Language Model Tokenization: Segmentation and Compression

Suchir Salhan
sas245@cam.ac.uk
24 November 2025

In this lecture, we focus on **language model tokenization**, scrutinising tokenization algorithms from a linguistic perspective. [1]

Key Objectives:

- How do information-theoretic segmentation algorithms relate to the task of tokenization?
- How can we use morphological segmentation to understand popular tokenization algorithms and assess their limitations?
- How can we intrinsically evaluate tokenizers and their morphological alignment?
- How can we draw on ideas from morphological typology – alongside more general linguistic considerations– to improve tokenization to make language models more cross-lingually equitable?

> Even when more data is available, Linguistics can be of value: it can inform [...] **more language-balanced tokenization** (Creutz and Lagus 2002; Limisiewicz et al. 2024; Fusco et al. 2024). From Opitz et al (2025), *Natural Language Processing RELIES on Linguistics*

This lecture focus on connecting linguistically-informed perspectives to modern language model tokenization research. This forms an important subset of research activity related to tokenization; more general (and less linguistic) issues related to tokenization will also be surveyed.

[1] This Handout is associated with MPhil Advanced Computer Science /Part III Computer Science Tripos Module – **L95 Introduction to Natural Language Syntax and Parsing**. These notes are copyright Suchir Salhan, unless otherwise indicated.

Language Model Tokenization Algorithms

An Introduction to Tokenization and Subwords in Natural Language Processing

Mielke et al. [2021] offer a comprehensive historical survey of tokenization in Natural Language Processing. A persistent challenge in NLP is how best to segment text – partially because of the difficulty of deterministically characterising the notion of **wordhood**.[2] **Tokens** therefore serve as approximations for word-forms – a "non-empty contiguous sequence of graphemes in a document".

In Latin-script languages, tokens often correspond to contiguous sequences of non-whitespace and non-punctuation characters. However, there

[2] Some linguists suggest there might be no quantal shift from word (morphotactic) to phrasal (syntactic) structure, but rather a smooth cline, morphology-like to syntax-like combination. See e.g., Haspelmath, M. (2011). The indeterminacy of word segmentation and the nature of morphology and syntax. *Folia Linguistica*, *45*(1), 31-80. https://doi.org/10.1515/flin.2011.002

is no one-to-one correspondence between tokens and word-forms: multiple tokens may form a single word-form (e.g., *sine die*) or a single token may represent multiple word-forms (e.g., English *don't = do + not*). [3]

Tokenization is a process of encoding strings into tokens of a fixed vocabulary size $\mathcal{V}$. A common way to formalize the tokenization problem is as a compression task that minimizes the ratio of tokens produced when tokenizing the input data.

Before we get into the details of popular tokenization algorithms, I briefly summarise some important conceptual ideas about **multi-granular representations in NLP** that are surveyed from "classical" (i.e. pre-LLM!) segmentation literature.

Limitations of Word-Level Models: Traditional word-level models operate under a *closed vocabulary* assumption: they cannot handle rare or unseen words, commonly referred to as out-of-vocabulary (OOV) items. Historically, these words were replaced with a special UNK token during training. While simple, this approach presents several drawbacks. First, it prevents natural language generation of novel words. Second, it loses semantic information for rare words, reducing representational richness. Third, it can inflate probability estimates for language models, particularly in morphologically rich languages. These limitations motivate the development of character- and subword-level modeling techniques, which address rare and novel words while maintaining interpretability.

Open-vocabulary modeling remains crucial for natural language generation and low-resource scenarios, motivating continued integration of character- and subword-level representations. Two-level models combining word- and character-level RNNs (e.g., Mielke & Eisner, 2018) allow generation of novel words, while cache mechanisms (e.g., Kawakami et al., 2017) facilitate handling of bursty or frequent function words. Unlike input-side open-vocabulary embeddings, generation requires reserving probability mass for unseen sequences, making open-vocabulary modeling an essential component for both practical applications and robust language understanding.

We need multi-granular representations – word Representations need to be augmented with character information: Open-vocabulary modelling extends beyond input embeddings: it allows models to generate novel words at test time. Character, subword, and word-level features complement each other for rare and novel words, providing a multi-granular approach that addresses OOV problems while preserving interpretability. Traditional word-level models suffer from the closed-vocabulary problem, failing to represent rare or unseen words and relying on UNK tokens, which discard semantic information and hinder generation. Approahces using hybrid word-character embeddings (Dos Santos & Zadrozny, 2014; Kim et al., 2016) and character n-gram methods such as fastText (Bojanowski et al., 2017) demonstrate that combining multiple levels of representation improves modelling of rare words.

Modern models like CharacterBERT further integrate subword units to enhance robustness, illustrating the complementary benefits of character, subword, and word-level features.

Typological Variation impacts Tokenization – *uniform tokenization will lead to cross-lingual inconsistency* Optimal segmentation varies across languages, morphological systems, and downstream tasks, requiring flexible, language-aware strategies. Unsupervised morphological segmen-

[3] As Mielke et al. [2021] note, this is what the Universal Dependencies guidelines refer to as **"multitoken words"** and **"multiword tokens"** (see More et al. [2018]). Both phenomena can interfere in non trivial ways (e.g. French *à l'instar* du = *à_l'instar_de + le*).

tation approaches, such as Morfessor (Creutz & Lagus, 2002–2020) or MDL-based methods (Brent et al., 1995; Goldsmith, 2001), highlight that one-size-fits-all tokenization can be suboptimal. Similarly, rule-based and language-specific tokenizers (FSTs, Porter stemmer) provide effective solutions for morphologically complex languages, particularly when combined with statistical subword methods. These findings underscore that tokenization choices should consider linguistic structure, morphology, and task requirements.

You can marginalize over possible (or non-canonical) segmentations. Treating segmentation as a latent variable provides a conceptually powerful framework, allowing models to marginalize over all possible segmentations. Character-level models with skip mechanisms (HM-RNNs, Chung et al., 2017) attempt to learn word boundaries automatically, but face challenges such as training instability and suboptimal segmentations without explicit whitespace. Segmental neural language models (Kong et al., 2016; Sun & Deng, 2018) and approximate inference methods (Chan et al., 2017; Buckman & Neubig, 2018) demonstrate practical approaches to this marginalization, making latent segmentation tractable while capturing uncertainty over word boundaries.

Bayesian and morphological models provide theoretically grounded and interpretable alternatives to purely statistical methods. Bayesian non-parametric models, including hierarchical Pitman-Yor and Dirichlet Process models (Teh, 2006; Wood et al., 2011), support infinite lexicons and principled smoothing. Two-stage models for word generation and reuse (Goldwater et al., 2006b; Johnson et al., 2007) not only enable unsupervised word segmentation but also align with cognitive phenomena, such as child language acquisition, providing insight into how humans might discover and generalize word forms.

Some Tokenization Notation

Let us start with a **string** s—a sequence of characters (or bytes) drawn from some base **alphabet** Σ.

We also have a **vocabulary** $\mathcal{V}$, which is a finite set of **pieces**. Each piece $p \in \mathcal{V}$ is a sequence of symbols from $\Sigma \cup \Sigma_{\text{res}}$, where Σ_{res} is a finite set of reserved symbols (e.g., whitespace markers, start/end tokens, etc.). During **tokenization**, our goal is to convert the string s into a sequence of tokens $t = (t_1, t_2, \ldots, t_n)$, where each $t_i \in \mathcal{V}$. We call this sequence a **segmentation** of s, which is just a different way of representing the original string.

A **tokenization algorithm** defines a mapping

$$f_\theta : \text{Strings} \to \text{Token sequences}$$

and a method for learning the parameters θ of this mapping. Applying f_θ to a string is often called **tokenizing** the string.

We assume that the original string s can be reconstructed from its token sequence t using a **detokenization function** g, often as simple as string concatenation with some special handling (e.g., whitespace):

$$s = g(t_1, t_2, \ldots, t_n).$$

From now on, we consider $\mathcal{V}$ fixed, as part of the model specification. All probabilities over strings and segmentations are defined with respect to this fixed vocabulary.

Even with a fixed vocabulary $\mathcal{V}$, there may be multiple valid token sequences t that detokenize to the same string s.[4] Formally, we define the set of all valid segmentations as the **set-valued inverse** of g:

$$\mathcal{T}(s) = \{t : g(t) = s\}.$$

This non-injectivity is important in practice [Meister, 2025]:

- **It's handled differently by different tokenization schemes** – we'll see one algorithm (UnigramLM) samples from a posterior over segmentations. Other tokenization algorithms optimise for minimum token sequence length [Meister, 2025].
- Non-injectivity gives rise to a property of tokenization called **non-canonicality**.

We typically distinguish between the **canonical tokenization** (the standard choice) and **non-canonical tokenizations** (other valid segmentations).

