and Meister, 2024; Phan et al., 2025).

Notably, a tokeniser's behaviour is constrained by its **vocabulary**: a finite set of subwords which it may produce. Given a character-string, a tokeniser deterministically maps it to a sequence of subwords from its vocabulary. Different tokenisers (with different vocabularies) thus map character-strings to different subword-strings (e.g., map "*hello*" to either $\langle he, llo \rangle$ or $\langle hello \rangle$). While an ideal LM would assign the same probability to a string regardless of tokenisation (§3.2), in practice, tokenisation is an influential design choice for model performance (Rust et al., 2021; Toraman et al., 2023; Ali et al., 2024, *inter alia*). How tokenisation affects trained models, however, remains poorly understood.

In this work, we first define the effect of a tokenisation choice on model behaviour as **tokenisation bias**. We then quantify one particular type of tokenisation bias: the effect of including or not a subword (e.g., $\langle hello \rangle$) in a tokeniser's vocabulary on the log-probability a trained model assigns to its characters (i.e., "$hello$"). While tokenisation bias may be an intuitive concept, estimating it is not straightforward. We cannot simply compare a model's log-probabilities on in- *vs.* out-of-vocabulary words (e.g., "$hello$" *vs.* "$appoggiatura$"); as vocabularies are built based on, e.g., frequency, there are systematic differences between these groups. Further, we cannot compare log-probabilities on the same word under different tokenisations (e.g., $\langle he, llo \rangle$ or $\langle hello \rangle$); as the model only sees one of them during training.[2]

To quantify tokenisation bias, we first frame it as a **causal effect**, asking: how would a word's log-probability change if its subword was removed from the vocabulary (see Fig. 1)? This question requires comparing two values: (i) an **observed value**, the log-probability the model assigns the word in question; (ii) a **counterfactual value**, the log-probability the model *would* assign the word had it not been in the vocabulary. We estimate this causal effect by noting that tokenisers' vocabularies are typically built incrementally. This effectively creates a ranking over subwords, which is arbitrarily cut off by a fixed vocabulary size: e.g., the first 32k subwords under this ranking are included in the vocabulary; later ones are excluded. Regression discontinuity (RD) design (Thistlewaite and Campbell, 1960) then gives us a principled way to estimate our causal effect: we can compare subwords near the cutoff—as they are bound to have similar features—using subwords before the cutoff to estimate our observed values, and the ones after the cutoff to input the counterfactual value.

Experimentally, we show that tokenisation consistently affects LM performance across vocabulary sizes, model scales, and tokenisers. Text represented as a single subword receives more log-probability (on average) than if split into two. Further, this bias grows during training and in models under 100M parameters, it reaches 2.88 nats for BPE and 2.51 for WP. In 850M-parameter models, this bias is around 1 nat: a subword's characters would thus be assigned roughly 2.7 times less probability if it were not in the tokeniser's vocabulary.

[2] A brute-force approach—e.g., training separate models under the different possible tokenisers—is (i) computationally impractical and (ii) estimates expected bias for a LM family, rather than the tokenisation bias in a specific model instance.

2 Tokenisation

Let $c \in \Sigma$ be a **character**, where Σ is an alphabet representing either a set of bytes, or the graphemes in a language (including whitespace and punctuation). Further, let $\mathbf{c} \in \Sigma^*$ be a **character-string**, i.e., a finite sequences of characters; we will represent these strings as $\mathbf{c} = \text{“}c_1 c_2 \cdots c_{|\mathbf{c}|}\text{”}$. The goal of language modelling is to learn a language's probability distribution over character-strings. We write these probabilities autoregressively as:

$$p(\mathbf{c}) = p(\mathsf{eos} \mid \mathbf{c}) \prod_{t=1}^{|\mathbf{c}|} p(c_t \mid \mathbf{c}_{<t}) \qquad (1)$$

where $\mathsf{eos} \notin \Sigma$ is a special symbol denoting the end of a string.[3] Most modern LMs, however, do not model eq. (1) directly. Rather, they model distributions over **subword-strings** $\mathbf{v} \in \mathcal{V}^*$:

$$p(\mathbf{v}) = p(\mathsf{eos} \mid \mathbf{v}) \prod_{t=1}^{|\mathbf{v}|} p(v_t \mid \mathbf{v}_{<t}) \qquad (2)$$

In words, subword-strings are finite sequences of subwords $v \in \mathcal{V}$, where $\mathcal{V}$ is a finite set typically called a vocabulary; we thus represent these strings as $\mathbf{v} = \langle v_1, ..., v_{|\mathbf{v}|} \rangle$. As we explain later, subwords represent character spans, denoted here as $\mathbf{c}_v = v$. How do eqs. (1) and (2) connect? This is the job of a tokeniser, which we describe next.

2.1 Characters-strings ↔ Subword-strings

Tokenisers map character-strings to subword-strings and back, and can be formally defined as a tuple $\mathbb{T} \overset{\text{def}}{=} (\mathcal{V}, \tau, \perp)$. The first item in this tuple is a **vocabulary**: a finite set of character-spans $\mathcal{V} \subset \Sigma^+$, whose elements are called subwords $v \in \mathcal{V}$.[4] The second item is known as the **tokenisation function**: a function $\tau \colon \Sigma^* \to \mathcal{V}^*$ which maps character-strings to subword-strings. Finally, the third item in $\mathbb{T}$ is a **detokenisation function**: a function $\perp \colon \mathcal{V}^* \to \Sigma^*$ which maps subword-strings back to character-strings.

[3] The eos symbol is required to define probability distributions over strings autoregressively (Du et al., 2023).

[4] To ensure any character-string can be represented with $\mathcal{V}$, the original alphabet is typically included in this set: $\Sigma \subseteq \mathcal{V}$.

As subwords represent character-spans, detokenisation is typically defined as the simple concatenation of the subwords in a subword-string:

$$\perp(\mathbf{v}) \stackrel{\text{def}}{=} v_1 \circ v_2 \circ \cdots \circ v_{|\mathbf{v}|} \tag{3}$$

e.g., $\perp(\langle he, llo\rangle) = \text{“}hello\text{”}$. Further, to guarantee that a tokeniser is lossless, τ is designed such that detokenising its output returns the original character-string: i.e., $\mathbf{c} = \perp(\tau(\mathbf{c}))$. We describe two popular tokenisation functions in App. A.

2.2 Language Modelling

Language modelling is the task of defining a probability distribution over character-strings, arguably being the most prominent use case of tokenisation. During LM training, we typically start with a dataset of character-strings: $\mathcal{D} = \{\mathbf{c}_n\}_{n=1}^N$, where $\mathbf{c}_n \sim p(\mathbf{c})$. We then use a tokeniser $\mathbb{T}$ to convert each of these into a subword-string: $\mathbf{v}_n = \tau(\mathbf{c}_n)$, which are the actual inputs to our model. Given this setup, we can deduce a distribution over subword-strings from the tokeniser:

$$p^{\mathbb{T}}(\mathbf{v}) = \begin{cases} p(\mathbf{c}) & \text{if } \mathbf{v} = \tau(\mathbf{c}) \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where we denote this distribution as $p^{\mathbb{T}}$ now to make its dependence on $\mathbb{T}$ explicit. Eq. (4) thus connects eqs. (1) and (2). Notably, training an LM on strings $\mathbf{v}_n$ produces a model $p_{\boldsymbol{\theta}}^{\mathbb{T}}(\mathbf{v})$—which ideally should be a good approximation of $p^{\mathbb{T}}(\mathbf{v})$. We can then recover a character-level distribution $p_{\boldsymbol{\theta}}^{\mathbb{T}}(\mathbf{c})$ from this model by marginalising over all $\mathbf{v}$ which are detokenised to this character-string:

$$p_{\boldsymbol{\theta}}^{\mathbb{T}}(\mathbf{c}) = \sum_{\mathbf{v} \in \mathcal{V}^*} p_{\boldsymbol{\theta}}^{\mathbb{T}}(\mathbf{v})\; \mathbb{1}\{\mathbf{c} = \perp(\mathbf{v})\} \tag{5}$$

Notably, while this equation depends on an infinite sum over $\mathcal{V}^*$, it can be approximated efficiently (Phan et al., 2024, 2025; Vieira et al., 2024).

3 Tokenisation Bias

Now, consider tokeniser $\mathbb{T} = (\mathcal{V}, \tau, \perp)$ and assume we use it for training a LM; out data-generating distribution is thus $p^{\mathbb{T}}$ (from eq. (4)) and our LM is $p_{\boldsymbol{\theta}}^{\mathbb{T}}$. Further, consider a specific subword v in this tokeniser's vocabulary. We can measure how well the model predicts this subword's associated characters $\mathbf{c}_v$ in a given context $\mathbf{c}_{<t}$ as how large of a probability it assigns to this sequence of characters $p_{\boldsymbol{\theta}}^{\mathbb{T}}(\mathbf{c}_v \mid \mathbf{c}_{<t})$. We are interested in studying how this probability would change had v not been in $\mathcal{V}$. More generally, we can ask: how does our choice of tokeniser $\mathbb{T}$ bias our trained model $p_{\boldsymbol{\theta}}^{\mathbb{T}}$? We term this tokenisation bias.[5]

Definition 1. *The **tokenisation bias** induced by a* property of the tokeniser *is its effect on a model's ability to predict some character-string* $\mathbf{c}$.

We can define different *kinds* of tokenisation bias, considering different properties of a tokeniser. The property we will focus on here is the inclusion of a subword in its vocabulary, i.e., $v \in \mathcal{V}$, and how it affects predictions on its characters, $\mathbf{c}_v$.

