

Performance of Grapheme-Based Tokenizers on Word-level and Natural Language Understanding tasks for Indic Languages

Shagun Dwivedi¹, Mohammad Saquib¹, Kaushik Gopalan^{1,2}

¹Centre for Interdisciplinary AI, FLAME University, India

²School of Computing and Data Sciences, FLAME University, India

Correspondence: {shagun.dwivedi, mohammad.saquib, kaushik.gopalan}@flame.edu.in

Abstract

Tokenization is one of the factors which influences the multilingual downstream performance of a language model. This paper demonstrates that vocabulary constrained grapheme cluster based tokenization leads to improved performance on Natural Language Understanding and Word-level tasks. We assess three different grapheme cluster based tokenizers, which are the Grapheme Cluster tokenizer, the Grapheme Pair Encoding, and a vocabulary constrained version of Grapheme Pair Encoding, across eleven Indic languages, namely Assamese, Bengali, Gujarati, Hindi, Kannada, Malayalam, Marathi, Odia, Punjabi, Tamil, and Telugu. We present the intrinsic performance of three grapheme cluster based tokenizers using Fertility, Rényi Efficiency, and Percentile Frequency. We also compare the performance of these grapheme based tokenizers with seven popular multilingual and Indic-focused tokenizers on downstream tasks. We find that the vocabulary constrained Grapheme Pair Encoding tokenizer exhibits competitive and sometimes superior downstream performance in comparison to other grapheme cluster based tokenizers.

1 Introduction

Multilingual language models that have been trained with hundreds of languages exhibit major inter-language disparity (Petrov et al., 2023). Advances in Indic language models are mostly based on adapting pretrained language models with large datasets. These datasets are sometimes created by translating non-Indic text using machine translation. A major advantage of these models is avoiding pre-training from scratch due to their ability to generalise. (Doddapaneni et al., 2023; Dabre et al., 2021; Khanuja et al., 2021; Sarvam AI, 2025; Bendale et al., 2024)

While a majority of work focuses on adapting established model architectures to Indic languages, tokenization is an under-studied aspect of a model’s

performance. Subword tokenizers that operate on subword or byte level can split words at orthographically unnatural boundaries, producing fragmented text that does not make sense to a native speaker. Dwivedi and Gopalan (2026) have introduced the Grapheme Cluster tokenization method for Hindi and Marathi, which relies on the orthographic boundaries of the *Devanāgarī* script, splitting the words into grapheme cluster. Further, Velayuthan and Sarveswaran (2025) introduce Grapheme Pair Encoding (referred to as GPE henceforth) tokenizer, an improvement to the BPE algorithm by incorporating grapheme-level splitting at the pretokenization step, for Tamil, Sinhala, and Hindi.

We adapt these two tokenizers to 11 Indian languages, namely Assamese, Bengali, Gujarati, Hindi, Kannada, Malayalam, Marathi, Odia, Punjabi, Tamil, and Telugu, which are all languages written using Indic scripts. These scripts belong to the abugida family; an abugida or an alphasyllabary writing system consists of consonant-vowel syllables, which sometimes make up one visual unit, with a diacritic form of the vowel (known as a *mātrā*) attached to the consonant, or a single consonant and its inherent vowel. Indic scripts also contain a large number of conjunct characters (or *samyuktākshara*), where two or more consonants are joined using *halantas* (a diacritic which is used to suppress the inherent vowel), and sometimes merged with a *mātrā* to form one syllable. These consonant-vowel pairs and conjuncts are known as grapheme clusters. Table 1 shows how different tokenizers tokenize

As an improvement to the GPE tokenizer, we introduce the GPE-Constrained (GPE-C) tokenizer, which retains the K most frequent grapheme clusters in the base vocabulary, and adds M grapheme cluster sequences to the vocabulary by frequency-based iterative merging. This provides us with a tunable vocabulary size consisting of base vocabulary, merges, and fallback characters, which are

eleven Indic languages: Hindi, Marathi, Gujarati, Bengali, Assamese, Odia, Punjabi, Telugu, Kannada, Tamil, and Malayalam. The grapheme-based tokenizers were trained using 100,000 samples per language from IndicCorpV2 (Doddapaneni et al., 2023). Next, the samples were transliterated to IPA and ITRANS using Aksharamukha (Rajan, 2018), resulting in a total of 3.3 million samples in the training dataset of the tokenizers.

3.1 Implementation

The implementation of the Grapheme Cluster segmentation is based on the work of Dwivedi and Gopalan (2026), but we extend the same algorithm to Gujarati, Bengali, Gurumukhi, Tamil, Telugu, Kannada, and Odia scripts by adding *mātras* from these scripts to the *combining_marks* and adding the *virāma* symbols from each script.

But in some languages, such as Tamil, and Malayalam, there are certain *mātras* which, when attached to a consonant, can not be described as one visual unit even though they are vowels dependent on the preceding consonant or conjunct. These *mātras* are not added to the *combining_marks* set. For example, the Tamil syllable லா (*lā*), is written as two visually distinct glyphs ல (*la*) and ா (*ā*) respectively. These also include *mātras* such as ு (*au*) which appear on both sides of a consonant; for example, in the syllable லௌ (*lau*), the *mātra* appears on both sides of ல, and the syllable contains three visually distinct glyphs.

The GPE-C tokenizer is parameterized by two values: base vocabulary size K , and merge count M . Beyond the top- K grapheme clusters, all characters are decomposed into its constituent Unicode codepoints. For example, if the Malayalam grapheme cluster സ്കി (*ski*) is not in the top- K elements, it is decomposed into the following Unicode codepoints: ണ (U+0D38), ള് (U+0D4D), ി (U+0D15), and ി (U+0D3F).

Next, each entry in the base vocabulary is re-encoded into Private Use Area (referred to as PUA henceforth) codepoints with whitespace insertion at the end if the original entry ended with a whitespace, denoting the word boundary. We experimented with four different values for K and M , deciding on $K = 2000$ and $M = 2000$ based on the intrinsic performance of the tokenizer, in comparison with GPE and the Grapheme Cluster tokenizer (refer to Section 3.2.1).

As baselines, we use seven out-of-the-box tokenizers, namely Gemma3 (Team et al., 2025), GPT-OSS (OpenAI, 2025), IndicBertv2 (Doddapaneni et al., 2023), MuRIL (Khanuja et al., 2021), Sarvam-1 (Sarvam AI, 2024), Sarvam-M (Sarvam AI, 2025), and XLM-R (Conneau et al., 2020). These tokenizers have multilingual and Indic-focused vocabularies, however they were not trained on the same dataset as the grapheme cluster-based tokenizers and may contain tokens from other languages.

3.2 Evaluation

The tokenizers are evaluated using information-theoretic intrinsic metrics and based on their downstream performance on several word-level and Natural Language Understanding (referred to as NLU henceforth) tasks¹. We note that due to the limited availability of training data, the experiments are limited to relatively small model and tokenizer sizes as larger models may saturate under low-resource conditions.

3.2.1 Intrinsic Evaluation

For the intrinsic evaluation of the tokenizers, we present the Fertility (Ali et al., 2024), Rényi efficiency and Percentile Frequency (Zouhar et al., 2023). Fertility refers to the average number of subword tokens produced per tokenized word. Rényi efficiency is the Rényi entropy of the token-frequency distribution at $\alpha = 2.5$, normalized by $\log_2$ of the number of unique observed tokens. Percentile Frequency is the cumulative probability mass of tokens whose frequency rank lies in the 3rd – 83rd percentile. Higher values of Rényi efficiency have been shown to improve the downstream performance of language models on some tasks (Zouhar et al., 2023), however they have also been shown to be weakly correlated to other tasks (Dwivedi and Gopalan, 2026; Cognition et al., 2024).