[4] This point is highlighted in Clara Meister's blogpost: **UnigramLM: An Attempt at Writing The Missing Manual ,** **https://cimeister.github.io/blog/unigramlm/**

Byte-Pair Encoding (BPE) and Compression

The leading tokenization algorithm is **Byte Pair Encoding (BPE)** [Sennrich et al., 2016]. In **byte-pair encoding (BPE)**, the mapping f_θ is parameterized by a list of merge pairs θ, and the algorithm learns these merges. At inference, starting from the string represented as individual symbols from Σ, BPE repeatedly merges the most frequent adjacent symbol pairs into new tokens.

Key Idea

BPE is conceptually simple: beginning with characters as the basic symbols, the algorithm repeatedly merges the most frequently co-occurring pair of adjacent symbols in the corpus. Each merge introduces a new symbol (the concatenated pair), which then participates in future frequency calculations. The process continues until a specified number of merges is reached or no pair has frequency >1.

Key points about BPE:

- It performs **lossless compression**: merged symbols replace recurring substrings, reducing the number of tokens needed to encode text.
- It does **not cross whitespace boundaries**, so orthographic words remain the upper bound of segmentation.
- Subwords grow in length over time, eventually approaching whole word forms.

Below we provide a summarised formalisation of BPE following Zouhar et al. [2023b].

Definition 1 (Merge)**.** *Let Σ be an alphabet. A* merge *is either a symbol $\sigma \in \Sigma$ or a pair $[\mu', \mu'']$ of merges. A* merge sequence *$\mu = \langle \mu_1, \ldots, \mu_N \rangle$ is valid if each non-trivial merge uses only previously defined merges or symbols.*

Definition 2 (Compression Utility)**.** *For a string $x \in \Sigma^*$, the compression utility of a merge sequence μ is*

$$\kappa_x(\mu) = |x| - |APPLY_\mu(x)|,$$

where APPLY iteratively applies merges to x.

Definition 3 (BPE Optimization Problem). *Find a merge sequence $\mu^\star$ of length M maximizing compression utility:*

$$\mu^\star = \arg\max_{\mu \in \mathcal{M}_{\Upsilon_\Sigma}, |\mu|=M} \kappa_x(\mu).$$

BPE uses a Greedy Algorithm to iteratively pick the most frequent adjacent pair:

$$\mu^\dagger = \text{GreedyBPE}(x, M)$$

Theorem 1 (Approximation Guarantee). *The greedy BPE is*

$$\frac{\kappa_x(\mu^\dagger)}{\kappa_x(\mu^\star)} \geq \frac{1}{\sigma(\mu^\star)} \left(1 - e^{-\sigma(\mu^\star)}\right),$$

where $\sigma(\mu^\star)$ is the total backward curvature. Empirically, the bound ≈ 0.37.

Worked Example:
String: `picked pickled pickles`, $M = 5$ merges:

$$\text{Merges: } [p, i], [c, k], [pi, ck], [e, d], [pick, l]$$

Final tokenization: `pick ed pickl ed pickl e s`.

Original greedy BPE: $O(NM)$, with N string length, M merges.

At test time, the same procedure of iteratively merging all occurences of the pair can be performed by executing all recorded merges in the order in which they were conducted during training of the tokenization model.

Zouhar et al. [2023b] offer an improved linked-list + max-heap implementation reduces complexity to $O(N \log M)$. Exact BPE via memoization enumerates all valid merge sequences. Though exponential, it is faster than brute-force enumeration of all $O(\min(|\Sigma|^{2M}, NM))$ sequences.

The BPE compression algorithm is from a family of algorithms called **macro-schemas**, where substrings are replaced with references to them [Gage, 1994].[5]

- There is a **Byte-Level BPE** algorithm (as the name indicates, merges are applied to bytes). [6]

[5] Note, though, that the idea originates from James A. Storer and Thomas G. Szymanski. 1982. Data compression via textual substitution. J. ACM, 29(4):928–951.

[6] Changhan Wang, Kyunghyun Cho, and Jiatao Gu. 2019. Neural machine translation with bytelevel subwords.

Is Compression All You Need?

Key Idea

The structural properties of subwords discovered during BPE compression correlate with the morphological typology of the language. (Hypothesis of Gutierrez-Vasques et al. [2023])

One of the central insights in linguistic typology is that languages differ significantly in how they encode information. Some languages distribute meanings across many short, loosely connected chunks (e.g., English), while others condense meaning into long, morphologically complex words (e.g., Inuit languages). Gutierrez-Vasques et al. [2023] undertake a **large-scale cross-linguistic analysis of BPE merges across 47 typologically-diverse languages**.

The authors focus on three concepts:

- **Productivity:** the number of word types that contain a subword (weighted by frequency). **Productive subwords** appear across many word types (high type frequency, moderate token frequency).

- **Idiosyncrasy**: How concentrated a subword is in a few word types (high = appears in few types but with large cumulative frequency). **Idiosyncratic subwords** appear in few types but with high token counts.
- **Cumulative frequency**: How often all instances of a subword occur in the corpus.

The authors collect multiple parallel corpora (e.g., Bible Translations) from 47 languages, and extract BPE subwords up to $\tilde{2}00$ merges. By tracking these properties across successive merges, they construct **language vectors encoding the distribution of productivity vs. idiosyncrasy in a language's most redundant patterns**. This allows them to construct language vectors that summarise each language's profile.

Here are the key findings:

- Languages with **rich inflectional morphology (Inuit, Slavic)** tend to produce early BPE merges that are **highly productive morpheme-like subwords**, similar to inflectional morphemes appearing across many word types.
- Languages with **less inflectional morphology (Vietnamese)** exhibit early merges dominated by **idiosyncratic subwords**, which appear in only a few word types but with high overall token frequency.
- **200 BPE merges are enough** to stabilize typological profiles.
- **BPE language space correlates with morphological clusters in typological databases** (WALS, the World Atlas of Linguistic Structure), significantly above chance.

The WordPiece Tokenization Algorithm

Word segmentation for **language without whitespace (e.g., Chinese, Japanese and Vietnamese)** has been a notoriously challenging issue for Natural Language Processing. In these languages, there are no explicit word boundary markers in the surface form.

There has been sustained interest amongst computational linguists in developing unsupervised methods for **Chinese Word Segmentation**.[7]

Schuster and Nakajima [2012] develop the WordPiece tokenization algorithm for Japanese and Korean text. Likewise, reliance on space-separated tokens is impossible; text is written without spaces. WordPiece merges **pairs that increase the likelihood that an *n*-gram LM trained with updated vocabulary**.

[7] Discriminative Models rely on **goodness measures for candidate segmentations** (e.g, Mutual Information), whilst Generative models focus on finding the optimal segmentation of highest generative probability [Sun and Deng, 2018]

Key Idea

WordPiece, a major alternative used in models like BERT, also builds subword vocabularies, but:

- It selects merges based on **maximum-likelihood improvement under an n-gram language model**, not raw pair frequency.
- After vocabulary construction, WordPiece uses a **longest-match-first** tokenization strategy.

The WordPiece algorithm is an unsupervised subword tokenization method originally developed for Japanese and Korean text, where the absence of whitespace makes word-level segmentation non-trivial [Schuster

and Nakajima, 2012]. Similar to early work on Chinese Word Segmentation, WordPiece aims to learn a vocabulary of subword units that balances expressiveness with compactness. Unlike character-based segmentation, which leads to overly long sequences, or whole-word segmentation, which suffers from severe out-of-vocabulary issues, WordPiece constructs a lexicon of variable-length units that approximate morphemes while maintaining full coverage of all Unicode text.

The training procedure begins with a vocabulary initialized to all individual Unicode characters. Given a large text corpus, WordPiece iteratively merges symbol pairs in a greedy bottom-up manner. At each iteration, the algorithm identifies the pair of adjacent symbols whose merge would result in the greatest increase in the likelihood of an n-gram language model trained on the corpus with the updated vocabulary. This likelihood-based criterion distinguishes WordPiece from purely frequency-driven approaches such as Byte-Pair Encoding (BPE): the objective is not simply to merge the most common pairs, but to merge those pairs that make the resulting tokenization statistically easier to model. The process continues until a target vocabulary size is reached, producing a subword lexicon optimized for language-model consistency rather than raw frequency.

At inference time, WordPiece tokenizes each word using a greedy longest-match-first strategy, sometimes referred to as MaxMatch. Given a fixed vocabulary and a single input word, the algorithm scans from left to right and repeatedly selects the longest prefix of the remaining string that occurs in the vocabulary. Subword units that begin in the middle of a word are marked with a special suffix indicator (e.g., in BERT), ensuring a clear distinction between word-initial and non-initial pieces. If at any point no valid vocabulary token matches the current prefix, the entire word is mapped to a designated unknown symbol. This greedy decomposition is deterministic and efficient, making it suitable for large-scale inference.

