What Tokens are Learned when Tokenization is Optimized Jointly with Language Modeling?

Saketh Reddy Vemula
IIIT Hyderabad, India
saketh.vemula@research.iiit.ac.in

Parameswari Krishnamurthy
IIIT Hyderabad, India
param.krishna@iiit.ac.in

Abstract

Tokenization is a fundamental component of language modeling pipelines. Despite its importance, it is often fixed, even though it significantly impacts model performance across languages. In this work, we analyze what tokens are learned when tokenization is jointly optimized with language modeling. We compare *tokenizer-free* approaches such as SSLMs and H-Nets with fixed tokenizers across 18 typologically and script-diverse languages. Our results show that joint optimization fundamentally alters token structure. SSLMs recover morphologically aligned and contextually efficient tokens, whereas H-Nets prioritize byte-level efficiency, producing longer tokens with very low overlap with standard subword vocabularies. We further show that tokenization behavior varies across language typologies. Agglutinative languages exhibit more dynamic segmentation patterns while learning. Through downstream evaluation, with pretrained-then-finetuned BERT models, we find that SSLM-based pretokenization consistently reduces language modeling perplexity and achieves competitive downstream performance despite distinct vocabularies. Overall, *tokenizer-free* approaches optimize for contextual and computational efficiency rather than strict morphological structure, resulting in fundamentally different yet effective vocabularies for downstream NLP.

1 Introduction

Tokenization is a fundamental preprocessing step in language models (LMs). In transformer-based LMs (Vaswani et al., 2023), it is often performed by learning a tokenizer through subword algorithms such as Byte-pair Encoding (BPE, Gage (1994); Sennrich et al. (2016)), Unigram Language Model (ULM, Kudo (2018)), or WordPiece (WPC, Schuster and Nakajima (2012)). These algorithms are built on simple statistical priors, such as the assumption that frequently co-occurring characters

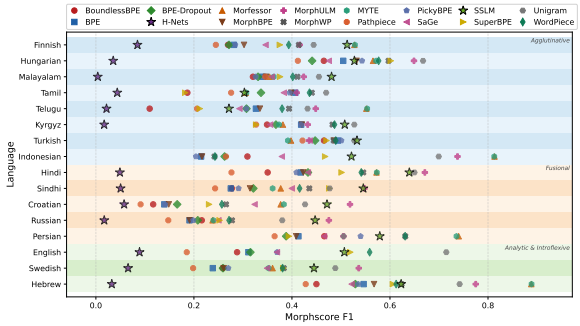


Figure 1: Morphological alignment evaluation of *tokenizer-free* (H-Nets, SSLMs) and fixed-tokenizer approaches. SSLMs show consistently higher alignment, whereas H-Nets trade morphological alignment for longer, computationally efficient tokens.

should be part of the same token (Sennrich et al., 2016). Despite their widespread adoption, it is unclear whether such approaches provide best performance, especially when working with languages across diverse morphological complexities, typologies, and scripts. This concern is further amplified from the standard *tokenization-LM-detokenization* pipeline, where a tokenizer is learned independently of language model training and remains static. We refer to these as *fixed-tokenizer* approaches. Assessing the best tokenizer for such LMs is both difficult and expensive, since the tokenizer remains fixed prior to training and cannot be modified thereafter.

Various modifications and novel algorithms have been proposed to learn a tokenizer. This includes advances in pre-tokenization (Banerjee and Bhattacharyya, 2018; Hu et al., 2025), robustness (Kudo and Richardson, 2018), morphological alignment (Hofmann et al., 2022; Libovický and Helcl, 2024; Zhu et al., 2025), vocabulary pruning (Cognetta et al., 2024) or modifications (Chizhov et al., 2024), and cross-token[1] approaches (Schmidt et al., 2025;

[1]Tokens that can span multiple words, for example "by the way".

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 53028–53048
October 24-29, 2026 ©2026 Association for Computational Linguistics

Liu et al., 2025). However, no single approach can be concluded to be the "best", as conclusions depend on various factors such as: 1) languages chosen and their morphological properties, 2) experimental setup like model architecture, dataset size and other hyperparameters (Zhu et al., 2019; Poelman et al., 2025).

Recently, new architectures have been proposed that overcome the aforementioned *tokenization-LM-detokenization* pipeline. These are often referred to as *tokenizer-free* LMs. Architectures such as Dynamic Token Pooling Transformers (Nawrot et al., 2023) and H-Nets (Hwang et al., 2025) incorporate data-dependent boundary prediction (i.e., tokenization) and byte-level language modeling into a single end-to-end pipeline. H-Nets, in particular, provide a stable architecture. On the other hand, models such as subword segmental language model (SSLM, Meyer and Buys (2022, 2025)) use long short-term memory (LSTMs, Hochreiter and Schmidhuber (1997)) or transformers (Vaswani et al., 2023) to marginalize and optimize over all possible segmentations. Despite being computationally heavy and unfeasible at large scale, SSLMs provide us with a joint architecture which we can utilize for analysis. These *tokenizer-free* approaches avoid any statistical or heuristic priors, thus providing us with a tokenization that is jointly learned with the LM.

In this work, we investigate "*what tokens are learned when tokenization is optimized jointly with language modeling*." Our contributions:

1. We analyze *how* and *what* tokens are learned through *tokenizer-free* approaches, namely SSLM and H-Net, by evaluating their intrinsic and linguistic properties across a diverse set of languages.
2. We compare these approaches with *fixed-tokenizer* methods, such as BPE and ULM, to assess *how similar* their learned vocabularies are.
3. We evaluate the impact of these tokenization approaches on downstream task performance to assess *whether* jointly learned tokenizers offer improvements.

2 Related Work

Prior work analyzing tokens learned by *tokenizer-free* LMs remains limited. Recently, Meyer and Buys (2025) studied the learning dynamics of tokens learned by SSLMs (Meyer and Buys, 2022). They tracked subword learning dynamics from a linguistic perspective by evaluating morphological alignment, productivity, idiosyncrasy, and fertility, and identified distinct stages of subword learning. However, because their study focused on learning dynamics, the analysis was limited to three languages. Moreover, critical confounding factors such as dataset size and disparities across languages were not accounted for when drawing conclusions across languages.

On the other hand, many studies have analyzed fixed tokenizers and their modifications. These studies have largely focused on the impact of subword algorithms on downstream LM performance. For example, Bostrom and Durrett (2020), Vemula et al. (2025) and Brahma et al. (2026) compared different algorithms, showing that ULM performs better on downstream tasks and recovers subwords that are more morphologically aligned. While, works such as Uzan et al. (2024) argued that inference strategy can matter as or more than the tokenizer construction algorithm.