Definition 2. *The **tokenisation bias of** $v \in \mathbb{T}$ is its effect on a model's ability to predict* $\mathbf{c}_v$.[6]

We now frame this tokenisation bias as a **causal effect**: it compares the performance of an *observed* model $p_{\boldsymbol{\theta}}^{\mathbb{T}}$, with the performance of a *counterfactual* model $p_{\boldsymbol{\theta}'}^{\mathbb{T}'}$. This counterfactual model represents what our model *would* have been if trained with tokeniser $\mathbb{T}' = (\mathcal{V}', \tau', \perp')$ which did not include subword v. We define this causal effect formally in the next section. First, however, note that:

$$\underbrace{v = \tau(\mathbf{c}_v),}_{\text{observed subword}} \qquad \underbrace{\mathbf{v}'^{\mathbf{c}_v} = \tau'(\mathbf{c}_v)}_{\text{counterfactual subwords}} \tag{6}$$

where $\mathbf{v}'^{\mathbf{c}_v}$ is the sequence of subwords which would be used to represent $\mathbf{c}_v$ in the counterfactual case. Assuming no tokenisation mismatch (Phan et al., 2024), i.e., that $\tau(\mathbf{c}_{<t})$ is a prefix of $\tau(\mathbf{c})$ in this context (and similarly under τ'), we then have:

$$p_{\boldsymbol{\theta}}^{\mathbb{T}}(\mathbf{c}_v \mid \mathbf{c}_{<t}) = p_{\boldsymbol{\theta}}^{\mathbb{T}}(v \mid \tau(\mathbf{c}_{<t})) \tag{7}$$

$$p_{\boldsymbol{\theta}'}^{\mathbb{T}'}(\mathbf{c}_v \mid \mathbf{c}_{<t}) = \prod_{t'=1}^{|\mathbf{v}'^{\mathbf{c}_v}|} p_{\boldsymbol{\theta}'}^{\mathbb{T}'}(\mathbf{v}'^{\mathbf{c}_v}_{t'} \mid \tau'(\mathbf{c}_{<t}) \circ \mathbf{v}'^{\mathbf{c}_v}_{<t'})$$

In words, we can compute $p_{\boldsymbol{\theta}'}^{\mathbb{T}'}(\mathbf{c}_v \mid \mathbf{c}_{<t})$ by multiplying the probabilities of subwords in $\mathbf{v}'^{\mathbf{c}_v}$.

3.1 A Subword's Causal Effect

We now define tokenisation bias in terms of potential outcomes (Rubin, 1974, 2005); this framework allows us to formally describe the causal effect of a **treatment** on some target **outcome**. In our case,

[5] We note that Phan et al. (2024) use the term tokenisation bias to describe an issue arising when one applies τ to a prefix of $\mathbf{c}$: namely, when $\tau(\mathbf{c}_{<t})$ is not necessarily a prefix of $\tau(\mathbf{c})$. We interpret this not as a bias, but as a misuse of tokenisers by the LM's users. We will thus term Phan et al.'s analysed phenomenon tokenisation mismatch here, instead of bias.

[6] We use the shorthand notation $v \in \mathbb{T}$ to denote that v is in the tokeniser's vocabulary $\mathcal{V}$.

we want to estimate the causal effect of including a subword v in tokeniser $\mathbb{T}$ on a model $p_{\boldsymbol{\theta}}^{\mathbb{T}}$'s ability to predict the character-string $\mathbf{c}_v$. Before defining the effect of interest, however, we must formally introduce its treatment and outcome. We define a **treatment assignment variable** as:

$$W_v \overset{\text{def}}{=} \mathbb{1}\{v \in \mathbb{T}\} \tag{8}$$

and we define a **potential outcome variable** as:

$$Y_W(v) \overset{\text{def}}{=} \mathop{\mathbb{E}}_{\mathbf{c}_{<t}} \left[\log p_{\boldsymbol{\theta}}^{\mathbb{T}}(\mathbf{c}_v \mid \mathbf{c}_{<t})\right] \tag{9}$$

where we quantify a model's ability to predict a character-string $\mathbf{c}_v$ as its log-probability averaged across contexts $\mathbf{c}_{<t}$.[7] Given these definitions, we can write the causal effect of interest as follows.

Definition 3. *The **causal effect of a subword** v on a model's ability to predict character-string $\mathbf{c}_v$ is:*

$$\psi_v \overset{\text{def}}{=} \underbrace{Y_1(v)}_{\substack{\textit{performance on } \mathbf{c}_v \\ \textit{when trained with } v}} - \underbrace{Y_0(v)}_{\substack{\textit{performance on } \mathbf{c}_v \\ \textit{when trained without } v}} \tag{10}$$

Importantly, a model is trained with a tokeniser that either includes v or does not. Thus, for any model, we can only observe either $Y_1(v)$ or $Y_0(v)$ with the other term being counterfactual. The causal analysis literature provides us with methods—and the relative assumptions that need to be met—to impute the counterfactual term from observable outcomes in a principled way. Definition 3 represents an individual treatment effect (ITE), as it defines the causal effect of a specific subword. Methods to estimate an ITE, however, require strong assumptions, if at all applicable (Lu et al., 2018). To avoid this requirement, we thus focus on average effects, as is common in the econometrics literature (Angrist and Pischke, 2015).

Definition 4. *The **expected effect of a subword** on the model's ability to predict its character-string is:*

$$\psi \overset{\text{def}}{=} \mathop{\mathbb{E}}_{v} [Y_1(v) - Y_0(v)] \tag{11}$$

In other words, we simplify the causal estimation problem by focusing on the expected causal effect across a population of subwords. The causal effect for each specific subword v might thus be larger or smaller than this expected effect; the expected effect we measure, however, gives us a best guess (in terms of mean squared error) of what their effect would be, given no additional information.

[7] Our framework also generalises beyond mean effects across contexts. For instance, replacing the expectation in eq. (9) with a standard deviation yields the tokenisation bias of $v \in \mathbb{T}$ on a model's variability when predicting $\mathbf{c}_v$. Similarly, defining $Y_W^{\mathbf{c}_{<t}}(v) = \log p_{\boldsymbol{\theta}}^{\mathbb{T}}(\mathbf{c}_v \mid \mathbf{c}_{<t})$ allows us to study a context-specific tokenisation bias instead of aggregated ones.

3.2 Tokenisation Bias in Perfect and in Untrained Models

We now analyse this causal effect under **perfect models**: LMs for which $p_{\boldsymbol{\theta}}^{\mathbb{T}}$ exactly matches the data-generating distribution $p^{\mathbb{T}}$.

Theorem 1. *Assume we have a training process that—regardless of the choice of tokeniser $\mathbb{T}$—always returns perfect models, i.e., models for which $p_{\boldsymbol{\theta}}^{\mathbb{T}}(\mathbf{v}) = p^{\mathbb{T}}(\mathbf{v})$. In this case, the causal effect of a subword v is always zero, i.e., $\psi_v = 0$.*

Proof. See the proof in App. B.1. $\square$

If the ideal effect is always zero, why bother estimating it? In practice, LMs are imperfect, and biases—both inductive and learned—may lead models to assign different probabilities to the same character-string under different tokenisations (all else held equal). Measuring this effect in practice is thus important. As an example, we show that at initialisation this causal effect is relatively large.

Theorem 2. *Assume that at initialisation our language model outputs the uniform distribution $p_{\boldsymbol{\theta}}^{\mathbb{T}}(v \mid \mathbf{v}_{<t}) = \frac{1}{|\mathcal{V}|}$.[8] In this case, the causal effect of a subword v at initialisation is $\psi_v \gtrapprox \log |\mathcal{V}|$.*

Proof. See the proof in App. B.2. $\square$

4 Estimating Tokenisation Bias

Now, we get to estimating the causal effect we defined; this is typically done in three steps. First, we define a **causal estimand**, the theoretical quantity that represents the effect of interest; this is the value in Definition 4. Second, we rewrite this causal estimand in terms of quantities that we can actually observe, thus defining a **statistical estimand**. Finally, we define an **estimator**, a statistical procedure to approximate the statistical estimand.

Importantly, the causal estimand in eq. (11) is non-trivial to estimate, as it contains a counterfactual; this is exposed by a simple decomposition:

$$\psi = \underbrace{\mathop{\mathbb{E}}_{v} [Y_1(v)]}_{①} - \underbrace{\mathop{\mathbb{E}}_{v} [Y_0(v)]}_{②} \tag{12}$$

[8] See support for this in, e.g., Chang and Bergen (2022).

Notably, we can observe the potential outcomes in expectation ① only for subwords in the given tokeniser ($v \in \mathbb{T}$), while expectation ② is observable only for subwords not in that tokeniser ($v \notin \mathbb{T}$). Naively comparing a model's average performance in these two groups of in- and out-of-vocabulary items would return a biased estimate of the causal effect. This is due to the way tokenisers are developed: subwords are not randomly included in the vocabulary, but carefully selected under some objective. As a subword's assigned treatment is not random, the two subgroups are not comparable, and there may thus be confounders.

Luckily, a popular class of tokenisation algorithms—which includes byte-pair encoding (BPE, Sennrich et al., 2016) and WordPiece (WP, Schuster and Nakajima, 2012)—select subwords iteratively, one-at-a-time, to compose their vocabularies. This creates a ranking among subwords which deterministically defines their treatment assignment (i.e., whether $v \in \mathbb{T}$ or not); we will leverage the non-randomness in this treatment assignment to estimate our desired effect. We describe these methods next.

