TokCollate: A Comprehensive Tool for Tokenizer Evaluation and Visualization across Languages

Dušan Variš and Abishek Stephen and Jindřich Libovický
Charles University, Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
Malostranské náměstí 25, 118 00 Praha 1, Czech Republic
{varis,stephen,libovicky}@ufal.mff.cuni.cz

Abstract

Tokenization quality varies significantly across languages, contributing to disparities in LLM performance and cost for speakers of less-resourced languages – a phenomenon known as the "token premium" problem. Despite growing research interest, no existing tool provides a comprehensive intrinsic evaluation of tokenizers paired with interactive visualization. We present *TokCollate* (pronounced similarly to *chocolate*), a Python-based evaluation framework combined with a JavaScript visualization interface that addresses this gap. TokCollate implements a wide range of intrinsic metrics, including monolingual measures such as average token length and Rényi/Shannon efficiency, and cross-lingual measures such as vocabulary overlap, Jensen-Shannon divergence, alignment-based Eflomal scores, and length ratios. It further enables analysis across language groups defined by genealogical families, scripts, geographic regions, speaker populations, and estimated data availability. TokCollate is open-source under the MIT license and available on GitHub.[1]

1 Introduction

Tokenization is a fundamental preprocessing step in modern natural language processing (NLP), converting raw text into discrete units from a final set that serve as the basic inputs to language models (Sennrich et al., 2016; Kudo and Richardson, 2018). In practice, however, tokenization quality varies dramatically across languages, with practical consequences for the performance and accessibility of large language models (LLMs).

A particularly pressing manifestation of this disparity is the so-called *token premium* (Ahia et al., 2023): because tokenizers trained predominantly on English and other high-resource languages tend to segment low-resource language text into longer subword sequences, speakers of those languages face higher computational costs and API prices for the same amount of information. Longer tokenizations have also been linked to degraded model performance (Rust et al., 2021; Petrov et al., 2023), as models must process more tokens to represent equivalent semantic content, straining context windows and attention mechanisms.

Interest in tokenization has grown substantially in recent years, with work examining cross-lingual tokenization quality (Rust et al., 2021), the relationship between tokenization and translation quality (Domingo et al., 2019), and cross-lingual transfer (Limisiewicz et al., 2023). However, tools for systematic tokenizer evaluation remain fragmented and do not focus on structured cross-lingual comparison.

We present *TokCollate*, a comprehensive tool that combines multiple intrinsic evaluation metrics, interactive visualization, and cross-language and cross-tokenizer comparisons in a single unified framework. TokCollate consists of a Python-based evaluation CLI and a web-based visualization interface with a React + TypeScript frontend and a Python Express backend.

It implements both monolingual metrics, such as average token length, Shannon entropy, and Rényi efficiency (Zouhar et al., 2023), and cross-lingual metrics including vocabulary overlap, Jensen-Shannon divergence (Limisiewicz et al., 2023), alignment-based Eflomal scores (Hämmerl et al., 2025), and length ratios between parallel texts.

Users can aggregate and explore results by language characteristics such as script, genealogical family, or resource level, making it straightforward to identify which language communities are most affected by a given tokenizer design.

TokCollate is released as open-source software under the MIT license.

[1] git@github.com:ufal/TokCollate.git

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), pages 418–427

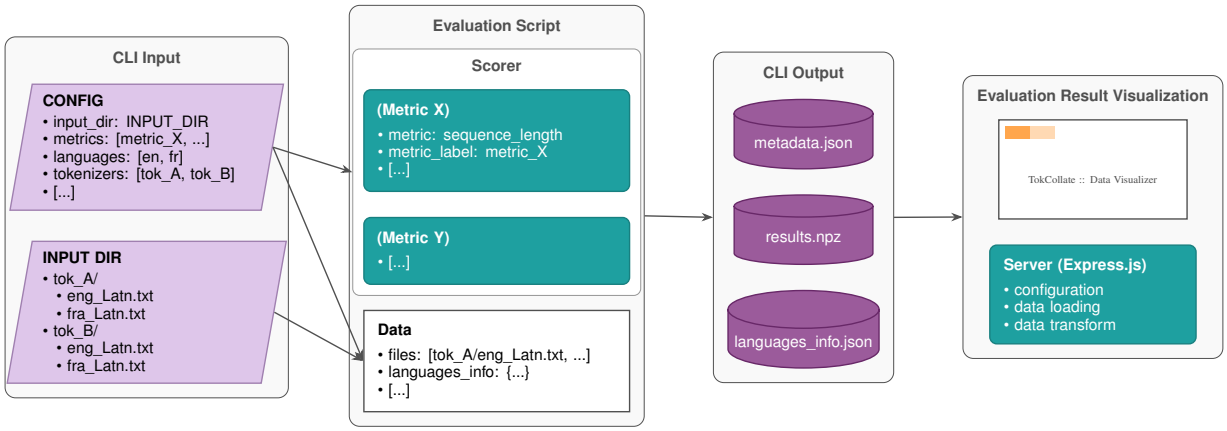


Figure 1: TokCollate architecture overview and its components: evaluation framework and the evaluation result visualizer.