Language Coverage
Agglutinative: Finnish, Hungarian, Malayalam, Tamil, Telugu, Kyrgyz, Turkish, Mongolian, Indonesian (**9 languages**)
Analytic & Introflexive: English, Swedish, Hebrew (**3 languages**)
Fusional: Sanskrit, Hindi, Sindhi, Croatian, Russian, Persian (**6 languages**)

Table 1: List of typologically and script diverse languages included in this study. Detailed version of this table can be found in Table 11 in the Appendix.

3 Methodology

3.1 Languages and Datasets

Languages vary widely in morphology, typology, and script, which may affect the best tokenization approach. We therefore study a broad, representative set of 18 languages spanning this diversity, listed in Table 1. Detailed version can be found in Table 11 in Appendix.

An ideal comparison involving multiple languages requires parallel training and test data. However, it is difficult to find such parallel corpora across many languages. Therefore, we proceed with non-parallel monolingual corpora while carefully controlling the data source and size. To balance the data sizes across languages, we extract 250,000 English sentences and measure their size in bytes. Then, for other languages, we scale the data size

according to *byte-premiums*[2] (BP) (Arnett et al., 2024) and extract the desired number of sentences[3]. We choose WMT News Crawl corpora[4] (Chelba et al., 2013) as the data source, thereby fixing the domain to News for most languages. And if it is not available for a specific language, we fallback to the NLLB corpus (Team et al., 2022). Statistics of the resultant pretraining data along with available morphologically annotated data is listed in Table 12 of the Appendix.

3.2 Tokenization Approaches

We include two distinct *tokenizer-free* approaches from recent literature:

- Transformer-based version of subword segmental language model; **SSLM** (Meyer and Buys, 2025); that marginalize over all possible segmentations while jointly optimizing for language modeling.
- End-to-end **H-Net**s (Hwang et al., 2025) that jointly perform data-dependent boundary prediction along with byte-level language modeling.

To compare these approaches with *fixed-tokenizer* approaches, we include both standard and improved algorithms. We include standard algorithms such as **BPE** (Sennrich et al., 2016), **ULM** (Kudo, 2018), and **WPC** (Schuster and Nakajima, 2012). We also study methods such as **SaGe** (Yehezkel and Pinter, 2023), which prefers subword units that occur in fewer distinct contexts relative to its frequency, and unsupervised morphological tokenizers such as **Morfessor** (Smit et al., 2014). In addition, we also include methods that modify inference (such as **BPE-dropout** (Provilkov et al., 2020) and **PathPiece** (Schmidt et al., 2024)), pretokenization (such as **MorphBPE** (Banerjee and Bhattacharyya, 2018)), encoding (such as **MYTE** (Limisiewicz et al., 2024)) and vocabulary construction (such as **PickyBPE** (Chizhov et al., 2024)). We further include cross-token approaches such as **SuperBPE** (Liu et al., 2025) and **BoundlessBPE** (Schmidt et al., 2025), which have been shown to be more efficient. Table 13 in Appendix describes these approaches and § B.3 lists their hyperparameters.

[2]Byte premium (BP) is the ratio of bytes required to encode a comparable amount of information or parallel text in one language relative to another language (typically designated as English).

[3]This is necessary for our comparative analysis involving UTF-8 encoding, as it is known to produce disparities across languages with diverse scripts (Arnett et al., 2024).

[4]https://data.statmt.org/news-crawl/

3.3 Evaluation

We evaluate both intrinsic properties and extrinsic performance of the mentioned tokenization approaches. We consider intrinsic metrics such as *contextual exponence* and *effective vocabulary size*, as well as linguistic metrics such as morphological alignment. Table 14 in the Appendix provides a comprehensive description of the metrics we evaluate. This broad intrinsic evaluation provides us with a meaningful assessment of the tokens produced by each tokenization approach.

To measure the impact of each tokenization approach on downstream performance, we pretrain BERT (Devlin et al., 2019) models[5] at a scale of 12M parameters[6] and finetune them on tasks: Sentiment Analysis, POS Tagging, Named Entity Recognition (NER), and Dependency Parsing. We limit the downstream evaluation to three typologically distinct languages: English, Hindi, and Telugu; to keep the computational cost manageable.

3.4 Experimental Setup

We take a backward approach from the ideal experimental setup as noted by Poelman et al. (2025), allowing us to better isolate the effect of the tokenizer. We train SSLMs, H-Nets, and *fixed-tokenizer* algorithms on the train subset of the dataset. Training is monitored on the validation subset, and all tokenization approaches are evaluated on the test subset. Total parameters count of H-Net models remains at approximately 3M, and that of SSLMs at 2M. This ensures that the ratio of token-to-parameter count is at least 2 across languages (see Table 12). While H-Nets do not require any vocabulary for initialization, SSLMs require a fixed lexicon provided beforehand. We consider $10,000$ most frequent words for this, following Meyer and Buys (2025), and set maximum token length[7] as 5. For downstream evaluation, we scrape a larger dataset containing 10M sentences from same data source; NewsCrawl (Chelba et al., 2013). This ensures a reasonable data size for 2M-30M parameter BERT models which we analyze. Detailed hyperparameters for each models are listed in the Appendix B.

[5]Note that this requires treating *tokenizer-free* approach as pretokenization. We discuss the reasons behind this in § 4.4.

[6]We perform an ablation study at different model scales and discuss the results in § 4.4 and in the Appendix.

[7]Measured in terms of unicode character length, and not as length in raw UTF-8 bytes

English		Hebrew		Hindi		Tamil		Hungarian		Indonesian	
reaping		משטרתיים		कार्यकर्ताओं		வகுத்து		brigádokkal		ceritakan	
reap ing		משטרתי ים		कार्यकर्ता ओं		வகு த்து		brigád okkal		cerita kan	
Segmentation	*F1*	*Segmentation*	*F1*	*Segmentation*	*F1*	*Segmentation*	*F1*	*Segmentation*	*F1*	*Segmentation*	*F1*
rea ping	0	משט רתיים	0	का र्यकर ◌ताओं	0	வக ◌ுத்து	0	bri g ádo kkal	0	ceri takan	0
re ap ing	0.67	מ שטר תי ים	0.5	कार्य कर्ता ओं	0.67	வகு த்து	1	br igá d okkal	0.5	c erita kan	0.67
reap ing	1	משטר תי ים	0.67	कार्य कर्ता ओं	0.67	வகு த்து	1	br igá dok kal	0	c erita kan	0.67
reap ing	1	משטר ת י ים	0.5	कार्य कर्ता ओं	0.67	வ குத்த ◌ு	0	b rigád okkal	0.67	c erita kan	0.67
reap ing	1	משטר ת י ים	0.5	कार्य कर्ता ओं	0.67	வ குத்த ◌ு	0	b rigád okkal	0.67	c erita k an	0.5

Table 2: Evolution of segmentation of few candidate wordforms produced by *tokenizer-free* approach SSLM with morphological alignment F1-scores. English, Hebrew, and Hindi are fusional or analytical languages, while Tamil, Hungarian, and Indonesian are agglutinative languages. More examples in Table 10 in Appendix.