The intrinsic metrics were calculated using the *devtest* subset of the FLORES-101 parallel corpus (Goyal et al., 2022) (CC BY-SA 4.0 license), which consists of 1012 sentences per language.

3.2.2 Extrinsic Evaluation

The extrinsic performance of a tokenizer is based on the downstream performance of a language model that uses the tokenizer. The downstream performance of the grapheme-based tokenizers is

¹The source code is available at <https://github.com/flame-cai/constrained-gpe>

compared with the seven out-of-the box tokenizers on NLU and Word-level tasks. The NLU tasks primarily assess the models’ ability to grammatical structuring and meaning representation; for this their performance is evaluated on three tasks: (i) *Named Entity Recognition (NER)*, (ii) *Morphological Inflection Generation*, based on the SIGMORPHON–UniMorph 2022 Shared Task 0: Generalization and Typologically Diverse Morphological Inflection (Kodner et al., 2022), and (iii) *Lemmatization*(de Marneffe et al., 2021).

We also compare the performance of the tokenizers on word-level tasks related to pronunciation and spelling of text; their performance is assessed on two benchmarks for this: (i) *grapheme-to-phoneme (G2P) conversion*, based on the SIGMORPHON 2020 shared Task 1 (Gorman et al., 2020) and (ii) *single-word transliteration*.

For conducting the extrinsic evaluation, one encoder-decoder based T5 model (Raffel et al., 2020) was pretrained from scratch for each of the aforementioned tasks. For a given task, each model was trained using the training samples from all the languages. Each model consists of 4 layers, other sub-layers and embeddings have the dimensions $d_{model} = 512$, the feed-forward neural network in each of the layer has the output dimensions $d_{ff} = 2048$ followed by ReLU non-linearity, and the key-value matrices have the dimensions $d_{kv} = 32$. For regularization, the *dropout_rate* is set to 0.1. The attention mechanisms of the models have 8 heads. The model was trained using the AdaFactor optimizer (Shazeer and Stern, 2018) with adaptive learning rate and a *weight_decay* of 0.01. During training the *batch_size* was set to 16. The model was trained for 10 epochs with early stopping on validation loss. All runs use *random_seed* = 0. The training took approximately 10 hours on three 48 GB NVIDIA RTX A6000 GPUs.

The multilingual NER, we use the Naamapadam NER dataset (Mhaske et al., 2023) for the 11 languages, under the Creative Commons CC0 license. The number of samples is restricted to 10,000 per language in the training set, resulting in 55,000 training samples for the task. For the Morphological Inflection Generation task, the inflection dataset for Assamese, Bengali, Hindi, Kannada, and Telugu was taken from Universal Morphology Project (UniMorph) repository (Batsuren et al., 2022). The maximum number of samples is limited to 10,000 samples per language for each of the language; the Assamese and Hindi datasets have 10,000 sam-

ples, whereas there are 3,999 samples for Bengali, 6,755 for Gujarati, 4,997 for Kannada and 1,225 samples for Telugu, resulting in a total of 36,976 samples. For Lemmatization, we use the Universal Dependencies (de Marneffe et al., 2021) Lemmatization Tree Bank (Hindi, Tamil, Marathi in-domain tree banks for training and evaluation, and Hindi-Parallel Universal Dependencies (PUD) and Tamil-Modern Written Tamil Treebank (MWTT) for same-language transfer evaluation). The training samples are limited to 10,000 per language for this task as well. This resulted in a total of 13,577 samples in the training dataset with 10,000 Hindi samples, 937 Marathi samples, and 2,640 Tamil samples.

The G2P task is trained and evaluated using the WikiPron broad phonemic transcriptions data (Lee et al., 2020), which is released under the CC BY-SA 4.0 license, capping the training set to a maximum of 10,000 samples per language. The training set contains 2,633 Assamese, 6,000 Bengali, 1,854 Gujarati, 10,000 Hindi, 796 Kannada, 6,390 Malayalam, 2,413 Marathi, 634 Punjabi, 6,213 Tamil, and 2,967 Telugu training samples, with a total of 39,900 training samples. For transliteration, we use the Dakshina dataset (Roark et al., 2020), with 25,000 samples per language in the training set (licensed under CC BY-SA 4.0). The transliteration is evaluated using character error rate (CER), whereas the G2P tasks is evaluated with word error rate (WER) and phoneme error rate (PER).

4 Results

Tokenizer	Fertility ($\downarrow$)	Rényi Eff. ($\uparrow$)	Percentile Freq. ($\uparrow$)	Vocab. Size
GraphemeCluster	3.420	0.635	0.509	68,214
GPE (K_{all}, M_{32k})	2.434	0.698	0.613	100,107
GPE (K_{32k}, M_{32k})	2.439	0.699	0.616	64,343
GPE (K_{2k}, M_{32k})	2.602	0.747	0.696	34,678
GPE (K_{all}, M_{2k})	3.261	0.659	0.538	70,362
GPE-C (K_{500}, M_{500})	5.420	0.703	0.762	2,062
GPE-C (K_{1k}, M_{1k})	4.662	0.751	0.794	2,942
GPE-C (K_{2k}, M_{2k})	3.981	0.775	0.790	4,824
GPE-C (K_{5k}, M_{5k})	3.329	0.750	0.723	10,714

Table 2: Intrinsic evaluation of the grapheme cluster based tokenizers on FLORES (*devtest*) evaluation benchmark dataset consisting of 1012 sentences in each of the eleven languages.

In this section, we present and discuss the results of our experiments on the grapheme cluster based tokenizers. The performance of the grapheme cluster based tokenizers on intrinsic metrics is reported in Table 2. The table shows the performance of four GPE-C configurations based on the base vocabu-

Tokenizer	as (n=51)	bn (n=607)	gu (n=1076)	hi (n=867)	kn (n=1019)	ml (n=974)	mr (n=1080)	or (n=994)	pa (n=993)	ta (n=758)	te (n=847)	$F1_w$ (n=9266)	$F1_\mu$ (n=9266)	$F1_m$ (n=9266)
GPE-C	51.16	60.42	<u>67.39</u>	66.04	64.66	69.40	<u>72.39</u>	18.14	58.18*	48.98	70.24	59.70*	62.65*	58.82*
GPE	38.29	55.31	65.45	60.39	62.42	65.84	69.38	15.57	53.52	47.66	67.69	56.45	59.537	54.68
GraphemeCluster	<u>45.45</u>	56.23	66.84	61.76	64.90	65.43	70.63	14.81	<u>55.39</u>	44.53	68.01	57.10	60.34	55.82
Gemma-3	31.82	57.34	64.24	63.53	61.37	65.69	71.01	14.19	41.28	47.28	64.92	55.02	58.64	52.97
GPT-OSS	38.30	56.62	66.84	62.91	<u>65.67</u>	68.60	69.16	<u>17.57</u>	45.07	47.50	67.72	56.85	59.84	55.09
IndicBERTv2	40.00	61.08	67.16	64.63	65.65	<u>68.68</u>	70.36	11.55	52.92	51.69	65.13	57.80	61.47	56.26
MuRIL	32.56	60.57	64.29	64.90	64.88	66.27	69.05	12.54	52.92	48.94	65.28	56.82	60.27	54.75
Sarvam-1	39.02	<u>60.77</u>	67.93	64.11	64.04	68.19	69.98	12.86	51.90	49.72	67.23	57.60	61.17	55.98
Sarvam-M	38.09	59.33	64.46	61.22	64.17	67.16	70.83	13.61	45.99	46.67	65.28	55.85	59.24	54.25
XLM-R	43.24	57.82	66.93	<u>65.94</u>	66.17	67.53	72.94	17.42	51.79	<u>49.88</u>	<u>68.32</u>	<u>58.59</u>	<u>61.82</u>	<u>57.09</u>

