# How long is a piece of string?
# A brief empirical analysis of tokenizers

**Anonymous Authors**[1]

## Abstract

Frontier LLMs are increasingly utilised across academia, society and industry. A commonly used unit for comparing models, their inputs and outputs, and estimating inference pricing is the token. In general, tokens are used as a stable currency, assumed to be broadly consistent across tokenizers and contexts, enabling direct comparisons. However, tokenization varies significantly across models and domains of text, making naïve interpretation of token counts problematic. We quantify this variation by providing a comprehensive empirical analysis of tokenization, exploring the compression of sequences to tokens across different distributions of textual data. Our analysis challenges commonly held heuristics about token lengths, finding them to be overly simplistic. These findings show that native token counts are not a model-independent measurement unit, complicating comparisons of context length, inference cost, throughput, and benchmark sequence length across foundation models. We hope the insights of our study add clarity and intuition toward tokenization in contemporary LLMs.

## 1. Introduction

Large language models (LLMs) are ubiquitous across contemporary AI research. As model capabilities continue to improve, LLMs are capturing attention more broadly, in society and industry. Frontier models follow instructions with sufficient consistency to robustly use tools, enabling them to act as agents capable of performing longer horizon tasks (Kwa et al., 2025) and economically valuable activity such as software engineering (Miserendino et al., 2025; Xu et al., 2024), scientific research (Schmidgall et al., 2025), and web-based economic tasks (Liu & Quan, 2025).

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

Fundamental to LLMs is an often overlooked process: *tokenization*. Tokenization describes the learned conversion between text characters and discrete "tokens" represented by unique numeric IDs. These token IDs correspond to items in a vocabulary—a database of all possible tokens a model can interpret and generate. When autoregressively generating new tokens an LLM samples from a probability distribution over this vocabulary. Tokenization is necessary as it enables the transformation of human-readable text characters into a numeric model-readable format. Each token ID corresponds to an embedding—a high-dimensional numeric representation that aims to capture its semantic meaning.

The token is a near-universally used unit of text sequences for describing numerous language modelling metrics, including sequence lengths, context limits, pricing, latency, and sizes of training corpora. Token counts are utilised to directly compare models, inference costs and provider platforms. Although different model series are known to implement bespoke tokenizers, and therefore tokenize text differently, tokens are used as though they are a consistent unit with differences that are trivial or averaged out over sufficient samplings or length of text. This assumption is evident in the widespread use of general heuristics for token lengths (such as one token is roughly 4 characters or 0.75 words (*e.g.*, (OpenAI, 2025a; Google, 2025))); or the comparison of API endpoint performance using number of tokens per second or usage metered as number of dollars per token.

However, as we demonstrate, token counts are not a stable unit of length: counts vary non-trivially for different domains of text or tokenizer. Therefore, when used naïvely, token counts provide an inadequate basis of comparison. In Fig. 1, we show clear variations in token boundaries and the number of tokens used by different tokenization schemes. Among frontier LLMs, a word like *antidisestablishmentarianism* is tokenized by Claude into nearly twice as many tokens (9) as Gemini or Grok (5). As our experimentation demonstrates, these differences in tokenization can result in significant differences in token counts over longer sequences.

As LLM-based systems become more prevalent, there is an emerging need for clarity around tokenization. Simply


*Figure 1.* **Tokenization schemes vary significantly**. Token boundaries are represented as shaded colours and token counts are shown in **purple**. *Antidisestablishmentarianism* is tokenized many different ways, using between **4** (Fuyu) and **9** tokens (early Llama models). Variation in tokenization is also observable for words containing fewer unique letters and more vowels, such as the Hawaiian word *humuhumunukunukuāpua'a* (Reef Triggerfish). Small changes to tokenizers within model families result in different tokenization, even in relatively simple sentences. *From (Berlin, 2013).

comparing sequence lengths and context limits based on model-specific tokenization is insufficient and misleading. Moreover, when accessed via an API (as is the case for most frontier LLMs), usage is metered by the token: accurate accounting requires a clear understanding of how a text sequence is mapped to tokens for a given text sequence.

Prior studies have explored tokenization for specific domains (Roberts et al., 2025), languages (Ahia et al., 2023) or strategies (Sälevä & Lignos, 2023), painting a valuable though incomplete picture. We extend these works and offer a comprehensive empirical analysis of tokenizer efficiencies across models and domains. We provide robust quantification of domain-specific tokenizer character compression ratios and estimates of word compression ratios as a function of word frequency and distribution. Building on these empirical findings, we examine model context limits, suggesting domain-specific character counts or model-agnostic token counts offer more directly comparable context limits. Our study is intentionally brief and focused, with clear goals of adding clarity to tokenization, quantifying differences in tokenization between models and domains, and imparting intuition on how to interpret token-based metrics.

## 2. Tokenization

During inference, an input text string is encoded into a sequence of token IDs by the tokenizer. These token IDs map to embeddings, which the LLM ingests along with positional information as input and iteratively generates a sequence of output token IDs. Finally, the tokenizer decodes this output into a human-readable text string. The exact nature of the mapping between formats and the level of compression between text characters and tokens is a trade-off balancing *efficiency* and *meaning*. Tokenizing at the character-level ensures a relatively small vocabulary, constrained by the number of unique characters, resulting in smaller embedding/output matrices at the expense of longer sequences and semantic meaning (information density) per token ('c' carries less meaning than 'car'). On the other hand, tokenizing at the word-level (or longer) results in fewer tokens that capture more semantic meaning at the expense of a larger vocabulary; this approach is subject to out-of-vocabulary issues. To compromise these conflicting objectives, tokenizers in LLMs are typically trained to tokenize at the subword level.

Tokenization typically consists of two phases: *training* and *segmentation*. **Training**: Several methods are commonly used to train tokenizers (Sennrich et al., 2016; Schuster & Nakajima, 2012; Kudo, 2018; Kudo & Richardson, 2018). Fundamental to all is the use of occurrence frequency of text strings within a training corpus. For methods like byte-pair encoding (BPE) (Sennrich et al., 2016) or WordPiece (Schuster & Nakajima, 2012), training involves iteratively merging bigrams in the corpus until the tokens defined by the merges fill the vocabulary to a desired size. **Segmentation**: Training creates a vocabulary of allowed tokens but

| Domain | Source | Description |
|---|---|---|
| *Essay* | Paul Graham essays (Graham, 2025) | Tech essays in English |
| *Technical* | arXiv* | 2x Oct-25 papers per arXiv category |
| *Code* | Transformers repository (Wolf et al., 2020) | Python code (transformers 4.46) |
| *Number* | $\pi$ | First 100k digits of $\pi$ |
| *Emojis* | Emoji dataset (Abdullah, 2024) | $\sim$5k emojis |
| *Alphanumeric* | UUIDs | Dictionary of UUID pairs |
| *Structured* | FinTabNet (Apoidea, 2024) | Financial data extracted from tables |
| *Web* | Popular webpages | Rendered HTML from 10 websites |

*Table 1.* **Text domains used in our experiments.** Unless referenced, all data was sourced by the authors; additional details can be found in the Appendix. *https://arxiv.org/

does not directly enable tokenization—rules are needed to segment text sequences into tokens such as determining the order of tokenization and where to draw token boundaries when potential token strings overlap. For most methods, this follows the merge rules and occurrence probabilities determined during training, and thus is implicitly influenced by the structure of the training corpus.

Differences in the tokenization of a text sequence occur along two axes: the *tokenizer* and the *domain of the text*. The choice of tokenizer directly introduces differences as tokenizers are typically trained using slightly differing methods on different training corpora and have different segmentation algorithms, resulting in non-identical tokenization schemes. Indirect differences are introduced by the domain of the text—distributions of text containing more commonly occurring sequences of characters are tokenized at a higher rate; a general-purpose essay will likely correspond to fewer tokens than a technical paper of equal character or word length.