4 Experiments & Results

4.1 Q1: How are the tokens learned?

We analyze the learning dynamics of SSLMs similar to Meyer and Buys (2025), but with broader language coverage, script variation, and, more importantly, dataset sizes adjusted according to *byte-premiums* (Arnett et al., 2024). We evaluate and track morphological alignment against MorphScore (Arnett and Bergen, 2025; Arnett et al., 2025) data, along with intrinsic properties such as effective vocabulary size and number of distinct neighbouring tokens, i.e., contextual exponence (Yehezkel and Pinter, 2023)[8]. Our results confirm the findings of Meyer and Buys (2025) with inclusion of more diverse languages.

Morphological Alignment

Figure 2 plots the variation in morphological alignment of tokens produced by SSLM, as its training progress, for a subset of languages; a similar plot containing all 18 languages is plotted in figure 9 in Appendix.

Findings: Across all languages, we observe four distinct phases in the morphscore dynamics: 1) **Rapid initial evolution** of morphemes, 2) **Fluctuation** for a brief period which is more consistent and pronounced in case of Agglutinative languages, 3) **Convergence** to a fixed alignment, and 4) **Saturation**, where alignment remains static implying convergence to a solution. Overall, higher alignment in Templatic and Fusional languages as compared to Agglutinative languages. This implies that a jointly learned tokenizer effectively identifies the root-and-pattern structures (as in Hebrew) and the inflectional suffixes (as in Hindi) very early in training. Despite being highly synthetic, Tamil

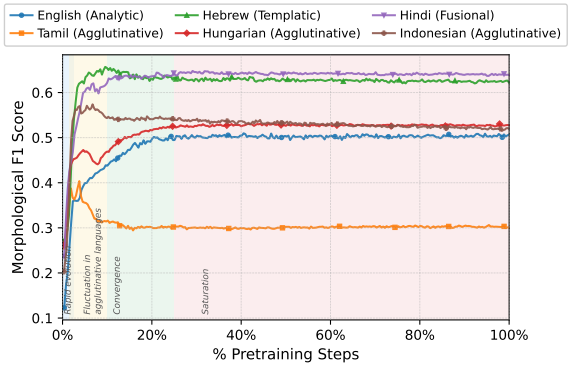
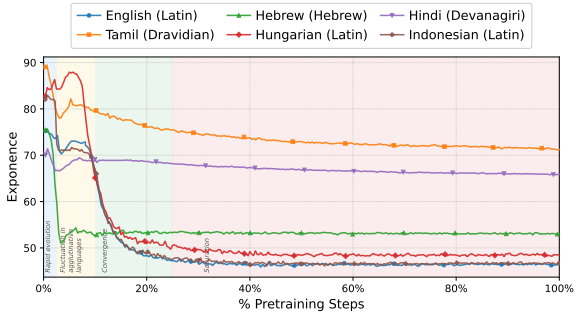


Figure 2: Evolution of morphological alignment (F1-score) of *tokenizer-free* approach SSLM shows that agglutinative languages show larger fluctuations compared to other languages.

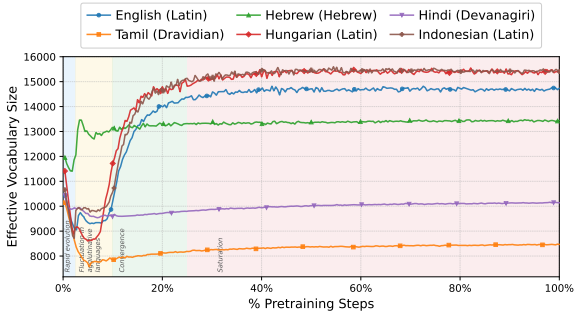
converges to comparatively much lower F1-score (≈0.3). Interestingly, the tokens reach mid-range alignment, but then regresses to a lower alignment. This is observed in both Tamil and Indonesian. Hungarian also shows similar trends like Tamil, but converges to higher alignment following another fluctuation. English, on the other hand, reaches a stable, mid-range alignment (≈0.5) without any major fluctuations. This may reflect its relatively simple morphology.

We identify and list candidate data-points from morphscore dataset (see Table 2), which depicts the overall trend. For Analytic and Fusional languages, we observe how quickly the jointly learned tokenizer identifies the morphemes. In English, it instantly progresses from nonsensical split ("rea ping", at step 2200) to a perfect morphologically aligned segment ("reap ing", at step 2950). Similarly in Hindi, it converges to a stable segmentation ("कार्य+कर्ता+ओं", at step 800) capturing the core semantic units. This implies that in languages where word structures are comparatively less complex, it converges on a stable "lexicon" of subwords. However, in Agglutinative languages such as Tamil,

[8]Refer to Table 14 in Appendix for brief description of these metric.



(a) Contextual Exponence



(b) Effective Vocabulary size

Figure 3: Evolution of number of distinct neighbouring tokens (i.e., contextual exponence) and effective vocabulary size of *tokenizer-free* approach SSLM shows a script-wise similarity. Latin languages saturates at lower contextual exponence through higher effective vocabulary size.

it reaches a perfect alignment ("வகு+த்து", at step 450), but then converges on an inaccurate split ("வ+குத்த+ ◌ு", at step 7150), implying that the jointly learned tokenizer found it beneficial to tradeoff morphological alignment in such languages.

Intrinsic Properties

We perform similar analysis for intrinsic properties such as contextual exponence and effective vocabulary size on test split.

Findings: Figure 3a and 3b reveal script-wise similarities in dynamics. Latin-script languages such as English, Hungarian, and Indonesian show largest effective vocabulary sizes, reaching between $14,000$ to $15,500$ unique tokens. This suggests that jointly learned tokenizer combines frequent character sequences into larger dedicated units to optimize contextual efficiency. We thereby see a steep drop in contextual exponence in these languages.