Table 3: F1 scores(%) of the multilingual NER models trained using different tokenizers on the Naamapadam NER dataset. $F1_w$, $F1_\mu$, and $F1_m$ are the weighted, micro and the unweighted (macro) average F1 scores across the 11 languages. The best score per column is highlighted in **bold**, statistically significant improvement with $p < 0.05$ is marked with an **asterisk***, and the second-best is underlined.

lary size, K , and number of merges, M . We find that the GPE-C tokenizer trained with the limited size of 2,000 for both K and M shows the highest Rényi efficiency, without much degradation of the other intrinsic metrics. Pushing the vocabulary size to 10,000 with $k = 5,000$ and $M = 5,000$ leads to a slight reduction in fertility but yields diminishing returns on the distribution efficiency. GPE-C (K_{2k} , M_{2k}) achieves a higher Rényi efficiency than either partially constrained variant: GPE (K_{all} , M_{2k}), which limits only the merges, and GPE (K_{2k} , M_{32k}), which limits only the base vocabulary. The GPE (K_{all} , M_{32k}) tokenizer exhibits the best Fertility score, but struggles with the Rényi Efficiency and Percentile Frequency, both of which are much lower than the various GPE-C tokenizers. Constraining one while leaving the other large is therefore not sufficient. The gain in distributional efficiency arises from constraining the base vocabulary and the number of merges together. The Grapheme Cluster tokenizer exhibits the lowest Rényi Efficiency and Percentile Frequency. The language-wise intrinsic metrics are available in Appendix A.

Table 3 reports the F1 scores of the grapheme cluster based tokenizers compared to other tokenizers on the NER task. It presents the $F1_w$, the per-language F1 score weighted by test-set size, and $F1_\mu$, the micro-F1 score which is obtained by pooling true positives, false positives, and false negatives across all 11 test sets, and $F1_m$ is the unweighted (macro) mean of the 11 per-language F1 scores. The model trained with the GPE-C tokenizer achieves the highest weighted and micro-F1 score. XLM-R achieves the second-best average scores, while the GPE and Grapheme Cluster tokenizer based models show mediocre performance on this task. No models show a significantly poor F1 scores across the languages, except for Assamese where the model using the Gemma-3 tokenizer exhibits a low F1 score of 31.82 in comparison to the GPE-C based model’s score of 51.16.

The model performance on the Morphological Inflection Generation task is presented in Table 4. On average, the Sarvam-M tokenizer shows the best Accuracy and CER for this task, followed closely by the GPE-C tokenizer. However, the per-language metrics show much more variance for this task.

Tokenizer	Inflection Accuracy ($\uparrow$)							CER ($\downarrow$)						
	as (n=4684)	bn (n=222)	gu (n=375)	hi (n=2721)	kn (n=277)	te (n=68)	Avg _w (n=8347)	as (n=4684)	bn (n=222)	gu (n=375)	hi (n=2721)	kn (n=277)	te (n=68)	Avg _w (n=8347)
GPE-C	78.29	94.14	90.93	99.12*	<u>74.37</u>	94.12*	86.07	5.35	22.90	1.87	0.11*	11.90	2.40	3.79
GPE	73.762	<u>95.49</u>	88.27	93.05	73.65	86.77	81.38	7.69	14.94	2.77	1.09	11.69	5.60	5.33
GraphemeCluster	76.79	95.95	88.80	96.69	72.56	92.65	84.32	6.82	12.10	2.77	<u>0.52</u>	11.85	4.32	4.57
Gemma-3	70.92	86.04	85.33	92.65	70.40	89.71	79.19	8.01	3.90	3.32	1.25	12.37	4.48	5.35
GPT-OSS	<u>80.46</u>	92.79	88.53	94.41	76.53	88.24	85.64	<u>5.33</u>	<u>2.18</u>	2.38	1.00	16.29	<u>3.20</u>	3.81
IndicBERTv2	1.09	18.47	5.33	43.40	23.10	4.41	16.29	87.61	45.51	68.64	19.51	61.71	58.88	60.14
MuRIL	61.64	77.93	84.00	93.90	55.24	66.18	73.42	11.53	6.61	4.80	1.07	20.62	12.80	7.57
Sarvam-1	75.11	91.89	88.80	94.78	65.34	76.47	82.27	7.16	2.66	2.69	1.28	13.70	8.32	4.91
Sarvam-M	81.68	93.69	93.60	<u>96.73</u>	72.20	83.82	87.14*	4.95	1.54	1.21	0.56	11.90	7.52	3.32
XLM-R	73.06	89.64	89.07	97.28	65.34	75.00	81.87	7.59	3.07	2.14	1.05	40.67	8.00	5.68

Table 4: Results from the Morphological Inflection Generation task using the UniMorph data, reported as percentages (%). Avg_w denotes the weighted average across the languages. The best score per column is highlighted in **bold**, statistically significant improvement with $p < 0.01$ is marked with an **asterisk***, and the second-best is underlined.

Tokenizer	Accuracy ($\uparrow$)						CER ($\downarrow$)					
	hi (n=5378)	hi_pud (n=5223)	mr (n=204)	ta (n=987)	ta_mwtt (n=858)	Avg (n=12650)	hi (n=5378)	hi_pud (n=5223)	mr (n=204)	ta (n=987)	ta_mwtt (n=858)	Avg (n=12650)
GPE-C	84.64	83.88*	62.26	65.86	47.44	79.98*	6.58	8.44	20.19	20.46	36.37	10.56
GPE	79.29	76.53	<u>64.22</u>	67.98	47.09	74.84	8.83	11.35	21.69	19.28	35.94	12.62
GraphemeCluster	83.62	<u>80.70</u>	66.67	64.84	<u>47.20</u>	<u>78.21</u>	5.88	8.27	15.78	23.94	44.94	10.96
Gemma-3	45.44	40.05	50.49	45.49	26.22	42.00	50.93	58.30	35.73	54.87	84.46	56.26
GPT-OSS	62.57	58.15	57.84	59.07	39.16	58.81	27.18	34.59	22.62	34.00	43.37	31.84
IndicBERTv2	37.30	30.25	42.16	34.85	16.08	32.84	80.09	94.92	67.63	76.76	82.79	86.09
MuRIL	43.46	35.94	45.59	37.69	19.11	38.28	62.83	64.20	52.67	66.39	82.96	64.82
Sarvam-1	64.32	55.81	57.84	50.56	30.07	57.30	44.78	51.20	54.64	84.36	128.76	56.17
Sarvam-M	74.23	72.81	59.31	62.72	46.27	70.61	24.07	25.95	29.70	23.91	38.68	25.82
XLM-R	72.46	68.52	62.74	63.12	37.18	67.56	24.34	27.20	26.80	29.70	61.94	28.35

Table 5: Accuracy (%) and Character Error Rate (%) of the multilingual Lemmatization models. The best score per column is highlighted in **bold**, statistically significant improvement with $p < 0.01$ is marked with an **asterisk***, and the second-best is underlined. The reported average is weighted by the test set size.