4.1 Sequential Vocabulary Construction

Let $\mathcal{D} = \{\mathbf{c}_n\}_{n=1}^{N}$ be a dataset of character-strings. Bottom-up tokenisers typically define an **objective function**, $\phi(v^{[1]}, v^{[2]}, \mathcal{D})$, which measures the benefit of including a new subword $v^{[\text{new}]} = v^{[1]} \circ v^{[2]}$ in the vocabulary. Given a target vocabulary size $K + |\Sigma|$, their algorithm initialises a vocabulary as $\mathcal{V}_0 = \Sigma$ and a dataset as $\mathcal{D}_0 = \mathcal{D}$. For K iterations, it then selects the pair of subwords $v_k^{[1]}, v_k^{[2]}$ which maximises objective ϕ over the current dataset, adds it to the vocabulary $\mathcal{V}_k = \mathcal{V}_{k-1} \cup \{v_k^{[\text{new}]}\}$, and applies it to $\mathcal{D}_k = \{\text{merge}_{v_k^{[1]} \circledcirc v_k^{[2]}}(\mathbf{v}) \mid \mathbf{v} \in \mathcal{D}_{k-1}\}$, where merge replaces consecutive occurrences of $v_k^{[1]}, v_k^{[2]}$ with $v_k^{[\text{new}]}$. The algorithm then returns $\mathcal{V}_K$.

While the choice of objective function is arbitrary, most tokenisers employed by current LMs use either the objective from BPE or WP:

$$\phi_{\text{bpe}}(v^{[1]}, v^{[2]}, \mathcal{D}_k) \overset{\text{def}}{=} \#(\langle v^{[1]}, v^{[2]} \rangle, \mathcal{D}_k) \tag{13}$$

$$\phi_{\text{wp}}(v^{[1]}, v^{[2]}, \mathcal{D}_k) \overset{\text{def}}{=} \frac{\#(\langle v^{[1]}, v^{[2]} \rangle, \mathcal{D}_k)}{\#(\langle v^{[1]} \rangle, \mathcal{D}_k)\, \#(\langle v^{[2]} \rangle, \mathcal{D}_k)}$$

where we use $\#(\mathbf{v}, \mathcal{D})$ to denote the number of occurrences of subword-string $\mathbf{v}$ in dataset $\mathcal{D}$. At each iteration, BPE's objective (ϕ_{bpe}) selects the most frequent subword-pair in dataset $\mathcal{D}$, while WP's (ϕ_{wp}) selects the subword-pair with the largest pointwise mutual information in it.

4.2 The Regression Discontinuity Design

We define a principled estimator for tokenisation bias based on the regression discontinuity (RD) design (Thistlewaite and Campbell, 1960).[9] Let us consider running the algorithm above (§4.1) for $K_+ \gg K$ iterations and obtain a list of subwords:

$$[v_1^{[\text{new}]}, v_2^{[\text{new}]}, ..., v_K^{[\text{new}]}, ..., v_{K_+}^{[\text{new}]}] \tag{14}$$

The index $k \in \{1, ..., K_+\}$ of a subword in this sequence represents the iteration at which it would be selected by the tokenisation algorithm. Index k thus determines a natural ordering of how subwords are added to a bottom-up tokeniser. Now, let us denote it as an index-valued random variable γ_v; this variable deterministically defines a subword's treatment assignment:

$$W_v = \mathbb{1}\{\gamma_v \leq K\} \tag{15}$$

In words, a subword is thus included in the vocabulary if its index γ_v is smaller than K. In RD design, γ_v is termed a **running variable** and K is the **cutoff** of this running variable.

For RD design to be applicable, this cutoff must be predetermined and exogenous—i.e., independent of potential outcomes—ensuring an unbiased treatment assignment. This holds for bottom-up tokenisers, where vocabulary sizes are set beforehand. Notably, we still cannot directly compare in- and out-of-vocabulary subwords, as they have different values of the running variable: by definition these groups have, respectively, running variable values of $\gamma_v \leq K$ or $\gamma_v > K$. This makes them systematically different and introduces bias. The RD design addresses this issue by focusing on a **local treatment effect** instead:

$$\psi_{\gamma_v} \overset{\text{def}}{=} \mathop{\mathbb{E}}_{v} [Y_1(v) - Y_0(v) \mid \gamma_v] \tag{16}$$

Notably, ψ_{γ_v} is thus a localised causal effect representing the effect expected for subwords added to the vocabulary at a specific index γ_v.

We now leverage the cutoff point K to estimate this local causal effect. In short, since treatment assignment is deterministic and the cutoff is exogenous, this assignment mechanism is "as good as random" (Angrist and Pischke, 2015). If the outcome $Y(v)$ is a continuous function of the running

[9] RD design is a quasi-experimental paradigm that exploits a deterministic relationship between an observed characteristic, known as the running variable, and the treatment assignment. Unlike a true experiment, a quasi-experiment does not randomly assign participants to control and treatment groups.

variable, thus, it should transition smoothly across the cutoff in the absence of treatment, making any discontinuity at this point solely due to treatment.[10]

4.3 The Regression Discontinuity Estimand

Before introducing a statistical estimand, we need to introduce the functional relationship between potential outcomes and the running variable. We can write this formally as:

$$\mathbb{E}_{v}[Y_0(v) \mid \gamma_v] = f(\gamma_v) \tag{17a}$$
$$\mathbb{E}_{v}[Y_1(v) \mid \gamma_v] = f(\gamma_v) + \psi_{\gamma_v} \tag{17b}$$

where eq. (17b) immediately follows from eq. (16). As typical in RD, we posit a smooth relationship between the running variable and potential outcomes.

Assumption 1 (Continuity Assumption)**.** *The conditional expectation of a potential outcome is continuous at the cutoff. That is, both limits exist:*

$$\lim_{k \to K} \mathbb{E}_{v}[Y_0(v) \mid \gamma_v = k] \tag{18a}$$
$$\lim_{k \to K} \mathbb{E}_{v}[Y_1(v) \mid \gamma_v = k] \tag{18b}$$

Now, we note that eq. (15) implies that there is no value of the running variable for which we can *observe* both potential outcomes, where we write an **observed outcome** as:[11]

$$Y_{\text{obs}}(v) = \begin{cases} Y_1(v) & \text{if } W_v = 1 \\ Y_0(v) & \text{otherwise} \end{cases} \tag{19}$$

Luckily, Hahn et al. (2001) show that the only required assumption to identify the causal estimand in eq. (16) is that Assump. 1 holds. We can write a statistical estimand at the cutoff as:

$$\psi_{\text{RD}} = \lim_{k \to K^-} \mathbb{E}_{v}[Y_{\text{obs}}(v) \mid \gamma_v = k] - \lim_{k \to K^+} \mathbb{E}_{v}[Y_{\text{obs}}(v) \mid \gamma_v = k] \tag{20}$$

In words, the RD estimand is the discontinuity in the conditional expectation function at the cutoff.[12]

[10]Notably, our running variable γ_v is discrete, and thus not continuous by definition. In our experiments, we consider large enough windows around the cutoff and thus assume our variable is fine-grained enough to allow for such an estimation. Alternatively, we could have used other running variables for our experiments instead, such as, e.g., the objective function ϕ. We also note that, while the RD design typically assumes continuous running variables, it can be applied to discrete ones, such as the rank in our case (Cattaneo et al., 2024).

[11]Assuming consistency, i.e., assuming that a potential outcome under the observed treatment assignment equals the observed outcome (Cole and Frangakis, 2009).

[12]If we could estimate eq. (20)'s expectations, we would have a principled (non-parametric) estimator for ψ_{γ_v}. Obtaining such estimates, however, is tricky (Angrist and Pischke, 2009, chapter 6). First, working in a small neighbourhood of the cutoff means we would have little data. Second, sample averages are biased when estimated on one side of a boundary.

4.4 The Regression Discontinuity Estimator

In the limit of eq. (20), the functional terms $f(\gamma_v)$ in the observed effects cancel out. However, we do not have a large number of samples exactly at the cutoff. In practice, thus, RD is estimated by extrapolating over values of the running variable in a window around the cutoff; in this case, these functional terms $f(\gamma_v)$ do not cancel out. To control for them, we first take a conditional expectation of $Y_{\text{obs}}(v)$, which we write in terms of eq. (17):

$$\mathbb{E}_{v}[Y_{\text{obs}}(v) \mid \gamma_v] = f(\gamma_v) + \psi_{\gamma_v} W_v \tag{21}$$

We then assume a parametric form for f (e.g., linear in γ_v) and ψ_{γ_v} (e.g., a constant). Given a set of observed subwords $\mathcal{K}$ in a window around the cutoff, we obtain an estimator of ψ_{γ_v} by minimising the squared error of the functional fit of f and ψ_{γ_v}.

Estimator 1. *Given a parametric function f and a set of observed subwords around the cutoff $\mathcal{K}$, the **regression discontinuity estimator** $\widehat{\psi}_{\text{RD}}$ is:*

$$\operatorname*{argmin}_{f, \psi_{\text{RD}}} \sum_{v \in \mathcal{K}} (Y_{\text{obs}}(v) - f(\gamma_v) - \psi_{\text{RD}} W_v)^2 \tag{22}$$

This is an unbiased estimator of the local effect ψ_{γ_v} at $\gamma_v = K$ under Assump. 1 and assuming f provides an adequate description of expected potential effects (Angrist and Pischke, 2009).

Notably, the validity of an RD estimator hinges on the correct specification of f and on the size of the window used to obtain the set $\mathcal{K}$. On the one hand, a larger window allows for a bigger sample size but relies more on the correct estimation of f. On the other hand, a smaller window depends less strongly on f but may not have enough samples for precise statistical estimation. Thus, RD requires careful tuning of the window size to control this bias-variance trade-off. In Fig. 2, we show an example of this method in practice.

5 Experimental Setup

We conduct experiments on a suite of Llama-style transformer models of varying scales, trained with different tokenisation schemes and vocabulary sizes.[13] More details in App. C.