2 System Overview

The current version of TokCollate[2] consists of two independent parts: the *evaluation script* and the *evaluation result visualizer*. The intended use is to first run the evaluation framework on the given set of tokenizer outputs, and then inspect the evaluation results using the visualizer. Figure 1 shows the basic workflow.[3] Although the evaluation and visualization components are meant to be used together, they can also be used independently since the visualizer accepts any evaluation results that conform to the expected output data structure.

Evaluation Script is implemented in Python and used to evaluate tokenization outputs (we do not currently support direct tokenization execution) from a set of user-defined tokenizers and languages (including their respective tokenization output files) against a selected set of metrics. Using YAML syntax, the user provides a configuration for the *Scorer* object, including the evaluation input (the location of the tokenized outputs), the output directory, tokenizer labels, languages, and metrics. Based on the configuration, the tokenizer outputs are loaded by the *Data* handler object, which provides an interface for TokCollate to access the outputs. The scorer then computes results for each *Metric* instance defined in the configuration file (a single metric can be instantiated multiple times with varying parameters), producing a matrix of evaluation results.

TokCollate currently provides a wide range of metric implementations and supports straightforward extension via the *Metric* abstract class interface.

Lastly, the evaluation results are stored in the output directory, including the evaluation metadata (JSON file), the result matrices (a single NPZ file), and (optional) language information metadata.

Appendix A provides more details on the configuration of the evaluation script and the configuration file structure.

Evaluation Result Visualizer is a web application to visualize the tokenizer evaluation results collected by the evaluation framework. The visualizer frontend is a React + TypeScript single-page app, and the Express.js server handles data loading and transformations.

In the web app, the user selects the directory containing the evaluation results via the *Import Data* button, which loads them into the visualizer. Then, they can select from the available visualizations (called *Figure Type*) and select the visualization parameters, including the set of tokenizers and languages. If the import directory contains `languages_info.json` file with language details, they can further filter selected languages based on their characteristics. The current version of TokCollate supports tabulating evaluation results and visualizing the correlation between metric pairs. The modular design enables straightforward extension to additional visualization styles by subclassing the *GraphType* class. After selecting the data directory, the user can visualize the individual metric results using the available figure types. A key feature of the visualizer is robust language filtering, which allows detailed inspection of metric behavior across language families, writing scripts, and other group-

[2] https://github.com/ufal/TokCollate

[3] A live visualizer demo is currently available at https://quest.ms.mff.cuni.cz/tokcollate/.

ings. While providing robust language filtering, the user can define their own language information files and pass them to TokCollate if needed.

The current figure configuration can be saved as a PNG file using the *Export Graph* button.

2.1 Evaluation Metrics

TokCollate implements two classes of intrinsic evaluation metrics: monolingual metrics computed per language and cross-lingual metrics computed over language pairs.

Supervised intrinsic metrics such as morphological alignment (Uzan et al., 2024; Batsuren et al., 2024) or cognitive plausibility (Beinborn and Pinter, 2023) are outside our scope, as they require gold-standard annotations, which are unavailable for most of the languages we target.

Monolingual Metrics. *Corpus Token Count (CTC)* measures the total number of tokens required to encode a given corpus (Schmidt et al., 2024); we also report character-per-token and fertility (average tokens per whitespace-delimited word; Rust et al., 2021) as normalized variants.

Rényi Efficiency and *Shannon Entropy* are implemented based on the tokenization scorer of Zouhar et al. (2023).[4] Rényi Efficiency measures the optimality of the token frequency distribution, parameterized by α, which controls sensitivity to distributional tails. Higher α values have been reported to correlate with downstream machine translation quality, though this relationship is not reliable across settings (Cognetta et al., 2024; Libovický and Helcl, 2024).

Compression-based metrics have been empirically linked to downstream performance (Goldman et al., 2024; Gallé, 2019), though this correlation is not universal (Schmidt et al., 2024).

