Comparative Analysis of the Intrinsic Metrics for Tokenizers and their effect on Downstream Tasks for Hindi and Marathi

Shagun Dwivedi[1], **Kaushik Gopalan**[1,2]

[1]Centre for Interdisciplinary AI, FLAME University, India
[2]School of Computing and Data Sciences, FLAME University, India
Correspondence: {shagun.dwivedi, kaushik.gopalan}@flame.edu.in

Abstract

Various studies have pointed out that the performance of language models is poor in non-English or non-European languages. One of the factors affecting this performance is the effectiveness and suitability of the tokenization scheme used in the model. Indic scripts require multiple Unicode codepoints to represent a single visual unit to be encoded in the standard UTF-8 scheme. This paper investigates the effect of multiple tokenizers that use UTF-8 text input on the downstream performance of pretrained language models for Hindi and Marathi, languages written in *Devanāgari* script. We present the intrinsic performance of the tokenizers using Fertility, Rényi Efficiency and Percentile Frequency, and report the extrinsic performance of monolingual and multilingual models on question-answering tasks, using an automated parts-of-speech and sentence similarity based evaluation framework, and on word-level tasks such as grapheme-to-phoneme conversion and transliteration. We propose a grapheme cluster tokenizer for the script which shows performance better than or competitive with other popular tokenizers. We also find that the Rényi Efficiency metric is highly correlated to downstream performance on question answering.

1 Introduction

Large transformer-based language models have advanced their linguistic skills significantly in languages such as English, French or Spanish, which has led to their widespread usage across the globe. ChatGPT has an active user base of 900 million, with more than 2.5 billion messages being sent everyday, which shows its popularity as the go-to AI tool for assistance for a wide range of tasks. But numerous studies (Maity et al., 2024; Toraman et al., 2023; Zhang et al., 2023) outline how the performance of multilingual language models, such as ChatGPT, lag behind in languages such as Chinese, Turkish, Hindi, Marathi, and Swahili, especially in understanding the nuanced aspects of each language, often evident even in short-length sentences. Zhang et al. (2023) show that GPT model exhibited a substantial degree of subordinate multilingualism—many of its responses seem to result from translating the input into English to formulate a response, which is then translated back into the original language—due to which the resulting accuracy was lower for non-English inputs in comparison to English inputs. They hypothesise that this is due to GPT having developed a representation of knowledge and communication strongly biased towards English.

Studies by Zhu et al. (2024), Zhang et al. (2023), and Shi et al. (2023) point out disparities in the performance of LLMs in languages dissimilar to English which can undermine the potential of the utility of LLMs for the majority of the non-English speaking population. Pretrained language models (PLMs) with their ability to generalise show immense promise in furthering the development in multilingual language models, but there remains a disbalance in the performance of such PLMs in different languages (Blasi et al., 2022; Ahuja et al., 2023).

One way to counter this imbalance, which is usually overlooked in work focusing on improvement of multilingual capabilities of language models, is through more inclusive word embedding representation techniques, or tokenization. Apart from necessary configurations such as model architecture and training datasets, a crucial step for building such language models is tokenization, the process of breaking down and mapping raw input text to subword token representations based on an algorithm. Many of the popular tokenization algorithms are designed for Unicode encoded text, which lead to good performance for languages that use the Roman script, where each character is mapped to one Unicode codepoint. However, such algorithms lead to poor representation of languages like Hindi and Marathi, where multiple codepoints might be needed to en-

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 22652–22663

code one visual character unit.

Hindi and Marathi are Indic languages that are usually written in *Devanāgari*, an abugida or an alphasyllabary writing system which contain consonant-vowel pairs making up one visual unit with a diacritic form of the vowel (*mātrā*) attached to the consonant, and a large number of conjunct characters where two or more consonants, joined using *halantas* (diacritic used to suppress the inherent vowel that otherwise occurs with every consonant letter), and a *mātrā* merges together to form one visual unit; over two hundred conjunct units exist in Hindi itself.

These units are called grapheme clusters, and while one Latin character is represented in Unicode with a single codepoint, common Indic characters may require upto 10 codepoints to represent one visual character. This means that during the tokenization of Devanagari characters using Unicode character based tokenization schemes, one or more Unicode codepoints from the same visual unit may get fragmented into multiple tokens. This undermines the compressional ability of the tokenizers, and plausibly could lead to poor learning by the language model.

In this paper, we compare the performance of five existing tokenizers that use UTF-8 inputs, and we study how, *ceteris paribus*, different tokenization schemes affect the ability of language models for question-answering, transliteration, grapheme-to-phoneme conversion, and their robustness to noise. We also propose a novel grapheme cluster tokenizer, a form of visual character unit level tokenizer for *Devanāgari*. We assess whether the performance of tokenizers on intrinsic evaluation metrics translates to the downstream performance of models trained using those tokenizers.

2 Related Work

A wide array of literature explores strategies to fully utilise the built-in capabilities of LLMs to improve its multilingual performance. Efforts have been made to mitigate the disbalance in multilingual performance through release of more multilingually representative datasets (de Gibert et al., 2024; Conneau et al., 2018; Kumar et al., 2022), monolingual or multilingual models (NLLB Team et al., 2022, Yang et al., 2023, Huang et al., 2024, Workshop et al., 2022, Dabre et al., 2022), multilingual instruction tuning (Lai et al., 2024), continued training with monolingual data (Cui et al., 2024; Yang et al., 2023) as well as multilingual metrics for evaluation of multilingual language models (Guo et al., 2025), however, there is still a long way to go before language models achieve parity in different languages.