Among the subset of languages here, non-Latin languages such as Tamil and Hindi show smaller effective vocabulary size and higher contextual exponence. This suggests that jointly learned tokenizer prefers to learn a small, flexible set of fundamental morphemes rather than memorizing long, specific wordforms in such scripts. Dravidian-script languages such as Tamil combine many parts together, so one learned subword can still work with many different neighboring units because words often have many added affixes. This is observed through higher contextual exponence and lower vocabulary size. In contrast, Hebrew reaches its stable effective vocabulary size much earlier than other scripts. This shows again the ability of jointly learned tokenizers to quickly identify the root-and-pattern (introflexive) structures inherent in the script.

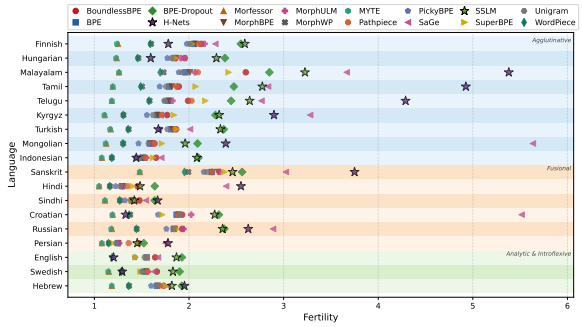
4.2 Q2: What tokens are learned?

In this section, we evaluate both *tokenizer-free* approaches[9]; H-Nets and SSLMs; and compare them with *fixed-tokenizer* approaches mentioned in § 3.2. We focus on metrics listed in Table 14 of Appendix. **Note:** H-Nets and SSLMs can, by design, produce fundamentally different types of tokens; H-Nets can produce cross-tokens (or superwords) frequently, while SSLMs are bounded to produce only subwords. This is reflected in the further results and discussions.

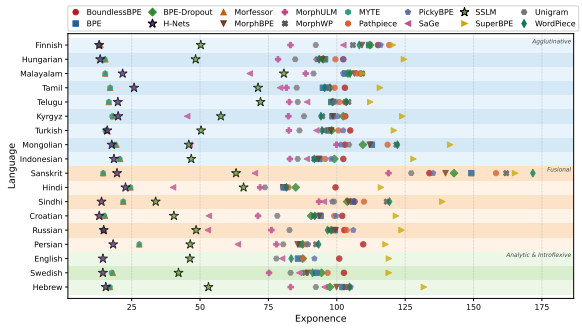
Morphological Alignment

Findings: As observed in figure 1, we identify a substantial difference among tokens produced by the two *tokenizer-free* approaches. SSLMs tradeoff computational efficiency for high linguistic fidelity, while H-Nets prioritize byte-level computational efficiency at the direct expense of morphological alignment. This can be observed by H-Net's low morphscore-F1 ranging below 0.1 across all languages. On the other hand, joint optimization approach (i.e., SSLM) showcase significantly higher alignment ranging from ≈ 0.26 in Telugu to ≈ 0.65 in Hindi. Notably, SSLMs show consistently higher alignment as compared to popular subword algorithms such as BPE or WPC, even converging to the most aligned tokens in languages such as Malayalam and Turkish. However in other languages it tends to align less compared to approaches such as ULM, MorphULM, MYTE, or Morfessor. In Hebrew, despite such *fixed-tokenizer* approaches providing nearly perfect alignment with morphology, SSLMs show comparatively lower alignment ($\approx$ 0.63). These findings suggest that the desirable morphological alignment of jointly learned tokenizer is indeed dependent on a language's morphological properties.

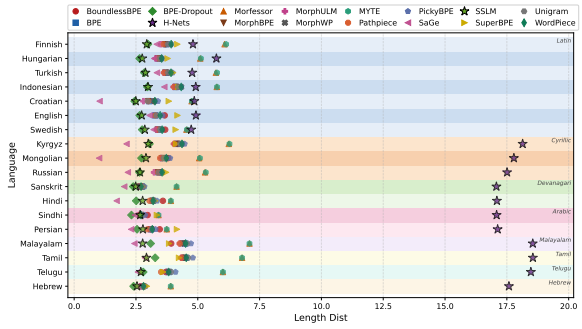
[9]We consider the tokens corresponding to the best checkpoint, i.e., the checkpoint corresponding to the least validation loss.



(a) Fertility



(b) Contextual exponence



(c) Mean Token Length

Figure 4: Intrinsic evaluation (fertility, contextual exponence, and mean token length) of *tokenizer-free* approaches (SSLMs and H-Nets) and *fixed-tokenizer* approaches. *Tokenizer-free* approaches show comparatively higher fertility and lower contextual exponence. Mean token length of H-Nets are significantly higher for non-Latin languages.

Intrinsic Properties

Findings: As observed in figure 4a, both *tokenizer-free* approaches exhibit higher fertility than *fixed-tokenizer* baselines, especially in agglutinative languages such as Malayalam, Tamil, and Telugu; this effect is strongest in H-Nets (≈5.0 to ≈5.5), while analytic or fusional languages remain below ≈2.0.

Across languages, *tokenizer-free* approaches consistently yield lower contextual exponence than *fixed-tokenizer* methods (see figure 4b), with SSLM and H-Net averaging fewer than 75 neighbouring tokens. This likely arises because minimizing language-modeling loss favors units with more predictable grammatical roles and fewer valid neighbours, improving semantic quality and downstream performance (Yehezkel and Pinter, 2023).

Interestingly, mean token length reveals strong divergence between *tokenizer-free* approaches (see figure 4c), especially across scripts. In non-Latin languages, H-Nets produce exceptionally long tokens (≈17 to 18.5 characters), likely because either data-dependent boundary prediction favors long phrases to minimize byte-level loss or struggles to identify boundaries in complex scripts. In contrast, SSLMs and most fixed tokenizers maintain shorter, stable lengths (≈2.5 to 5.0 characters) which follows from its design of explicit maximum token-length constraints.