Both the Grapheme Cluster and GPE tokenizers have the best and second-best accuracy scores for Bengali, but they exhibit a mediocre CER for the language. IndicBERTv2 shows poor performance for this task. MuRIL shows very imbalanced performance across the languages (55%-93% accuracy, and 1.07% to 20.62% CER).

The accuracy and character error rate achieved on the Lemmatization task are presented in Table 5. We find that all the grapheme based tokenizers show the highest performance on both the metrics. On average, the models trained with both the GPE-C and Grapheme Cluster tokenizer outperform other mod-

els. On the other hand, IndicBERTv2 and MuRIL based models showed the worst performance across all the languages for this task.

Table 6 and Table 7 report the performance of the tokenizers (phoneme error rate and word error rate, respectively) on the Grapheme to Phoneme task. The GPE-C tokenizer also demonstrates the lowest average PER for the G2P task; it shows the best performance for all tasks except Assamese and Hindi, where it shows the second-best performance, being outperformed by the Sarvam-M tokenizer by a small percentage. The Grapheme Cluster tokenizer exhibits a decent third best performance, closely

Tokenizer	as (n=146)	bn (n=333)	gu (n=102)	hi (n=1263)	kn (n=44)	ml (n=355)	mr (n=134)	pa (n=35)	ta (n=345)	te (n=164)	Avg (n=2921)
GPE-C	<u>8.92</u>	18.20	8.40	<u>5.12</u>	8.09	9.67*	7.41	41.18	1.08*	2.58	7.17*
GPE	16.74	21.73	19.08	8.36	29.36	13.10	10.39	52.94	4.37	8.12	11.31
GraphemeCluster	14.27	25.56	13.93	8.01	22.55	<u>13.99</u>	9.83	53.59	3.46	8.69	11.25
Gemma-3	16.32	33.28	21.18	24.71	27.23	21.36	23.93	48.37	17.39	11.37	23.23
GPT-OSS	10.01	21.11	22.52	10.23	27.23	18.38	11.68	50.33	4.15	6.88	12.52
IndicBERTv2	43.21	58.03	84.35	91.39	89.79	70.62	65.53	109.80	74.30	68.10	78.39
MuRIL	47.87	48.98	54.96	47.76	48.09	48.89	52.99	56.86	32.02	43.65	46.30
Sarvam-1	14.54	27.61	40.65	22.16	51.49	23.62	26.78	75.82	19.21	23.11	23.87
Sarvam-M	8.09	<u>19.58</u>	10.12	4.89	13.62	14.11	8.55	42.48	2.51	4.87	8.29
XLM-R	18.66	<u>34.77</u>	39.31	27.41	64.25	34.42	33.33	81.70	23.24	24.64	29.63

Table 6: Phoneme Error Rate (%) of the models trained on the Grapheme to Phoneme task across all languages. The reported average is pooled by phoneme count. The best score per column is highlighted in **bold**, statistically significant improvement with $p < 0.01$ is marked with an **asterisk***, and the second-best is underlined.

Tokenizer	as (n=146)	bn (n=333)	gu (n=102)	hi (n=1263)	kn (n=44)	ml (n=355)	mr (n=134)	pa (n=35)	ta (n=345)	te (n=164)	Avg (n=2921)
GPE-C	35.62	<u>65.17</u>	30.39	27.71	22.73	28.17*	32.09	85.71	5.80*	10.98	29.82
GPE	46.56	72.07	52.94	34.60	50.00	36.90	40.29	94.29	14.20	22.56	38.51
GraphemeCluster	42.47	79.28	41.18	34.92	40.91	41.41	39.55	91.43	15.36	31.71	39.85
Gemma-3	40.41	78.68	55.88	48.69	47.73	54.93	52.98	88.57	42.03	29.27	51.49
GPT-OSS	<u>30.82</u>	63.66	48.04	30.25	38.64	50.42	35.08	<u>85.71</u>	14.49	20.12	35.74
IndicBERTv2	73.29	92.79	98.04	99.37	95.45	94.08	90.30	97.14	96.23	93.90	95.45
MuRIL	96.58	99.70	100.00	99.92	100.00	98.87	97.76	100.00	100.00	98.78	99.45
Sarvam-1	37.67	72.67	62.74	45.05	63.64	52.11	54.48	97.14	40.29	48.78	50.29
Sarvam-M	26.71	66.67	<u>33.33</u>	20.03*	34.09	40.84	<u>34.33</u>	88.57	<u>11.01</u>	<u>15.85</u>	29.07
XLM-R	58.90	89.19	75.49	77.44	81.82	76.90	79.85	91.43	71.59	62.20	76.52

Table 7: Word Error Rate (%) of the models trained on the Grapheme to Phoneme task across all languages. The reported average is weighted by per-language test size. The best score per column is highlighted in **bold**, statistically significant improvement with $p < 0.01$ is marked with an **asterisk***, and the second-best is underlined.

Tokenizer	bn (n=2, 500)	gu (n=2, 500)	hi (n=2, 500)	kn (n=2, 500)	ml (n=2, 500)	mr (n=2, 500)	pa (n=2, 500)	ta (n=2, 500)	te (n=2, 500)	Avg (n=22, 500)
GPE-C	15.13*	14.20*	11.82*	5.65*	4.73*	8.67*	14.84	9.52*	9.41*	9.94*
GPE	25.16	20.55	19.70	11.56	12.37	14.95	19.79	15.28	16.94	16.84
GraphemeCluster	<u>18.23</u>	<u>16.88</u>	<u>15.49</u>	<u>8.72</u>	<u>8.02</u>	<u>10.87</u>	<u>15.13</u>	<u>10.87</u>	<u>14.68</u>	<u>12.76</u>
Gemma-3	55.79	30.14	49.44	21.09	23.43	36.04	26.96	35.00	25.97	32.82
GPT-OSS	33.10	27.78	32.58	18.77	17.82	22.99	24.39	18.42	21.40	23.38
IndicBERTv2	89.82	85.66	86.47	96.00	88.69	84.20	83.50	91.02	90.15	88.78
MuRIL	82.43	76.29	80.46	74.04	80.69	82.46	71.97	81.37	79.07	78.92
Sarvam-1	42.68	38.15	42.35	33.60	28.20	35.69	35.12	37.08	36.42	36.15
Sarvam-M	25.80	18.49	23.23	10.89	8.91	16.19	20.50	15.09	17.59	16.71
XLM-R	45.49	38.22	44.79	36.35	42.21	39.57	37.51	39.66	40.17	40.34

Table 8: Character Error Rate (%) of the models trained on the Transliteration task using the benchmark Dakshina dataset. The reported average is pooled by character count. The best score per column is highlighted in **bold**, statistically significant improvement with $p < 0.01$ is marked with an **asterisk***, and the second-best is underlined.

followed by the GPE tokenizer. The IndicBERTv2 tokenizer leads to the worst performance based on the PER. The MuRIL tokenizer has the worst WER with an average of 99.45%, though it shows a comparatively less poor PER, averaging to 46.30%. It also has a 100% word error rate in multiple languages.