Tokenizers and their impact on the downstream performance of LLMs has been the subject of extensive research. Gastaldi et al. (2025) have described and covered the evolution of tokenizers as a crucial step in natural language processing. The most widely used tokenization algorithms for training language models are Byte Pair Encoding (Sennrich et al., 2016), Unigram (Kudo, 2018) and WordPiece (Wu et al., 2016).

Extrinsic evaluation of tokenizers is computationally expensive, and it also suffers from a dearth of datasets when it comes to low-resource languages. However, there have been several studies that quantify the impact of the tokenizer used on the model performance in downstream tasks. Gowda and May (2020) have studied how the vocabulary length of a tokenizer affects the downstream performance of a language model. Rust et al. (2021) showed the impact of replacing the original multilingual tokenizer with the specialised monolingual tokenizer on the downstream performance of multilingual models. Bostrom and Durrett (2020) have compared masked language models pretrained with BPE and Unigram tokenizers, suggesting the use of Unigram over BPE for language model training.

The recent years have seen development of tokenizers designed for non-English languages. Graën et al. (2018) presented Cutter, a framework for tokenization that makes use of language-specific and language independent token identification rules. There is scope to study how this method might work with Indic languages. A major proportion of research in the field is focused on languages that use Roman script, such as English, or languages without the nuances specific to Indic languages, such as the logo-syllabic Chinese or Japanese (Yang, 2024; Si et al., 2023; Fujii et al., 2023).

In the context of Indic languages, Husain et al. (2024) introduced RomanSetu where they study the impact of two transliteration methods, ITRANS and IndicXlit, on various NLU, NLG, and MT tasks in multiple Indic languages. Brahma et al. (2026) introduce Constrained BPE, an extension to the traditional BPE algorithm that incorporates script-specific constraints which facilitate morphology-aware segmentation, incorporating sandhi splitting to enhance the subword tokenization for Hindi and

Marathi. Velayuthan and Sarveswaran (2025) propose Grapheme Pair encoding, a modification to Byte Pair Encoding using grapheme based character extraction as the pre-tokenization method. However, they only analyse compression and parity of Grapheme pair Encoding, and not the downstream performance of an LLM. Goel and Sadat (2025) study the performance of tokenizers on downstream performance on sentiment analysis and named entity recognition tasks for Hindi using the subword tokenizers BPE, Unigram, WordPiece and SentencePiece BPE, showing poor performance of the BPE tokenizer. In their opinion paper, Gorman and Pinter (2025) point out the adverse effects of inconsistent encoding of diacritized characters on NLP systems, using the example of dependency parsing task in Hindi.

Though not used for language models, Ghimire et al. (2023) proposed a pronunciation aware syllable tokenizer for the Automatic Speech Recognition in Nepali language. There have been attempts at tokenization at a syllable level in other languages too (Shikali et al., 2019; Atuhurra et al., 2024). We use a similar method for tokenization of *Devanāgari*, referred to as grapheme cluster tokenizer. Chincholikar et al. (2025) use Grapheme Error Rate as an analogue of the Unicode Character Error Rate to provide a more faithful evaluation of errors in Handwritten Text Recognition of Sanskrit manuscripts at the level of visual units.

There has been some development in the intrinsic evaluation metrics for tokenizers, Zouhar et al. (2023) proposed to define the efficiency of a tokenizer in terms of Rényi entropy and showed its high correlation with BLEU scores for machine translation. They also show high absolute Pearson correlation of machine translation performance with the sum of token frequencies from the 3rd to 83rd percentile, referred to as Percentile Frequency. Fertility, defined as the average number of tokens that are required to represent a piece of text, is a common metric to evaluate a tokenizer's performance (Ali et al., 2024; Rust et al., 2021). Petrov et al. (2024) introduced the notion of parity which assesses how fairly a tokenizer treats equivalent sentences in different languages.

Evaluating language generation tasks poses a significant challenge since string based answer assessment metrics don't align well with human judgement. Efforts in this domain are limited, especially for low resource languages (Si et al., 2021; Bulian et al., 2022; Li et al., 2024; Fabbri et al., 2021). However, Ferreira et al. (2016) propose a sentence similarity assessment metric which is calculated using lexical, semantic, and syntactic similarity components of a sentence. We use a mixture of this similarity metric and parts-of-speech tags based evaluation of answers based on seven major question types for evaluating the question answering tasks.

3 Methodology

We assess the performance of six different tokenizers: Byte Pair Encoding (BPE), Unigram, WordPiece, grapheme cluster tokenizer, Grapheme Pair Encoding(GPE), and Byte Pair Encoding with transliteration during the pretokenization step (ITR+BPE)[1]. The BPE tokenizer employs whitespace splitting at pretokenization, decomposes the words into Unicode characters to form the base vocabulary, and then iteratively merges the most frequent adjacent character pairs to create the vocabulary. The Unigram tokenizer also employs whitespace-only pretokenization; it starts with a large vocabulary of Unicode character sequences and then iteratively prunes tokens that least affect corpus likelihood to form the final vocabulary. The WordPiece tokenizer splits on both whitespace and punctuation during pretokenization, decomposes the words into characters, which are merged to form tokens based on likelihood scores.

All the tokenizers were trained using the Wikipedia articles dataset containing ≈138 thousand articles scraped from the Hindi Wikipedia webpages, ≈60 thousand articles from the Marathi Wikipedia webpages and the contexts from SQuAD (Rajpurkar et al., 2016) for English. To ensure comparability, we first trained the monolingual and multilingual grapheme cluster tokenizers, and then used the resulting vocabulary sizes to fix the vocabulary sizes while training rest of the tokenizers. The design of the grapheme cluster tokenizer, which segments text into *Devanāgari* visual units, is explained in the following paragraph.