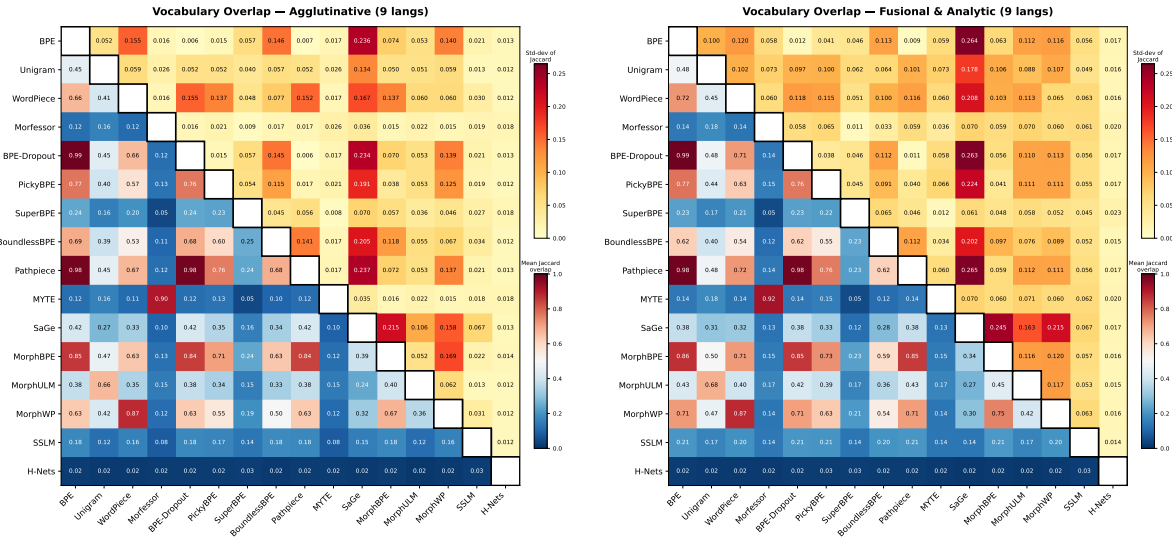
4.3 Q3: How similar are the learned tokens?

We measure pairwise Jaccard overlap on the test split to compare tokens learned by *tokenizer-free* and *fixed-tokenizer* approaches across language typologies. Figure 5 shows the corresponding heatmap.

Findings: H-Nets exhibit near-zero overlap with fixed tokenizers across typologies, indicating that end-to-end data-dependent boundary prediction learns token sets fundamentally different from frequency-based subword methods by prioritizing byte-level sequence efficiency over standard subword boundaries. SSLMs, on the other hand, show comparatively higher (though still low) overlap with fixed tokenizers, particularly with frequency-based methods such as BPE and WPC; the strongest similarity occurs between SSLMs and BPE variants across all typologies. Overlap is lowest for agglutinative languages (≈0.12 to 0.18 Jaccard), suggesting that productive morphology generates many possible subwords, encouraging joint optimization to learn smaller atomic morphemes while fixed tokenizers merge them into larger but linguistically arbitrary units. BPE, BPE-dropout, PathPiece, and MorphBPE form a high-overlap cluster (greater than 0.80 Jaccard), indicating that changes to pretokenization or inference do not fundamentally alter learned vocabularies. In contrast, SaGe shows lower overlap, suggesting that prioritizing contextual exponence substantially changes the vocabulary.

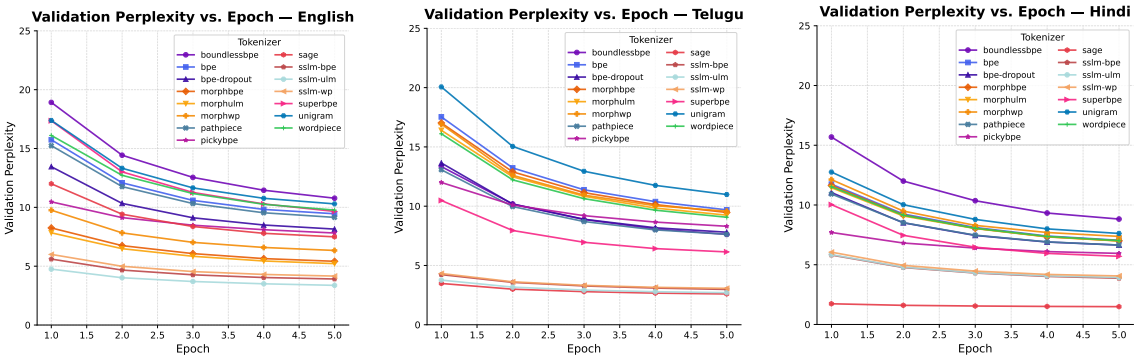
4.4 Q4: Are the learned tokens better?

To determine if these fundamentally distinct tokens learned through *tokenizer-free* approaches improve language understanding, we analyze the downstream performance of BERT (Devlin et al.,



(a) Agglutinative Languages **(b)** Analytic and Fusional Languages

Figure 5: Jaccard token overlap between *tokenizer-free* approaches such as SSLM and H-Net, and fixed-tokenizer approaches on the test split. The lower triangle of the heatmap denotes the Jaccard overlap, while the upper triangle denotes the standard deviation of the overlap.



(a) English (Analytic) **(b)** Telugu (Agglutinative) **(c)** Hindi (Fusional)

Figure 6: Variation in the validation perplexity of BERT models during pretraining. Models trained with tokenizers using SSLM as pretokenization consistently achieve the lowest perplexity.

2019) models pretrained with each tokenizer and finetuned for the tasks listed in § 3.3, with results reported in Table 3. As noted, *tokenizer-free* approaches do not have a fixed vocabulary, while BERT models require static, fixed, pre-defined vocabulary. Therefore, we use *tokenizer-free* approaches[10] as pretokenization and learn a standard fixed tokenizer on the pre-segmented corpus. This approach fixes the model architecture and provides a fair downstream comparison. We refer to these with prefix SSLM, i.e., SSLM-BPE, SSLM-ULM, and SSLM-WPC.

Findings: As illustrated in Figure 6 and summarized in Table 4, *tokenizer-free* approaches consistently achieve the lowest validation perplexity across all three evaluated languages and converge earlier compared to fixed tokenizers. Only in Hindi, we observe SaGe outperforming *tokenizer-free* approaches with nearly perfect perplexity of 1. This improvement in perplexity is even more pronounced as compared to other pretokenization-based modifications such as that of Morfessor. We also observe that Telugu has the highest perplexity without pretokenization, but the lowest perplexity when the *tokenizer-free* SSLM is used for pretokenization.

[10]We avoid considering pretokenization with H-Nets, as it is observed that it mostly segments at whitespaces and it is not straightforward to limit and fix its vocabulary due to its cross-token flexibility.