Although early implementations of WordPiece used a naïve longest-prefix lookup with worst-case quadratic complexity, modern systems represent the vocabulary as a trie and employ optimized matching algorithms. Recent linear-time implementations—such as LinMaxMatch—extend the trie with failure links and failure pops inspired by the Aho–Corasick automaton [Song et al., 2021]. These auxiliary structures allow the tokenizer to fall back to shorter candidate matches without re-scanning previously processed characters, thereby guaranteeing strictly $\mathcal{O}(n)$ time for a word of length n. Such efficiency improvements are crucial for industrial-scale deployments, where tokenization becomes a computational bottleneck in systems processing billions of queries.

In summary, WordPiece provides a principled compromise between character-based and word-based representations by leveraging LM-based merge criteria during training and deterministic greedy segmentation at test time. Its language-model-aware vocabulary construction, robustness to out-of-vocabulary sequences, and efficient decoding have made it a core component of modern transformer-based NLP architectures, including BERT. Furthermore, its origins in segmenting unspaced languages highlight the longstanding connection between subword modeling and unsupervised word segmentation research.

UnigramLM

Kudo [2018] introduce the **UnigramLM Tokenization Algorithm**. The UnigramLM algorithm provides a probabilistic framework for tokenizing text, treating tokenization as inference in a latent-variable generative model over strings.

The description of UnigramLM follows a very comprehensive exposition from Clara Meister's (very!) recent blogpost on UnigramLM – all credit for this summary should be attributed to the blogpost![8]

At its core, UnigramLM assumes that each observed string arises from a latent sequence of tokens drawn independently from a unigram distribution over a fixed vocabulary.

Formally, let X denote a random variable over tokens in vocabulary $\mathcal{V}$ with probabilities $\theta_x = P(X = x)$, and let $X = (X_1, \ldots, X_n)$ represent sequences of such tokens. A length prior $P(L)$ may optionally be incorporated to model token sequence lengths. The data-generating distribution over strings S is then obtained as a pushforward of the token sequence distribution via the deterministic mapping from token sequences to strings.

Inference in UnigramLM involves finding the most probable segmentation of a string under the learned unigram model, often using a Viterbi-style dynamic program. While the length prior can influence the segmentation, it is often ignored in practice, and probabilities are computed as the product of token probabilities along candidate segmentations.

The model parameters θ and vocabulary $\mathcal{V}$ are unknown and must be estimated from observed text. Maximum likelihood estimation (MLE) of θ would require the complete dataset of token sequences, which is latent. Instead, the algorithm maximizes the observed data log-likelihood via the Expectation-Maximization (EM) algorithm. In the E-step, the expected complete-data log-likelihood is computed under current parameter estimates, effectively weighting all valid segmentations of each string by their posterior probabilities. The M-step then updates the token probabilities using these expected counts, analogous to multinomial MLE but with fractional counts derived from the posterior over segmentations.

A distinctive feature of UnigramLM is its iterative pruning of the initial vocabulary. Starting from an overcomplete vocabulary, the algorithm alternates EM updates with pruning steps that remove tokens contributing least to the expected log-likelihood, gradually shrinking the vocabulary to the desired size. Token importance can be approximated by the change in corpus likelihood if the token is removed, often computed via the best alternative segmentation. This pruning heuristic avoids repeated full forward-backward evaluations for each token while retaining high-probability segmentations.

As Clara's blog post highlights, practical implementations, notably in the SentencePiece library, introduce additional design choices.

- The initial vocabulary is generated via frequent substring mining rather than exhaustive substrings, EM is run for a fixed number of sub-iterations per pruning step,
- A Bayesian variant of the M-step applies a Dirichlet-like prior to penalize low-count tokens.
- Normalization, whitespace handling, and prefix/suffix markers are also applied to constrain feasible segmentations, further shaping the learned token vocabulary.

[8] Check out Clara's Blogpost: **UnigramLM: An Attempt at Writing The Missing Manual** , `https://cimeister.github.io/blog/unigramlm/`

Collectively, these choices define the widely used, practical instantiation of the UnigramLM tokenization algorithm.

Pretokenization

Modern tokenization pipelines rely on pre-tokenization, which splits text into "pretokens" (commonly full words). This creates two issues:

- **Skewed token distributions:** Most common words become single tokens, limiting variability and uniformity in token frequency.
- **Limited vocabulary utilization:** Expanding vocabularies does not significantly change token distributions since low-frequency tokens contribute little.

BoundlessBPE [Schmidt et al., 2025] Schmidt et al. [2025] introduce BOUNDLESSBPE to **overcome the pretokenization barrier to enable more uniform token frequency distributions and higher compression efficiency, and better vocabulary utilisation**. This extends standard Byte Pair Encoding (BPE) with two main innovations.

First, it introduces superwords via supermerges: adjacent pretokens, each represented by a single token, can merge into multi-word tokens, e.g., `[' of'] + [' the']` $\rightarrow$ `[' of the']`, reducing the frequency of overly common tokens like `" the"` by distributing occurrences among superwords. Superwords are frequency-driven and not necessarily semantically cohesive, unlike PMI-based multi-word expressions.

Second, it employs an aggressive deletion strategy (a variant of PickyBPE), where intermediate tokens primarily used within higher-frequency merges are deleted and broken down into single bytes, allowing re-merging into higher-frequency tokens and freeing up vocabulary space.

During training, regular merges, supermerges, and deletions are considered at each step, with the highest-frequency operation applied first, and supermerges restricted to cases where both pretokens are single tokens. BOUNDLESSBPE also introduces the `BOUNDLESS_PATTERN` regex for improved pre-tokenization, particularly for code (e.g., `XMLHttpRequest`, `snake_case`, `camelCase`), named entities, and words with contractions or optional spaces, enabling fine-grained subcomponents to form superwords if frequency warrants, e.g.,

`{'XMLHttpRequest', 'snake_case', 'camelCase', 'CONSTANT'}` $\rightarrow$ `['XML', 'Http', 'Request', ' snake', '_case', ' camel', 'Case', ' CONSTANT']`.

Empirical results show more uniform token distributions, with supermerges constituting $\sim$36.6% of merges, rising to $\sim$50.8% in later stages, reducing counts of extremely common tokens, increasing bytes per token by 9–15%, improving Rényi efficiency by 3–5%, and achieving higher vocabulary utilization. Ablation studies indicate minimal impact from deletion style but significant effects from regex choice. Compared to related multi-word token strategies (e.g., SuperBPE, MWE tokenizers), BOUNDLESSBPE operates in a single pass, provides explicit control over pretokens eligible for merging, and integrates supermerges and deletions within the standard BPE loop.[9]

[9] The regex improvements can probably naturally generalise beyond natural language, but there are risks of semantically incoherent superwords.

SuperBPE [Liu et al., 2025] **Tokenization in modern language models typically assumes that tokens should be subwords—units that never**

cross whitespace boundaries. While intuitive, this assumption is questionable because whitespace does not reliably mark semantic units. Many expressions ("by the way"), multi-word constructions that act as single concepts, and languages without spaces (e.g., Chinese) challenge the idea that subwords should be restricted to within-word boundaries. This motivates exploring tokenization approaches that move beyond subwords to capture larger semantic units.

SuperBPE introduces a simple but powerful modification to the standard BPE algorithm: a two-stage pretokenization curriculum. In stage one, the tokenizer behaves like regular BPE and learns subwords using whitespace pretokenization. In stage two, the whitespace boundary is removed, allowing the tokenizer to learn "superword" tokens that span multiple words. This approach dramatically increases encoding efficiency—up to 33% fewer tokens than BPE for the same text with a 200k vocabulary—because it can merge frequent multi-word expressions. When training 8B-parameter models with fixed architecture, vocabulary size, and compute, SuperBPE consistently outperforms BPE, yielding a +4.0% average improvement across 30 downstream tasks (including +8.2% on MMLU) while reducing inference compute by 27%.

Analysis shows that SuperBPE models benefit from more uniform token difficulty and better semantic segmentation. By merging frequent multi-word expressions into single tokens, SuperBPE reduces very easy and very hard predictions, creating a smoother loss distribution. These tokens often correspond to semantically cohesive units—prepositional phrases, fixed expressions, or collocations—that BPE otherwise fragments. The result is shorter token sequences, more semantically meaningful units, and ultimately more efficient and capable LMs. Because SuperBPE is a local change requiring no architectural modifications, it offers a simple path to building faster, stronger language models.

SuperBPE tokenizers encode text much more efficiently than BPE, and this advantage grows with larger vocabulary size.