A grapheme cluster is a visually identifiable unit, which might consist of a consonant or a conjunct character, usually carrying an attached vowel in a diacritic form. Consider the word वाक्य (*vākya*, "sentence"), in Unicode, the word consists of 5 characters, व, ◌ा, क, ◌्, and य, but visually the word is composed of two characters, namely वा (*vā*) and

[1]The source code and dataset is available at https://github.com/flame-cai/tokenizer-comparative-analysis

Algorithm 1 Grapheme Cluster Tokenization

Input: Devanāgari string S
Output: A list of grapheme clusters *graphemes*

$combining\_marks \leftarrow$ { ◌ु, ◌ॅ, ◌ं, ◌ः, ◌ा, ◌ि, ◌ी, ◌ु, ◌ू, ◌ृ, ◌ॄ, ◌ॆ, ◌े, ◌ै, ◌ॊ, ◌ो, ◌ौ, ◌ँ, ◌ॅ, ◌ॉ}
$graphemes \leftarrow [\,]$
$temp \leftarrow$ ""
for all $char$ in S **do**
  if $char \notin combining\_marks$ **then**
    if $temp \neq$ "" **and** $temp$ does not end with ◌् **then**
      $graphemes.append(temp)$
      $temp \leftarrow char$
    else
      $temp \leftarrow temp + char$
  else
    $temp \leftarrow temp + char$
if $temp \neq$ "" **then**
  $graphemes.append(temp)$
return $graphemes$

Figure 1: Algorithm for segmenting Hindi text into grapheme clusters. A similar method can also be used for other Indic languages.

क्य (*kya*). When such a word is tokenized using a tokenizer such as BPE, the resulting tokens most often separate the attached vowel from the consonant; the tokenizer for GPT4 tokenized the aforementioned word as: व, ◌ाक, and ◌्य.

This tokenization not only results in tokens that are incoherent to humans, it also demonstrates a lower compressional ability or higher fertility for the language. On the other hand, the grapheme cluster tokenizer breaks down text into visual character units, using the algorithm as defined in Fig. 1. This tokenization method considers a consonant or a conjunct consonant with one or more vowels attached to it as one single token. Each unique grapheme cluster thus segmented from the training data is added as a token in the tokenizer's vocabulary. The vocabulary size of the tokenizer is the total number of unique grapheme clusters added to the vocabulary. The vocabulary size of the monolingual tokenizer thus built using the Wikipedia dataset resulted in total of ≈12,850 tokens for the Hindi tokenizer, ≈10500 tokens for the Marathi tokenizer, while the multilingual counterpart had a vocabulary size of ≈16,800 tokens. For the grapheme-to-phoneme (G2P) task, we train the tokenizers with additional Hindi and Marathi Wikipedia texts transliterated to IPA using Aksharamukha (Rajan, 2018); this tokenizer had a vocabulary size of ≈17,400 tokens.

For out-of-vocabulary grapheme clusters, the tokenizer finds the longest substring that exists in the vocabulary in a left-to-right scan of the text. Hence, it emits an <unk> token only for substrings with no corresponding token in the vocabulary, and not the entire grapheme cluster.

Further, to test how pre-tokenization might impact a model's downstream performance, we add the step of transliteration to ITRANS, before tokenization using BPE algorithm as described above. For our study, we chose ITRANS over other transliteration schemes such as IAST, because of its lossless reconstruction into ASCII, as it does not employ any special characters. For brevity, this tokenization method has been referred to as the ITR+BPE tokenization henceforth. Finally, we also study the grapheme pair encoding tokenizer, as per the implementation described by Velayuthan and Sarveswaran (2025).

3.1 Evaluation

The tokenizers are assessed using information-theoretic intrinsic evaluation metrics, as well as through the extrinsic evaluation of the performance of a language model pre-trained using different tokenizers. The investigation of correlation between the intrinsic and extrinsic metrics is also reported.

3.1.1 Intrinsic Evaluation

For the purpose of intrinsic evaluation, three metrics, namely Fertility, Rényi Frequency, and Percentile Frequency were used. Fertility refers to the average number of subword tokens produced per tokenized word. The minimum value for fertility can be 1. It denotes that each word in a document is a token. Rust et al. (2021) show that a high fertility leads to degradation of multilingual performance of a language model. However, the lowest possible fertility of 1 is also not desirable for a tokenizer due to vocabulary scalability issues in morphologically rich languages, as it may cause poor handling of rare words and morphological variants. Hence, an ideal tokenizer should strike a balance between a low fertility score and a small vocabulary size.

Rényi efficiency, an index of diversity, penalizes distributions with either very high or very low-frequency tokens. It is calculated using the order $\alpha = 2.5$, as suggested by Zouhar et al. (2023). The order allows one to penalise the use of long codes to varying degrees. Percentile frequency is defined (Zouhar et al., 2023) as the sum of token frequencies from the 3rd to 83rd percentile. Higher values of Rényi efficiency and percentile frequency are preferable for a tokenization method. These metrics are calculated on the Hindi, Marathi, and English sub-

sets of Meta's FLORES-101 evaluation benchmark dataset (Goyal et al., 2022)(CC BY-SA 4.0 license), comprising 3001 sentences which were not used for building the vocabulary of the tokenizers.