[13]Since our method requires knowing which subwords would be added after the cutoff, we could not use open-source

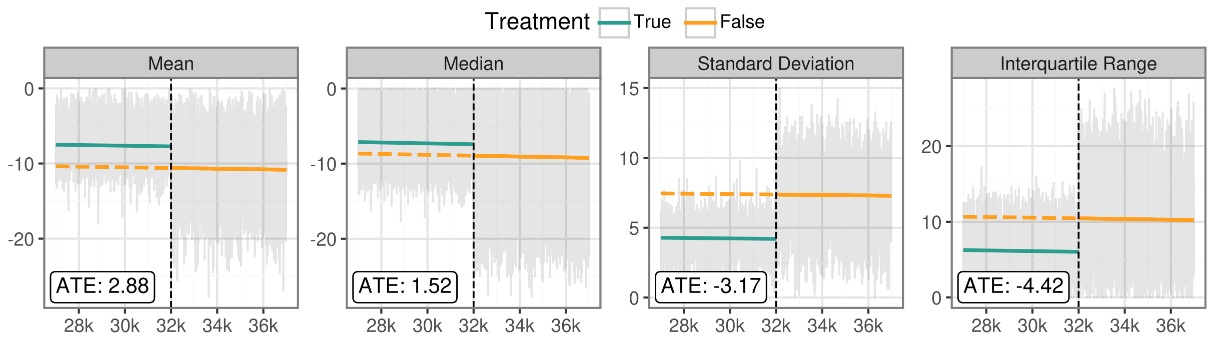

Figure 2: Tokenisation bias of $v \in \mathbb{T}$ for the last checkpoint of a model trained with BPE tokeniser and $K = 32$k. The x-axis shows the running variable γ_v, and the y-axis shows different outcome variables: mean, standard deviation, median, and interquartile range of a $\mathbf{c}_v$'s log-probability across contexts. Subwords on the left-hand side of the cutoff are in the tokeniser. The dashed orange line indicates the estimated counterfactual outcome.

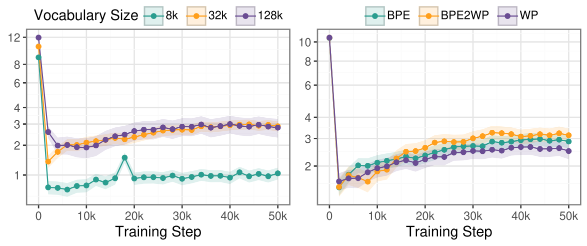

Figure 3: Tokenisation bias ψ_{γ_v} of 57M across training for (left) BPE across vocabulary sizes 8k, 32k, and 128k; and for (right) vocabulary size 32k across tokenisers BPE, WP, and BPE2WP.

Models and Tokenisers. Unless otherwise specified, we run our experiments with a BPE tokeniser with vocabulary size $K = 32$k, and a Llama (Touvron et al., 2023) LM with 57M non-embedding parameters trained on the MiniPile (Kaddour, 2023) dataset for 50k steps. To explore how tokenisation bias changes as a function of: (i) model size, we further train Llama variants with approximately 340M, and 850M non-embedding parameters; (ii) vocabulary size, we experiment with BPE's with K equal 8k, 32k, and 128k; (iii) tokenisation algorithm, we experiment with a standard WP tokeniser, and a tokeniser using WP's tokenisation function but BPE's vocabulary, which we call BPE2WP; (iv) embedding tying and training dataset, we train a 100M parameter model with and without tied input-output embeddings on 20B tokens from the Fineweb-Edu corpus (Penedo et al., 2024).

Evaluation and Data Collection. All models are evaluated on the MiniPile validation set. To compute $Y_{\text{obs}}(v)$, we collect subwords' log-probabilities and aggregate them across the contexts in which they appear. For our regression discontinuity estimation, we focus on a window of 5k subwords around the tokeniser's vocabulary cutoff.

LMs, as this information is typically not published even for open-source models like Pythia (Biderman et al., 2023) and OLMo (Groeneveld et al., 2024). Further, their tokeniser construction details are often unreproducible. We therefore train new tokenisers and models from scratch.

6 Results

In this section, we report the estimated tokenisation bias ψ_{γ_v} in our models. Fig. 2 (left) presents tokenisation bias in BPE with default parameters. We estimate this bias to be 2.88 nats, with treated character-strings $\mathbf{c}_v$ having a log-probability of -7.72 and untreated ones -10.60. This means a $\mathbf{c}_v$ can be assigned roughly 17 times less probability due solely to tokenisation, implying LMs are highly susceptible to tokenisation bias. We next examine this effect in detail.

Effect Across Training. Fig. 3 (left; orange line) shows the causal effect evolving through training. As predicted (Thm. 2), this effect is large at the start of training. Notably, it drops sharply by step 2k and then slowly grows again. This is counterintuitive. Since an ideal model would be unaffected by tokenisation (Thm. 1), one would expect a decline in ψ_{γ_v} across training. Instead, our results suggest models become more biased as they improve.

Comparison Across Vocabulary Sizes. Fig. 3 (left) also shows how tokenisation bias varies with vocabulary size ($K \in \{8\text{k}, 32\text{k}, 128\text{k}\}$). We find weaker biases when using smaller vocabularies. This might be because for smaller K, the character-strings $\mathbf{c}_v$ near the cutoff will be more frequent; untreated strings $\mathbf{c}_v$ will thus see more training, which potentially reduces tokenisation bias. Beyond a critical size, the effect stabilises, with 128k and 32k showing similar biases despite a large difference in vocabulary size (see also Fig. 6, App. D).

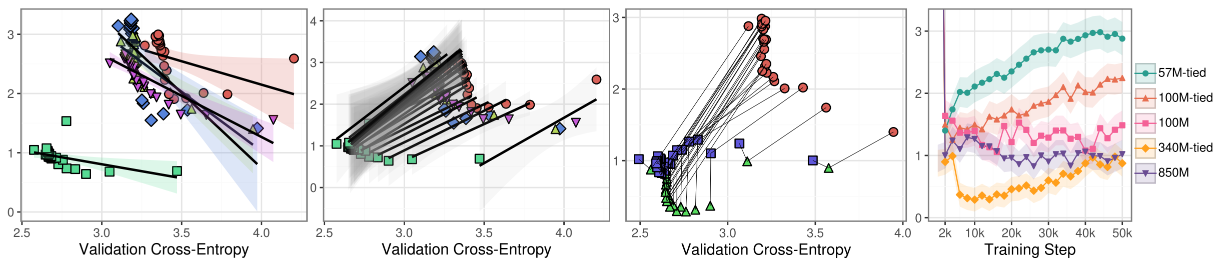

Figure 4: (Left to Center-right) Tokenisation bias ψ_{γ_v} vs. validation cross-entropy of trained models. Each point represents a training step of a model trained with a specific tokeniser. (Left) Linear fits per tokeniser and across training steps, size fixed to 57M; (Center-left) Linear fits per training step and across tokenisers, size fixed to 57M; (Center-right) Linear fits per training step and across model sizes, tokeniser fixed as BPE with $K = 32\text{k}$. (Right-most) Tokenisation bias ψ_{γ_v} across training of models with different sizes and BPE tokeniser with $K = 32\text{k}$.

Comparison Across Tokenisers. Fig. 3 (right) shows how the causal effect changes as a function of the used tokenisation and objective functions, displaying results for BPE (with $\tau_{\uparrow}$, ϕ_{bpe}), BPE2WP (with $\tau_{\uparrow}$, ϕ_{wp}), and WP (with τ_{long}, ϕ_{wp}). In this figure, we see that tokenisation bias seems consistent across these tokenisers (see also Fig. 8, App. D).

Comparison Across Model Sizes. Fig. 4 (right) shows how tokenisation bias changes as a function of model size. In this figure, we see that bias is smaller on larger (e.g., 340M parameters) than in small models (e.g., 57M). Among our largest models, however (with either 340M or 850M parameters), we see little difference in tokenisation bias. Further, even in the largest models, bias does not seem to decrease across training. Together, these results suggest that tokenisation bias may be a persisting property of LMs.

Tokenisation Bias vs. Model Quality. We next explore in more detail the previous observation that tokenisation bias grows across training (Fig. 3)—suggesting an inverse-scaling effect where tokenisation bias strengthens as models improve (McKenzie et al., 2023). Fig. 4 plots the estimated ψ_{γ_v} against model quality (quantified as cross-entropy). As can be seen, two opposing trends emerge: across training, better models show larger causal effects (shown on the left); across tokenisers with different vocabulary sizes, better models show smaller effects (shown on the center-left). Further, this relationship seems to change across model scales (shown on the center-right). This highlights that tokenisation bias is a complex phenomenon, which can either improve or worsen as models get better.

Other Causal Effects. Our framework also allows us to study tokenisation-related biases beyond average model performance. Here, we examine the effect of tokenisation bias on the stability of model outputs across contexts. To this end, we define new potential outcome variables:

$$Y_W^{\mathbb{M}}(v) \overset{\text{def}}{=} \underset{\mathbf{c}_{<t}}{\mathbb{M}} \left[\log p_{\boldsymbol{\theta}}^{\mathbb{T}}(\mathbf{c}_v \mid \mathbf{c}_{<t})\right] \tag{23}$$

where we consider standard deviation, median, and interquartile range as $\mathbb{M}$. Results are in Fig. 2 (also in Fig. 6 and 8, App. D). In particular, standard deviation results show that whether a subword is present in an LM's vocabulary significantly affects output stability: $\mathbf{c}_v$ of in-vocabulary subwords exhibit far less variation in log-probability across contexts than out-of-vocabulary ones. This again underscores the strong influence of tokenisation on model behaviour.