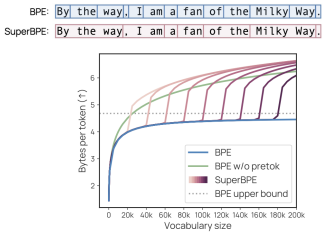

Figure 1: **From Liu et al. [2025]:** The graph compares encoding efficiency (bytes per token) of BPE versus SuperBPE. SuperBPE encodes text using fewer tokens, achieving higher efficiency that continues to improve with larger vocabularies by adding common word sequences. BPE's efficiency plateaus early, limited by whitespace-delimited words. SuperBPE also outperforms a naive BPE variant without whitespace pretokenization, with transitions from subword to superword tokens giving immediate gains.

Tokenizer Robustness and Non-Canonicality

Most tokenization algorithms are deterministic to map text to a single **canonical token sequence**, but a string s can be encoded through many non-canonical tokenizations.

The existence of non-canonical tokenizations affects how we compute string probabilities under a language model.

Tokenisation and Marginal Likelihood. Modern neural language models rely on subword tokenisers such as BPE or unigram models to provide an open

vocabulary. However, Cao and Rimell [2021] highlight that these tokenisers introduce ambiguity, since many surface strings admit multiple valid segmentations. Standard evaluation uses only the tokeniser's one-best segmentation, implicitly assuming that the chosen tokenisation is both correct and representative. This practice **conflates tokeniser quality with model quality**: if the tokeniser chooses a suboptimal segmentation—especially in out-of-domain settings—the measured perplexity may significantly misrepresent a model's true ability. Cao and Rimell [2021] suggest that a more principled approach evaluates the model on its *marginal likelihood over all possible tokenisations*, thereby isolating intrinsic model performance from tokeniser-induced artifacts.

Estimating the Marginal Likelihood. Computing the exact marginal likelihood is infeasible because the number of tokenisations grows exponentially with sequence length. Instead, the marginal can be approximated via importance sampling, using the probabilistic unigram tokeniser as the proposal distribution. Each sampled segmentation receives an importance weight proportional to the ratio between the language model probability and tokeniser probability. To reduce estimator variance, the paper investigates sampling without replacement using adaptations of stochastic beam search, incorporates consistent tokenisation of repeated word types within a document, and compares these approaches to an n-best Viterbi approximation. Experiments show that with an appropriate estimator and a modest number of samples, marginal likelihood can be reliably approximated and often yields substantially lower perplexity than the one-best baseline.

Tokeniser Uncertainty, Caching Effects, and Implications. Empirical results reveal that the *marginal gap*—the improvement of marginal perplexity over one-best perplexity—grows with tokeniser uncertainty, quantified via the entropy of the segmentation lattice. Higher entropy indicates many plausible tokenisations, and correspondingly large discrepancies between the model's preferred segmentations and the tokeniser's choices. Moreover, language models exhibit strong caching effects: if a word reappears with the same segmentation, its loss drops significantly, but changes in tokenisation eliminate this advantage. These findings imply that tokenisers are a key bottleneck for model generalisation, especially out-of-domain. Evaluating or even training models with multiple sampled segmentations may therefore improve robustness and better reflect the true competence of modern language models.

Formal Perspectives on Tokenization

- **Tokenization is NP-Complete:** Recent work by Whittington et al. [2025] rigorously proves that both direct and bottom-up tokenisation, when formulated as the problem of compressing a dataset to a fixed number of symbols, are NP-complete. Direct tokenisation seeks an optimal vocabulary of subwords to compress a dataset, while bottom-up tokenisation constructs a sequence of merge operations achieving the same goal.[10]

Practically, this means that no efficient algorithm can guarantee optimal tokenisation in general, validating the widespread use of heuristic methods like BPE or UnigramLM. For NLP researchers, this establishes a theoretical

[10] This is achieved by reductions from the NP-hard max-2-SAT problem and also discuss variants such as deduplicated datasets, single long strings, and hybrid approaches.

ceiling on tokenisation optimization and highlights the importance of designing approximate algorithms and proxy objectives (e.g., compression, unigram likelihood) to balance computational feasibility, downstream performance, and multilingual fairness in model training.

Tokenizer Evaluation

Morphological Segmentation v {BPE, WordPiece, UnigramLM}

TL;DR: Findings comparing linguistically-motivated segmentations v data-driven ones are inconclusive

- Gutierrez-Vasques et al. [2023] find that early WordPiece merges do not correspond to typological patterns, unlike BPE. The algorithm has a strong preference for pairs with high joint probability, even if individually rare; thus, creating a tendency to merge rare-character patterns early on. WordPiece also produces longer subwords (more stem-like) earlier.

We focus on the intrinsic evaluation benchmark proposed in**?**. It consists of four information measures: Morphological Alignment, Cognitive Plausibility, Rényi Efficiency, and Fertility:

Goldman et al. [2024] conduct experiments that show there correlation between tokenizers' compression and models' downstream performance, suggesting that compression is a reliable intrinsic indicator of tokenization quality.

Lotz et al. [2025] introduce a set of intrinsic tokenization metrics that correlate more strongly with multilingual downstream performance, moreso than traditional text prediction. These intrinsic tokenizer metrics provide complementary perspectives on the statistical properties of a tokenizer's output. **Token Cardinality (CARDINALITY)** quantifies the overall vocabulary size, reflecting the granularity of tokenization and its potential impact on model capacity and generalization. **Rank-Frequency AUC (AUC)** summarizes the distribution of token frequencies across ranks, capturing how closely the tokenizer approximates the expected heavy-tailed distribution of natural language. **Slope of Linear Fit (SLOPE)** explicitly measures the adherence of the token distribution to a Zipfian power-law, providing insight into whether frequent and rare tokens are represented in proportionate ways. Finally, **Deviation from Power Law (POWER LAW)** quantifies the discrepancy between observed token frequencies and the ideal Zipfian fit, highlighting systematic deviations that may indicate suboptimal token splitting. Together, these metrics were selected to capture both the size and structure of the token distribution, balancing vocabulary richness with statistical naturalness.

Tokenization as Channel Efficiency: Rényi efficiency [Zouhar et al., 2023a]

Rényi efficiency

Rényi efficiency is a measure that penalises vocabularies consisting of many low-frequency or high-frequency tokens. It captures efficient channel usage and was found to correlate highly with BLEU scores [Zouhar et al., 2023a].

Zouhar et al. [2023a] introduces the concept that a good tokenizer should use the channel efficiently. Here, tokenization is framed as preparing text for transmission over a noiseless communication channel. Efficient tokenizers produce balanced token distributions, making optimal use of the available vocabulary.

Efficiency is important because token distributions with extremely frequent tokens waste capacity—the model repeatedly sees trivial pieces. Conversely, distributions with many rare tokens are harmful, as the model sees too few examples to learn them effectively. Therefore, a tokenizer should strike a balance, avoiding both extremely common and extremely rare tokens. The central claim is that better tokenizers yield token distributions that serve as "efficient" encodings of the text.

Shannon Entropy is Insufficient Shannon entropy measures the average code length under optimal compression. However, Shannon-optimal encodings assign very short codes to frequent tokens and extremely long codes to rare tokens. While theoretically optimal for compression, this encourages the presence of many rare tokens, which impairs model learning. Empirically, Shannon entropy correlates only weakly with downstream performance, for example, BLEU scores in machine translation show a correlation of approximately 0.22.

Rényi Entropy as a Superior Metric To overcome the limitations of Shannon entropy, the authors propose using Rényi entropy with a tunable parameter α.

Let Δ be a token alphabet and p_Δ the unigram distribution over Δ. The **Rényi entropy** of order $\alpha > 0$ is defined as:

$$H_\alpha(p_\Delta) = \frac{1}{1-\alpha} \log \sum_{\delta \in \Delta} p_\Delta(\delta)^\alpha. \tag{1}$$

Rényi entropy generalizes Shannon entropy, allowing non-linear weighting of token probabilities, which is useful for designing tokenizers that balance frequent and rare tokens.

- Low α behaves similarly to Shannon entropy.
- High α penalizes distributions with rare tokens more aggressively.

Special cases include:

- $\alpha \to 1$: $H_1(p_\Delta)$ reduces to the Shannon entropy $H(p_\Delta)$.
- $\alpha \to 0$: $H_0(p_\Delta) = \log |\Delta|$, corresponding to uniform code lengths.
- $\alpha \to \infty$: $H_\infty(p_\Delta) = \max_{\delta \in \Delta} - \log p_\Delta(\delta)$, dominated by the rarest token.

The study finds that Rényi entropy with $\alpha \approx 2.5$ correlates strongly with BLEU scores (correlation 0.78). This choice of α reflects the intuition that rare tokens should be penalized strongly, extremely frequent tokens should be discouraged, and the distribution should be balanced around mid-frequency tokens. This principle underlies the *Compression Principle*: downstream model performance should correlate with the Rényi efficiency of the tokenizer, i.e., its normalized, discounted expected code length.