The results of the transliteration task are presented in Table 8. For this task, both GPE-C and Grapheme Cluster based models significantly outperform other models. The GPE-C based model shows the highest average performance, followed by the model using the Grapheme Cluster tokenizer as the second best performer for this task. The GPE tokenizer and Sarvam-M tokenizer are comparable in performance on this task. On the other hand, the models using the IndicBERTv2 tokenizer and the MuRIL tokenizer show a very high character error rate in transliteration for all languages.

Averaged over all languages, the results are significant for the Morphological Inflection Generation (weighted accuracy with $p < 0.05$), NER (weighted, macro and micro F1 scores with $p < 0.05$), Lemmatization (weighted accuracy with $p < 0.01$), G2P (weighted PER with $p < 0.01$), and Transliteration (weighted CER with $p < 0.01$) tasks. However, we observe that the results are not consistently significant across all the languages for the tasks. For example, the performance of the model trained with GPE-C tokenizer on Punjabi for Transliteration, Tamil MWT for Lemmatization, Telugu for Morphological Inflection, and Punjabi, Oria, and Malayalam for NER were found to not be statistically significant in comparison to the second best performing models for the respective task and language.

While all errors made by a model can not be attributed to the tokenizer, in Table 9, we present a qualitative analysis of the outputs of the models, which can be caused due to the tokenizer used. For

the transliteration task, we observe that the errors made by the grapheme based tokenizers are due to the model transliterating English words written in Indic scripts to a version which is faithful to the pronunciation, but does not match with the English spellings. The Sarvam-M based model makes similar mistakes but the transliterations are more corrupted. However, in the case of models trained with other tokenizers, they often predict an unrelated English word, or merge an English subword with the first half of the transliteration.

Model	Input	Ground Truth	Prediction
<i>Transliteration</i>			
GraphemeCluster	बावटेल्	cocktail	kaktel
GPE-C	एंजाइम	enzyme	enjaaim
Gemma-3	आवाज	aavaz	dinos
GPT-OSS	अशोक	ashok	ashometer
Sarvam-1	अस्त्र	astr	taavrat
Sarvam-M	ऑपरेटिंग	operating	aaperingering
<i>Lemmatization</i>			
GraphemeCluster	बाँधना	मैं बाँधूँगी	मैं बाँध
Gemma-3	एवढे	एवढा	एप्लीडे
GPT-OSS	93	93	सम्म ग्राक
MuRIL	अडचनें	अडचन	अडडडडडड
Sarvam-1	ഉണ്ടാകുക	ഉണ്ടാകും	...വാർ
XLM-R	കൊடுക്കുക	കൊடு	൭൭൭൭൭൭...൭൭൭൭൭൭
XLM-R	எண்ணால்	நூண்	मध्यप्रदेश ाल
<i>Morphological Inflection Generation</i>			
Gemma-3	बेचना	मैं बेचूँगी	मैं बे बचगी
Gemma-3	इच्छा इच्छा	इच्छा श्रवो	इच्छा ॐ
GPT-OSS	सोचना	हम सोचेंगे	हम सोगे
GPT-OSS	चाहना	ये चाहेंगे	योहेंगे
IndicBERTv2	तिथि	तिथिदेर	तिथिदेर ॐ
IndicBERTv2	उछलना	हम उछलें	हम उछलें ॐ
MuRIL	ಹೇಳು	ನಾನು	ಹೇಳು ॐ
Sarvam-1	ದೊಂಬು	ದೊಂಬು	ವೆ ॐ
Sarvam-M	ಖುಡಲಿಡೆರೆ	ಖುಡಲಿಡೆರೆ	ಖುಡಲಿಡೆರೆ
XLM-R	उछलना	हम उछलें	हम ॐ

Table 9: Incorrect predictions made by different models on the Transliteration, Lemmatization and Morphological Inflection Generation tasks.

For the lemmatization task, we find that the Sarvam-1 based model often outputs broken bytes

instead of meaningful characters, Models using Gemma-3, GPT-OSS, and Sarvam-M struggle with lemmatization of digits, predicting text with mixed scripts, We also observed that the XLM-R based model often predicts mixed scripts of tokens. We find that many of the tokenizers predict broken text for the Morphological Inflection Generation task. The grapheme based tokenizers do not lead to such broken/missing tokens for any of the tasks; in the case of these tokenizers, the common errors contain repeated tokens or correctly spelled but wrong output for a given input. The GPT-OSS based model often predicts multiple *mātras* attached to the same consonants, which do not form any valid syllables.

Tokenizer	Samples with Undertrained Tokens	Samples with Undertrained Rows
<i>Transliteration</i>		
GraphemeCluster	50.84	15.44
GPE	72.76	27.85
GPE-C	6.79	1.45
<i>Grapheme to Phoneme</i>		
GraphemeCluster	42.62	15.33
GPE	60.08	25.16
GPE-C	11.09	2.95
<i>Lemmatization</i>		
GraphemeCluster	30.7	10.31
GPE	53.9	22.56
GPE-C	3.24	1.0
<i>Morphological Inflection Generation</i>		
GraphemeCluster	11.14	0.46
GPE	16.46	0.71
GPE-C	1.51	0.06
<i>NER</i>		
GraphemeCluster	45.58	2.28
GPE	54.28	4.08
GPE-C	1.39	0.05

Table 10: Percentage of both the embedding rows which received less than 25 gradient updates during training and the input tokens at inference which are represented by these under-trained rows.

To assess why constraining the tokenizer vocabulary impacts the downstream performance of a model trained with a grapheme cluster based tokenizer, we investigate the embeddings of each of the models since they are dependent on the tokenizer used by the model. We analyse the number of embedding rows trained for each model and for each downstream task. We count the number of embedding rows which have received gradient updates less than 25 times during training, and how many input tokens at the time of inference are represented using those rows. The higher the number of samples and tokens represented using the under-trained embedding rows, the greater is the reliance of the

model on the poorly trained representation at inference. We find that this number is much higher for the GPE and Grapheme cluster tokenizer than the GPE-C tokenizer for all the tasks.

Table 10 reports the percentage of samples and percentage of tokens encountered at inference for each task which were represented using the under-trained token embedding rows. We observe that the other two tokenizers lead to such embedding rows representing a larger number of samples at inference (3.8x to 39.1x across the downstream tasks). This indicates that the models trained using these tokenizers rely more frequently on under-trained rows of their embedding matrices than the model trained with GPE-C tokenizer. Across all tasks, the GPE-C tokenizer consistently leads to a smaller fraction of inference samples and tokens being represented by embeddings which received less than 25 gradient updates.

For example, we present some samples from the test set which were represented using such under-trained tokens, and led to poor outputs in Table 11. The table shows the task, language, input, ground truth, the tokenizer, minimum of the gradient update count for each embedding row used for the input, the tokenization of the input and the prediction by the downstream model on the given input. The minimum number of gradient updates received by any of the token embeddings which represent the input tokens for the model using the GPE-C tokenizer is much higher than for the other two tokenizers. In the Lemmatization example, the model using the other tokenizers received a minimum of only 2 gradient updates whereas the GPE-c model received 128 updates. Due to the Unicode fallback in the GPE-C tokenizer, the input can be represented through token embeddings which have undergone a substantially higher number of gradient updates than the other tokenizers. Hence GPE-C avoids relying on under-trained token embeddings and produces the correct prediction.