3.1.2 Extrinsic Evaluation

As the language models are ultimately employed for downstream tasks, we conduct extrinsic evaluation to determine whether the tokenizers' performance on intrinsic metrics translate to the extrinsic measures as well. We evaluate their performance on question answering tasks in monolingual and multilingual setups for Hindi, Marathi, and English. Next, we assess the robustness of the tokenizers in dealing with noise corruption in the input of the question answering task with two synthetic noising schemes outlined by Xue et al. (2022): (*i*) *drop* (each unicode codepoint has a 2.5% chance of being dropped) and (*ii*) *repetitions* (each unicode codepoint has 5% chance of being repeated 1-3 times with equal probability).

We compare the performance of grapheme cluster tokenizer with the subword level tokenizers on word-level tasks related to pronunciation and spelling of text; for this we evaluate their performance on two benchmarks: (*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 this extrinsic evaluation, we pretrained the encoder-decoder T5 model (Raffel et al., 2020) using the Huggingface framework. The question answering model consists of 6 layers (4 layers for word-level tasks), the feed forward network in each layer has an output dimensionality of d_{ff} = 2048 followed by ReLU nonlinearity. The key and value matrices have a dimensionality of d_{kv} = 32, and all attention mechanisms have 12 heads (8 heads for word-level tasks). All other sub-layers and embeddings have the dimensionality of d_{model} = 512, and for regularization, the dropout rate of 0.1 is applied in the model. The model was trained using the AdaFactor optimiser with adaptive learning rate and a weight decay of 0.01. While training, the input length was set to 512 (128 for word level tasks), the maximum target length was 50 (128 for word level tasks), whereas the maximum generation length at inference is allowed to be 64 tokens. The models were trained for 12 epochs (15 epochs for word-level tasks), and the weights from the best model are used for the evaluation. The models are directly trained on the respective tasks.

For multilingual G2P conversion, we extract the G2P mappings from the WikiPron dictionary(Lee et al., 2020), which is released under an Apache 2.0 license, for Hindi (≈31,000 pairs), Marathi (4,126 pairs), and English ((≈53,000 pairs). For Hindi, the development and test set shared in SIGMORPHON 2020 Task 1 are removed from the training, and used to present the final results. For Marathi and English, 450 grapheme-phoneme pairs are stratified sampled for the development set and randomly sampled for the test set. The training set has 58,034 samples—30,000 Hindi pairs, 3,034 Marathi pairs, and 25,000 English pairs (stratified sampled from extracted mappings). For transliteration, we use the Dakshina benchmark(Roark et al., 2020), with 50,000 samples in train, 5,000 in development and 2,500 in test set (licensed under CC BY-SA 4.0).

We use an augmented version of the Wikipedia dataset which was used to train the tokenizers, removing any multilingual noise, incomplete sentences, and truncating the article length to maximum of ten sentences, to create a question-answering dataset in the SQuAD format (Rajpurkar et al., 2016) using OpenAI's GPT4o-mini (similar to the synthetic data creation workflow of Singh et al. (2025)); the consequent dataset contains 50 thousand contexts for Hindi and 70 thousand contexts for Marathi, and one to three question-answer pairs for each context based on the length of the articles. The articles cover a wide range of topics, including general knowledge, science, culture, and health available on Wikipedia until the 2019 dataset release. For English, we used the SQuAD v1.1 dataset(Rajpurkar et al., 2016). During inference, we evaluate the models using benchmark datasets for Hindi, Marathi, and English, with the Hindi subset of XQuAD (Artetxe et al., 2019)(CC BY-SA 4.0 license), MahaSQuAD (Ghatage et al., 2023)(CC BY-NC-SA 4.0 license), and the English subset of TyDiQA-GoldP(Clark et al., 2020)(Apache License 2.0).

Reference based evaluation metrics, such as ROUGE-L recall or token based F1 score, which rely on string matching, can not be directly used for evaluating answers to different types of questions. Different questions may demand distinct answer formats or may have multiple correct answers. Hence, we assess the performance of the models by evaluating each type of question through different natural language processing techniques. We first classify the questions into 7 types—'Which/Who', 'When', 'Where', 'How much', 'What', 'Why', and 'How'.

We assess the agreement of this framework with human evaluation for a sample of 625-700 questions (≈100 questions for each question type) for each of the three languages. Two human evaluators, who were fluent in the respective languages, evaluated each answer from different models in each language. They were instructed to mark the predicted answers correct, incorrect, or invalid based on the context and question; out of the 700 questions, the ambiguous or unanswerable questions given the context were labelled as invalid and were not considered for the assessment. The inter-evaluator agreement was very high; the Cohen's kappa was 0.96 for Hindi, 0.95 for Marathi, and 0.97 for English. We then evaluated the accuracy of evaluation framework using the human judgments, for questions where both evaluators agreed, as the true evaluation. The automated framework achieved 84.89% accuracy for Hindi, 88.43% for Marathi, and 91.25% for English. Most of the incorrect evaluations can be classified as Type 2 errors (false negatives). This indicates that the framework is stricter than human evaluator and may underestimate rather than overestimate model performance.