Additionally, TokCollate supports supervised metrics that allow a comparison of the tokenized text with reference segmentation. For the evaluation of morphological plausibility, we provide a subset of languages from the FLORES-200 dataset, segmented automatically using morphosegmentation models described by John et al. (2026).

Cross-lingual Metrics. *Vocabulary Overlap* measures the proportion of subword types shared between the vocabularies induced for two languages; low overlap suggests that the tokenizer vocabulary serves the two languages largely independently, which may hinder cross-lingual transfer (Limisiewicz et al., 2023).

Jensen-Shannon Divergence (JSD) is a symmetric measure of similarity between token frequency distributions of two languages (Limisiewicz et al., 2023) that, unlike vocabulary overlap, accounts for how frequently shared tokens are actually used.

Eflomal Alignment Score uses word alignment (Hämmerl et al., 2025) to estimate the degree to which tokens in parallel sentences correspond across languages, with higher scores indicating more consistent segmentation, which is generally beneficial for cross-lingual transfer. Because computing word alignment between all language pairs might be too slow for many languages, we also add mean cross-lingual point-wise mutual information as a fast approximation of cross-lingual token alignability.

Length Ratio on parallel text measures the ratio of token counts between source and target sentences (Petrov et al., 2023), with large deviations from 1.0 directly quantifying the token premium (Ahia et al., 2023).

All metrics can be computed over any parallel corpus. We recommend FLORES+ for its broad language coverage, which enables consistent cross-language and cross-tokenizer comparison across a large number of languages. Metrics requiring only tokenized text can be computed for any language in the corpus; alignment-based metrics additionally require parallel sentence pairs and are therefore restricted to language pairs present in the chosen corpus.

2.2 Language Information

We assembled a language knowledge base derived from the FLORES+ dataset (Perez-Ortiz et al., 2024), which serves as the primary source of language coverage in TokCollate. Languages are identified by their ISO 639-3 codes and ISO 15924 script codes; where language variants require further disambiguation, we additionally use Glottolog codes (Nordhoff and Hammarström, 2011). Speaker population estimates and continent-of-origin annotations are sourced from WikiData (Vrandecic and Krötzsch, 2014).

Genealogical classification is retrieved via the Glottolog API, grouping languages into the following families: Afro-Asiatic, Artificial Language, Atlantic-Congo, Austroasiatic, Austronesian, Aymaran, Dravidian, Indo-European, Japonic,

[4] https://github.com/zouharvi/tokenization-scorer

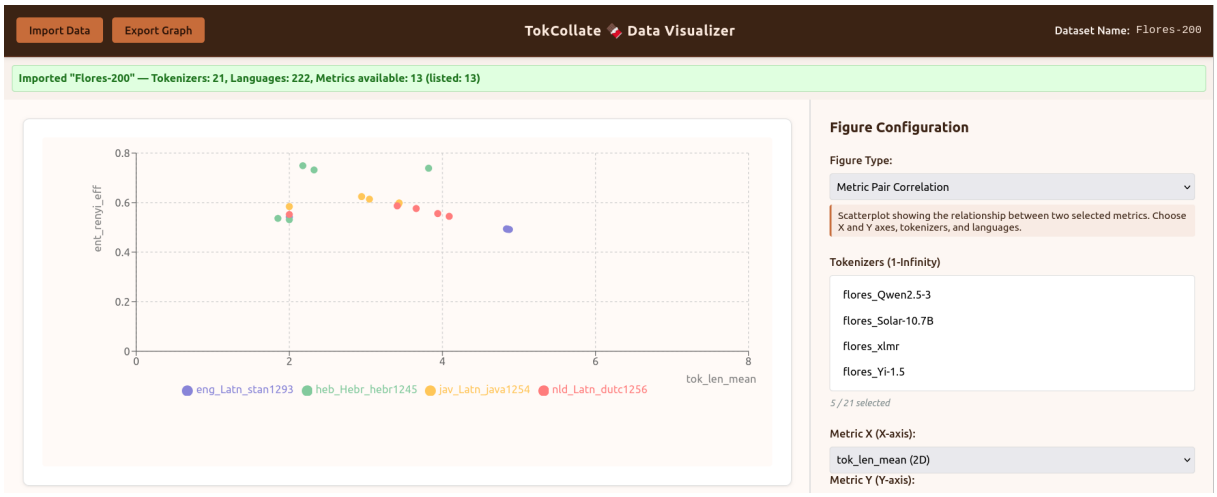


Figure 2: Screenshot from the visualization: A plot comparing average token length and Rényi entropy of the tokenizers and four languages.