These results are consistent with the intrinsic evaluations, in which the GPE-C tokenizer exhibits the most balanced token distribution as compared to the other tokenizers. Across all the downstream tasks, the models trained with GPE-C have the smallest fraction of input tokens being represented by embeddings that are under-trained; i.e. embeddings which only received a few gradient updates while training the model. This suggests that a balanced token distribution can lead to a better trained token embedding matrix for the tokens encountered

Language	Input	Gold	Tokenizer	Min. Gradient Updates	Input Tokens	Prediction
Transliteration						
hi	अकापुल्को	akapulko	GraphemeCluster	2	अ/का/पु/ल्को	akapulcon
			GPE	2	अ/का/पु/ल्को	akapurikko
			GPE-C	148	अ/का/पु/ल/क/ो	akapulko
bn	ফরেস্তি	forestry	GraphemeCluster	6	ফ/রে/স্তি	forestri
			GPE	6	ফ/রে/স্তি	forestory
			GPE-C	99	ফ/রে/স/ট/ি	forestry
te	చట్టబద్ధ	chattabaddha	GraphemeCluster	11	చ/ట్ట/బ/ద్ధ	chattabadda
			GPE	11	చ/ట్ట/బ/ద్ధ	chattabada
			GPE-C	79	చ/ట్ట/బ/ద్/ద	chattabaddha
Grapheme to Phoneme						
ml	റ്റാറ്റാ	t t a: t t a:	GraphemeCluster	5	ട/റ്റാ	t a: t t a:
			GPE	5	റ്റാ/റ്റാ	t a: t t a:
			GPE-C	131	റ്റ/ാ/റ്റ/ാ	t t a: t t a:
Lemmatization						
ta	ரத்தன்குமார்	ரத்தன்குமார்	GraphemeCluster	2	ர/த்த/ன்/கு/ம/ா/ர/்	ரத்தன்குமாரம்
			GPE	2	ர/த்த/ன்/கு/மா/ர/்	ரத்தன்கும்
			GPE-C	128	ர/த்த/ன/்க/ு/மா/ர்	ரத்தன்குமார்

Table 11: Examples of inputs represented by untrained row embeddings leading to incorrect predictions by models trained with GPE and Grapheme Cluster tokenizer. Token boundaries are marked “/”. Correct prediction is highlighted in **bold**.

at inference, providing an explanation for the improved downstream performance.

5 Conclusion

In this paper, we demonstrate the impact of constraining the vocabulary size of the GPE tokenizer on intrinsic and extrinsic metrics. We find that optimizing the GPE vocabulary to contain the 2,000 most frequent grapheme clusters in the base vocabulary and 2,000 merges demonstrates the best balance of Fertility and Rényi Efficiency scores. This constrained vocabulary also leads to a competitive and sometimes superior performance in the three NLU and two word-level tasks in comparison to other multilingual and Indic-focused tokenizers.

Through this work, we also compare the performance of a vocabulary constrained GPE tokenizer to various other tokenizers on extrinsic metrics for 11 Indian languages. We find that the grapheme based tokenizers lead to the best downstream performance for three Natural Language Understanding tasks and two word level tasks. For some of the tasks, the model using the Sarvam-M tokenizer also shows competitive or superior performance, whereas its performance is middling for other tasks. We also find that IndicBERTv2 and MuRIL tokenizers show the poorest performance on average across all tasks, except for the Named Entity Recognition task, where the different models do not show a significant difference in their performance.

We observe that the performance of the tokenizers on the intrinsic metrics does not trans-

late to downstream performance. While the GPE-C (K_{2k} , M_{2k}) tokenizer exhibits superior performance on both the Rényi Efficiency and most of the downstream tasks, the same relationship does not hold for the GPE and Grapheme Cluster tokenizers. The GPE tokenizer outperforms the Grapheme Cluster tokenizer on all three of the intrinsic metrics, but it is outperformed by the latter on all downstream tasks.

Future work in this direction should entail evaluation of the tokenizers on more complex downstream tasks, such as question answering, long context understanding, and reasoning, for which the availability of large scale training data remains a bottleneck in low resource settings, especially for comparisons with state-of-the-art pretrained models.

6 Limitations

This work has some limitations, which are outlined in this section. The paper presents comparison of the downstream performance of the tokenizers on a limited number of NLU and word-level tasks. Evaluating the tokenizers on larger and more complex downstream tasks would require substantially greater computational resources and large-scale task-specific training corpora. This is primarily due to the fact that tokenizer adaptation requires pretraining a model from scratch instead of parameter-efficient adaptation techniques.

Further, all the out-of-the-box tokenizers studied in this paper were not trained on identical corpora, which may introduce a degree of imbalance in their

comparison. However, the primary objective of the comparison is to evaluate the impact of tokenization on downstream task performance rather than direct comparison of intrinsic information-theoretic abilities of the pretrained tokenizers. As the downstream evaluation intends to assess the impact of tokenization on task-specific model learning, we believe the work remains informative despite these weaknesses.

7 Acknowledgements

The authors acknowledge the usage of Claude Code for implementation of the downstream tasks, running the experiments, and generation of the results.

References

- Mehdi Ali, Michael Fromm, Klaudia Thellmann, Richard Rutmann, Max Lübbering, Johannes Leveling, Katrin Klug, Jan Ebert, Niclas Doll, Jasper Buschhoff, Charvi Jain, Alexander Weber, Lena Jurkschat, Hammam Abdelwahab, Chelsea John, Pedro Ortiz Suarez, Malte Ostendorff, Samuel Weinbach, Rafet Sifa, and 2 others. 2024. [Tokenizer choice for LLM training: Negligible or crucial?](#) In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3907–3924, Mexico City, Mexico. Association for Computational Linguistics.
- Khuyagbaatar Batsuren, Omer Goldman, Salam Khalifa, Nizar Habash, Witold Kieraś, Gábor Bella, Brian Leonard, Garrett Nicolai, Kyle Gorman, Yustinus Ghanggo Ate, Maria Ryskina, Sabrina Mielke, Elena Budianskaya, Charbel El-Khaissi, Tiago Pimentel, Michael Gasser, William Abbott Lane, Mohit Raj, Matt Coler, and 76 others. 2022. [UniMorph 4.0: Universal Morphology](#). In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 840–855, Marseille, France. European Language Resources Association.
- Abhijit Bendale, Michael Sapienza, Steven Ripplinger, Simon Gibbs, Jaewon Lee, and Pranav Mistry. 2024. [Sutra: Scalable multilingual language model architecture](#). *Preprint*, arXiv:2405.06694.
- Maharaj Brahma, N J Karthika, Atul Kumar Singh, Devaraja Adiga, Smruti Bhate, Ganesh Ramakrishnan, Rohit Saluja, and Maunendra Sankar Desarkar. 2026. [MorphTok: Morphologically grounded tokenization for indic languages](#). In *Tokenization Workshop*.
- Marco Cогnetta, 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.
- Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. [Unsupervised cross-lingual representation learning at scale](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.
- Raj Dabre, Himani Shrotriya, Anoop Kunchukuttan, Ratish Puduppully, Mitesh M. Khapra, and Pratyush Kumar. 2021. [IndicBart: A pre-trained model for natural language generation of indic languages](#). *Preprint*, arXiv:2109.02903.
- Marie-Catherine de Marneffe, Christopher D. Manning, Joakim Nivre, and Daniel Zeman. 2021. [Universal Dependencies](#). *Computational Linguistics*, 47(2):255–308.
- Sumanth Doddapaneni, Rahul Aralikkatte, Gowtham Ramesh, Shreya Goyal, Mitesh M. Khapra, Anoop Kunchukuttan, and Pratyush Kumar. 2023. [Towards leaving no Indic language behind: Building monolingual corpora, benchmark and models for Indic languages](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12402–12426, Toronto, Canada. Association for Computational Linguistics.
- Shagun Dwivedi and Kaushik Gopalan. 2026. [Comparative analysis of the intrinsic metrics for tokenizers and their effect on downstream tasks for Hindi and Marathi](#). In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 22652–22663, San Diego, California, United States. Association for Computational Linguistics.
- Takuro Fujii, Koki Shibata, Atsuki Yamaguchi, Terufumi Morishita, and Yasuhiro Sogawa. 2023. [How do different tokenizers perform on downstream tasks in scriptio continua languages?: A case study in Japanese](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pages 39–49, Toronto, Canada. Association for Computational Linguistics.
- Juan Luis Gastaldi, John Terilla, Luca Malagutti, Brian DuSell, Tim Vieira, and Ryan Cotterell. 2025. [The foundations of tokenization: Statistical and computational concerns](#). In *The Thirteenth International Conference on Learning Representations*.
- Rashi Goel and Fatiha Sadat. 2025. [Studying the effect of Hindi tokenizer performance on downstream tasks](#). In *Proceedings of the First Workshop on Natural Language Processing for Indo-Aryan and Dravidian Languages*, pages 44–49, Abu Dhabi. Association for Computational Linguistics.
- Google DeepMind. 2026. [Gemma 4: Byte for byte, the most capable open models](#).