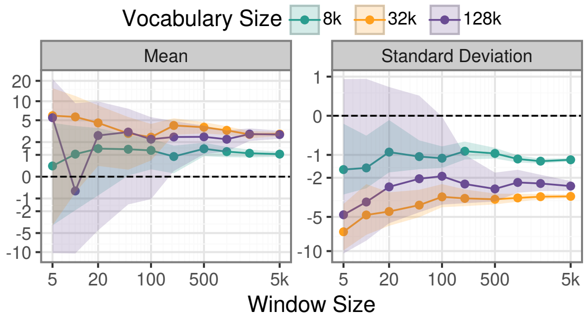

Figure 5: Estimated effect $\widehat{\psi}_{\text{RD}}$ (y-axis) vs. window size used to estimate it (x-axis). Results for fully trained model with BPE and $K \in \{8\text{k}, 32\text{k}, 128\text{k}\}$. Shaded regions correspond to standard errors.

Robustness of the RD Estimation. Finally, we analyse the robustness of our estimates with respect to: (i) window size used to select $\mathcal{K}$, and (ii) functional form of f used for the regression fit. First, Fig. 5 shows that the estimated effect is unstable for window sizes smaller than 500, but stabilises when using at least 1k subwords on both sides of the cutoff (see also Fig. 7, App. D). Second, Fig. 9 (in App. D, due to space constraints) shows that more flexible specifications of f return similar estimates to our linear choice, though fitting noise.

7 Tokenisation Bias in Context

We now place our findings in the broader context of existing work on tokenisation, highlighting theoretical implications, potential links to known biases, and opportunities for improved fairness.

7.1 Implications of Tokenisation Bias

Theoretical Underpinnings of Tokenisation. Our results align with recent theoretical work on the role of tokenisation in language modelling. Rajaraman et al. (2024) show that subword vocabularies endow Transformers with inductive biases suited to modelling higher-order Markov sources, while character-level models lag behind. This is supported by our finding that representing a character-string as two subwords instead of one reliably lowers its log-probability.

Length Bias. Tokenisation bias may help explain the well-documented length bias in LMs (Murray and Chiang, 2018; Stahlberg and Byrne, 2019), where longer subword sequences are less likely to be generated. We find that character-strings tokenised as multiple subwords are systematically assigned lower probabilities, which may cause models to prefer generating shorter sequences.

Multilingual Fairness. Tokenisation can also contribute to performance disparities across languages. Petrov et al. (2023) show that under typical multilingual tokenisers, lower-resource languages tend to have longer tokenisations. While originally noted for its cost implications, we show this may also hurt performance as, *ceteris paribus*, (i) longer subword-strings receive lower probability due to tokenisation bias and, thus, (ii) they may reduce the model's likelihood of generating low-resource language content, reinforcing data sparsity effects. These effects echo Rust et al.'s (2021) findings, which show that language-specific tokenisers close much of the performance gap between monolingual and multilingual models, highlighting tokenisation as a key factor in multilingual performance.

7.2 Going Forward: Uses for ψ_{γ_v}

This paper provides new empirical evidence that tokenisation choices bias model predictions. Prior studies have largely focused on observational evidence of these effects, often through adversarial prompts or indirect correlates (Chai et al., 2024; Wang et al., 2025). Our estimator, by contrast, enables a more principled study of how vocabulary construction choices affect a LM's behaviour. We sketch two concrete use cases for ψ below.

Measuring Impact on Lexical Generalisation. Adding subwords to the vocabulary causes models to treat the corresponding character span as an indivisible unit. This can be beneficial (e.g., in English, $\langle pl \rangle$ and $\langle ay \rangle$ do not have meanings on their own, but $\langle play \rangle$ does), but may hinder generalisation across morphological variants (e.g., $\langle play \rangle$ and $\langle plays \rangle$). Tokenisation may thus impact model generalisation across orthographically similar lexical items. Prior work already hints at this tradeoff: Schäfer et al. (2024) show that duplicated entries in a vocabulary may hurt generalisation, while Toraman et al. (2023) and Schmidt et al. (2024) show that vocabulary expansion helps only under morphology-aware tokenisation. These results suggest that adding too many subwords to the vocabulary can hurt lexical generalisation, a hypothesis our estimator could be adapted to test.

Optimising Tokenisation. Finding an optimal tokeniser is NP-complete (Whittington et al., 2025; Kozma and Voderholzer, 2024). Further, existing heuristic metrics for tokeniser selection, such as subword frequency thresholds (Gowda and May, 2020) or Rényi efficiency (Zouhar et al., 2023), are often poorly correlated with downstream model performance (Schmidt et al., 2024). Our causal estimator offers a more grounded alternative. If a practitioner aims to optimise held-out perplexity, a positive ψ would signal that expanding the vocabulary (i.e., including more subwords) could help. Conversely, a negligible or negative ψ might justify shrinking the vocabulary to gain efficiency without hurting performance. This moves us beyond heuristic selection toward a more systematic, model-level approach to tokeniser design.

8 Conclusion

We study a phenomenon we call **tokenisation bias**: the extent to which a model's outputs are affected by whether a subword appears in its tokeniser's vocabulary. We propose a new method to measure it without re-training the model. We empirically show that character-strings tokenised as a single subword receive significantly more probability than when split, and that this effect intensifies over training.

Limitations

This work estimates tokenisation bias: the extent to which a model's output depends on whether a subword appears in its tokeniser's vocabulary. Unfortunately, due to the costs associated with training large LMs, most of our experiments focused on relatively small models trained in a single language (English). Investigating whether other model architectures, training procedures, and natural languages result in similar causal effects would be important to strengthen our conclusions. Moreover, our method estimates a local causal effect (i.e., the effect for subwords in a window around the cutoff). However, our results suggest that the estimated effect can be extrapolated further beyond the window size, as our regression estimates show an almost flat trend with respect to the running variable.

Acknowledgments

Pietro and Andreas received funding from the European Research Council (ERC) under the European Union's Horizon 2020 Research and Innovation programme grant AVeriTeC (Grant agreement No. 865958). We thank the anonymous reviewers for their helpful questions and comments. We also thank Gregor Bachmann for his feedback on an earlier version of this paper.

References

Mehdi Ali, Michael Fromm, Klaudia Thellmann, Richard Rutmann, Max Lübbering, Johannes Leveling, Katrin Klug, Jan Ebert, Niclas Doll, Jasper Buschhoff, Charvi Jain, Alexander Weber, Lena Jurkschat, Hammam Abdelwahab, Chelsea John, Pedro Ortiz Suarez, Malte Ostendorff, Samuel Weinbach, Rafet Sifa, Stefan Kesselheim, and Nicolas Flores-Herr. 2024. Tokenizer choice for LLM training: Negligible or crucial? In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3907–3924, Mexico City, Mexico. Association for Computational Linguistics.

Joshua Angrist and Jörn-Steffen Pischke. 2015. *Mastering 'Metrics: The Path from Cause to Effect*. Princeton University Press.

Joshua D. Angrist and Jörn-Steffen Pischke. 2009. *Mostly harmless econometrics: An empiricist's companion*. Princeton university press.

Stella Biderman, Kieran Bicheno, and Leo Gao. 2022. Datasheet for the Pile. *arXiv preprint 2201.07311*.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23.

Matias D. Cattaneo, Nicolas Idrobo, and Rocío Titiunik. 2024. *A Practical Introduction to Regression Discontinuity Designs: Extensions*. Cambridge University Press.

Yekun Chai, Yewei Fang, Qiwei Peng, and Xuhong Li. 2024. Tokenization falling short: On subword robustness in large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1582–1599, Miami, Florida, USA. Association for Computational Linguistics.

Tyler A. Chang and Benjamin K. Bergen. 2022. Word acquisition in neural language models. *Transactions of the Association for Computational Linguistics*, 10:1–16.

Stephen R. Cole and Constantine E. Frangakis. 2009. The consistency statement in causal inference: A definition or an assumption? *Epidemiology*, 20(1).

Li Du, Lucas Torroba Hennigen, Tiago Pimentel, Clara Meister, Jason Eisner, and Ryan Cotterell. 2023. A measure-theoretic characterization of tight language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9744–9770, Toronto, Canada. Association for Computational Linguistics.

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

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The Pile: An 800GB dataset of diverse text for language modeling. *arXiv preprint 2101.00027*.

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, Online. Association for Computational Linguistics.

Dirk Groeneveld, Iz Beltagy, Evan Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, Tushar Khot, William Merrill, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew Peters, Valentina Pyatkin, Abhilasha Ravichander, Dustin Schwenk, Saurabh Shah, William Smith, Emma Strubell, Nishant Subramani, Mitchell Wortsman, Pradeep Dasigi, Nathan Lambert, Kyle Richardson, Luke Zettlemoyer, Jesse Dodge, Kyle Lo, Luca Soldaini, Noah Smith, and Hannaneh Hajishirzi. 2024. OLMo: Accelerating the science of language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15789–15809, Bangkok, Thailand. Association for Computational Linguistics.

Jinyong Hahn, Petra Todd, and Wilbert Van der Klaauw. 2001. Identification and estimation of treatment effects with a regression-discontinuity design. *Econometrica*, 69(1):201–209.

Shengding Hu, Yuge Tu, Xu Han, Ganqu Cui, Chaoqun He, Weilin Zhao, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Xinrong Zhang, Zhen Leng Thai, Chongyi Wang, Yuan Yao, Chenyang Zhao, Jie Zhou, Jie Cai, Zhongwu Zhai, Ning Ding, Chao Jia, Guoyang Zeng, dahai li, Zhiyuan Liu, and Maosong Sun. 2024. MiniCPM: Unveiling the potential of small language models with scalable training strategies. In *First Conference on Language Modeling*.