Kartvelian, Koreanic, Mande, Mongolic-Khitan, Nakh-Daghestanian, Nilotic, Quechuan, Saharan, Sino-Tibetan, Tai-Kadai, Tupian, Turkic, and Uralic.

For resource stratification, we adopt the six-tier taxonomy of Joshi et al. (2020), which ranks languages by the availability of linguistic and raw text resources as of 2020. Tier 5 comprises seven high-resource languages (English, Spanish, German, Japanese, French, Modern Standard Arabic, and Mandarin Chinese), while Tier 0 languages have only minimal resources. We note that the original tier assignments are partially ambiguous, as the language names in Joshi et al. (2020) do not always correspond to the canonical English names used in Glottolog or Wikipedia, so we match some of the names manually.

We also include a coarse morphological typology classification, distinguishing agglutinative, fusional, and isolating languages. This classification was constructed manually from Wikipedia descriptions, as no comprehensive, machine-readable resource covering all languages in our knowledge base is currently available.

Finally, we include an estimate of each language's data availability, approximated by the amount of data for that language in the FineWeb2 corpus (Penedo et al., 2024). FineWeb2 is derived from the full Common Crawl and is broadly representative of multilingual web content, making it a suitable proxy for a language's presence in the pretraining data of contemporary LLMs.

2.3 Visualization Interface

The interface is divided into three sections: the top menu, the right-side figure configuration panel, and the left-side figure rendering area. Figure 2 shows the browser interface layout.

The top menu provides options for loading evaluation results and exporting figures. The right-side configuration menus are used to configure the rendered figure. The user can select a figure type and a subset of tokenizers and languages; these options are common to all figure types. Languages can be selected either directly from the language list or by applying a combination of filters. In the current release, TokCollate supports two figure types: Metric Table and Metric Pair Correlation.

Metric Table is the primary method for visualizing evaluation results. It displays a table for a given metric, with a selected set of tokenizers and languages (or language pairs for pairwise metrics). The user can sort the table by any selected row or column. Individual cells are color-coded by their values, making it easier to assess relative differences across tokenizers and languages.

Metric Pair Correlation displays a scatterplot that helps inspect the correlation between two selected metrics. In addition to the common configuration, the user can choose from various datapoint color groupings (tokenizers, languages, etc.) and toggle the display of the correlation trendline (global or per group).

Tokenized Text enables side-by-side comparison of texts tokenized by various tokenizer or across different languages. Besides the tokenizer/language selection, similar, to other visualization types, the

Figure 3: Screenshot from the visualization: Displaying selected lines (Czech Flores) tokenized by two different tokenizers (Gemma-2 and Gemma-3). Scrolling over a token highlights all its instances in the displayed text selection.

user can also select the span of lines to be displayed. Furthermore, hovering the mouse cursor over a token in the displayed texts shows all instances of this token, allowing better identification of the differences between the displayed tokenizations. Figure 3 shows the tokenized text visualization.

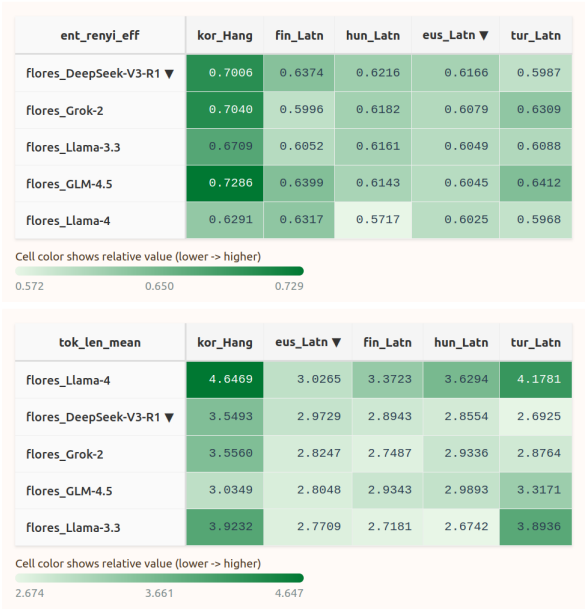
3 Use Cases and Target Audience

TokCollate is designed for three broad user groups: researchers, NLP engineers, and educators.

Researchers studying multilingual NLP can use TokCollate to systematically compare tokenizers across languages, identify which language communities are most affected by a given tokenizer design, and investigate the relationship between tokenization properties and model behavior. The combination of multiple metrics and language characteristic groupings makes it straightforward to test hypotheses, for example, about the effects of script or morphological type on fertility or compression rate.