## 3. Domains and Tokenizers

### Domains

Tab. 1 outlines the 8 distinct domains we use to curate a small test corpus. We consider these domains to cover most common types of text sequences relevant to LLMs, including standard natural language, technical language, numeric sequences, and programming languages.

### Tokenizers

We focus our experimentation on a 10-tokenizer subset of the models analysed in Fig. 1, considering these to represent a broad range of proprietary and open-weights models covering both frontier and weaker levels of capability. In many cases, tokenizers are reused across models within a series, rather than being bespoke. To increase the applicability of our results, we refer to model families used by each tokenizer and include Tab. 2 as a reference for specific models.

| Tokenizer | Models |
|---|---|
| Claude (Cloud, 2025a) | Claude 3/3.5/4/4.5 |
| DeepSeek (DeepSeek-AI, 2025) | DeepSeek V3/R1 |
| Gemini (Cloud, 2025b) | Gemini 2/2.5 |
| Grok (xAI, 2025) | Grok 3/4 |
| GPT (OpenAI, 2025b) | GPT 4o/4.1/4.5/5/oss; O-Series |
| Mistral (Tekken) (AI, 2025b) | Mistral-Nemo/Pixtral/Ministral |
| Llama (AI, 2025a) | 4 |
| Reka (AI, 2025c) | Reka Flash 3/3.1 |
| Jamba (Labs, 2025) | Jamba 1.5/1.6/1.7 |
| Qwen (Team, 2025) | Qwen 1.5/2/2.5/3/; QVQ/QWQ |

*Table 2.* **Tokenizer-model mapping**. Tokenizers are commonly shared across models. These mappings and all token counts analysis in this work are empirically-derived using the cited repositories/SDKs. Tokenizers mentioned hereafter refer to the corresponding model mappings.

We use the term *compression ratio* throughout to describe the efficiency with which characters in a text sequence are converted into tokens, *i.e.*, $c = n_{chars}/n_{tokens}$. In other works (Rust et al., 2021; Turuta & Maksymenko, 2025), token fertility is reported as the mean number of tokens per word; to offer a more direct comparison to the 0.75 words per token heuristic, we report words per token instead. There are numerous ways of measuring the length and number of characters in a text string. In this work, we count characters as the number of Unicode code points in a given string. This approach is chosen over bytes, which are dependent on an underlying encoding (*e.g.*, UTF-8), and grapheme cluster counts, which can vary with Unicode version and implementation.

## 4. Experiments

We present our core empirical analysis in 3 distinct sections, covering the following tokenization metrics: character compression (§4.1), word compression (§4.2), and sequence lengths (§4.3).

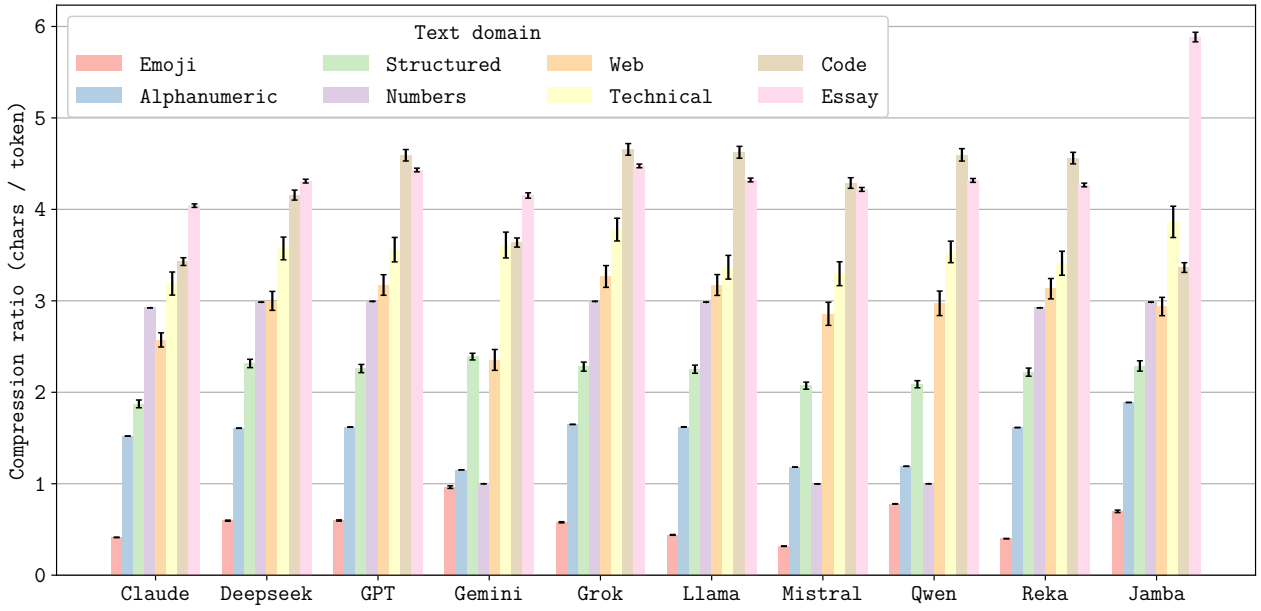


*Figure 2.* **Mean tokenizer compression ratio across different text domains**. Error bars are calculated using the standard error over 50 deterministic samplings (each ≥1000 characters).

### 4.1. Character compression

We calculate the compression ratio of 10 LLM tokenizers by averaging over 50 deterministic samplings for each of the 8 domain text corpora described above. Each sample contained at least 1000 characters and was randomly extracted from the corpus, ensuring clean token boundaries (*e.g.,* linebreaks/whitespace) at each end of the sequence. These results are displayed in Fig. 2.

**Tokenizer**. As expected due to differences in tokenizer training (corpus distribution and algorithm) and segmentation, compression ratios for a given domain vary significantly across tokenizers. The more useful insight is the degree of variation—taking into account the extreme values, compression ratios differ by 100% for the emoji and numbers domains and between ∼20-50% for the others.

**Domain**. Significant variation in compression ratio is observed across domains. For most models, compression ratios vary by nearly a factor of ten between the lowest (emojis) and highest (code/essay) compression ratios. Considering just natural language domains, the compression rate for essays is approximately 25% higher than that of technical text.

**Precision**. The size of the standard error conveys information about the breadth of the tokenizer vocabulary for text in each domain. Less structure in the text of the numeric, alphanumeric, and emoji domains results in fewer relevant tokens and a better-defined compression ratio. In the natural language domains (essay, web, technical and python code), the larger pool of different tokens produces more variation in the compression ratios across samples.

The macro-observation from these results is that **variation in compression ratios is non-trivial**: assuming near-consistent tokenization rates across domains and models is an oversimplification. Moreover, the analysis of the compression ratio of natural language essays of different languages in Fig. 3 also shows differences in the spread of compression ratios across tokenizers for a given language and notable disparity between languages. There is no significant correlation between the compression ratio and the common crawl prevalence for a given language.

### 4.2. Word compression

To investigate the common rule of thumb equating one token to approximately 0.75 words, we use a frequency-ranked set of the 10,000 most frequent English words (derived from Google's Trillion Word Corpus (first20hours; Brants & Franz, 2006)) as a representative sample of the "everyday" words the rule largely applies to. In Fig. 4, we plot a moving average of the words per token for the words in the corpus. Due to the influence of occurrence frequency in the tokenizer training distribution, the words per token decrease at higher frequency rank (as the words become less commonplace).