Jean Kaddour. 2023. The MiniPile challenge for data-efficient language models. *arXiv preprint 2304.08442*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*.

László Kozma and Johannes Voderholzer. 2024. Theoretical analysis of byte-pair encoding. *arXiv preprint 2411.08671*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Min Lu, Saad Sadiq, Daniel J Feaster, and Hemant Ishwaran. 2018. Estimating individual treatment effect in observational data using random forest methods. *Journal of Computational and Graphical Statistics*, 27(1):209–219.

Ian R. McKenzie, Alexander Lyzhov, Michael Martin Pieler, Alicia Parrish, Aaron Mueller, Ameya Prabhu, Euan McLean, Xudong Shen, Joe Cavanagh, Andrew George Gritsevskiy, Derik Kauffman, Aaron T. Kirtland, Zhengping Zhou, Yuhui Zhang, Sicong Huang, Daniel Wurgaft, Max Weiss, Alexis Ross, Gabriel Recchia, Alisa Liu, Jiacheng Liu, Tom Tseng, Tomasz Korbak, Najoung Kim, Samuel R. Bowman, and Ethan Perez. 2023. Inverse scaling: When bigger isn't better. *Transactions on Machine Learning Research*. Featured Certification.

Kenton Murray and David Chiang. 2018. Correcting length bias in neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 212–223, Brussels, Belgium. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. 2024. The FineWeb datasets: Decanting the web for the finest text data at scale. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

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

Buu Phan, Brandon Amos, Itai Gat, Marton Havasi, Matthew J. Muckley, and Karen Ullrich. 2025. Exact byte-level probabilities from tokenized language models for FIM-tasks and model ensembles. In *The Thirteenth International Conference on Learning Representations*.

Buu Phan, Marton Havasi, Matthew J. Muckley, and Karen Ullrich. 2024. Understanding and mitigating tokenization bias in language models. In *ICML 2024 Workshop on Theoretical Foundations of Foundation Models*.

Tiago Pimentel and Clara Meister. 2024. How to compute the probability of a word. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 18358–18375, Miami, Florida, USA. Association for Computational Linguistics.

Nived Rajaraman, Jiantao Jiao, and Kannan Ramchandran. 2024. An analysis of tokenization: Transformers under Markov data. In *Advances in Neural Information Processing Systems*, volume 37, pages 62503–62556. Curran Associates, Inc.

Donald B. Rubin. 1974. Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of Educational Psychology*, 66(5):688–701.

Donald B. Rubin. 2005. Causal inference using potential outcomes. *Journal of the American Statistical Association*, 100(469):322–331.

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

Anton Schäfer, Thomas Hofmann, Imanol Schlag, and Tiago Pimentel. 2024. On the effect of (near) duplicate subwords in language modelling. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 9580–9597, Bangkok, Thailand. Association for Computational Linguistics.

Craig W. Schmidt, Varshini Reddy, Haoran Zhang, Alec Alameddine, Omri Uzan, Yuval Pinter, and Chris Tanner. 2024. Tokenization is more than compression. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 678–702, Miami, Florida, USA. Association for Computational Linguistics.

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

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, Berlin, Germany. Association for Computational Linguistics.

Felix Stahlberg and Bill Byrne. 2019. On NMT search errors and model errors: Cat got your tongue? 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 3356–3362, Hong Kong, China. Association for Computational Linguistics.

Donald L. Thistlewaite and Donald T. Campbell. 1960. Regression-discontinuity analysis: An alternative to the ex-post facto experiment. *Journal of Educational Psychology*, 51(6):309–317.

Cagri Toraman, Eyup Halit Yilmaz, Furkan Şahinuç, and Oguzhan Ozcelik. 2023. Impact of tokenization on language models: An analysis for Turkish. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, 22(4).

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

Tim Vieira, Ben LeBrun, Mario Giulianelli, Juan Luis Gastaldi, Brian DuSell, John Terilla, Timothy J. O'Donnell, and Ryan Cotterell. 2024. From language models over tokens to language models over characters. *arXiv preprint 2412.03719*.

Dixuan Wang, Yanda Li, Junyuan Jiang, Zepeng Ding, Ziqin Luo, Guochao Jiang, Jiaqing Liang, and Deqing Yang. 2025. Tokenization matters! Degrading large language models through challenging their tokenization. *arXiv preprint 2405.17067*.

Kaiyue Wen, Zhiyuan Li, Jason S. Wang, David Leo Wright Hall, Percy Liang, and Tengyu Ma. 2025. Understanding warmup-stable-decay learning rates: A river valley loss landscape view. In *The Thirteenth International Conference on Learning Representations*.

Philip Whittington, Gregor Bachmann, and Tiago Pimentel. 2025. Tokenisation is NP-complete. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. 2022. Scaling vision transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12104–12113.

Vilém Zouhar, Clara Meister, Juan Gastaldi, Li Du, Mrinmaya Sachan, and Ryan Cotterell. 2023. Tokenization and the noiseless channel. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5184–5207, Toronto, Canada. Association for Computational Linguistics.

A Definitions of Tokenisation Functions

Merge-based Tokenisation Function. The most common tokenisation function to date—used by, e.g., byte pair encoding (BPE; Gage, 1994; Sennrich et al., 2016)—builds subword-strings by merging a string's symbols two-at-a-time in a fixed pre-determined order. The building blocks of this function are merges. A **merge** $m \in \mathcal{V} \times \mathcal{V}$ represents a pair of subwords; it can thus be written as: $m = v^{[1]} \circledcirc v^{[2]}$. Given a list of merges $\mathbf{m} = [m_1, m_2, \ldots, m_{|\mathbf{m}|}]$, this function is defined as:

$$\tau_{\uparrow}(\mathbf{c}) \overset{\text{def}}{=} \mathrm{merge}_{m_{|\mathbf{m}|}}(\cdots(\mathrm{merge}_{m_1}(\mathbf{c}))) \tag{24}$$

where $\mathrm{merge}_m : \mathcal{V}^* \to \mathcal{V}^*$ is a function which scans a string left-to-right, replacing any appearance of $v^{[1]}$ followed by $v^{[2]}$ with another subword $v^{[\mathrm{m}]}$ which represents their concatenation $v^{[\mathrm{m}]} = v^{[1]} \circ v^{[2]}$. E.g., $\mathrm{merge}_{he \circledcirc llo}(\langle he, llo\rangle) = \langle hello\rangle$.

Longest Prefix-match Tokenisation Function. A common alternative to eq. (24)—used by, e.g., WordPiece (WP; Schuster and Nakajima, 2012)—is to select the longest subword $v \in \mathcal{V}$ matching a prefix of $\mathbf{c}$, and then recursively tokenising the remaining string. This function is defined as:

$$\tau_{\mathsf{long}}(\mathbf{c}) \overset{\text{def}}{=} \left\langle \underset{\text{s.t. } \mathbf{c}_{1:|v|}=v}{\mathrm{argmax}_{v\in\mathcal{V}}} |v|, \tau_{\mathsf{long}}(\mathbf{c}_{|v|:|\mathbf{c}|}) \right\rangle \tag{25}$$

B Proofs

We provide the proofs for Thm. 1 and 2 here.

B.1 Proof of Thm. 1

Theorem 1. *Assume we have a training process that—regardless of the choice of tokeniser $\mathbb{T}$—always returns perfect models, i.e., models for which $p^{\mathbb{T}}_{\boldsymbol{\theta}}(\mathbf{v}) = p^{\mathbb{T}}(\mathbf{v})$. In this case, the causal effect of a subword v is always zero, i.e., $\psi_v = 0$.*

Proof. Note that tokenisation functions τ are necessarily injective and, thus, each character-string is mapped to a unique subword-string. For any tokeniser, we can thus rewrite eq. (4) as:

$$p(\mathbf{c}) = p^{\mathbb{T}}(\tau(\mathbf{c})) \tag{26}$$

Further, for perfect models $p^{\mathbb{T}}_{\boldsymbol{\theta}}(\mathbf{v}) = p^{\mathbb{T}}(\mathbf{v})$. We can thus show that:

$$\begin{aligned} p^{\mathbb{T}}_{\boldsymbol{\theta}}(\mathbf{c}) &= \sum_{\mathbf{v}\in\mathcal{V}^*} p^{\mathbb{T}}_{\boldsymbol{\theta}}(\mathbf{v})\; \mathbb{1}\{\mathbf{c} = \lrcorner(\mathbf{v})\} && \text{(27a)}\\ &= \sum_{\mathbf{v}\in\mathcal{V}^*} p^{\mathbb{T}}(\mathbf{v})\; \mathbb{1}\{\mathbf{c} = \lrcorner(\mathbf{v})\} && \text{(27b)}\\ &= p^{\mathbb{T}}(\tau(\mathbf{c})) && \text{(27c)}\\ &= p(\mathbf{c}) && \text{(27d)} \end{aligned}$$