NLP engineers selecting tokenizers for multilingual applications can use TokCollate to identify languages for which a candidate tokenizer performs poorly before committing to it in a production pipeline. The interactive interface allows filtering and sorting by language, metric, and grouping variable, making it practical to diagnose tokenization issues for a specific target language set.

Educators can use the visualization interface to demonstrate tokenization disparities interactively. The interface requires no local installation on the student side. The instructor deploys the visualizer once, after which students can access it directly in a web browser. This, combined with support for exploratory analysis, makes it suitable for classroom use or assignments in which students compare tokenizers across language families or resource levels.



ent_renyi_eff	kor_Hang	fin_Latn	hun_Latn	eus_Latn ▼	tur_Latn
flores_DeepSeek-V3-R1 ▼	0.7006	0.6374	0.6216	0.6166	0.5987
flores_Grok-2	0.7040	0.5996	0.6182	0.6079	0.6309
flores_Llama-3.3	0.6709	0.6052	0.6161	0.6049	0.6088
flores_GLM-4.5	0.7286	0.6399	0.6143	0.6045	0.6412
flores_Llama-4	0.6291	0.6317	0.5717	0.6025	0.5968

Cell color shows relative value (lower -> higher)
0.572 0.650 0.729

tok_len_mean	kor_Hang	eus_Latn ▼	fin_Latn	hun_Latn	tur_Latn
flores_Llama-4	4.6469	3.0265	3.3723	3.6294	4.1781
flores_DeepSeek-V3-R1 ▼	3.5493	2.9729	2.8943	2.8554	2.6925
flores_Grok-2	3.5560	2.8247	2.7487	2.9336	2.8764
flores_GLM-4.5	3.0349	2.8048	2.9343	2.9893	3.3171
flores_Llama-3.3	3.9232	2.7709	2.7181	2.6742	3.8936

Cell color shows relative value (lower -> higher)
2.674 3.661 4.647

Figure 4: A screenshot of table visualization from the interface showing Rényi efficiency and token length for Tier 4 agglutinative languages. The rows are order by the values of Basque (`eus_Latn`), the columns are ordered by the values for DeepSeek-V3.

4 Case Study

To demonstrate the capabilities of TokCollate, we investigate how the writing system constrains the token premium by comparing average bytes-per-token and Rényi Efficiency for agglutinative lan-

guages across all six tiers of the taxonomy in Joshi et al. (2020). We evaluate the following models: Llama-3.3 (Grattafiori et al., 2024), Llama-4 (Meta AI, 2025), DeepSeek-V3 (DeepSeek-AI et al., 2025), GLM-4.5 (Team et al., 2025), Yi-1.5 (AI et al., 2025), Mistral-3 (Mistral AI, 2025), and Grok-2 (xAI, 2024). Figure 4 shows the Rényi efficiency and average token length for Tier 4 agglutinative languages as a representative example.

For Tier 5 languages, Llama-3.3 and Llama 4, DeepSeek-V3, GLM-4.5, and Grok-2 allocate more than 4 bytes per token on average, yielding Rényi Efficiencies of over 0.5 for Japanese.

Llama-4 and Yi-1.5 encode Korean in the Hangul script, with 4.6 bytes per token in Tier 4. This is consistent with the fact that Korean syllable blocks each occupy exactly one Unicode code point (3 bytes in UTF-8), so a three-syllable word requires 9 bytes in UTF-8. The contrast between these two models appears in their byte-per-token allocation for agglutinative languages using Latin scripts: Llama-4 consistently allocates around 3.5 bytes per token, while Yi-1.5 drops to approximately 2 bytes per token. Despite this difference, their Rényi Efficiencies remain in a broadly similar range.

In Tier 3, DeepSeek-V3, Mistral-3, and Llama-4 exhibit broadly similar behaviour, allocating more bytes across all languages with a nuanced bias toward non-Latin scripts. Yi-1.5, by contrast, treats Tier 3 languages no differently from Tier 4, showing little sensitivity to script type. Tier 2, which consists entirely of Bantu languages written in Latin script, sees all models converging on a narrower range of roughly 2–3 bytes per token, which is unsurprising given the script's compact Unicode representation.

For Tiers 1 and 0, covering Sino-Tibetan, Turkic, and Dravidian languages, the models largely reiterate the Tier 3 patterns, again allocating more bytes to non-Latin scripts. Tamil characters span the Unicode range `U+0B80`–`U+0BFF`, requiring 3 bytes per code point; since each grapheme requires, on average, approximately 2 Unicode code points, a single Tamil grapheme costs roughly 6 bytes in UTF-8, twice the cost of a Korean syllable block.