Tokenizer	Sentiment Analysis (Acc)			POS Tagging (F1)			Named Entity Recognition (F1)			Dependency Parsing (LAS)		
Language	eng_latn	tel_telu	hin_deva	eng_latn	tel_telu	hin_deva	eng_latn	tel_telu	hin_deva	eng_latn	tel_telu	hin_deva
Standard Algorithms												
BPE	85.55	62.85	76.67	95.09	88.41	97.15	89.98	94.72	90.69	75.64	66.31	88.20
UnigramLM	83.61	64.88	**77.63**	95.11	89.72	97.20	89.61	94.80	91.08	75.95	68.88	**88.73**
WordPiece	84.86	65.91	76.10	95.22	89.51	**97.34**	88.37	**95.33**	90.73	**78.11**	66.31	88.63
Contextual Algorithms												
SaGe	85.32	63.22	73.61	91.68	81.99	91.67	83.32	91.88	88.32	62.17	55.14	70.07
Inference Modifications												
BPE-dropout	80.28	62.66	72.85	93.06	83.50	95.37	84.70	92.78	89.41	68.38	59.06	83.47
PathPiece	84.75	64.32	77.06	92.27	87.10	95.08	83.89	92.62	87.04	62.76	58.76	77.47
Vocabulary Modifications												
PickyBPE	73.16	60.44	58.70	78.74	76.57	77.54	68.96	88.61	84.27	49.96	45.77	56.66
Cross-token Algorithms												
BoundlessBPE	83.49	65.25	75.14	91.98	83.50	95.25	84.84	93.62	87.15	61.65	57.55	78.08
SuperBPE	83.83	64.51	74.76	93.82	85.17	96.17	88.93	93.05	88.21	68.32	62.69	80.85
Pre-tokenization using Morfessor												
MorphBPE	86.24	63.96	76.10	95.18	89.12	97.24	**90.63**	94.66	91.02	75.56	**69.18**	88.11
MorphULM	85.89	66.91	77.06	**95.28**	89.38	97.33	90.03	94.90	90.52	75.28	69.03	88.24
MorphWPC	**87.38**	**67.28**	75.91	95.26	90.03	97.23	88.10	94.95	90.75	77.65	69.03	88.35
Pre-tokenization using SSLM												
SSLM-BPE	85.66④	64.70⑦	77.06④	95.09⑥	88.42⑧	97.18⑥	88.93⑤	94.38⑨	91.03④	74.90⑧	66.77⑦	87.47⑧
SSLM-ULM	84.17⑨	63.40⑪	76.10⑧	95.06⑧	**90.17**①	97.16⑧	88.88⑥	94.58⑧	91.30②	73.35⑨	68.58⑥	86.95⑨
SSLM-WPC	84.75⑩	65.25⑤	74.38⑫	94.95⑨	90.10②	97.17⑦	87.05⑨	94.84⑥	**91.41**①	76.78③	68.73⑤	87.51⑦

Table 3: Downstream performance of 12M-parameter BERT models trained with different tokenizer variants. Best results for each task-language pair are in **bold**; superscripts denote tokenizer ranks with SSLM pretokenization. Scores are averaged over three finetuning runs (seeds 42, 43, 44). We report accuracy for Sentiment Analysis, F1 for POS tagging and NER, and LAS for Dependency Parsing. Variants improved by SSLM pretokenization are underlined.

Pretokenizer	eng_latn	tel_telu	hin_deva
None	9.84	9.92	7.23
SSLM	**3.81** (↓61.3%)	**2.92** (↓70.6%)	**3.96** (↓45.2%)

Table 4: Aligning tokens learned by fixed-tokenizer approaches with that learned by SSLM through pretokenization significantly reduces the perplexity, showcasing improvement in language modeling. The scores are averages of perplexities across BPE, ULM, and WPC.

The downstream results (see Table 3 above) showcase the practical implications of each approach. Despite having fundamentally distinct vocabulary, *tokenizer-free* approaches remain competitive. In contrast, the performance of many modified *fixed-tokenizer* approaches such as SaGe, PickyBPE, and BPE-dropout remains inconsistent and often significantly lower across different tasks and languages. Notably, SSLM-ULM achieves the highest F1-score for POS tagging in Telugu (90.17%), while SSLM-WPC achieves highest for NER in Hindi (91.41%). Pretokenization with Morfessor remains the best performing approach overall. It consistently improves the performance over standard approaches. Overall, the results suggest that jointly learned tokenizers learn meaningful tokens that transfer effectively to downstream NLP tasks.

We additionally perform an ablation study across different model scales and observe consistent trends. SSLM-based pretokenization continues to achieve lower validation perplexity and competitive downstream performance across scales, although the relative gains reduce slightly for larger models. Detailed experimental setup, hyperparameters, and complete results are provided in the § A of Appendix.

5 Conclusion

In this work, we showed that jointly learned tokenization fundamentally changes the structure and behavior of learned tokens across typologically diverse languages. In particular, SSLMs learn more morphologically aligned and contextually efficient segments than standard frequency-based tokenizers, while H-Nets prioritize byte-level efficiency and produce substantially different token structures. Although morphology-aware pretokenization remains the strongest overall approach, SSLM-based pretokenization consistently improves corresponding tokenizer families and achieves competitive downstream performance without relying on external linguistic resources. Overall, our findings show that jointly optimized tokenizers can learn linguistically meaningful token structures that remain effective for downstream NLP despite having substantially different vocabularies from standard tokenization approaches.

Code and Data Availability

Code and artifacts for this work are publicly available at `github.com/SakethReddyVemula/what-tokens-are-learned`. The release includes the tokenizer training, evaluation, and analysis pipeline, together with the trained tokenizers and the raw per-metric results underlying every table and figure reported above. The README of the repository links all remaining artifacts, including the training corpora, the pretrained SSLM, H-Net, and BERT checkpoints, the pre-segmented corpora, and the companion repositories used to train the *tokenizer-free* models.

Limitations

While we do our best to handle confounding factors such as dataset sizes, model sizes, and other experimental setup, a few limitations remain:

1. **Model and dataset sizes**: We focused on spanning an extensive set of tokenization approaches, languages, and their combinations. Therefore, our analysis was limited to models and dataset at relatively smaller scale. For instance, we limit *tokenizer-free* LMs to 3M parameters and dataset of 250,000 sentences. Downstream evaluation was performed on BERT models at 12M parameters and dataset of 10M sentences. Although we perform an ablation study with BPE and SSLM-BPE tokenization approaches, we leave scalability analysis of our findings as future work. In this work, we trade-off model and dataset sizes for larger coverage of languages and tokenization approaches.