Under this condition, we can rewrite the potential outcome in eq. (9) as:

$$\begin{aligned} Y_W(v) &= \mathop{\mathbb{E}}_{\mathbf{c}_{<t}} \left[\log p^{\mathbb{T}}_{\boldsymbol{\theta}}(\mathbf{c}_v \mid \mathbf{c}_{<t})\right] && \text{(28a)}\\ &= \mathop{\mathbb{E}}_{\mathbf{c}_{<t}} \left[\log \frac{p^{\mathbb{T}}_{\boldsymbol{\theta}}(\mathbf{c}_{<t} \circ \mathbf{c}_v \circ \Sigma^*)}{p^{\mathbb{T}}_{\boldsymbol{\theta}}(\mathbf{c}_{<t} \circ \Sigma^*)}\right] && \text{(28b)}\\ &= \mathop{\mathbb{E}}_{\mathbf{c}_{<t}} \left[\log \frac{p(\mathbf{c}_{<t} \circ \mathbf{c}_v \circ \Sigma^*)}{p(\mathbf{c}_{<t} \circ \Sigma^*)}\right] && \text{(28c)}\\ &= \mathop{\mathbb{E}}_{\mathbf{c}_{<t}} \left[\log p(\mathbf{c}_v \mid \mathbf{c}_{<t})\right] && \text{(28d)} \end{aligned}$$

where we denote as $p(\mathbf{c} \circ \Sigma^*)$ the sum of the probability assigned to all character-strings starting with $\mathbf{c}$. As the potential outcomes are a function of p, which does not depend on the used tokeniser, the values of $Y_0(v)$ and $Y_1(v)$ will be the same, and the causal effect ψ_v will thus be 0. □

B.2 Proof of Thm. 2

Theorem 2. *Assume that at initialisation our language model outputs the uniform distribution $p^{\mathbb{T}}_{\boldsymbol{\theta}}(v \mid \mathbf{v}_{<t}) = \frac{1}{|\mathcal{V}|}$. In this case, the causal effect of a subword v at initialisation is $\psi_v \gtrapprox \log|\mathcal{V}|$.*

Proof. For treatment assignment 1, we have $v \in \mathbb{T}$ and $\tau(\mathbf{c}_v) = v$; for any context, thus, the model will approximately assign this character-string probability $p^{\mathbb{T}}_{\boldsymbol{\theta}}(v \mid \mathbf{v}_{<t}) = \frac{1}{|\mathcal{V}|}$.[14] For treatment assignment 0, we have $v \notin \mathbb{T}$ and $\tau(\mathbf{c}_v) = \langle v^{[1]}, v^{[2]}, \ldots\rangle$, i.e., this subword now gets tokenised into multiple subwords instead; for any context, thus, the model will approximately assign $\mathbf{c}_v$ probability

$$\prod_{i=1}^{|\tau(\mathbf{c}_v)|} p^{\mathbb{T}}_{\boldsymbol{\theta}}(v^{[i]} \mid \mathbf{v}_{<t} \circ \tau(\mathbf{c}_v)_{<i}) = \frac{1}{|\mathcal{V}|}^{|\tau(\mathbf{c}_v)|} \tag{29}$$

For treatment assignment 0, we have $|\tau(\mathbf{c}_v)| \geq 2$. The ψ_{γ_v} is thus approximately lower-bounded by: $\log \frac{1}{|\mathcal{V}|} - \log \frac{1}{|\mathcal{V}|}^2 = \log|\mathcal{V}|$. □

C Implementation Details

We implement all experiments using PyTorch (Paszke et al., 2019) and implement variants of the Llama architecture using components implemented in the transformers library (Wolf et al., 2020).

[14] The character-string probabilities will only be approximately $\frac{1}{|\mathcal{V}|}$ since other subword-strings might also map to $\mathbf{c}_v$. However, these alternative subword-strings will be longer, and thus have exponentially less probability mass. Once subword-string probabilities are marginalised out, they will thus make little difference to results.

Data. We use the MiniPile dataset (Kaddour, 2023), a curated 6GB subset of the deduplicated Pile corpus (Gao et al., 2020; Biderman et al., 2022). The training set consists of 1M documents, which we use to train both tokenisers and models. Additionally, we train two 100M-parameter models on 20B tokens from Fineweb-Edu (Penedo et al., 2024). For evaluation, we use the 10k-document validation set to collect the subwords' in-context log-probabilities $p^{\mathbb{T}}_{\boldsymbol{\theta}}(\mathrm{c}_v \mid \mathrm{c}_{<t})$.

Tokenisers. Our method requires knowing which subwords would have been added if we allowed for a larger vocabulary. To identify which subwords would be added with a larger vocabulary, we construct a tokeniser with a vocabulary size of 320k, then truncate it to define smaller tokenisers while retaining the full ranked list of subwords. Tokenisers are built using the tokenisers library,[15] and we encourage others to report similar details to support reproducibility. We evaluate vocabularies of sizes 8,024, 32k, and 128k, which allows us to study how tokenisation bias varies with vocabulary size. As the vocabulary grows, the added subwords are naturally of lower frequency. We compare BPE and WP tokenisers, trained identically using byte-level pre-tokenisers, processors, decoders, and a byte-based alphabet.

Model Training. We use variants of the LLaMA 2 architecture. Our default model has 57M parameters, with 6 layers, 24 attention heads, a hidden size of 768, and tied input–output embeddings. Details for other model configurations are available in our repository.[16] Models are trained with AdamW (Kingma and Ba, 2015; Loshchilov and Hutter, 2019) using learning rate 6×10^{-4}, $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 1 \times 10^{-8}$, and weight decay 0.1. We adopt the warmup-stable-decay schedule (Zhai et al., 2022), which maintains a flat learning rate after warmup and applies cosine decay only during cooldown—an approach shown to be effective for small LMs (Hu et al., 2024; Wen et al., 2025) and which avoids requiring a fixed compute budget. Training uses a context size of 2,048 tokens, batch size 128, gradient clipping to 1.0, and runs for 50k steps with checkpoints saved every 2k steps. All experiments are run with the same seed for consistent data order and initialisation.

[15] github.com/huggingface/tokenisers.
[16] github.com/pietrolesci/tokenisation-bias.

Regression Model. Given observed outcomes and subword indices, we estimate ψ_{RD} by fitting the regression:

$$Y_{\mathsf{obs}}(v) = \alpha + \beta \frac{\gamma_v}{1000} + \psi_{\mathsf{RD}} W_v + \eta_v \tag{30}$$

where η_v is a zero-mean error term. To avoid confounding, we exclude subwords that are nested within larger subwords also present in the vocabulary (e.g., if both $\langle he \rangle$ and $\langle hello \rangle$ are included, only the latter is used).[17]

Hardware Details. We use a server with one NVIDIA A100 80GB PCIe, 32 CPUs, and 32 GB of RAM for all experiments. Below, we report a subset of the output of the lscpu command:

```
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Address sizes:         46 bits physical,
                       48 bits virtual
Byte Order:            Little Endian
CPU(s):                32
On-line CPU(s) list:   0-31
Vendor ID:             GenuineIntel
Model name:            Intel(R) Xeon(R)
                       Silver 4210R CPU
                       @ 2.40GHz
CPU family:            6
Model:                 85
Thread(s) per core:    1
Core(s) per socket:    1
Socket(s):             8
Stepping:              7
BogoMIPS:              4800.11
```

Reproducibility. We release all experimental artefacts as a HuggingFace Hub collection.[18] This includes: (i) the raw and tokenised data in model-consumed order; (ii) the full 320k-subword tokenisers; (iii) the reduced-vocabulary tokenisers used in our experiments; and (iv) all model checkpoints. We encourage others to adopt similar practices and make tokeniser design choices more transparent.

D Additional Plots

Additional plots follow on the next pages.

[17] Following standard causal analysis, we assume SUTVA: treating one unit does not affect others. This assumption is violated by overlapping subwords—i.e., subwords that are themselves part of larger subwords—which we thus exclude.
[18] https://huggingface.co/collections/pietrolesci/tokenisation-bias-66d5d0b40cb82a2d789b19db.