The overall findings indicate that the token premium is driven by the interaction between the writing system and tokenizer design, rather than by the resource tier alone. The tier taxonomy is a weak predictor of byte allocation; what matters is whether a language uses a script that maps efficiently onto a model's vocabulary.

The disparity between average bytes-per-token and Rényi Efficiency values also highlights an important distinction between tokenizer design and quality. For Japanese and Korean, higher bytes-per-token paired with higher Rényi Efficiency suggest a more diverse and well-utilized vocabulary: their writing systems compress substantial semantic content into dense characters, which helps the tokenizer recover linguistically motivated boundaries. For Tamil, however, Llama-4 and DeepSeek-V3 allocate 4–5 bytes per token on average and achieve a Rényi Efficiency of roughly 0.7, yet the tokenization may produce grapheme-level tokens that do not correspond to morpheme boundaries. These results illustrate that writing system properties, tokenizer training, and language morphology interact in non-trivial ways, with substantive implications for tokenizer quality.

5 Related Work

Several tools for tokenizer evaluation exist, but each addresses only a subset of the metrics and use cases that TokCollate targets.

TokEval[5] (Meister, 2025) is a Python command-line framework restricted to HuggingFace tokenizers. It implements basic metrics (compression rate, fertility, token length, type-token ratio), information-theoretic metrics (Rényi entropy, vocabulary utilization, average token rank), morphological metrics via MorphScore (Arnett et al., 2025), and a cross-lingual Gini coefficient for compression equity. It produces static plots and LaTeX tables but does not offer interactive visualization, and because it includes supervised morphological metrics, evaluation is tied to a fixed set of languages with available annotated data.

TkTkT[6] (Bauwens, 2025) is primarily a library of subword tokenizer implementations rather that implements a broad range of evaluation metrics. Its evaluation module covers fertility, morphological boundary precision and recall, Rényi efficiency, window-based metrics (MATTR), bigram metrics (accessor variety), and pairwise tokenizer comparisons.

Qtok[7] (Chelombitko et al., 2024) takes a vocabulary-centric approach: it classifies vocabulary tokens by Unicode range, script, and surface

[5] https://github.com/cimeister/tokenizer-analysis-suite

[6] https://github.com/bauwenst/TkTkT

[7] https://github.com/nup-csai/Qtok

form (space-initial vs. word-internal, alphabetic vs. other), and uses these distributions as proxies for multilingual coverage. It does not compute corpus-level metrics such as fertility, length ratios, or information-theoretic measures, and provides no interactive visualization.

Several single-metric tools have been released alongside individual papers. The tokenization scorer of Zouhar et al. (2023) implements Rényi efficiency and Shannon entropy; TokCollate adopts the same implementation. MorphScore[8] (Arnett et al., 2025) measures morpheme boundary precision and recall against Universal Dependencies treebanks across 71 languages, but requires gold morphological annotations and is therefore outside our scope. The Byte Premium Tool[9] (Arnett et al., 2024) computes the UTF-8 byte ratio between two languages encoding the same content, either from a precomputed database of 1,155 language pairs, from parallel text, or predicted from monolingual text via regression.

6 Conclusion

We presented TokCollate, a tokenizer evaluation framework comprising a Python multi-metric evaluation tool and a React single-page visualization interface with a Python Express backend.

The evaluation tool computes a range of intrinsic monolingual metrics (corpus token count, fertility, tokens per character, Rényi efficiency, and Shannon entropy) as well as cross-lingual metrics over parallel text (vocabulary overlap, Jensen-Shannon divergence, alignment-based Eflomal scores, and length ratios). All metrics can be computed over any parallel corpus.

The visualization interface supports sorting, filtering, and aggregation of results by language characteristics, including genealogical family, script, geographic region, speaker population, and estimated data availability making it an useful analysis tool for linguists studying LLMs.

Limitations

TokCollate implements only intrinsic evaluation metrics. The relationship between intrinsic metrics and downstream task performance is not straightforward: empirical correlations between metrics such as Rényi efficiency and translation quality have been reported but are inconsistent across settings (Cognetta et al., 2024; Libovický and Helcl, 2024), and no intrinsic metric reliably predicts extrinsic quality.