Analysis of the model lines relative to the 0.75 words/token (dashed) line challenge the rule of thumb. The rule could situationally provide a reasonable estimate for some mod-

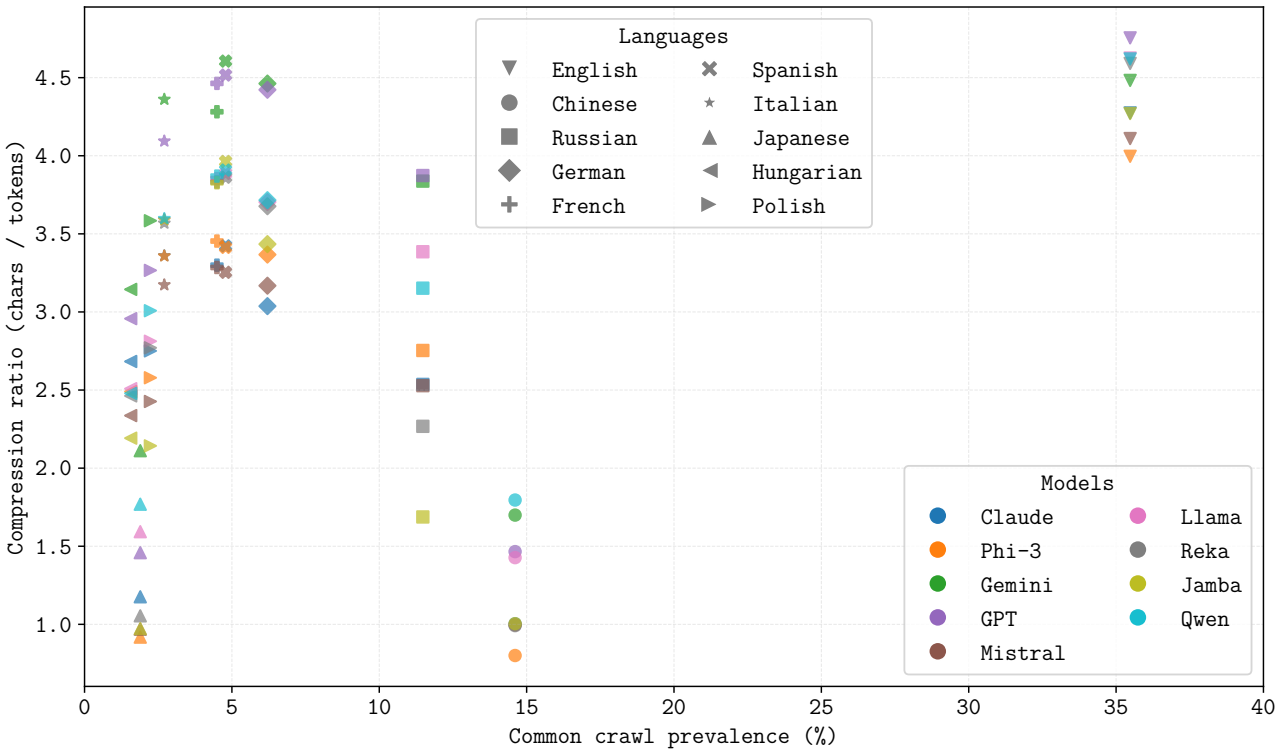


*Figure 3.* **Compression ratios of essays (Graham, 2025) translated into different languages.** Compression ratios are averaged over 11 essays; common crawl prevalence is estimated from (Abadji et al., 2022). Where necessary, arbitrary offsets have been added to the x-coordinates to reduce overlap between languages.

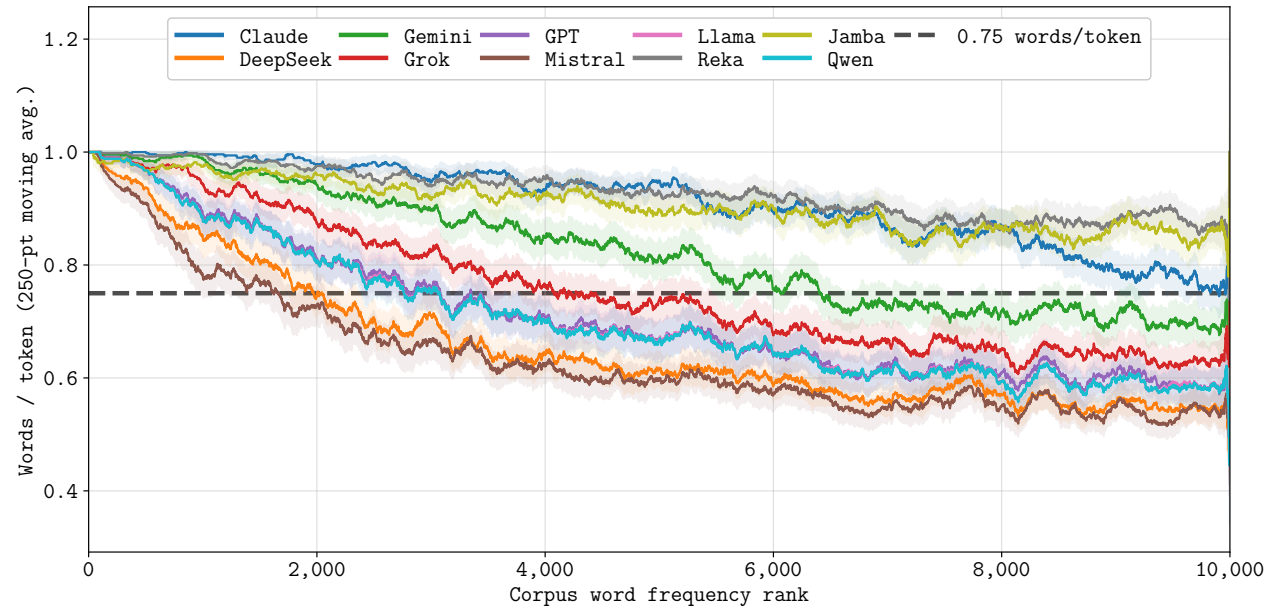


*Figure 4.* **Word to token compression variation for frequency-ranked English words**. Shaded regions show 95% confidence intervals. Mean words/token for randomly selected English words are far lower (0.35–0.45).