Characteristics of Good Tokenizers Rather than advocating specific algorithms (e.g., BPE, Unigram, WordPiece), the paper identifies statistical properties of effective tokenizers:

- Avoid very low-frequency tokens, which models cannot learn effectively.
- Avoid extremely high-frequency tokens, which waste capacity.
- Achieve mid-range dispersion controlled by Rényi entropy $\alpha \approx 2.5$.
- Produce distributions with high Rényi efficiency and balanced channel usage.

In practice, tokenizers that score well under $\alpha \approx 2.5$ typically have moderate vocabulary sizes, balanced subword frequency spectra, and learned segmentations that avoid both over-segmentation (character-like behavior) and under-segmentation (few very frequent long tokens). Examples likely to perform well include Unigram LM (e.g., SentencePiece Unigram) and appropriately tuned BPE, while WordPiece performance is intermediate depending on hyperparameters.

Two Counterexamples [Cognetta et al., 2024] Cognetta et al. [2024] introduces two counterexamples to illustrate that Rényi efficiency is not a perfect predictor:

RANDOM-DROP BPE This approach selects a subset of frequent subwords and decomposes them into smaller subwords. While this increases Rényi efficiency, it can decrease BLEU because sequences become longer and harder for the model to learn. Even though the efficiency metric rises, downstream performance may drop substantially.

DUPLICATION BPE Here, high-frequency tokens are duplicated in the vocabulary with different embeddings but identical surface forms. This also increases Rényi efficiency by spreading probability mass across duplicates. However, it often dramatically reduces BLEU, as the model treats duplicates as independent tokens, confusing learning.

Experimental Findings The experiments demonstrate that both RANDOM-DROP and DUPLICATION tokenizers can increase Rényi efficiency while reducing BLEU. Restricting decomposition to the most frequent tokens in RANDOM-DROP tends to lower BLEU despite higher efficiency. DUPLICATION shows that even small duplication factors can severely harm downstream performance. Other intrinsic metrics, such as percentile frequency or average sequence length, occasionally succeed where Rényi efficiency fails but also fail in certain cases, emphasizing the limitations of single-metric evaluation.

Key insights from the paper include:

- Rényi efficiency is a useful but brittle intrinsic metric.
- Tokenizers can "game" efficiency metrics without improving downstream performance.
- Effective tokenizer evaluation requires considering interactions between tokenization and model learning, not only unigram statistics.
- The paper provides a framework to design and test tokenizers that challenge intrinsic metrics, motivating the development of richer evaluation methods.

While Zouhar et al. [2023a] claim the best tokenizers are those whose token distributions maximize Rényi efficiency with $\alpha \approx 2.5$, balancing frequent and rare tokens and correlate strongly with improved downstream performance, counterexamples like RANDOM-DROP and DUPLICATION BPE reveal the limitations of relying solely on Rényi efficiency, underscoring the need for metrics that integrate both token statistics and model learning dynamics.

Other Intrinsic Tokenizer Evaluation Metrics

Fertility

Fertility is the average number of subwords produced per tokenised word, used by Ács [2019] to compare compression efficiency across languages for a multilingual tokeniser. A score of 1 indicates perfect compression; every word in the evaluation set exists in the tokeniser's vocabulary.

Standard tokenization metrics like fertility measure average tokens per word but obscure cross-lingual vocabulary allocation and fairness. Recently, Nayeem et al. [2025] introduce the **Single Token Retention Rate (STRR)** to quantify how many words are preserved as single tokens, offering an interpretable, fairness-sensitive diagnostic.

STRR analysis shows English consistently has high single-token retention, Chinese receives strong support, and Hindi suffers from fragmentation. Fertility alone misses these disparities, highlighting systematic prioritization of certain languages in multilingual tokenizers. The paper proposes a four-stage vocabulary-expansion pipeline—core vocabulary identification, vocabulary injection, pretraining, and multilingual instruction tuning—using STRR to guide interventions, aiming to reduce subword fragmentation and improve equitable token coverage across languages.

Morphological alignment

The alignment of tokenisers with gold-standard morphological segmentations of words.

The benchmark compares tokeniser alignment to morphological annotations from seven resources and returns a macro-averaged F1 score. The morphological data comes from LADEC (Gagné et al., 2019), MorphoLex (SánchezGutiérrez et al., 2018), MorphyNet (Batsuren et al., 2021), and DagoBert (Hofmann et al., 2020).

Uzan et al. [2024] further augment these datasets with morpheme segmentation data (Batsuren et al., 2022), novel blend structure detection data (Pinter et al., 2021), and compound separation data (Minixhofer et al., 2023).

Cognitive plausibility

A benchmark from Beinborn and Pinter [2023] who found that human reaction time and accuracy from a lexical decision task correlated negatively with the number of tokens produced by a tokeniser for words, and correlated positively for nonwords. The score consists of an average from correlation scores across the four conditions (words/nonwords, accuracy/reaction time) with a higher score indicating increased 'cognitive plausibility'.

Multilinguality, Morphological Alignment and Tokenization

Understanding tokenization quality is important for equalising model performance cross-lingually; it is an issue that disproportionately affects low-resourced languages [Petrov et al., 2023]. Tokenization plays a central role in modern NLP systems, yet widely used frequency-based methods such as BPE and UnigramLM tend to privilege high-resource languages. Because these algorithms optimize for global frequency statistics, languages with large corpora exert disproportionate influence on the learned vocabulary. As a result, lower-resource languages often receive longer or less coherent segmentations, and may even suffer increased rates of <UNK> tokens. These disparities not only raise computational costs for underrepresented languages but also reinforce broader inequities in multilingual NLP.

Multilingual language models rely heavily on subword tokenizers to represent input text across languages, yet the properties of these tokenizers remain poorly understood. Limisiewicz et al. [2023] introduces two core dimensions for evaluating multilingual tokenizers: *vocabulary allocation*, which measures how effectively a tokenizer dedicates its vocabulary to language-specific lexical units, and *vocabulary overlap*, which captures how much tokenization units are shared across languages. By examining tokenizers trained with Unigram LM, BPE, and two proposed methods for merging monolingual tokenizers (NOOVERLAP and TOKMIX), the study highlights substantial variation in how different tokenization strategies represent diverse languages:

- In the NOOVERLAP setting, Limisiewicz et al. [2023] train a separate Unigram tokenizer for each of the L languages, each with vocabulary size N/L, and during multilingual tokenization we apply the language-specific tokenizer independently so that the final vocabulary is a concatenation of L disjoint segments. As a result, identical character sequences across languages may receive different token IDs because tokenization is fully language-specific.
- In the TOKMIX setting, Limisiewicz et al. [2023] first train Unigram tokenizers for each of the L languages and then compute an averaged vocabulary distribution by weighting each monolingual tokenizer with w_i and forming $\hat{\theta} = \sum_{i=1}^{L} w_i \theta_i$, after which we sort these merged units by probability and truncate to a shared vocabulary of size N. Unlike NOOVERLAP, identical units across languages are merged into a single token with probability determined by the averaged distribution.

Experiments using masked language modeling and a range of downstream tasks—POS tagging, dependency labeling, NER, NLI, and cross-lingual sentence retrieval—reveal that tokenizer properties strongly influence model performance. High vocabulary allocation consistently improves word-level tasks, suggesting that richer language-specific segmentation leads to better lexical representations. In contrast, greater vocabulary overlap provides clear benefits for NER and sentence-level cross-lingual tasks such as retrieval and NLI, but harms syntactic tasks like POS and dependency labeling. Surprisingly, tokenizers that yield strong masked-LM performance often perform worse on downstream tasks, due in part to the increased lexical ambiguity associated with larger effective vocabularies.

These findings offer practical guidance for designing multilingual tokenizers prior to costly model pre-training. Allocation and overlap capture complementary aspects of multilingual lexical representation, and their effects vary systematically by task type: word-level syntactic tasks prefer high allocation and low overlap, while semantic and cross-lingual tasks benefit from shared subword units.

MorphScore: Evaluating Multilingual Morphological Alignment

One proposed dimension of tokenizer evaluation is **morphological alignment**, which measures the extent to which token boundaries correspond to morpheme boundaries. MorphScore [Arnett and Bergen, 2025] initially covered 22 languages and quantified alignment by comparing predicted tokenizations to gold-standard morpheme segmentations. Arnett et al. [2025b]expands MorphScore to 70 languages, adds parameter flexibility such as frequency weighting and inclusion/exclusion of one-token words, and incorporates additional linguistic information like part-of-speech and morphological features from Universal Dependencies.

Given a dataset of words annotated with their morpheme boundaries, MorphScore evaluates whether a tokenizer places token boundaries at those same positions.

For a word w with a linguistically annotated morpheme boundary at position b, define:

$$\text{MorphScore}(w) = \begin{cases} 1 & \text{if a tokenizer boundary occurs at } b, \\ 0 & \text{otherwise.} \end{cases}$$

Words consisting of a single token (i.e., the entire word is in the vocabulary) are *excluded* from evaluation so that the tokenizer is not penalized for producing no segmentation.