4 Results

Tokenizer	Fertility (↓)	Rényi Eff.(↑)	Percentile Freq.(↑)
Hindi			
GraphemeCluster	**2.432**	0.651	0.479
GraphemePairEncoding	2.921	**0.674**	**0.569**
WordPiece	3.353	0.604	0.492
Unigram	3.478	0.621	0.505
BytePairEncoding	2.890	0.261	0.329
ITR+BPE	3.433	0.294	0.304
Marathi			
GraphemeCluster	**3.269**	0.659	0.499
GraphemePairEncoding	3.408	**0.753**	**0.649**
WordPiece	3.621	0.648	0.621
Unigram	3.775	0.659	0.618
BytePairEncoding	3.642	0.316	0.357
ITR+BPE	4.264	0.350	0.342
English			
GraphemeCluster	-	-	-
GraphemePairEncoding	-	-	-
WordPiece	4.339	**0.759**	**0.684**
Unigram	3.574	0.562	0.480
BytePairEncoding	**3.473**	0.316	0.309
ITR+BPE	-	-	-
Mixed			
GraphemeCluster	3.537	0.579	0.339
GraphemePairEncoding	**3.258**	**0.723**	**0.576**
WordPiece	3.753	0.643	0.517
Unigram	3.595	0.596	0.521
BytePairEncoding	3.293	0.268	0.327
ITR+BPE	3.555	0.293	0.271

Table 1: Intrinsic evaluation for each tokenizer calculated using 3001 sentences of text in each language in the FLORES-101 evaluation benchmark dataset.

In this section, we report the results for the intrinsic and extrinsic evaluation experiments. Table 1 reports the performance of the six tokenization schemes on three intrinsic metrics, fertility, Rényi Efficiency and Percentile Frequency for text in Hindi, Marathi, English, and all three languages mixed. The fertility of grapheme cluster tokenizer is low for Hindi and Marathi, but increases relative to other tokenizers, with addition of English text in the mixed set. This makes sense since the tokenizer just functions as a character level tokenizer. The fertility of BPE is lower than other tokenizers for the mixed dataset, but is middling for each of the language separately. We note here that we do not report the performance of grapheme cluster tokenizer, GPE, and ITR+BPE on English datasets in monolingual setups either for intrinsic metrics or for downstream tasks as they are intended to be used for *Devanāgari* text.

The Rényi efficiency and percentile frequency metrics show a different trend, with BPE having a significantly lower score compared to other tokenization schemes for all the languages, showing a disparity between the distribution of the most frequent and infrequent tokens. Such disparity has been shown to result in inefficient training of language models. The tokenizers designed primarily for *Devanāgari*, GPE and grapheme cluster tokenizer, both perform well on all metrics for Hindi and Marathi; GPE performs well even for the mixed dataset, whereas the intrinsic performance of grapheme cluster tokenizer reduces on mixed data due to the addition of English text.

Further, we present the performance of the models for question answering tasks in Table 2. The results for the datasets with synthetic noise are produced using monolingual models. The grapheme cluster tokenizer and WordPiece demonstrate a competitive performance consistently for both Hindi and Marathi, GPE is more accurate for Marathi but not for Hindi, and BPE and ITR+BPE show the lowest accuracy.

For the noise addition task, the model using the grapheme cluster shows more robustness than other models for three of the four subtasks, but is slightly outperformed by WordPiece in unseen repetition subtask in the Hindi dataset. For the Marathi dataset, it is more robust than other models on three of the four subtasks, but is slightly outperformed by GPE in case of learnable drop subtask. Since the models looks at one grapheme cluster as one token, it can be more robust in finding dropped/repeated unicode codepoints and 'fixing' the sequence of tokens while output generation even when trained on clean data. The poor performance of ITR+BPE

Tokenizer	Monolingual	Multilingual	Unseen Noise		Learnable Noise	
			Drop	Repetitions	Drop	Repetitions
XQuAD (Hindi)	*(n=1184)*					
GraphemeCluster	**41.60**	**52.96**	**33.40**	23.80	**34.40**	**37.00**
GraphemePairEncoding	36.20	47.38	27.62	23.10	29.10	31.80
WordPiece	39.60	48.90	29.39	**25.00**	33.40	28.00
Unigram	34.60	48.90	24.92	21.60	29.90	31.90
BytePairEncoding	10.10	30.74	10.38	7.30	12.60	13.30
ITR+BPE	15.71	15.63	9.68	8.05	11.17	11.85
MahaSQuAD (Marathi)	*(n=486)*					
GraphemeCluster	47.74	51.23	**41.98**	**36.21**	39.30	**41.14**
GraphemePairEncoding	48.35	**53.29**	36.42	35.60	**39.41**	37.24
WordPiece	**50.82**	51.85	33.74	28.81	39.30	40.33
Unigram	44.03	49.59	37.86	33.13	32.18	37.86
BytePairEncoding	16.46	29.42	20.78	11.32	24.49	3.09
ITR+BPE	4.32	2.06	4.94	2.67	3.29	5.56

Table 2: Accuracy(%) of the Hindi and Marathi monolingual and multilingual models trained using different tokenizers on benchmark datasets. Learnable noise is added in training and test set of a dedicated monolingual model, whereas unseen noise only affects the test set of the monolingual model.

may be attributed to increased semantic and phonetic confusion in words or tokens belonging to different languages, or weakening of language identification due to ambiguous or overlapping tokens from different languages.

Tokenizer	Accuracy *(n=440)*
GraphemeCluster	44.54
GraphemePairEncoding	45.00
WordPiece	**46.59**
Unigram	42.04
BytePairEncoding	21.36
ITR+BPE	18.49

Table 3: Accuracy(%) of the multilingual models trained using different tokenizers on TyDiQA.

Table 3 shows the performance of the same multilingual models on TyDiQA. The performance of the grapheme cluster tokenizer does not seem to degrade relative to the other tokenizers for the English benchmark in a multilingual setup. Further, the BPE model also performs poorly on English.