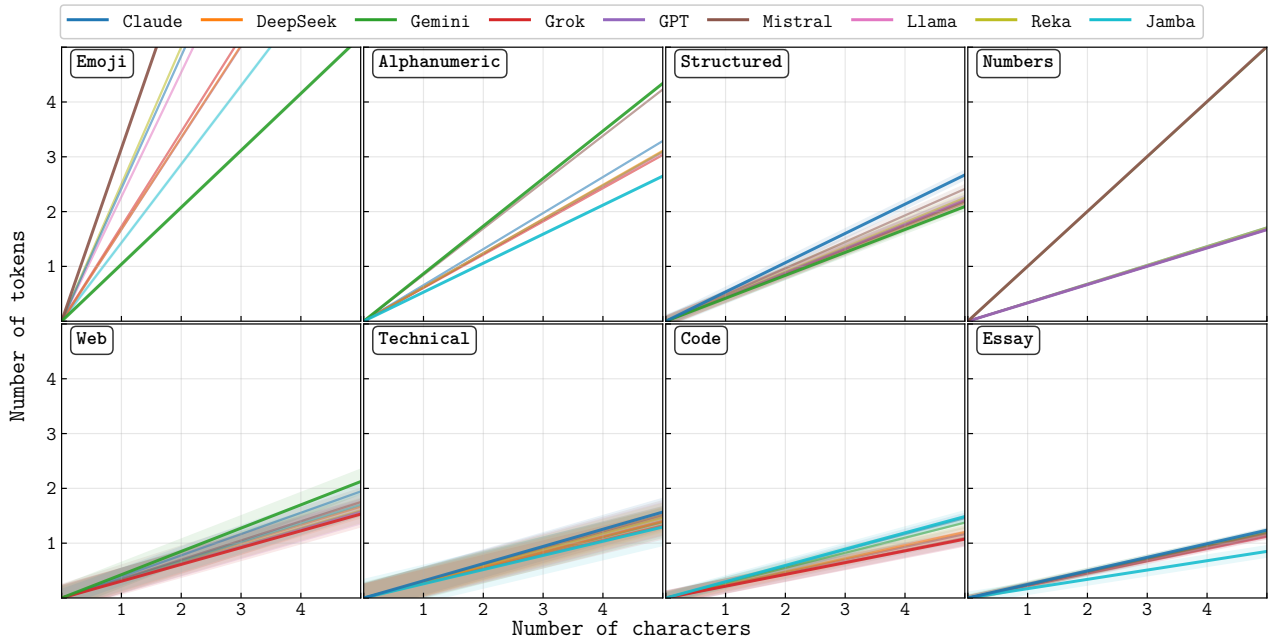


*Figure 5.* **Token-character mapping varies significantly across domains and tokenizers**. Lines are plotted using the inverse of the compression ratio as the gradient: steeper slopes require more tokens per character. Shaded regions show 95% confidence intervals.

els on text sequences that include a broad distribution of word frequency ranks. However, for widely used models such as Claude and Gemini, the average words/token is well above 0.75 for the first 5k most frequent words, and only just approaches 0.75 by the 10 thousandth. In these cases, the rule overestimates the number of tokens of a given text sequence. For other tokenizers, such as DeepSeek and Mistral, the rule is more of an underestimate, with the average words/token closer to 0.6 for the 10k most frequent words. The disparity between the rule and empirical values are more stark when considering randomly selected words from the English language. Average words/token for 10,000 words randomly sampled across the entire WordNet (Miller, 1995) English language lie in the range 0.35-0.45 for these models. A more detailed comparison of words/tokens for different text samples can be found in the Appendix. **On balance, we find the heuristic of 0.75 words/token to be an oversimplification**.

### 4.3. Context limits

A key length metric of LLMs is their *context limit* – the maximum number of input and output tokens they can process simultaneously – which is reported in model native tokens (*i.e.*, specific to the model tokenizer). As the compression ratio of text sequences into tokens varies across tokenizers and domains (Fig. 2), the "token" does not provide a consistent length measurement. This is further evident in the divergent lines of Fig. 5 that show mappings derived from the empirical compression ratios (§4.1). Therefore, using model native token counts is problematic and prevents direct comparisons between models.

In Fig. 6, we illustrate the differences in context limits for different models and domains using consistent length representations. Concretely, we initially use the empirical compression ratios and reported model context limits to derive a domain-specific *character context limit* for each model—this value is effectively tokenizer agnostic (see upper x-axis). We then use a set tokenization rate – specifically, the Llama 3 tokenization rate for natural language (essays) – to convert the domain-specific character context limits for each model into *equivalent Llama 3 essay token context limits*[1]. These "equivalent token" context limits should be interpreted as a text sequence of equal character length to an equivalent text sequence made up of as many Llama 3 essay tokens. A key observation from this plot is that context limits do not correspond to the same length text sequences across models and domains—even models with equal reported context limits have different length context limits in terms of characters or equivalent tokens. For almost all domains, the equivalent context limit in Llama essay tokens is far below the reported context limit. These results convey an important insight that reported context limits are nuanced: they refer to **the tokenization of a *specific model* on text of a *specific domain* and should not be directly compared**.

[1]The choice of reference tokenization scheme is arbitrary; we use Llama 3 essay tokenization given the popularity in open source research of Llama and similarity of essay text to ordinary natural language.

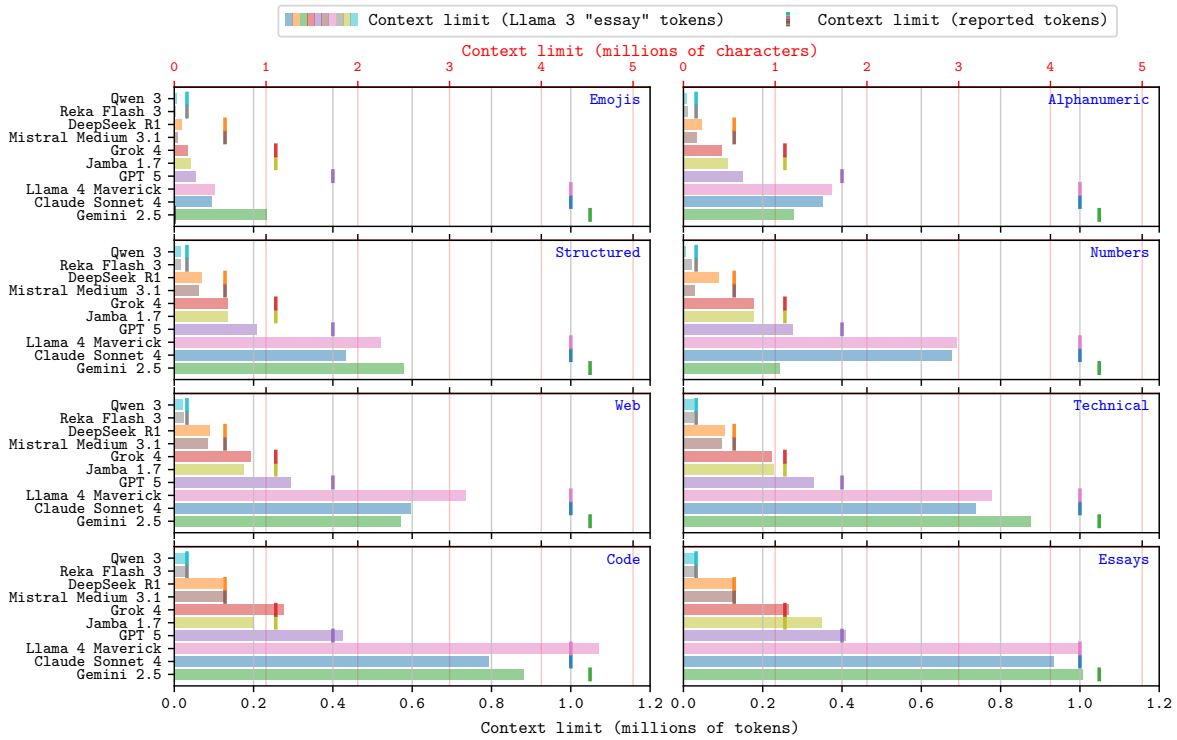


*Figure 6.* **Comparing context limits**. By converting reported "model-native" token context limits into a fixed unit, we can provide direct comparisons of context limits for given text domains. The lower (black) x-axis denotes the context limit in tokens, while the upper (red) axis displays the context limit in characters. Shaded bars represent the context limit in equivalent Llama 3 essay tokens, while the coloured lines denote the reported context limit in model native tokens.

### 4.4. Implications for model evaluation

The analysis we have presented has direct implications for the empirical evaluation and comparison of foundation models in both academic and production settings. The fundamental tension we highlight is that token counts are often used as if they were model-independent measurements of sequence length, context length, throughput, and inference cost. However, as some previous works have preliminarily shown – and we comprehensively demonstrate – tokenization varies across both tokenizers and text domains. Therefore, underspecified native token counts can obscure model comparisons rather than clarify them. A benchmark using a fixed number of model-native tokens could correspond to substantially different character lengths across models. Likewise, reported tokens-per-second metrics or provider cost-per-token values are not directly comparable unless a common tokenization scheme is used. In a similar manner, agentic evaluations of longer horizon tasks with token budgets are subject to the same limitations.