- Kyle Gorman, Lucas F.E. Ashby, Aaron Goyzueta, Arya McCarthy, Shijie Wu, and Daniel You. 2020. [The SIGMORPHON 2020 shared task on multilingual grapheme-to-phoneme conversion](#). In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 40–50, Online. Association for Computational Linguistics.
- Thamme Gowda and Jonathan May. 2020. [Finding the optimal vocabulary size for neural machine translation](#). In *Findings of the Association for Computational Linguistics: EMNLP 2020*. Association for Computational Linguistics.
- Naman Goyal, Cynthia Gao, Vishrav Chaudhary, Peng-Jen Chen, Guillaume Wenzek, Da Ju, Sanjana Krishnan, Marc’Aurelio Ranzato, Francisco Guzmán, and Angela Fan. 2022. [The Flores-101 evaluation benchmark for low-resource and multilingual machine translation](#). *Transactions of the Association for Computational Linguistics*, 10:522–538.
- Johannes Graën, Mara Bertamini, and Martin Volk. 2018. [Cutter - a universal multilingual tokenizer](#). In *Swiss Text Analytics Conference*.
- Simran Khanuja, Diksha Bansal, Sarvesh Mehtani, Savya Khosla, Atreyee Dey, Balaji Gopalan, Dilip Kumar Margam, Pooja Aggarwal, Rajiv Teja Nagipogu, Shachi Dave, Shruti Gupta, Subhash Chandra Bose Gali, Vish Subramanian, and Partha Talukdar. 2021. [Muril: Multilingual representations for indian languages](#). *Preprint*, arXiv:2103.10730.
- Jordan Kodner, Salam Khalifa, Khuyagbaatar Batsuren, Hossep Dolatian, Ryan Cotterell, Faruk Akkus, Antonios Anastasopoulos, Taras Andrushko, Aryaman Arora, Nona Atanalov, Gábor Bella, Elena Budianskaya, Yustinus Ghanggo Ate, Omer Goldman, David Guriel, Simon Guriel, Silvia Guriel-Agiashvili, Witold Kieraś, Andrew Krizhanovsky, and 11 others. 2022. [SIGMORPHON–UniMorph 2022 shared task 0: Generalization and typologically diverse morphological inflection](#). In *Proceedings of the 19th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 176–203, Seattle, Washington. Association for Computational Linguistics.
- Taku Kudo. 2018. [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, Melbourne, Australia. Association for Computational Linguistics.
- Jackson L. Lee, Lucas F.E. Ashby, M. Elizabeth Garza, Yeonju Lee-Sikka, Sean Miller, Alan Wong, Arya D. McCarthy, and Kyle Gorman. 2020. [Massively multilingual pronunciation modeling with WikiPron](#). In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4223–4228, Marseille, France. European Language Resources Association.
- Arnav Mhaske, Harshit Kedia, Sumanth Doddapaneni, Mitesh M. Khapra, Pratyush Kumar, Rudra Murthy, and Anoop Kunchukuttan. 2023. [Naamapadam: A large-scale named entity annotated data for Indic languages](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10441–10456, Toronto, Canada. Association for Computational Linguistics.
- OpenAI. 2025. [gpt-oss-120b gpt-oss-20b model card](#). *Preprint*, arXiv:2508.10925.
- Priyaranjan Pattnayak, Hitesh Patel, and Amit Agarwal. 2025. [Tokenization matters: Improving zero-shot ner for indic languages](#). In *2025 IEEE International Conference on Electro Information Technology (eIT)*, pages 456–462.
- Aleksandar Petrov, Emanuele La Malfa, Philip Torr, and Adel Bibi. 2023. [Language model tokenizers introduce unfairness between languages](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 36963–36990. Curran Associates, Inc.
- Roshini Priya R, M Subika, Venkatapathy Subramanian, R Manimegalai, and 1 others. 2025. [Agathiyam-sandhi-aware tokenization for tamil language](#).
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. [Exploring the limits of transfer learning with a unified text-to-text transformer](#). *Journal of Machine Learning Research*, 21(140):1–67.
- Vinodh Rajan. 2018. [Aksharamukha](#). Accessed: 2026-04-15.
- Brian Roark, Lawrence Wolf-Sonkin, Christo Kirov, Sabrina J. Mielke, Cibu Johny, Isin Demirsahin, and Keith Hall. 2020. [Processing South Asian languages written in the Latin script: the Dakshina dataset](#). In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 2413–2423, Marseille, France. European Language Resources Association.
- 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.
- Sarvam AI. 2024. [Sarvam 1: The first indian language llm](#).
- Sarvam AI. 2025. [Sarvam-m: Explorations in post training and inferencing optimizations for a hybrid indic llm](#).
- 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.
- Noam Shazeer and Mitchell Stern. 2018. [Adafactor: Adaptive learning rates with sublinear memory cost](#). In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4596–4604. PMLR.
- Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, Dipanjan Das, and Jason Wei. 2023. [Language models are multi-lingual chain-of-thought reasoners](#). In *The Eleventh International Conference on Learning Representations*.
- Chenglei Si, Zhengyan Zhang, Yingfa Chen, Fanchao Qi, Xiaozhi Wang, Zhiyuan Liu, Yasheng Wang, Qun Liu, and Maosong Sun. 2023. [Sub-character tokenization for Chinese pretrained language models](#). *Transactions of the Association for Computational Linguistics*, 11:469–487.
- Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 197 others. 2025. [Gemma 3 technical report](#). *Preprint*, arXiv:2503.19786.
- Aamod Thakur, Ajay Nagpal, Atharva Savarkar, Kundeshwar Pundalik, Siddhesh Dosi, Piyush Sawarkar, Viraj Thakur, Rohit Saluja, Maunendra Sankar Desarkar, and Ganesh Ramakrishnan. 2025. [The art of breaking words: Rethinking multilingual tokenizer design](#). *Preprint*, arXiv:2508.06533.
- Menan Velayuthan and Kengatharaiyer Sarveswaran. 2025. [Egalitarian language representation in language models: It all begins with tokenizers](#). In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 5987–5996, Abu Dhabi, UAE. Association for Computational Linguistics.
- Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, and 12 others. 2016. [Google’s neural machine translation system: Bridging the gap between human and machine translation](#). *Preprint*, arXiv:1609.08144.
- Jinbiao Yang. 2024. [Rethinking tokenization: Crafting better tokenizers for large language models](#). *Preprint*, arXiv:2403.00417.
- Xiang Zhang, Senyu Li, Bradley Hauer, Ning Shi, and Grzegorz Kondrak. 2023. [Don’t trust ChatGPT when your question is not in English: A study of multilingual abilities and types of LLMs](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7915–7927, Singapore. Association for Computational Linguistics.
- Wenhao Zhu, Hongyi Liu, Qingxiu Dong, Jingjing Xu, Shujian Huang, Lingpeng Kong, Jiajun Chen, and Lei Li. 2024. [Multilingual machine translation with large language models: Empirical results and analysis](#). In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2765–2781, Mexico City, Mexico. Association for Computational Linguistics.
- 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 Appendix