The overall MorphScore for language ℓ is the mean value:

$$\text{MorphScore}(\ell) = \frac{1}{N_\ell} \sum_{w \in \mathcal{D}_\ell} \text{MorphScore}(w),$$

where $\mathcal{D}_\ell$ is the set of annotated words for language ℓ.

MorphScore uses manually or semi-automatically annotated morphological datasets drawn from Universal Dependencies and UniMorph. For this evaluation:

- 22 languages were selected.
- Half were agglutinative and half fusional.
- Languages were balanced for resource level and diversity of language families.
- Each dataset consists of words with exactly one target morpheme boundary (except Korean, where the leftmost boundary is used).
- All datasets contain at least 100 items, sampled up to 2000 per language.

Table illustrates how MorphScore is computed.

Language	Word	Segmentation	Tokenizer Output / Score
Basque	*aldiz*	aldi + z	[al, diz] → 0 [aldi, z] → 1
Croatian	*suučešnika*	suučešnik + a	various segmentations → 0
Icelandic	*samráðs*	samráð + s	[samrað, s] → 1
Greek		+	[, , ,] → 1

Table 1: Example MorphScore calculations for several languages.

The dataset construction focuses on fusional and agglutinative languages, excluding non-concatenative and isolating languages due to complexities in determining gold morpheme segmentations. Each word is segmented into stem and affixes when possible, excluding irregular forms. The scoring system evaluates both boundary-level precision/recall and subword-level precision, recall, and F1, which penalizes oversegmentation and provides a more nuanced assessment of tokenization quality than raw accuracy.

Two predictions of the morphological alignment hypothesis are:

1. Agglutinative languages should have *lower* MorphScores than fusional languages.
2. Better MorphScores should correlate with better language model performance.

Empirical findings contradict both:

- Agglutinative languages show **higher** MorphScores (66.3% vs. 53.3%, $p = .008$).
- MorphScore does **not** correlate with perplexity ($p = 0.58$).

Further analyses show that this is not attributable to:

- longer average word length, or
- higher tokenization fertility,

even though both are indeed higher in agglutinative languages.

Where the original MorphScore evaluates tokenizer alignment using a single boundary-identification metric on small, carefully selected datasets, Arnett et al. [2025b] substantially extends the framework along three axes. First, it introduces a far broader multilingual evaluation set, constructing gold segmentations for 86 languages (70 after filtering) using UD morphology and explicit concatenative stem–affix reconstruction, while excluding

non-concatenative and irregular forms that would yield misleading gold boundaries. Second, it generalizes scoring by moving beyond MorphScore's single boundary-accuracy measure to a richer suite of metrics, including macro boundary precision/recall and micro/macro subword precision, recall, and F1, thereby penalizing oversegmentation and capturing alignment at both boundary and morpheme-span levels. Third, it systematically analyzes design choices absent from the original work—such as frequency scaling and the inclusion of single-token words—and empirically identifies default settings that are most predictive of downstream model performance. Together, these extensions transform MorphScore from a narrow diagnostic into a large-scale, configurable evaluation framework capable of quantifying morphological alignment across diverse languages, tokenizers, and modeling conditions.

While the expanded MorphScore provides a comprehensive framework for evaluating morphological alignment across 70 languages, its predictive power for model performance is limited. The study highlights the need to combine morphological alignment with other intrinsic metrics, such as compression or Rényi efficiency, to better understand tokenizer quality. The released datasets and evaluation code enable further multilingual research and aim to improve equity in language technology across diverse languages.

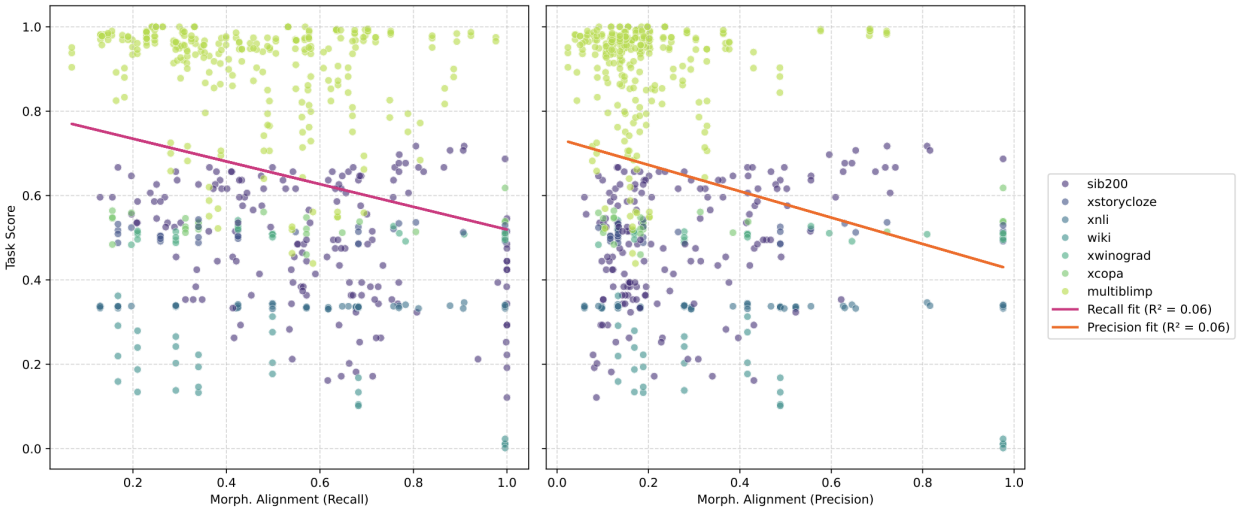

Figure 2: Correlation between model performance on different tasks (color-coded) for Morphological Alignment recall (left) and Morphological Alignment precision (right). Figure from Arnett et al. [2025b]

Poelman et al. [2025] investigates the relationship between language morphology and the difficulty of language modelling, arguing that prior mixed findings are largely due to confounding factors in experimental designs. This critiques prior work that suggests AGGLUTINATIVE LANGUAGES (ALs) are harder to model than FUSIONAL LANGUAGES (FLs), pointing out that differences in **tokenization alignment**, **efficiency**, and **dataset size** may be **conflated with morphological effects**. They aim to identify these confounding factors, re-evaluate existing hypotheses, and propose metrics that better capture the intrinsic difficulty of modeling different languages. Morphological complexity is often measured using metrics like mean word length, type-token ratio, and conditional entropy, which serve as proxies for word formation complexity. However, these metrics either ignore how text is

tokenized or rely on expert annotations, limiting their utility for large-scale language modeling studies. The authors emphasize the need for corpus-based metrics that operate on the same units used by language models.

- **Six Confounding Factors:** language selection, grouping of languages, choice of tokenization algorithm, vocabulary versus data size, corpus domain, and performance metrics. Each factor can bias conclusions; for instance, different tokenizers or unbalanced corpora can artificially inflate or deflate model perplexities. They argue that without controlling for these variables, prior claims about ALs versus FLs are unreliable, and experimental setups must carefully consider these factors to draw valid inferences.
- **New Metrics:** token bigram metrics—accessor variety (AV), accessor uniqueness (AU), and entropic efficiency ()—as gradient, corpus-based proxies for morphological complexity that do not require expert annotations. These metrics capture the diversity and predictability of token sequences, reflecting the intrinsic difficulty of next-token prediction for language models. Applying these metrics across multiple parallel and non-parallel corpora, they show that ALs generally have higher AV and , suggesting that the greater number of equally probable token continuations, rather than morphology per se, contributes to higher perplexity.

Parity-Aware BPE [Foroutan et al., 2025]

Parity-aware BPE is proposed as a corrective to these cross-lingual imbalances. Rather than selecting merges purely based on raw co-occurrence counts, this variant adapts BPE's merge rule to focus on the language currently experiencing the poorest compression rate. At each merge step, the algorithm chooses the operation that improves the worst-off language the most. This approach introduces a principled notion of fairness into the tokenization process: it intentionally sacrifices a small amount of global frequency optimality to ensure that no language is consistently disadvantaged.

Despite this shift in optimization priority, parity-aware BPE preserves much of the efficiency and effectiveness of standard BPE. In practice, it succeeds in equalizing per-language token counts while maintaining comparable overall compression performance. Importantly, downstream language model quality remains stable, indicating that fairness-oriented tokenization need not come at the cost of accuracy. Together, these results suggest that equitable tokenization is both feasible and beneficial for multilingual systems.

Token Premiums

Token premiums are cross-linguistic differences in the number of tokens needed to encode parallel text— these disparities persist even in *monolingual* tokenizers trained with identical data size, vocabulary size, and implementation. Token premiums matter because they reduce compression, increasing training and inference costs for certain languages. Some languages have extremely high byte premiums (Burmese: 3.51, Dzhongkha: 3.64; Shan: 3.94; Petrov et al. [2023]).