To overcome these drawbacks, we recommend that evaluations involving sequence length, long-context performance, pricing, or inference efficiency report at least one tokenizer-independent length measure alongside native token counts. Suitable choices of tokenizer-independent length metrics include characters or bytes; alternatively, a fixed reference-token equivalent (as shown in Fig. 6) can also be used. Ideally, this should use a reference tokenizer that is open-source or at least publicly accessible. For throughput and cost comparisons, the evaluated text domain should be stated, since compression ratios differ substantially between essays, code, technical text, numbers, structured data, and emojis. These reporting practices would make token-based comparisons more interpretable and reduce misleading conclusions caused by differing tokenization schemes.

### 4.5. Additional experiments

In the Appendix, we continue our empirical analysis of tokenizers beyond the preceding experiments to investigate compression ratio variation across additional languages and alphabets, geographic regions, and as text characters are poisoned. We test the hypothesis that compression ratio serves as a crude proxy for prevalence in the LLM training corpus and knowledge.

## 5. Related Work

A number of prior works have carried out empirically-driven analyses of tokenizers, broadly falling along two directions:

comparing the effect of (1) differing tokenization ratios on underrepresented languages and (2) tokenizer design on downstream tasks.

**Language**

A substantial body of research has centered on the differences in tokenization rates across languages. In (Tamang & Bora, 2024), compression ratios are compared for the 22 official Indian languages. Other studies have found downstream performance to be more robust to the effect of language imbalance in tokenizer training (Zhang et al., 2022), while compression optimized tokenization segmentation strategies are shown to potentially provide advantages for low-resource language applications (Raj et al., 2024). It is also demonstrated that tokenization methods fail to fairly represent underrepresented complex language scripts (Velayuthan & Sarveswaran, 2025). Several works compare the impacts of tokenizer choices on downstream multilingual tasks (Lotz et al., 2025; Rust et al., 2021). Another angle this research has taken is to compare costs: less represented languages in the training corpus often have lower tokenizer compression ratios, typically requiring more input and output tokens (Ahia et al., 2023; Petrov et al., 2023). A recent work (Turuta & Maksymenko, 2025) shares similarities with our work as it evaluates contemporary LLM tokenizers, however the focus is on efficiency for Ukranian language text.

**Downstream tasks**

Numerous works explore the impacts of tokenizer design on downstream LLM tasks (Goldman et al., 2024; Sälevä & Lignos, 2023; Dagan et al., 2024; Uzan et al., 2024). These range from comparing correlations between tokenizer compression ratios and downstream performance (Goldman et al., 2024), the effect of different byte-pair encoding merge operations on performance (Sälevä & Lignos, 2023), to performance metrics such as generation speed and effective context size (Dagan et al., 2024). Findings from (Schmidt et al., 2024), suggest compression does not provide a clear explanation of what makes a tokenizer effective for downstream tasks.

One recent work (Roberts et al., 2025) has eluded to the disparity between tokenizers of contemporary LLMs, demonstrating that differences in tokenization rates lead to inconsistency in defining sequence lengths. However, this analysis only covers the type of abstract alphanumeric text sequences commonly used in needle-in-a-haystack style experiments, thus offers limited widely applicable insights regarding tokenization. Our study provides a much broader and comprehensive evaluation of tokenization rates of commonly used tokenizers across many text domains and frequencies.

## 6. Conclusions

We conduct an empirical analysis of tokenization, focusing on the tokenizers used by frontier LLMs. We curate a small text corpus covering 8 distinct domains and compare the compression ratios of 10 tokenizers, finding significant variation between tokenizers and domains. We report similar variation in the compression ratios of text sequences translated into different languages. Our analysis also challenges the commonly held rule of thumb equating one token to 0.75 words, finding it to be inadequate and oversimplified. We demonstrate the inconsistencies of using tokens as a metric for measuring sequence lengths and comparing model context limits. More broadly, our results suggest that reliable model evaluations require tokenizer- and domain-aware reporting whenever tokens are used as a unit in length, cost, throughput, or context metrics.

## Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

## References

Abadji, J., Ortiz Suarez, P., Romary, L., and Sagot, B. Towards a Cleaner Document-Oriented Multilingual Crawled Corpus. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pp. 4344–4355, Marseille, France, June 2022. European Language Resources Association. URL https://aclanthology.org/2022.lrec-1.463/.

Abdullah, B. Emoji Dataset, 2024. URL https://huggingface.co/datasets/badrex/emoji-dataset. Accessed 2025-11-04.

Ahia, O., Kumar, S., Gonen, H., Kasai, J., Mortensen, D. R., Smith, N. A., and Tsvetkov, Y. Do all languages cost the same? tokenization in the era of commercial language models. *arXiv preprint arXiv:2305.13707*, 2023.

AI, M. meta-llama/Llama-4-Scout-17B-16E-Instruct — Model Card, 2025a. URL https://huggingface.co/meta-llama/Llama-4-Scout-17B-16E-Instruct. Hugging Face. Accessed 12 Nov 2025.

AI, M. mistralai/mistral-common: Official inference library for pre-processing of Mistral models (Tekken tokenizer), 2025b. URL https://github.com/mistralai/mistral-common. GitHub repository. Accessed 12 Nov 2025.

AI, R. RekaAI/reka-flash-3 — Model Card, 2025c. URL https://huggingface.co/RekaAI/reka-flash-3. Hugging Face. Accessed 12 Nov 2025.

Apoidea. FinTabNet-HTML, 2024. URL https://huggingface.co/datasets/apoidea/fintabnet-html. Accessed 2025-11-04.

Berlin, I. *The hedgehog and the fox: An essay on Tolstoy's view of history*. Princeton University Press, 2013.

Brants, T. and Franz, A. Web 1T 5-gram Version 1. LDC Catalog No. LDC2006T13, 2006. URL https://catalog.ldc.upenn.edu/LDC2006T13. Google Web 1T English n-gram counts.

Cloud, G. Count Tokens API: Generative AI on Vertex AI, 2025a. URL https://docs.cloud.google.com/vertex-ai/generative-ai/docs/multimodal/get-token-count. Product documentation. Accessed 12 Nov 2025.

Cloud, G. List and count tokens: Generative AI on Vertex AI, 2025b. URL https://docs.cloud.google.com/vertex-ai/generative-ai/docs/multimodal/list-token. Product documentation. Accessed 12 Nov 2025.

Dagan, G., Synnaeve, G., and Roziere, B. Getting the most out of your tokenizer for pre-training and domain adaptation. *arXiv preprint arXiv:2402.01035*, 2024.

DeepSeek-AI. deepseek-ai/DeepSeek-R1 — Model Card, 2025. URL https://huggingface.co/deepseek-ai/DeepSeek-R1. Hugging Face. Accessed 12 Nov 2025.

first20hours. google-10000-english: 10,000 Most Common English Words. https://github.com/first20hours/google-10000-english. GitHub repository. Accessed 11 Nov 2025.

Goldman, O., Caciularu, A., Eyal, M., Cao, K., Szpektor, I., and Tsarfaty, R. Unpacking tokenization: Evaluating text compression and its correlation with model performance. *arXiv preprint arXiv:2403.06265*, 2024.

Google. Understand and count tokens. https://ai.google.dev/gemini-api/docs/tokens, 2025. Accessed: 2025-11-09.

Graham, P. Essays. https://www.paulgraham.com/articles.html, 2025. Accessed 2025-11-04.

Kudo, T. Subword regularization: Improving neural network translation models with multiple subword candidates. *arXiv preprint arXiv:1804.10959*, 2018.