Tokenizer	hi	bn	mr	ta	te	kn	ml	pa	gu	or	as	Avg
GraphemeCluster	2.4007	3.2529	3.2943	4.7648	3.6563	4.1374	4.9699	2.7866	2.9255	3.3005	3.4036	3.4203
GPE (K_{all}, M_{32k})	1.7340	2.2084	2.3569	3.0710	2.8482	3.1842	3.5482	<u>1.8746</u>	2.2374	2.3329	2.3343	2.4344
GPE (K_{32k}, M_{32k})	<u>1.7409</u>	<u>2.2099</u>	<u>2.3624</u>	<u>3.0751</u>	<u>2.8506</u>	<u>3.1973</u>	<u>3.5531</u>	1.8728	<u>2.2449</u>	<u>2.3391</u>	<u>2.3358</u>	<u>2.4389</u>
GPE (K_{2k}, M_{32k})	1.8804	2.3520	2.5165	3.1578	3.0935	3.4356	3.7520	1.9821	2.4427	2.5319	2.4752	2.6021
GPE (K_{all}, M_{2k})	2.2899	3.0750	3.1240	4.3649	3.6162	4.0286	4.8189	2.5667	2.8261	3.1829	3.2441	3.2614
GPE-C (K_{500}, M_{500})	3.6439	4.9791	5.0903	6.6972	6.5040	7.4829	7.9468	4.0320	4.9090	5.4953	5.1487	5.4196
GPE-C (K_{1k}, M_{1k})	3.1648	4.2590	4.4194	5.7214	5.5974	6.2206	6.6841	3.5238	4.2753	4.9167	4.3826	4.6623
GPE-C (K_{2k}, M_{2k})	2.7855	3.6840	3.7693	4.8921	4.7141	5.2288	5.8692	2.9735	3.6668	4.0182	3.8008	3.9814
GPE-C (K_{5k}, M_{5k})	2.3921	3.0766	3.2697	4.1920	3.8813	4.2730	4.7514	2.4841	3.0762	3.2992	3.1992	3.3287

Table 12: Fertility across languages ($\downarrow$), Avg is the corpus-pooled average fertility. Best values are shown in **bold**; second-best values are underlined.

Tokenizer	hi	bn	mr	ta	te	kn	ml	pa	gu	or	as	Avg
GraphemeCluster	0.6329	0.6070	0.6452	0.5895	0.6736	0.6404	0.6418	0.6564	0.6444	0.6407	0.6097	0.6347
GPE (K_{all}, M_{32k})	0.6600	0.6982	0.7011	0.6807	0.7059	0.6886	0.7005	0.7287	0.7170	0.7053	0.6958	0.6983
GPE (K_{32k}, M_{32k})	0.6619	0.6987	0.7018	0.6793	0.7076	0.6882	0.6993	0.7286	0.7189	0.7063	0.6968	0.6989
GPE (K_{2k}, M_{32k})	0.7033	0.7408	0.7408	<u>0.7546</u>	0.7522	0.7550	0.7619	0.7518	<u>0.7760</u>	<u>0.7505</u>	0.7330	0.7473
GPE (K_{all}, M_{2k})	0.6614	0.6413	0.6779	0.6613	0.6661	0.6537	0.6516	0.6857	0.6665	0.6520	0.6327	0.6591
GPE-C (K_{500}, M_{500})	0.7321	0.7239	0.7461	0.6950	0.6498	0.6613	0.7039	0.7064	0.7066	0.6957	0.7146	0.7032
GPE-C (K_{1k}, M_{1k})	0.7932	<u>0.7606</u>	<u>0.7777</u>	0.7491	0.7028	0.7293	0.7595	<u>0.7549</u>	0.7609	0.7307	<u>0.7479</u>	<u>0.7515</u>
GPE-C (K_{2k}, M_{2k})	<u>0.7874</u>	0.7731	0.7925	0.7728	0.7447	0.7746	0.7847	0.7726	0.7955	0.7737	0.7501	0.7747
GPE-C (K_{5k}, M_{5k})	0.7444	0.7358	0.7482	0.7531	<u>0.7462</u>	<u>0.7550</u>	<u>0.7682</u>	0.7536	0.7719	0.7488	0.7235	0.7499

Table 13: Rényi efficiency (with $\alpha = 2.5$) across languages ($\uparrow$), Avg is the mean over the eleven languages. Best values are shown in **bold**; second-best values are underlined.

Tokenizer	hi	bn	mr	ta	te	kn	ml	pa	gu	or	as	Avg
GraphemeCluster	0.4868	0.4929	0.5161	0.4847	0.5384	0.4910	0.5078	0.5900	0.4989	0.5108	0.4864	0.5095
GPE (K_{all}, M_{32k})	0.5828	0.6152	0.6110	0.6274	0.5908	0.5796	0.6246	0.6749	0.6075	0.6228	0.6095	0.6133
GPE (K_{32k}, M_{32k})	0.5859	0.6158	0.6119	0.6313	0.5953	0.5831	0.6273	0.6754	0.6107	0.6242	0.6127	0.6158
GPE (K_{2k}, M_{32k})	0.6528	0.6842	0.6839	0.7160	0.6918	0.7102	0.7180	0.7080	0.7255	0.6979	0.6680	0.6960
GPE (K_{all}, M_{2k})	0.5186	0.5346	0.5576	0.5659	0.5194	0.5002	0.5267	0.6243	0.5238	0.5306	0.5210	0.5384
GPE-C (K_{500}, M_{500})	<u>0.8040</u>	0.7875	0.8153	0.7513	0.6832	0.7471	0.7647	0.7637	0.7697	0.7394	0.7608	0.7624
GPE-C (K_{1k}, M_{1k})	0.8253	<u>0.7821</u>	<u>0.8086</u>	0.8020	0.7294	0.8040	0.7915	0.7972	0.8286	<u>0.7853</u>	0.7815	0.7941
GPE-C (K_{2k}, M_{2k})	0.7935	0.7772	0.7939	<u>0.7980</u>	0.7430	<u>0.7962</u>	0.8095	0.8007	<u>0.8178</u>	0.7909	<u>0.7653</u>	<u>0.7896</u>
GPE-C (K_{5k}, M_{5k})	0.7040	0.7039	0.7097	0.7415	0.7063	0.7286	0.7479	0.7385	0.7600	0.7214	0.6902	0.7229

Table 14: Percentile-frequency scores across languages ($\uparrow$), Avg is the mean over the eleven languages. Best values are shown in **bold**; second-best values are underlined.