While all errors can not be attributed to the behaviour of the tokenizer, we will discuss some such errors observed during inference. We present some of these in Table 4. There are multiple ways to encode some diacritized Devanagari glyphs, one of them is a *nuqta* (़), which can be added to many characters (ज़, ड़, ढ़, फ़) to represent phonological distinction from the undiacritized letters (ज, ड, ढ, फ). These characters can be represented as both one unicode codepoint (diacritized) and a collection of codepoints (undicritized letter followed by a *nuqta*), corresponding to Unicode normalization forms NFC and NFD respectively. Without normalization to a consistent form, the two representations will fail string equality tests despite being visually and semantically identical. The XQuAD dataset uses both representations for this diacritic (for ढ़ and फ़). Models trained with the BPE tokenizer can answer questions where the glyph is represented as a combination of codepoints (17.64% accuracy on 153 questions), but only gets one question right (2.94% accuracy) from the second form (34 questions). WordPiece (8.82%) and Grapheme Cluster tokenizer (20.58%) show more skill for these questions.

Example 1	
Question:	थॉमस डेविस ने कितने फ़ोर्ड्स फ़म्बल किए थे? ?
Gold Answer:	चार
Prediction:	तीन ोर्ड्स।
Example 2	
Question:	प्रत्येक विस्तारित महानगरीय क्षेत्रों की आबादी में कितनी संख्या तक बढ़ोतरी हुई है?
Gold Answer:	पांच मिलियन
Prediction:	यह संख्या में पांच संख्या तक बोतरी हुई है।
Example 3	
Question:	DECnet का फ़ेज़ 2 क्या बन गया?
Gold Answer:	प्रकाशित विनिर्देशों युक्त ओपन स्टैंडर्ड थे, और कई कार्यों को DEC के बाहर विकसित किया गया था, जिसमें से एक Linux के लिए भी था
Prediction:	DECnet का ़ II (और बाद)।
Example 4	
Question:	चित्रपटगृहात जास्तीत जास्त लोक चित्रपट पाहतील याची खात्री करण्यासाठी काय प्रतिबंधित आहे?
Gold Answer:	ऑनलाइनवर त्वरित प्रवेश
Prediction:	जास्तीत जास्त लोकांना सिनेमाला जाण्याठी प्रोत्साहित झाली आहे.
Example 5	
Question:	आईन्स्टाईन माकच्या तत्त्वाचा पुरस्कार कसा करतात?
Gold Answer:	अंशतः
Prediction:	आईन्स्टाईन माकच्या गाईचा पुरस्कार करतात की दूरच्या कोरी जडत्वाचे स्पष्टीकरण देतात कारण ते गुरु

Table 4: Example of incorrect predictions by multilingual BPE model for questions from XQuAD Hindi and MahaSQuAD.

Further, we find that, out of the 100 most frequently occuring tokens in the Hindi and Marathi QA training set, a substantial portion of the BPE tokens are standalone *mātras*. BPE breaks up words at awkward boundaries, possibly stripping the words of their semantic meaning. We observed that some

of the incorrectly predicted answers do contain invalid combinations of such *mātras*.

Table 5 shows the performance of the multilingual models trained on the word-level tasks. The performance in the G2P task is evaluated using two metrics—word error rate and phoneme error rate. The grapheme cluster shows the highest performance for Hindi and Marathi. BPE performs better for this task than the WordPiece and Unigram tokenizers. For word-level tasks, WordPiece performs poorly for Hindi and Marathi; the likelihood-based merging seems to result in better semantic understanding but the model struggles at capturing orthographic complexity. On the other hand, the same over-fragmentation of BPE tokens that reduce its accuracy on longer text generation/information retrieval tasks might be helping it achieve a mediocre score.

Tokenizer	**G2P**		**Transliteration**
	WER	PER	CER
Hindi	***(n=450)***		***(n=2500)***
GraphemeCluster	**12.67**	**3.46**	**12.99**
GraphemePairEncoding	17.11	5.57	15.92
WordPiece	82.67	132.95	21.14
Unigram	30.44	6.45	21.28
BytePairEncoding	31.33	19.14	21.29
ITR+BPE	24.89	12.97	-
Marathi	***(n=450)***		***(n=2500)***
GraphemeCluster	**32.44**	**8.68**	**9.56**
GraphemePairEncoding	38.00	11.67	11.76
WordPiece	87.56	79.56	21.93
Unigram	43.56	9.84	20.71
BytePairEncoding	42.44	22.54	19.17
ITR+BPE	39.78	16.01	-
English	***(n=450)***		
GraphemeCluster	47.79	13.73	-
GraphemePairEncoding	50.00	17.08	-
WordPiece	46.90	13.76	-
Unigram	**43.14**	**11.66**	-
BytePairEncoding	52.43	15.83	-
ITR+BPE	62.39	22.36	-

Table 5: Results from multilingual word-level tasks.

Across different tasks, we observe different relationships between the intrinsic metrics and the downstream performance. For the various question answering tasks in the experimental setup, Rényi Efficiency is highly correlated to the model accuracy (Pearson correlation coefficient $r = 0.86 - 0.97$); the Percentile Frequency shows a positive, though weaker, correlation ($r = 0.79 - 0.93$); in contrast there is no correlation between fertility and question answering accuracy. For transliteration, fertility is positively correlated to the character error rate ($r = 0.82$ for Hindi, $r = 0.93$ for Marathi), whereas Rényi Efficiency shows mild negative correlation ($r = -0.51$ for Hindi, $r = -0.37$ for Marathi). Finally, for G2P task, none of the intrinsic metrics exhibit significant correlation to the error rates.