Byte premiums refer to **cross-lingual differences in the number of bytes required by a language to encode equivalent content**.

There have been some attempts to reduce token premiums in multilingual byte-based tokenizers:

MYTE [Limisiewicz et al., 2024] replaces long byte strings with unused bytes that represents meaningful word segments, morphemes. This method leads to significantly reduced token premium, or byte premium, effects, but it requires a predefined dictionary of morphemes.

MAGNET [Ahia et al., 2024] takes byte-level inputs and uses script-specific boundary-predictor modules to dynamically predict segmentation boundaries. Those boundaries are used to pool together byte-level representations before feeding them into the language model as input. This leads to more equitable segmentation across languages.

Why do these differences arise? And is it possible to mitigate token premiums in subword tokenizers without modifying a tokenizer algorithm?

[11] analyse how *byte premiums*, tokenizer algorithm (BPE vs. Unigram), and scaling training data by byte premium affect compression. They find that **BPE provides the best compression overall**, and that byte-premium scaling has no meaningful impact. Even under perfectly controlled training conditions, token premiums vary widely across languages.

[11] Catherine Arnett, Tyler A Chang, Stella Biderman, and Ben Bergen. Explaining and mitigating crosslingual tokenizer inequities. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a

This motivates the search for linguistic and tokenizer-design factors that explain compression variation. By training roughly **7,000 monolingual tokenizers** across **97 languages**, systematically varying tokenizer type, vocabulary size, and dataset size, this allows the authors to examine potential predictors of token premiums. Three factors explain substantial variance: **train–evaluation data similarity**, **mean token length in actual usage**, and **the proportion of whitespaces in a language**. Data similarity explains the most variance, though even it accounts for less than one-quarter of the differences. Whitespace frequency emerges as a key culprit because traditional pre-tokenization prevents merges across whitespace, disproportionately harming languages that rely heavily on space-delimited words.

Given these findings, the paper evaluates interventions for reducing token premiums. Training on **parallel data** slightly improves compression but does *not* meaningfully reduce token premiums. Adjusting **vocabulary size** helps substantially: each language has an "optimal" vocabulary size where compression becomes comparable across languages. The strongest improvements come from **SuperBPE tokenizers**, which allow merges across whitespaces. These superword tokenizers consistently achieve lower token premiums and reduced variance across languages. Finally, Arnett et al. [2025a] note that some residual disparities remain due to inherent linguistic and script-based factors, such as character length and UTF-8 encoding differences. While future work may address these issues—for instance, through alternative Unicode encodings—the present study provides the first large-scale, crosslinguistic analysis of monolingual token premiums. It shows that crucial design choices like vocabulary size and pre-tokenization significantly influence compression fairness in multilingual NLP systems.

Scripts-Diverse Languages

Tokenization methods often fail to fairly represent complex scripts like Tamil, Sinhala, and Hindi, primarily due to pre-tokenization choices [Velayuthan and Sarveswaran, 2025].

Existing tokenization schemes—either byte-based or Unicode codepoint-based—impose structural biases, such as penalizing non-Latin scripts due to UTF-8's variable-length encoding and enabling merges that produce tokens containing partial or invalid character sequences.

To address these issues, Land and Arnett [2025] propose SCRIPT, a new encoding system called Script Category Representation in PreTokenization. SCRIPT replaces UTF-8 bytes with a linguistically motivated representation based on two Unicode properties: script (e.g., Latin, Han) and supercategory (letters/marks, punctuation/symbols, numbers, separators, other). Each character is represented using exactly two tokens: one identifying its script-supercategory block and one identifying the character index within that block. This eliminates byte-length disparities between scripts and enforces consistent, meaningful character granularity.

SCRIPT naturally enables a simple, rule-based pretokenization strategy that groups consecutive characters by shared script and supercategory, bypassing the need for brittle regular expressions. The authors also introduce constrained BPE merging, which restricts merges to operations that maintain character integrity. This prevents the formation of tokens that mix full and partial characters—an issue common in byte-level BPE due to greedy merging across UTF-8 boundaries. These constraints reduce invalid or semantically incoherent tokens and simplify downstream processing. SCRIPT-BPE is evaluated in both monolingual and multilingual settings, comparing it to byte-based BPE using OpenAI's cl100k and o200k pretokenizers. Across languages, constrained merges consistently improved compression and eliminated partial-character tokens. SCRIPT-BPE showed strong, stable compression performance across diverse scripts, outperforming byte-based approaches in many languages where regex pretokenizers fail—such as Thai, Hindi, and Punjabi—while remaining competitive for languages where regex patterns are highly optimized (e.g., Chinese with o200k).

Current Perspectives

Tokenization is not "solved" in downstream NLP:

- Transformer layers don't fully fix tokenization issues, especially for multilingual, low-resource, or off-domain settings.
- Evaluation is challenging: intrinsic metrics (BPC, fertility) don't always correlate with downstream performance.

Yuval Pinter classifies three perspectives in the latest technical tokenizer research – Join Them (innovate within the confines of current practice), Beat Them (break the confines of tokenizers – *enter tokenizer-free language models*) and Fix Them (a more "restorative" philosophy). [12]

- **Beat Them:** We've seen a trend in byte/character level models that "beat" traditional tokenization (**CANINE, ByT5, MambaByte, PIXEL, Byte Latent Transformers, H-Net**). Sometimes bypassing tokens entirely (bytes/pixels) can outperform traditional subword methods?
- **Join Them:** Most people who use NLP models want a direct mapping of words/subwords to embeddings, and some code to convert raw text to embeddings. Ideally, we'd also like to **swap the tokenizer or subword vocabulary without changing the model itself**.

[12] **Beat them? Join them? Fix them? Tokenization Research in a Downstream World** (Yuval Pinter, ICML TokShop Keynote: https://icml.cc/media/icml-2025/Slides/48832.pdf

- **Fix Them:** Focuses on **improving existing BPE/unigram tokenizers**, focusing on the importance of **auxiliary improvements in the NLP pipeline**, not just the main model.
  - **BPE-Dropout, PathPiece, FLOTA, Trimmed-BPE, BPE-knockout, Picky-BPE** for inference/vocab improvements.[13]
  - **Splinter**: addresses nonconcatenative languages (Hebrew, Arabic, Malay) by learning a pre-processing step to linearize morphemes.
  - Also touches on improving pretokenization and handling rare/intermediate tokens.

[13] Tokenization improvements for inference and vocabulary: **BPE-Dropout** randomly skips merges to improve robustness; **PathPiece** keeps the vocabulary but makes inference maximally compressive; **FLOTA** selects the longest matching token during inference; **Trimmed-BPE** removes rare intermediate tokens post-vocabulary-building; **BPE-knockout** prunes tokens using morphological information; **Picky-BPE** selectively discards tokens during vocabulary construction when suboptimal.

Dynamic Patching and *Tokenizer-Free Language Models?* Well, there's "no tokenizer free lunch" (Arnett 2025)

Recently, a new wave of dynamic tokenization models has come about. This includes Byte Latent Transformer, Hierarchical Autoregressive Transformer, and H-Net.

Dynamic tokenization strategies like the Byte Latent Transformer (BLT), Hierarchical Autoregressive Transformer (HAT), and H-Net represent a shift away from traditional fixed-vocabulary tokenizers by learning representations over flexible, variable-length units rather than predefined subwords. In these approaches, the model typically starts with a small, fixed set of base units—often bytes or characters—each associated with a learned embedding. These base units are then grouped into higher-level structures such as chunks, patches, or latent units, whose embeddings capture contextual information over the sequence of base elements. By dynamically determining the boundaries and composition of these units, the models can adaptively represent rare words, morphological variations, or multilingual content without relying on a static vocabulary, which can reduce out-of-vocabulary issues and improve generalization across domains.

Despite being called "tokenizer-free," as Arnett notes in her HuggingFace Blogpost,[14] these models still rely on a minimal fixed vocabulary of base units, since embeddings must exist for the individual bytes or characters. The novelty lies in how these base embeddings are aggregated into higher-level representations that serve as the input to the main model. For instance, in BLT, sequences of bytes are compressed into latent embeddings via an autoencoding process; in HAT, hierarchical autoregressive modeling constructs representations over progressively larger spans; and H-Net similarly learns chunk embeddings that integrate contextual information across variable-length sequences. These dynamic embeddings function analogously to subword embeddings in standard transformers, allowing the model to process sequences flexibly while leveraging learned compositional structures, effectively bridging the gap between fine-grained byte representations and semantically meaningful units.