Kudo, T. and Richardson, J. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018.

Kwa, T., West, B., Becker, J., Deng, A., Garcia, K., Hasin, M., Jawhar, S., Kinniment, M., Rush, N., Von Arx, S., et al. Measuring ai ability to complete long tasks. *arXiv preprint arXiv:2503.14499*, 2025.

Labs, A. ai21labs/AI21-Jamba-Large-1.5 — Model Card, 2025. URL https://huggingface.co/ai21labs/AI21-Jamba-Large-1.5. Hugging Face. Accessed 12 Nov 2025.

Liu, Z. and Quan, Y. EconWebArena: Benchmarking Autonomous Agents on Economic Tasks in Realistic Web Environments. *arXiv preprint arXiv:2506.08136*, 2025.

Lotz, J. F., Lopes, A. V., Peitz, S., Setiawan, H., and Emili, L. Beyond Text Compression: Evaluating Tokenizers Across Scales. *arXiv preprint arXiv:2506.03101*, 2025.

Met Office. *Cartopy: a cartographic Python library with a Matplotlib interface*. Exeter, Devon, 2010 - 2015. URL https://cartopy.readthedocs.io.

Miller, G. A. WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11):39–41, 1995. doi: 10.1145/219717.219748.

Miserendino, S., Wang, M., Patwardhan, T., and Heidecke, J. SWE-Lancer: Can Frontier LLMs Earn $1 Million from Real-World Freelance Software Engineering? *arXiv preprint arXiv:2502.12115*, 2025.

OpenAI. Tokenizer tool — OpenAI Platform. https://platform.openai.com/tokenizer, 2025a. Accessed: 2025-11-05.

OpenAI. openai/tiktoken: A fast BPE tokeniser for use with OpenAI's models, 2025b. URL https://github.com/openai/tiktoken. GitHub repository. Accessed 12 Nov 2025.

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

Project Gutenberg. Project Gutenberg. https://www.gutenberg.org, 2024. Accessed: 2024-09-23.

Raj, B., Suri, G., Dewangan, V., and Sonavane, R. When every token counts: Optimal segmentation for low-resource language models. *arXiv preprint arXiv:2412.06926*, 2024.

Roberts, J., Han, K., and Albanie, S. Needle Threading: Can LLMs Follow Threads through Near-Million-Scale Haystacks? In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025.

Rust, P., Pfeiffer, J., Vulić, I., Ruder, S., and Gurevych, I. 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)*, pp. 3118–3135, 2021.

Sälevä, J. and Lignos, C. What changes when you randomly choose BPE merge operations? Not much. *arXiv preprint arXiv:2305.03029*, 2023.

Schmidgall, S., Su, Y., Wang, Z., Sun, X., Wu, J., Yu, X., Liu, J., Moor, M., Liu, Z., and Barsoum, E. Agent laboratory: Using llm agents as research assistants. *arXiv preprint arXiv:2501.04227*, 2025.

Schmidt, C. W., Reddy, V., Zhang, H., Alameddine, A., Uzan, O., Pinter, Y., and Tanner, C. Tokenization is more than compression. *arXiv preprint arXiv:2402.18376*, 2024.

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

Sennrich, R., Haddow, B., and Birch, A. 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)*, pp. 1715–1725, 2016.

SimpleMaps.com. World cities database, 2025. URL https://www.kaggle.com/dsv/11944536.

Tamang, S. and Bora, D. J. Evaluating Tokenizer Performance of Large Language Models Across Official Indian Languages. *arXiv preprint arXiv:2411.12240*, 2024.

Team, Q. Qwen/Qwen3-VL-2B-Instruct — Model Card, 2025. URL https://huggingface.co/Qwen/Qwen3-VL-2B-Instruct. Hugging Face. Accessed 12 Nov 2025.

Turuta, O. and Maksymenko, D. Tokenization efficiency of current foundational large language models for the Ukrainian language. *Frontiers in Artificial Intelligence*, 8:1538165, 2025.

Uzan, O., Schmidt, C. W., Tanner, C., and Pinter, Y. Greed is all you need: An evaluation of tokenizer inference methods. *arXiv preprint arXiv:2403.01289*, 2024.

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

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

xAI. xai-org/xai-sdk-python: The official Python SDK for the xAI API, 2025. URL https://github.com/xai-org/xai-sdk-python. GitHub repository. Accessed 12 Nov 2025.

Xu, F. F., Song, Y., Li, B., Tang, Y., Jain, K., Bao, M., Wang, Z. Z., Zhou, X., Guo, Z., Cao, M., et al. Theagentcompany: benchmarking llm agents on consequential real world tasks. *arXiv preprint arXiv:2412.14161*, 2024.

Zhang, S., Chaudhary, V., Goyal, N., Cross, J., Wenzek, G., Bansal, M., and Guzman, F. How Robust is Neural Machine Translation to Language Imbalance in Multilingual Tokenizer Training? *arXiv preprint arXiv:2204.14268*, 2022.

# Appendix

## A. Additional words per token results

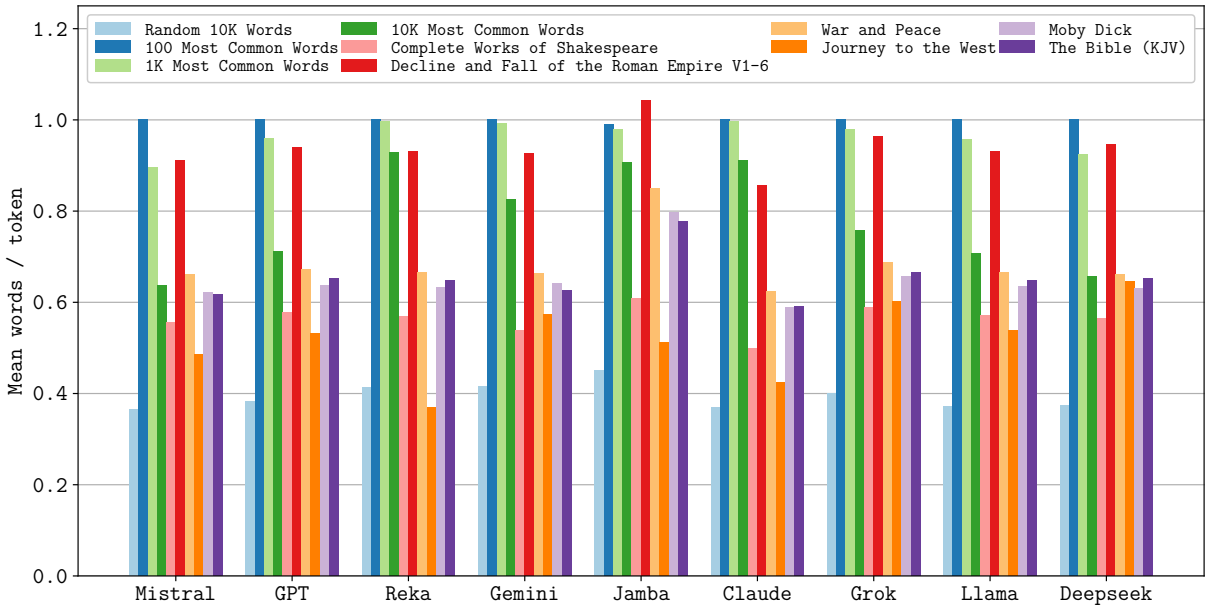


*Figure 7.* **Average words/token for different text corpora**. Random 10k Words are randomly sampled from WordNet (Miller, 1995) English words and the 100, 1K, 100K Most Commons Words are derived from the most frequent words in Google's Trillion Word Corpus (first20hours; Brants & Franz, 2006). The books were sourced from Project Gutenberg (Project Gutenberg, 2024).