5 Conclusion

In this paper, we assessed the performance of various tokenizers on intrinsic and extrinsic metrics. We find that the tokenizers with poor performance on Rényi Efficiency demonstrate poor downstream accuracy on question answering tasks but the metric is poorly correlated with downstream performance on word-level tasks. The fertility and percentile frequency metrics do not show a strong correlation with downstream model performance.

We proposed a grapheme cluster tokenizer for *Devanāgari* text and compared its performance to popular subword tokenizers such as Unigram, and WordPiece, and variations of BPE. It exhibits high Rényi efficiency and fertility, and has competitive or superior performance on extrinsic metrics. The character-level Byte Pair Encoding tokenizer consistently exhibits a poor performance on monolingual and multilingual question answering tasks where the tokenizer is trained on *Devanāgari* text. The Unigram and WordPiece tokenizers demonstrate performance comparable to or better than the grapheme cluster based tokenizers on the question answering tasks, but they lag behind on the word-level tasks, showing poor orthographic understanding.

This analysis can help inform the choice of tokenizers in LLMs, as even popular model families such as GPT and Llama use byte-pair encoding, a tokenization scheme which consistently demonstrates poor downstream performance on tasks involving *Devanāgari* text. The grapheme cluster tokenizer can also be adapted to other languages that employ the abugida writing system, particularly Indic scripts, such as Bengali, Gujarati, and Tamil.

6 Limitations

The study suffers from some limitations which are summarised in this section. While there are innumerable ways of training a tokenizer, such as using different datasets, implementing different parameter settings, or performing different number of merges, this case study only takes into account only a limited subset of the possibilities. Further, while the automated evaluations employ various semantics based techniques, they are not fool-proof; the actual accuracy for the models can be expected to be higher if manually evaluated. Also, the accuracy of the evaluation depends on extraneous factors such as the quality of the POS tagging and dependency

parsing models. However, we prioritise detecting incorrect answers accurately, reducing the Type I error.

Another concern regarding the study stems from the research done by Shumailov et al. (2024) demonstrating that training on samples from another generative model can induce a distribution shift, which over time may cause Model Collapse, where the model may mis-perceive the underlying learning task. To partially counter this, we have used the context based question-answering task, limiting the question-answer generation to be strictly based on content from open source Hindi and Marathi Wikipedia articles from before the rise of artificially generated content on the internet.

Further, the use of Wikipedia articles as the training corpus may introduce potential biases in topic coverage, writing style and domain transferability. However, it enables controlled and fair comparison of the tokenization methods and aligns with standard practices. Evaluating tokenizers on more diverse corpora remains important future work.

7 Acknowledgements

The authors wish to express their sincere thanks to Ajay Dwivedi, Anushka Verma, Anushree Bhuskute, and Paul M Kallarackal for their prompt assistance with the manual evaluation, Kartik Chincholikar for discussing and reviewing the manuscript, and the anonymous reviewers for their constructive feedback.

References

Kabir Ahuja, Harshita Diddee, Rishav Hada, Millicent Ochieng, Krithika Ramesh, Prachi Jain, Akshay Nambi, Tanuja Ganu, Sameer Segal, Mohamed Ahmed, Kalika Bali, and Sunayana Sitaram. 2023. MEGA: Multilingual evaluation of generative AI. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4232–4267, Singapore. Association for Computational Linguistics.

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.

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2019. On the cross-lingual transferability of monolingual representations. *CoRR*, abs/1910.11856.

Jesse Atuhurra, Hiroyuki Shindo, Hidetaka Kamigaito, and Taro Watanabe. 2024. Introducing syllable tokenization for low-resource languages: A case study with swahili. *Preprint*, arXiv:2406.15358.

Damian Blasi, Antonios Anastasopoulos, and Graham Neubig. 2022. Systematic inequalities in language technology performance across the world's languages. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5486–5505, Dublin, Ireland. Association for Computational Linguistics.

Kaj Bostrom and Greg Durrett. 2020. Byte pair encoding is suboptimal for language model pretraining. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4617–4624, Online. Association for Computational Linguistics.

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*.

Jannis Bulian, Christian Buck, Wojciech Gajewski, Benjamin Börschinger, and Tal Schuster. 2022. Tomayto, tomahto. beyond token-level answer equivalence for question answering evaluation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 291–305, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Kartik Chincholikar, Shagun Dwivedi, Kaushik Gopalan, and Tarinee Awasthi. 2025. A case study of handwritten text recognition from pre-colonial era Sanskrit manuscripts. In *Computational Sanskrit and Digital Humanities - World Sanskrit Conference 2025*, pages 52–69, Kathmandu, Nepal. Association for Computational Lingustics.

Jonathan H. Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. 2020. Tydi qa: A benchmark for information-seeking question answering in typologically diverse languages. *Transactions of the Association for Computational Linguistics*.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485, Brussels, Belgium. Association for Computational Linguistics.

Yiming Cui, Ziqing Yang, and Xin Yao. 2024. Efficient and effective text encoding for chinese llama and alpaca. *Preprint*, arXiv:2304.08177.

Raj Dabre, Himani Shrotriya, Anoop Kunchukuttan, Ratish Puduppully, Mitesh Khapra, and Pratyush Kumar. 2022. Indicbart: A pre-trained model for indic natural language generation. In *Findings of the Association for Computational Linguistics: ACL 2022*. Association for Computational Linguistics.