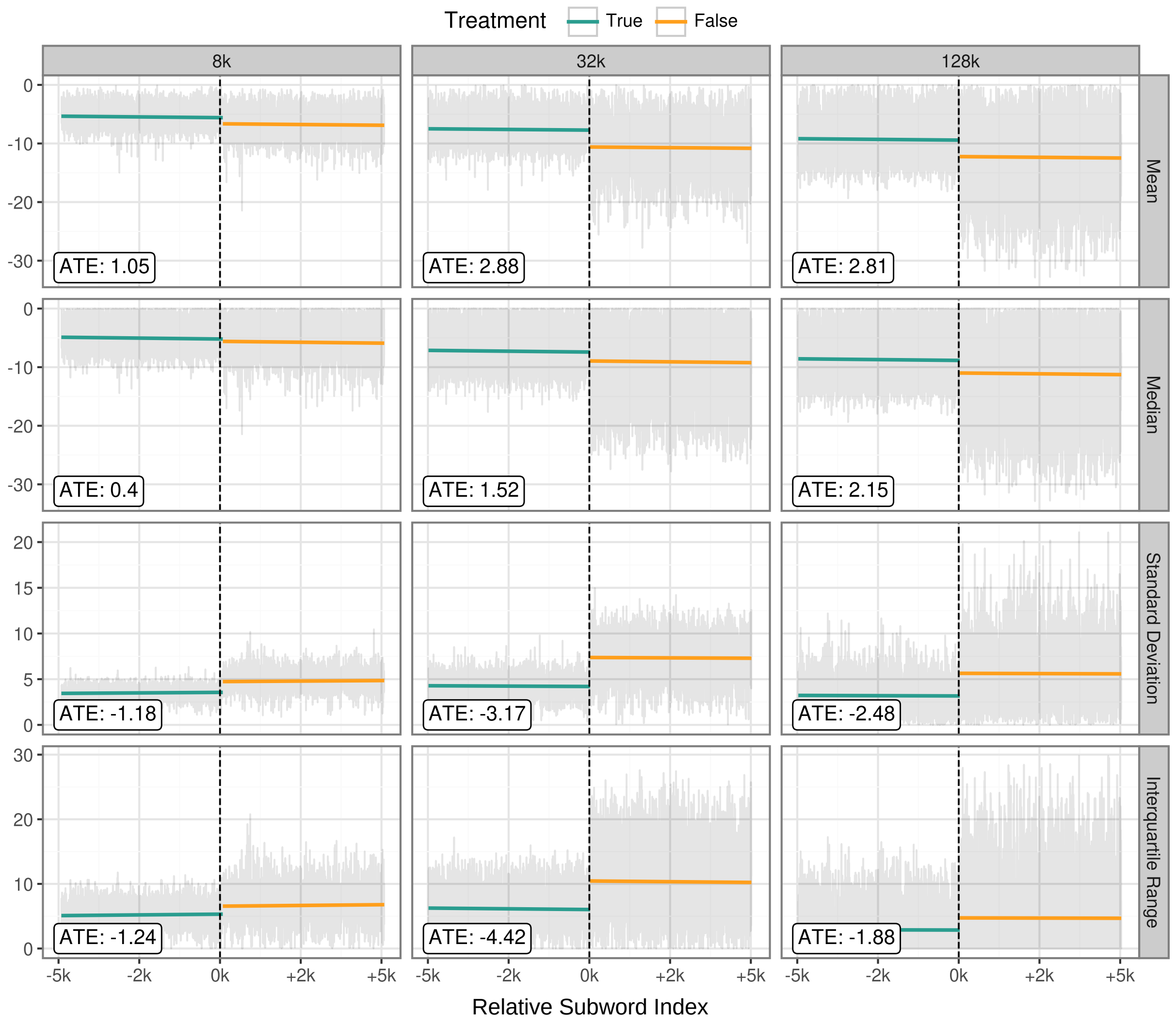

Figure 6: Average treatment effect for BPE with $K \in \{8\text{k}, 32\text{k}, 128\text{k}\}$ at the last model checkpoints. Each row refers to a different outcome variable: mean, standard deviation, median, and interquartile range of a $\mathbf{c}_v$'s log-probability across contexts. Subwords on the left-hand side of the cutoff are treated (i.e., added to the vocabulary).

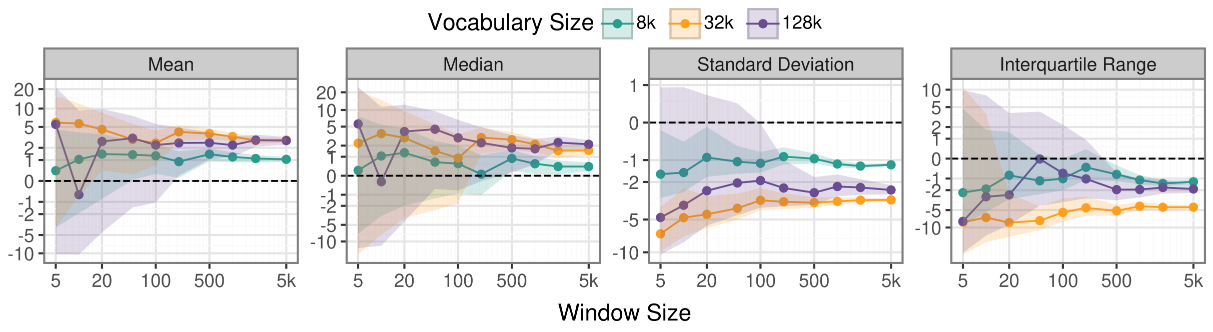

Figure 7: Estimated effect $\widehat{\psi}_{\text{RD}}$ (y-axis) vs. window size used to estimate it. Results for fully trained model with BPE and $K \in \{8\text{k}, 32\text{k}, 128\text{k}\}$. Shaded regions correspond to standard errors. The columns refer to different outcome variables: mean, standard deviation, median, and interquartile range of a $\mathbf{c}_v$'s log-probability across contexts.

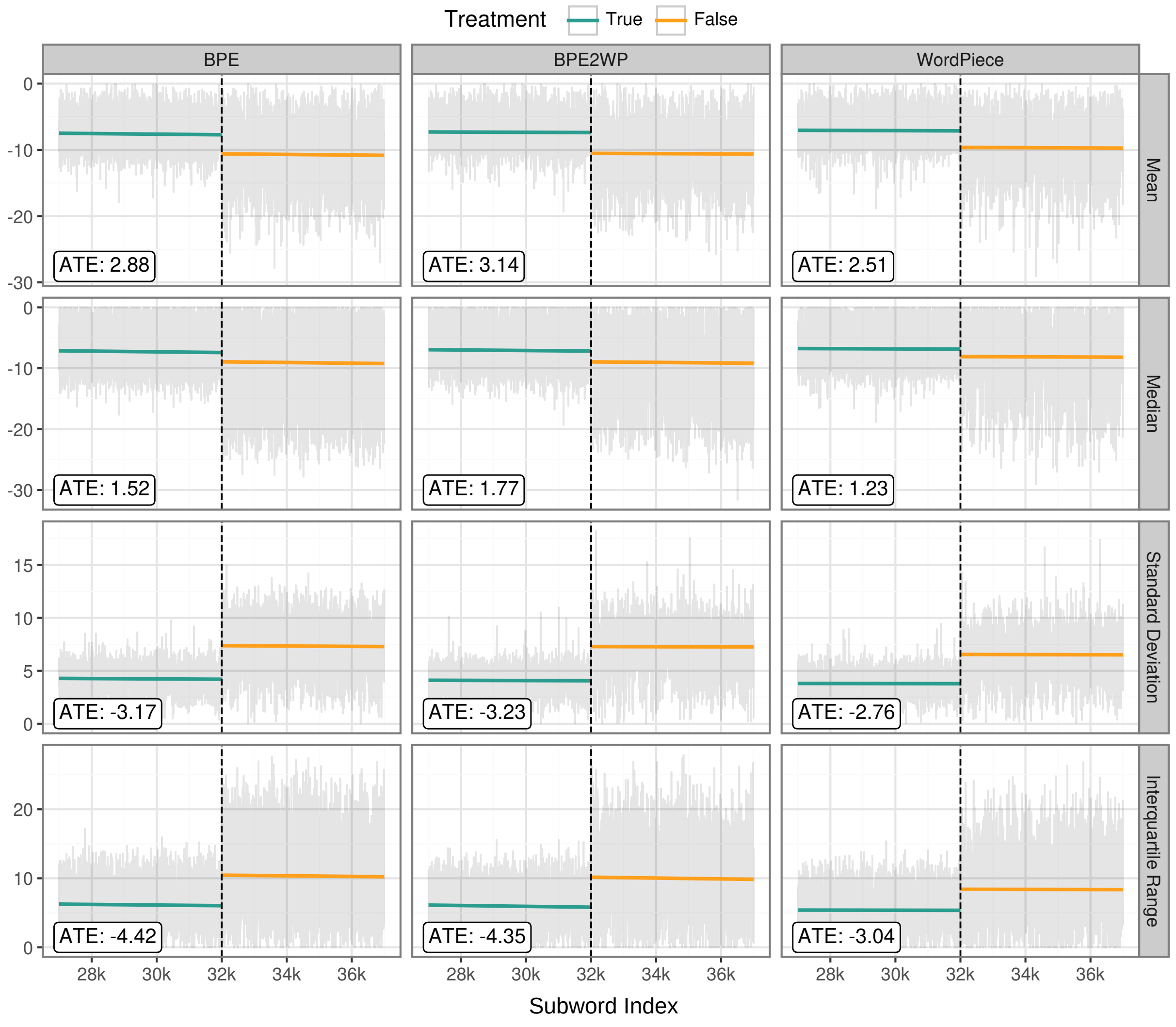

Figure 8: Average treatment effect for fully trained models with BPE, BPE2WP, and WP and $K = 32$k. Each row refers to a different outcome variable: mean, standard deviation, median, and interquartile range of a $\mathbf{c}_v$'s log-probability across contexts. Subwords on the left-hand side of the cutoff are treated (i.e., added to the vocabulary).

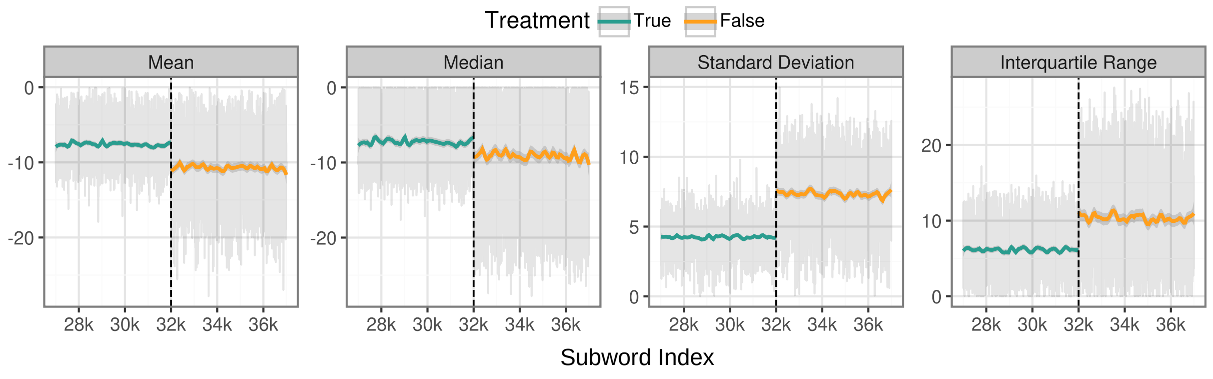

Figure 9: Stability of the average treatment effect with respect to the functional form of f, for fully trained models with BPE and $K = 32$k. Columns refer to different outcome variables: mean, standard deviation, median, and interquartile range of a $\mathbf{c}_v$'s log-probability across contexts. Subwords on the left-hand side of the cutoff are treated (i.e., added to the vocabulary). Conditional mean lines and confidence intervals are computed using the LOESS (locally estimated scatterplot smoothing) method.