## B. Languages Compression Ratios

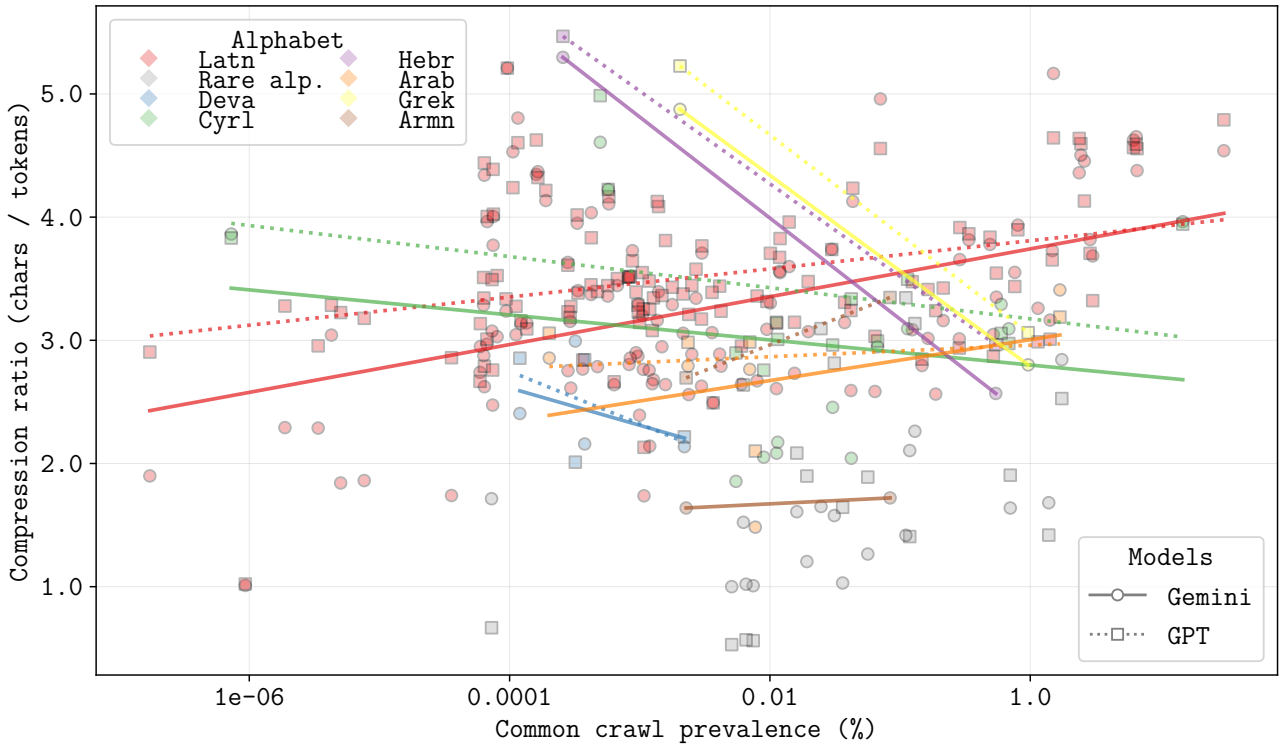


*Figure 8.* **Compression ratios for a natural language essay translated into languages present in Common Crawl**. Each data point represents a language text tokenized by the Gemini (circle) or GPT (square) tokenizers. Language common crawl prevalence is estimated from (Abadji et al., 2022). Translated data was derived using GPT-4o to translate the essay "What to do" (Graham, 2025).

## C. Geospatial

The geospatial experiments in this subsection investigate hypotheses related to potential geographic bias in the tokenizer training corpora. Figs. 9-11 display the GPT tokenizer compression ratios for country, region and city/town names, respectively.

Fig. 12 shows the haversine distance error between the latitude and longitude positions predicted by Gemini 2.5 Flash and ground truth of 48k settlements worldwide (see Fig. 11). When plotted against the compression ratios of the settlement names, a negative correlation is observable: distance error decreases with higher compression ratios. We hypothesize this is tied to the distribution of training data. Assuming the tokenizer training data follows a similar distribution to the LLM training data, words or subwords that are more prevalent will both have a higher compression and represent concepts the LLM has more accurate knowledge of.

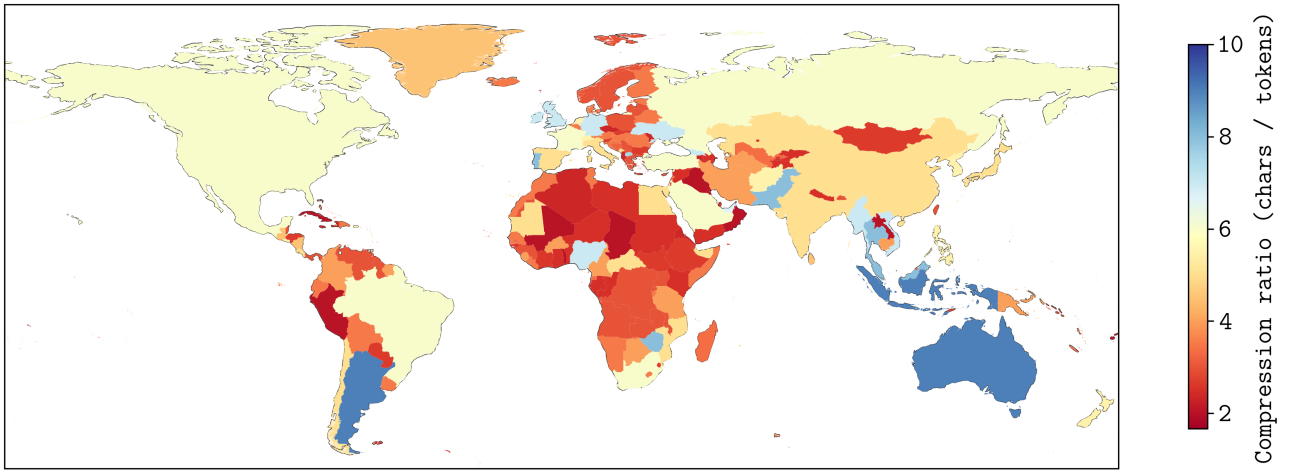


*Figure 9.* **Country name tokenization**. Compression ratios are displayed for the GPT tokenizer using country names and geometries derived from cartopy (Met Office, 2010 - 2015).

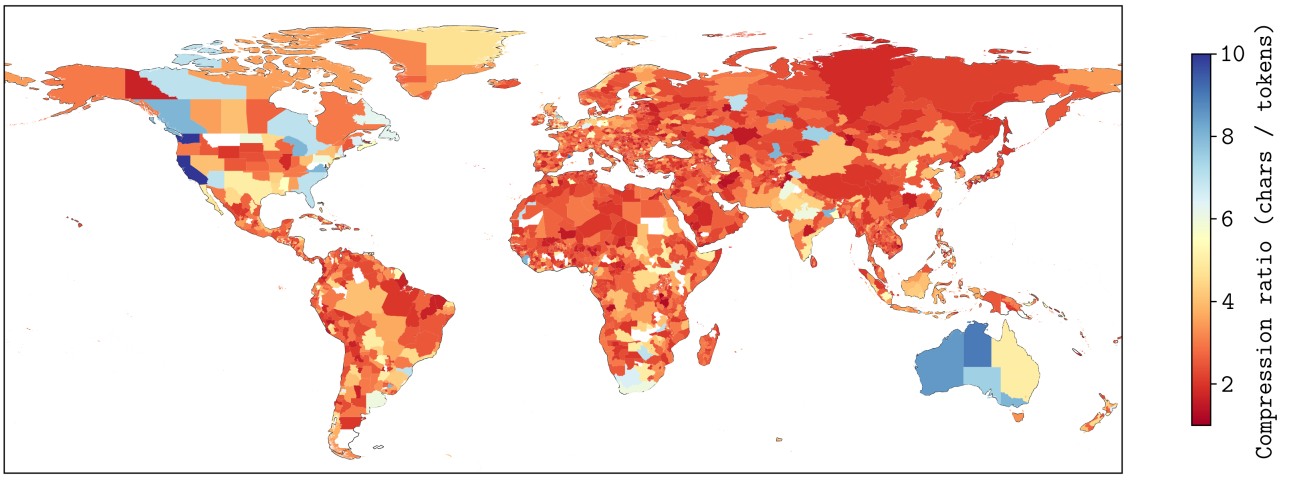


*Figure 10.* **Region name tokenization**. Compression ratios are displayed for the GPT tokenizer using region names and geometries derived from cartopy (Met Office, 2010 - 2015).