To maximize language coverage, TokCollate currently supports mostly unsupervised intrinsic metrics that require only raw tokenized text. The current design also provides limited support for metrics that use reference segmentation for comparison. In the future, this support should be further expanded to include supervised metrics such as morphological boundary alignment (Arnett et al., 2025; Uzan et al., 2024) and cognitive plausibility measures (Beinborn and Pinter, 2023), which capture aspects of tokenization quality not reflected in the metrics we implement.

The language knowledge base currently covers approximately 200 languages, limited by FLORES+'s coverage. Cross-lingual metrics requiring parallel text are further restricted to language pairs present in the chosen corpus, and languages outside the supported set cannot be evaluated without extending the underlying data sources.

Ethics Statement

TokCollate is intended to support the analysis of tokenization disparities across languages, making it easier to identify which language communities are disproportionately affected by a given tokenizer design. The tool runs entirely locally; no user data or evaluation results are transmitted externally. TokCollate is released as open-source software under the MIT license, allowing full inspection of all metric implementations and reproducibility of all reported results.

Acknowledgments

This work was funded by the Czech Science Foundation (GA ČR, grant no. 25-16242S) and partially supported by the Charles University projects (GA UK, grant no. 101924) and SVV (grant no. 260 821).

References

Orevaoghene Ahia, Sachin Kumar, Hila Gonen, Jungo Kasai, David Mortensen, Noah Smith, and Yulia Tsvetkov. 2023. Do all languages cost the same? tokenization in the era of commercial language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*,

[8] https://github.com/catherinearnett/morphscore

[9] https://github.com/catherinearnett/byte-premium-tool

pages 9904–9923, Singapore. Association for Computational Linguistics.

01. AI, :, Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Guoyin Wang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, Kaidong Yu, Peng Liu, Qiang Liu, Shawn Yue, Senbin Yang, Shiming Yang, and 14 others. 2025. Yi: Open foundation models by 01.ai. *Preprint*, arXiv:2403.04652.

Catherine Arnett, Tyler A. Chang, and Benjamin K. Bergen. 2024. A bit of a problem: Measurement disparities in dataset sizes across languages. In *Proceedings of the Annual Meeting of the Special Interest Group on Under-Resourced Languages*.

Catherine Arnett, Marisa Hudspeth, and Brendan O'Connor. 2025. Evaluating Morphological Alignment of Tokenizers in 70 Languages. In *Proceedings of the ICML 2025 Tokenization Workshop (TokShop)*.

Khuyagbaatar Batsuren, Ekaterina Vylomova, Verna Dankers, Tsetsuukhei Delgerbaatar, Omri Uzan, Yuval Pinter, and Gábor Bella. 2024. Evaluating subword tokenization: Alien subword composition and oov generalization challenge. *Preprint*, arXiv:2404.13292.

Thomas Bauwens. 2025. Tktkt: the tokeniser toolkit.

Lisa Beinborn and Yuval Pinter. 2023. Analyzing cognitive plausibility of subword tokenization. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4478–4486, Singapore. Association for Computational Linguistics.

Iaroslav Chelombitko, Egor Safronov, and Aleksey Komissarov. 2024. Qtok: A comprehensive framework for evaluating multilingual tokenizer quality in large language models. *CoRR*, abs/2410.12989.

Marco Cognetta, Vilém Zouhar, Sangwhan Moon, and Naoaki Okazaki. 2024. Two counterexamples to tokenization and the noiseless channel. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 16897–16906, Torino, Italia. ELRA and ICCL.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, and 181 others. 2025. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

Miguel Domingo, Mercedes García-Martínez, Alexandre Helle, Francisco Casacuberta, and Manuel Herranz. 2019. How much does tokenization affect neural machine translation? In *Computational Linguistics and Intelligent Text Processing - 20th International Conference, CICLing 2019, La Rochelle, France, April 7-13, 2019, Revised Selected Papers, Part I*, volume 13451 of *Lecture Notes in Computer Science*, pages 545–554. Springer.

Matthias Gallé. 2019. Investigating the effectiveness of BPE: The power of shorter sequences. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1375–1381, Hong Kong, China. Association for Computational Linguistics.

Omer Goldman, Avi Caciularu, Matan Eyal, Kris Cao, Idan Szpektor, and Reut Tsarfaty. 2024. Unpacking tokenization: Evaluating text compression and its correlation with model performance. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2274–2286, Bangkok, Thailand. Association for Computational Linguistics.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Katharina Hämmerl, Tomasz Limisiewicz, Jindřich Libovický, and Alexander Fraser. 2025. Beyond literal token overlap: Token alignability for multilinguality. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 756–767, Albuquerque, New Mexico. Association for Computational Linguistics.