Ona de Gibert, Graeme Nail, Nikolay Arefyev, Marta Bañón, Jelmer van der Linde, Shaoxiong Ji, Jaume Zaragoza-Bernabeu, Mikko Aulamo, Gema Ramírez-Sánchez, Andrey Kutuzov, Sampo Pyysalo, Stephan Oepen, and Jörg Tiedemann. 2024. A new massive multilingual dataset for high-performance language technologies. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 1116–1128, Torino, Italia. ELRA and ICCL.

Alexander R. Fabbri, Wojciech Kryściński, Bryan McCann, Caiming Xiong, Richard Socher, and Dragomir Radev. 2021. Summeval: Re-evaluating summarization evaluation. *Transactions of the Association for Computational Linguistics*, 9:391–409.

Rafael Ferreira, Rafael Dueire Lins, Steven J. Simske, Fred Freitas, and Marcelo Riss. 2016. Assessing sentence similarity through lexical, syntactic and semantic analysis. *Computer Speech & Language*, 39:1–28.

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*.

Ruturaj Ghatage, Aditya Ashutosh Kulkarni, Rajlaxmi Patil, Sharvi Endait, and Raviraj Joshi. 2023. MahaSQuAD: Bridging linguistic divides in Marathi question-answering. In *Proceedings of the 20th International Conference on Natural Language Processing (ICON)*, pages 497–505, Goa University, Goa, India. NLP Association of India (NLPAI).

Rupak Raj Ghimire, Bal Krishna Bal, Balaram Prasain, and Prakash Poudyal. 2023. Pronunciation-aware syllable tokenizer for Nepali automatic speech recognition system. In *Proceedings of the 20th International Conference on Natural Language Processing (ICON)*, pages 36–43, Goa University, Goa, India. NLP Association of India (NLPAI).

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.

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.

Kyle Gorman and Yuval Pinter. 2025. Don't touch my diacritics. 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 285–291, Albuquerque, New Mexico. 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*.

Yanzhu Guo, Simone Conia, Zelin Zhou, Min Li, Saloni Potdar, and Henry Xiao. 2025. Do large language models have an English accent? evaluating and improving the naturalness of multilingual LLMs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3823–3838, Vienna, Austria. Association for Computational Linguistics.

Zixian Huang, Wenhao Zhu, Gong Cheng, Lei Li, and Fei Yuan. 2024. Mindmerger: Efficiently boosting llm reasoning in non-english languages. In *Advances in Neural Information Processing Systems*, volume 37, pages 34161–34187. Curran Associates, Inc.

Jaavid Aktar Husain, Raj Dabre, Aswanth M, Jay Gala, Thanmay Jayakumar, Ratish Puduppully, and Anoop Kunchukuttan. 2024. RomanSetu: Efficiently unlocking multilingual capabilities of large language models via Romanization. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15593–15615, Bangkok, Thailand. 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.

Aman Kumar, Himani Shrotriya, Prachi Sahu, Amogh Mishra, Raj Dabre, Ratish Puduppully, Anoop Kunchukuttan, Mitesh M. Khapra, and Pratyush Kumar. 2022. IndicNLG benchmark: Multilingual datasets for diverse NLG tasks in Indic languages. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5363–5394, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Wen Lai, Mohsen Mesgar, and Alexander Fraser. 2024. LLMs beyond English: Scaling the multilingual capability of LLMs with cross-lingual feedback. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 8186–8213, Bangkok, Thailand. 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.

Zongxia Li, Ishani Mondal, Huy Nghiem, Yijun Liang, and Jordan Lee Boyd-Graber. 2024. PEDANTS: Cheap but effective and interpretable answer equivalence. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 9373–9398, Miami, Florida, USA. Association for Computational Linguistics.

Subhankar Maity, Aniket Deroy, and Sudeshna Sarkar. 2024. How ready are generative pre-trained large language models for explaining bengali grammatical errors? *Preprint*, arXiv:2406.00039.

NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, and 20 others. 2022. No language left behind: Scaling human-centered machine translation. *Preprint*, arXiv:2207.04672.

Aleksandar Petrov, Emanuele La Malfa, Philip H.S. Torr, and Adel Bibi. 2024. Language model tokenizers introduce unfairness between languages. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

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.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

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.

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.

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 multilingual chain-of-thought reasoners. In *The Eleventh International Conference on Learning Representations*.

Casper S. Shikali, Zhou Sijie, Liu Qihe, and Refuoe Mokhosi. 2019. Better word representation vectors using syllabic alphabet: A case study of swahili. *Applied Sciences*, 9(18).

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Yarin Gal, Nicolas Papernot, and Ross Anderson. 2024. The curse of recursion: Training on generated data makes models forget. *Preprint*, arXiv:2305.17493.

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.

Chenglei Si, Chen Zhao, and Jordan Boyd-Graber. 2021. What's in a name? answer equivalence for open-domain question answering. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9623–9629, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Abhishek Kumar Singh, Vishwajeet Kumar, Rudra Murthy, Jaydeep Sen, Ashish Mittal, and Ganesh Ramakrishnan. 2025. INDIC QA BENCHMARK: A multilingual benchmark to evaluate question answering capability of LLMs for Indic languages. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 2607–2626, Albuquerque, New Mexico. Association for Computational Linguistics.

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

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.

BigScience Workshop, Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, and 1 others. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

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.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. ByT5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306.

Jinbiao Yang. 2024. Rethinking tokenization: Crafting better tokenizers for large language models. *Preprint*, arXiv:2403.00417.

Wen Yang, Chong Li, Jiajun Zhang, and Chengqing Zong. 2023. Bigtranslate: Augmenting large language models with multilingual translation capability over 100 languages. *arXiv preprint arXiv:2305.18098*.

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.