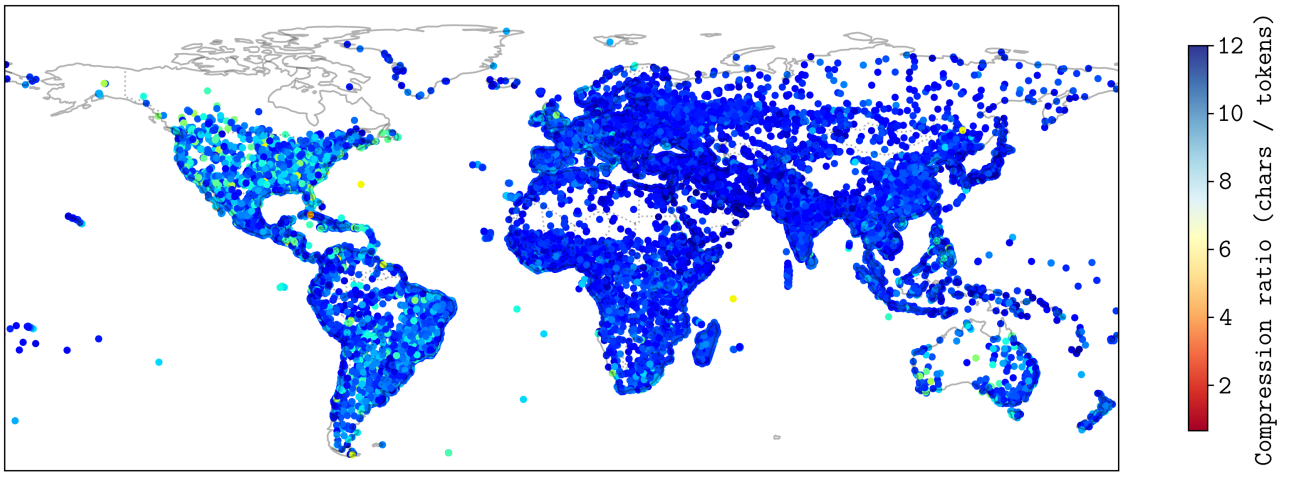


*Figure 11.* **Settlement name tokenization**. Compression ratios are displayed for the GPT tokenizer for ∼48k town and city names and geometries from (SimpleMaps.com, 2025).

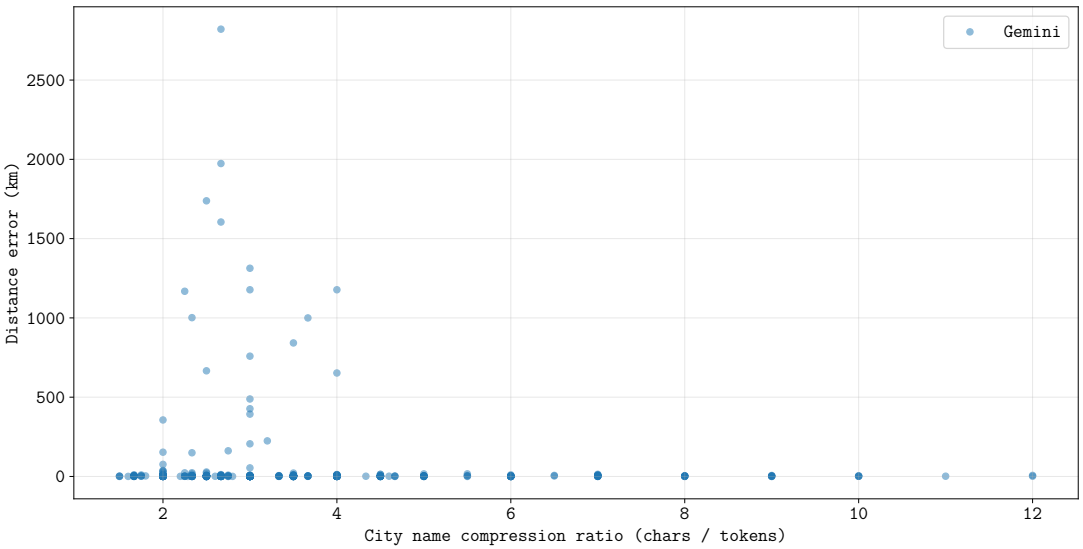


*Figure 12.* **Haversine distance errors** between the positions (latitude, longitude) predicted by Gemini 2.5 Flash and ground truth for 48k settlements worldwide (SimpleMaps.com, 2025).

## D. Character Poisoning

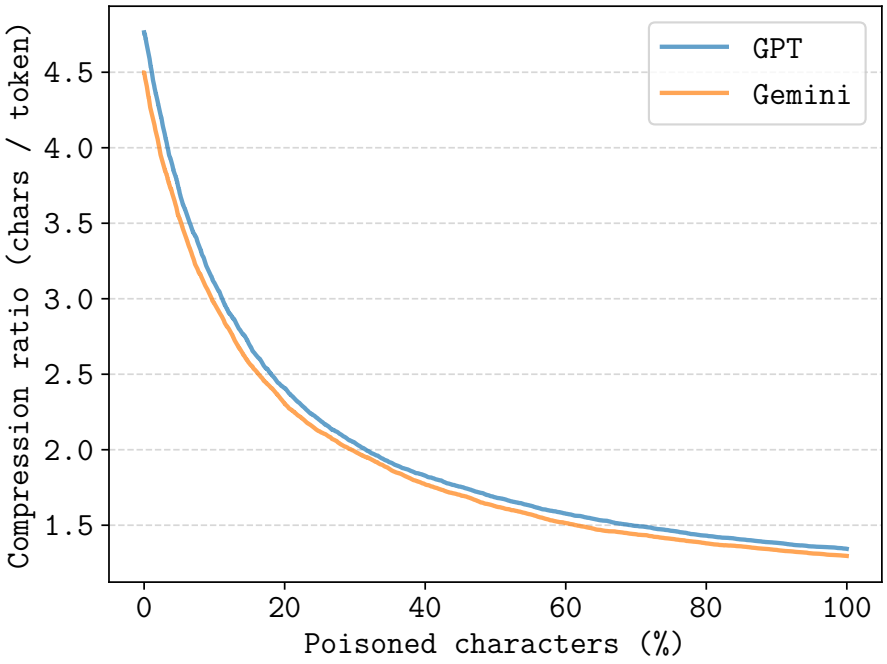


*Figure 13.* **Compression ratios decay exponentially as characters are randomly poisoned** for text from "What to do" (Graham, 2025). Incrementally, characters were selected at random and replaced with a different character. The compression ratio was averaged across the entire text each iteration.