[14] Check out Catherine's Blogpost on HuggingFace: **There is no such thing as a tokenizer-free lunch,** `https://huggingface.co/blog/catherinearnett/in-defense-of-tokenizers`

More General, Less-Linguistic Tokenizer Issues

- **Multimodal Tokenization:** Tokenization strategies for images, audio, video, and other modalities, including methods to align representations across different types of data. **Pixel Language Modelling** and Tokenization for **Omni-Models** (Speech, Text, Image/Video) are avenues that also explore **alternative input representations**

- **Tokenizer Transfer and Post-Training:** What methods can be developed to update tokenizers after model training to boost performance and/or efficiency without retraining from scratch?
- **Tokenization and Safety:** *Lots* of important research activity here...

Noncanonical tokenizations are both semantically meaningful and capable of bypassing safety measures, underscoring the brittleness of current alignment approaches and motivating the integration of safety considerations into LLM pre-training. One common strategy, retokenizing inputs, can mitigate the vulnerability only if the tokenizer is bijective, which is not the case for many deployed models. Recently, Geh et al. [2025] highlight that since LLMs retain semantics of non-canonical tokenizations, this raises the potential for exploiting tokenizations as an axis for adversarial attacks. This leads the authors to introduce **adversarial tokenization**, where malicious prompts are retokenized without altering their text. This allows attackers to bypass safety and alignment mechanisms while still eliciting meaningful responses from models.

They formulate tokenization distance using a Multi-Rooted Multi-valued Decision Diagram (MRMDD) to efficiently sample and explore noncanonical tokenizations. They empirically show that semantic signal is preserved even as tokenizations move farther from canonical forms. Leveraging this property, Geh et al. [2025] design AdvTok, a greedy local search algorithm that iteratively explores small neighborhoods of tokenizations to maximize the probability of generating a target response. AdvTok is evaluated across three adversarial scenarios—jailbreaking, evading safety classifiers, and prompt injection—and demonstrates that adversarial tokenizations alone are sufficient to outperform or enhance existing state-of-the-art attack methods without changing the underlying text. The authors argue that a core safety issue lies in the safety post-training stage, which is limited in scope compared to pre-training, allowing adversarial tokenizations to exploit unaligned probability mass in the token distribution.

References

Orevaoghene Ahia, Sachin Kumar, Hila Gonen, Valentin Hofmann, Tomasz Limisiewicz, Yulia Tsvetkov, and Noah A Smith. Magnet: Improving the multilingual fairness of language models with adaptive gradient-based tokenization. *Advances in Neural Information Processing Systems*, 37:47790–47814, 2024.

Catherine Arnett and Benjamin Bergen. Why do language models perform worse for morphologically complex languages? In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert, editors, *Proceedings of the 31st International Conference on Computational Linguistics*, pages 6607–6623, Abu Dhabi, UAE, January 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-main.441/.

Catherine Arnett, Tyler A Chang, Stella Biderman, and Ben Bergen. Explaining and mitigating crosslingual tokenizer inequities. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a.

Catherine Arnett, Marisa Hudspeth, and Brendan O'Connor. Evaluating morphological alignment of tokenizers in 70 languages. In *Proceedings of the TokShop: Workshop on Tokenization at ICML 2025*, Vancouver, Canada, 2025b.

Lisa Beinborn and Yuval Pinter. Analyzing cognitive plausibility of subword tokenization. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4478–4486, Singapore, December 2023. Association for Computational Linguistics. DOI: 10.18653/v1/2023.emnlp-main.272. URL https://aclanthology.org/2023.emnlp-main.272/.

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, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 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/.

Marco Cognetta, Vilém Zouhar, Sangwhan Moon, and Naoaki Okazaki. Two counterexamples to tokenization and the noiseless channel. In Nicoletta Calzolari, Min-Yen Kan, Veronique Hoste, Alessandro Lenci, Sakriani Sakti, and Nianwen Xue, editors, *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 16897–16906, Torino, Italia, May 2024. ELRA and ICCL. URL https://aclanthology.org/2024.lrec-main.1469/.

Negar Foroutan, Clara Meister, Debjit Paul, Joel Niklaus, Sina Ahmadi, Antoine Bosselut, and Rico Sennrich. Parity-aware byte-pair encoding: Improving cross-lingual fairness in tokenization. *arXiv preprint arXiv:2508.04796*, 2025.

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

Renato Geh, Zilei Shao, and Guy Van Den Broeck. Adversarial tokenization. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 20738–20765, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. DOI: 10.18653/v1/2025.acl-long.1012. URL https://aclanthology.org/2025.acl-long.1012/.

Omer Goldman, Avi Caciularu, Matan Eyal, Kris Cao, Idan Szpektor, and Reut Tsarfaty. Unpacking tokenization: Evaluating text compression and its correlation with model performance. *arXiv preprint arXiv:2403.06265*, 2024.

Ximena Gutierrez-Vasques, Christian Bentz, and Tanja Samardžić. Languages through the looking glass of bpe compression. *Computational Linguistics*, 49(4):943–1001, 2023.

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

Sander Land and Catherine Arnett. Bpe stays on script: Structured encoding for robust multilingual pretokenization. *arXiv preprint arXiv:2505.24689*, 2025.

Tomasz Limisiewicz, Jiří Balhar, and David Mareček. Tokenization impacts multilingual language modeling: Assessing vocabulary allocation and overlap across languages. *arXiv preprint arXiv:2305.17179*, 2023.

Tomasz Limisiewicz, Terra Blevins, Hila Gonen, Orevaoghene Ahia, and Luke Zettlemoyer. Myte: Morphology-driven byte encoding for better and fairer multilingual language modeling. *arXiv preprint arXiv:2403.10691*, 2024.

Alisa Liu, Jonathan Hayase, Valentin Hofmann, Sewoong Oh, Noah A. Smith, and Yejin Choi. Superbpe: Space travel for language models. In *Proceedings of the Conference on Language Modeling (COLM)*, To appear, 2025. Association for Computational Linguistics.

Jonas F Lotz, António V Lopes, Stephan Peitz, Hendra Setiawan, and Leonardo Emili. Beyond text compression: Evaluating tokenizers across scales. *arXiv preprint arXiv:2506.03101*, 2025.

Clara Meister. Unigramlm: An attempt at writing the missing manual. `https://cimeister.github.io/blog/unigramlm/`, 2025. Accessed: 2025-11-23.

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

Amir More, Özlem Çetinoğlu, Çağrı Çöltekin, Nizar Habash, Benoît Sagot, Djamé Seddah, Dima Taji, and Reut Tsarfaty. CoNLL-UL: Universal morphological lattices for Universal Dependency parsing. In Nicoletta Calzolari, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Koiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, Stelios Piperidis, and Takenobu Tokunaga, editors, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA). URL `https://aclanthology.org/L18-1608/`.

Mir Tafseer Nayeem, Sawsan Alqahtani, Md Tahmid Rahman Laskar, Tasnim Mohiuddin, and M Saiful Bari. Beyond fertility: Analyzing strr as a metric for multilingual tokenization evaluation. *arXiv preprint arXiv:2510.09947*, 2025.

Aleksandar Petrov, Emanuele La Malfa, Philip Torr, and Adel Bibi. Language model tokenizers introduce unfairness between languages. *Advances in neural information processing systems*, 36:36963–36990, 2023.

Wessel Poelman, Thomas Bauwens, and Miryam de Lhoneux. Confounding factors in relating model performance to morphology. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng, editors, *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 7273–7298, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. DOI: 10.18653/v1/2025.emnlp-main.369. URL `https://aclanthology.org/2025.emnlp-main.369/`.

Craig W Schmidt, Varshini Reddy, Chris Tanner, and Yuval Pinter. Boundless byte pair encoding: Breaking the pre-tokenization barrier. *arXiv preprint arXiv:2504.00178*, 2025.

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

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

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, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 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/.

Zhiqing Sun and Zhi-Hong Deng. Unsupervised neural word segmentation for Chinese via segmental language modeling. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4915–4920, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. DOI: 10.18653/v1/D18-1531. URL https://aclanthology.org/D18-1531/.

Omri Uzan, Craig W. Schmidt, Chris Tanner, and Yuval Pinter. Greed is all you need: An evaluation of tokenizer inference methods. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 813–822, Bangkok, Thailand, August 2024. Association for Computational Linguistics. DOI: 10.18653/v1/2024.acl-short.73. URL https://aclanthology.org/2024.acl-short.73/.

Menan Velayuthan and Kengatharaiyer Sarveswaran. Egalitarian language representation in language models: It all begins with tokenizers. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 5987–5996, 2025.

Philip Whittington, Gregor Bachmann, and Tiago Pimentel. Tokenisation is NP-complete. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 28133–28153, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. DOI: 10.18653/v1/2025.acl-long.1365. URL https://aclanthology.org/2025.acl-long.1365/.

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

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, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 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/.

Judit Ács. Exploring bert's vocabulary. Blog post, February 2019. URL https://juditacs.github.io/2019/02/19/bert-tokenization-stats.html.