Vojtěch John, Zdeněk Žabokrtský, and Benjamin Reeves. 2026. Modeling word-internal structures: Morphological segmentation across 58 languages. In *Proceedings of the Workshop on Structured Linguistic Data and Evaluation (SLiDE) at LREC-COLING 2026*, pages 180–190, Palma, Mallorca, Spain.

Pratik Joshi, Sebastin Santy, Amar Budhiraja, Kalika Bali, and Monojit Choudhury. 2020. The state and fate of linguistic diversity and inclusion in the NLP world. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6282–6293, Online. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Jindřich Libovický and Jindřich Helcl. 2024. Lexically grounded subword segmentation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7403–7420, Miami,

Florida, USA. Association for Computational Linguistics.

Tomasz Limisiewicz, Jiří Balhar, and David Mareček. 2023. Tokenization impacts multilingual language modeling: Assessing vocabulary allocation and overlap across languages. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5661–5681, Toronto, Canada. Association for Computational Linguistics.

Clara Meister. 2025. Tokeval: A tokenizer analysis suite.

Meta AI. 2025. The Llama 4 herd: The beginning of a new era of natively multimodal AI innovation. https://ai.meta.com/blog/llama-4-multimodal-intelligence/. Blog post.

Mistral AI. 2025. Mistral medium 3. https://mistral.ai/news/mistral-medium-3. Blog post.

Sebastian Nordhoff and Harald Hammarström. 2011. Glottolog/langdoc: Defining dialects, languages, and language families as collections of resources. In *First International Workshop on Linked Science 2011-In conjunction with the International Semantic Web Conference (ISWC 2011)*.

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

Juan Antonio Perez-Ortiz, Felipe Sánchez-Martínez, Víctor M. Sánchez-Cartagena, Miquel Esplà-Gomis, Aaron Galiano Jimenez, Antoni Oliver, Claudi Aventín-Boya, Alejandro Pardos, Cristina Valdés, Jusèp Loís Sans Socasau, and Juan Pablo Martínez. 2024. Expanding the FLORES+ multilingual benchmark with translations for Aragonese, aranese, Asturian, and Valencian. In *Proceedings of the Ninth Conference on Machine Translation*, pages 547–555, Miami, Florida, USA. Association for Computational Linguistics.

Aleksandar Petrov, Emanuele La Malfa, Philip H. S. Torr, and Adel Bibi. 2023. Language model tokenizers introduce unfairness between languages. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

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

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

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

5 Team, Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, Kedong Wang, Lucen Zhong, Mingdao Liu, Rui Lu, Shulin Cao, Xiaohan Zhang, Xuancheng Huang, Yao Wei, and 152 others. 2025. Glm-4.5: Agentic, reasoning, and coding (arc) foundation models. *Preprint*, arXiv:2508.06471.

Omri Uzan, Craig W. Schmidt, Chris Tanner, and Yuval Pinter. 2024. Greed is all you need: An evaluation of tokenizer inference methods. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 813–822, Bangkok, Thailand. Association for Computational Linguistics.

Denny Vrandecic and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

xAI. 2024. Grok-2 beta release. https://x.ai/news/grok-2. Blog post.

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

A Example Evaluation Configuration File

The configuration uses YAML syntax that mirrors the components of the evaluation framework. The following is an excerpt from an example configuration file.

```
scorer:
input_dir: data/flores-200
output_dir: output/flores-200
metrics:
- metric: token_length
metric_label: tok_length
[...]
languages_info: null
languages:
- en
- fr
```

```
[...]
tokenizers:
- tokenizer_A
- tokenizer_B
[...]
```

The central component of the configuration is `scorer`. It requires a list of `languages` (language codes), `tokenizers`, and `metrics`, along with input and output directories. Other optional parameters can also be provided.[10] The `<scorer.input_dir>/` must contain `<tokenizer>/<languages>.txt` files for each tokenizer and language combination. The `metric` instances are defined as individual sets of attributes. Each metric definition must contain a `metric` attribute indicating the metric class type and a unique `metric_label` identifying a specific instance of that metric and its parameters.

Optionally, the user can also include a `scorer.languages_info` definition file. During evaluation, this file is used as a sanity check, restricting evaluation to language codes in `languages` that also appear in the `scorer.languages_info` file. This ensures compatibility with the visualization component.

Individual configuration values can be overridden at the CLI call using OmegaConf-style syntax by directly specifying their keys, e.g., `scorer.output_dir=<my_dir>` to override the output directory.

[10] See the `scorer.py` definition for a full list.