2. **Generalizability of downstream results**: For every combination of tokenization approach and language, we had to pretrain a model. Hence, our downstream evaluation was limited to three typologically distinct languages: English, Hindi, and Telugu. We do not evaluate downstream performance across all 18 languages, in order to keep computational cost manageable. We focus on evaluating all tokenization approaches for these selected languages in a controlled experimental setting.

3. **Hyperparameters of *tokenizer-free* LMs**: We do not sweep over certain hyperparameters of *tokenizer-free* LMs, for example, maximum token length in SSLM. We fix them following heuristics from previous works, since sweeping over these hyperparameters require significant computation resources, which we limit.

Ethics Statement

This work focuses on the analysis of tokenization strategies across languages and does not involve human subjects or the use of sensitive personal data. All experiments are conducted on publicly available text corpora. However, differences in dataset quality, and linguistic coverage across languages may introduce biases that affect the observed behavior of tokenization methods.

Acknowledgement

We would like to thank Language Technologies Research Center (LTRC), IIIT Hyderabad for providing essential compute resources and support.

References

Md Shad Akhtar, Ayush Kumar, Asif Ekbal, and Pushpak Bhattacharyya. 2016. A hybrid deep learning architecture for sentiment analysis. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 482–493, Osaka, Japan. The COLING 2016 Organizing Committee.

Sawsan Alqahtani, Mir Tafseer Nayeem, Md Tahmid Rahman Laskar, Tasnim Mohiuddin, and M Saiful Bari. 2026. Stop taking tokenizers for granted: They are core design decisions in large language models. *Preprint*, arXiv:2601.13260.

Catherine Arnett and Benjamin Bergen. 2025. Why do language models perform worse for morphologically complex languages? In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 6607–6623, Abu Dhabi, UAE. Association for Computational Linguistics.

Catherine Arnett, Tyler A. Chang, and Benjamin Bergen. 2024. A bit of a problem: Measurement disparities in dataset sizes across languages. In *Proceedings of the 3rd Annual Meeting of the Special Interest Group on Under-resourced Languages @ LREC-COLING 2024*, pages 1–9, Torino, Italia. ELRA and ICCL.

Catherine Arnett, Marisa Hudspeth, and Brendan O'Connor. 2025. Evaluating morphological alignment of tokenizers in 70 languages. *Preprint*, arXiv:2507.06378.

Tamali Banerjee and Pushpak Bhattacharyya. 2018. Meaningless yet meaningful: Morphology grounded subword-level NMT. In *Proceedings of the Second Workshop on Subword/Character LEvel Models*, pages 55–60, New Orleans. Association for Computational Linguistics.

Khuyagbaatar Batsuren, Gábor Bella, and Fausto Giunchiglia. 2021. MorphyNet: a large multilingual database of derivational and inflectional morphology. In *Proceedings of the 18th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 39–48, Online. 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, Rajat Verma, Nagasai Saketh Naidu, Rohit Saluja, Maunendra Sankar Desarkar, and Ganesh Ramakrishnan. 2026. Multilingual tokenization through the lens of Indian languages: Challenges and insights. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 32614–32632, San Diego, California, United States. Association for Computational Linguistics.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. Technical report, Google.

Pavel Chizhov, Catherine Arnett, Elizaveta Korotkova, and Ivan P. Yamshchikov. 2024. BPE gets picky: Efficient vocabulary refinement during tokenizer training. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 16587–16604, Miami, Florida, USA. Association for Computational Linguistics.

Marco Cognetta, Tatsuya Hiraoka, Rico Sennrich, Yuval Pinter, and Naoaki Okazaki. 2024. An analysis of BPE vocabulary trimming in neural machine translation. In *Proceedings of the Fifth Workshop on Insights from Negative Results in NLP*, pages 48–50, Mexico City, Mexico. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Sumanth Doddapaneni, Rahul Aralikatte, 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.

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

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Valentin Hofmann, Hinrich Schuetze, and Janet Pierrehumbert. 2022. An embarrassingly simple method to mitigate undesirable properties of pretrained language model tokenizers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 385–393, Dublin, Ireland. Association for Computational Linguistics.

Yifan Hu, Frank Liang, Dachuan Zhao, Jonathan Geuter, Varshini Reddy, Craig W. Schmidt, and Chris Tanner. 2025. Entropy-driven pre-tokenization for byte-pair encoding. *Preprint*, arXiv:2506.15889.

Sukjun Hwang, Brandon Wang, and Albert Gu. 2025. Dynamic chunking for end-to-end hierarchical sequence modeling. *Preprint*, arXiv:2507.07955.

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.

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

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

Tomasz Limisiewicz, Terra Blevins, Hila Gonen, Orevaoghene Ahia, and Luke Zettlemoyer. 2024. MYTE: Morphology-driven byte encoding for better and fairer multilingual language modeling. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15059–15076, Bangkok, Thailand. Association for Computational Linguistics.

Alisa Liu, Jonathan Hayase, Valentin Hofmann, Sewoong Oh, Noah A. Smith, and Yejin Choi. 2025. Superbpe: Space travel for language models. *Preprint*, arXiv:2503.13423.

Francois Meyer and Jan Buys. 2022. Subword segmental language modelling for nguni languages. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6636–6649, Abu Dhabi, United

Arab Emirates. Association for Computational Linguistics.

Francois Meyer and Jan Buys. 2025. The learning dynamics of subword segmentation for morphologically diverse languages. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pages 647–661, Mumbai, India. The Asian Federation of Natural Language Processing and The Association for Computational Linguistics.

Sandeep Sricharan Mukku and Radhika Mamidi. 2017. ACTSA: Annotated corpus for Telugu sentiment analysis. In *Proceedings of the First Workshop on Building Linguistically Generalizable NLP Systems*, pages 54–58, Copenhagen, Denmark. Association for Computational Linguistics.

Piotr Nawrot, Jan Chorowski, Adrian Lancucki, and Edoardo Maria Ponti. 2023. Efficient transformers with dynamic token pooling. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6403–6417, Toronto, Canada. Association for Computational Linguistics.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal Dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4034–4043, Marseille, France. European Language Resources Association.

Joakim Nivre and Chiao-Ting Fang. 2017. Universal Dependency evaluation. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*, pages 86–95, Gothenburg, Sweden. Association for Computational Linguistics.

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual name tagging and linking for 282 languages. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1946–1958, Vancouver, Canada. Association for Computational Linguistics.

Wessel Poelman, Thomas Bauwens, and Miryam de Lhoneux. 2025. Confounding factors in relating model performance to morphology. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 7262–7287, Suzhou, China. Association for Computational Linguistics.

Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. 2020. BPE-dropout: Simple and effective subword regularization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1882–1892, Online. Association for Computational Linguistics.

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

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

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *International Conference on Acoustics, Speech and Signal Processing*, pages 5149–5152.

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.

Peter Smit, Sami Virpioja, Stig-Arne Grönroos, and Mikko Kurimo. 2014. Morfessor 2.0: Toolkit for statistical morphological segmentation. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 21–24, Gothenburg, Sweden. Association for Computational Linguistics.

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.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need. *Preprint*, arXiv:1706.03762.

Saketh Reddy Vemula, Sandipan Dandapat, Dipti Sharma, and Parameswari Krishnamurthy. 2025. Rethinking tokenization for rich morphology: The dominance of unigram over BPE and morphological alignment. In *The 14th International Joint Conference on Natural Language Processing and The 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pages 232–252, Mumbai, India. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Shaked Yehezkel and Yuval Pinter. 2023. Incorporating context into subword vocabularies. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 623–635, Dubrovnik, Croatia. Association for Computational Linguistics.

Qingyang Zhu, Xiang Hu, Pengyu Ji, Wei Wu, and Kewei Tu. 2025. Unsupervised morphological tree tokenizer. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 22299–22312, Vienna, Austria. Association for Computational Linguistics.

Yi Zhu, Benjamin Heinzerling, Ivan Vulić, Michael Strube, Roi Reichart, and Anna Korhonen. 2019. On the importance of subword information for morphological tasks in truly low-resource languages. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 216–226, Hong Kong, China. 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 Ablation Study: Model Scale

In our downstream evaluation, we fix the size of BERT models at approximately 12M parameters. This was scaled accordingly with respect to the data size of 10M sentences. Here, we perform an ablation study by changing the size of our models while keeping the data size same. We perform additional evaluation of BERT models at approximately 2M and 30M parameters. To limit the computational resources, we limit our analysis to two tokenization approaches: BPE and SSLM-BPE. This allows us to focus on the effect of pretokenization using *tokenizer-free* approach SSLM.

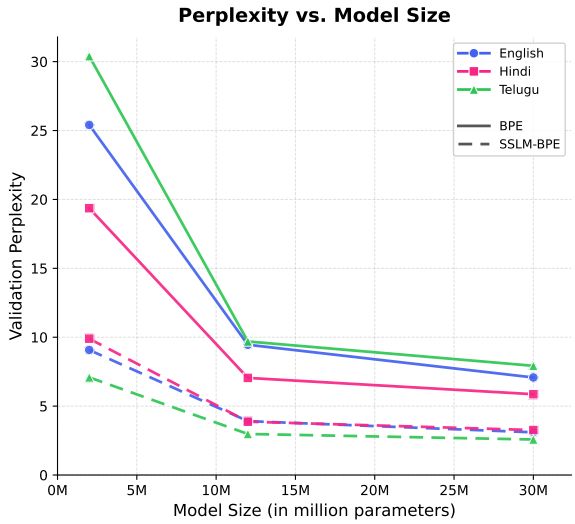


Figure 7: Validation perplexity of BERT models at different scales. Aligning tokens with *tokenizer-free* approaches through pretokenization consistently results in lower perplexity.

Findings: As observed in Figure 7, models trained using SSLM as pretokenization consistently show lower perplexity across all model sizes. However, as model size increases, the improvement saturates. This indicates that SSLM-BPE generally provides more efficient language modeling. We also perform evaluation on downstream tasks at different scales. The results are listed in Table 5 and plotted in Figure 8. Across all three languages, SSLM-BPE often matches BPE on downstream tasks such as POS tagging and NER. We observe that whenever model benefits with larger size, the gap between SSLM-BPE and BPE reduces, especially for POS tagging and NER tasks. **Note**: Our *tokenizer-free* LMs were trained 250,000 sentence corpora, rather than on the entire 10M sentence corpora which we use for downstream evaluation. This was due to high computation requirement of such models (approximately 10× slower than transformers). We leave further scaling analysis accounting for these factors as our future work.

B Hyperparameters & Experimental Setup

B.1 H-Nets

Table 6 details the primary hyperparameter configurations used for pretraining the Hierarchical Network (H-Net) models across all evaluated languages. The model employs a structural layout of m_1 at the bottom level and T_2 at the top level, utilizing an embedding and hidden dimension of 256 for both levels. The bottom level (m_1) operates purely without an FFN block, while the top level (T_2) uses an intermediate feed-forward dimension of 640. This setup operates directly on byte sequences (vocabulary size of 256) and results in a lightweight model with approximately 3M parameters. We trained the model using an AdamW optimizer (with $\beta_1 = 0.9, \beta_2 = 0.95$) and a batch size of 128 for a maximum of 70 epochs. Optimization was guided by a cosine learning rate scheduler with a peak base learning rate of 3×10^{-4} after an initial 10% linear warmup phase. To prevent overfitting, early stopping was employed with a patience of 5 epochs alongside a weight decay of 0.01. The H-Net specific structural compression constraints were maintained constantly throughout training, with the ratio loss scale set to 1.0 and zero warmup compression epochs.

B.2 SSLMs

Table 7 details the comprehensive set of hyperparameters utilized during the pre-training of our Subword Segmental Language Models (SSLMs) across all evaluated languages. We adopted a lightweight Transformer decoder architecture tailored to subword segmental modeling, bringing the total parameter count to 2,105,836. Key architectural adjustments include scaling down the number of layers, embedding dimensions, and feed-forward dimensions to ensure fast and efficient training with a limited parameter budget. The maximum segment length was constrained to 5 spatial units to balance compositional flexibility and computational overhead. Optimization was performed using the Adam optimizer coupled with an inverse square root learning rate scheduler and a linear warmup phase.

Model Scale	Sentiment Analysis (Acc)			POS Tagging (F1)			Named Entity Recognition (F1)			Dependency Parsing (LAS)		
Language	eng_latn	tel_telu	hin_deva	eng_latn	tel_telu	hin_deva	eng_latn	tel_telu	hin_deva	eng_latn	tel_telu	hin_deva
Standard Algorithm: BPE												
2M	77.75	59.15	65.97	89.16	62.22	92.57	70.79	86.15	75.90	54.78	56.04	74.12
12M	85.55	62.85	76.67	95.09	88.41	97.15	89.98	94.72	90.69	75.64	66.31	88.20
30M	**88.42**	**65.43**	**78.01**	95.97	**92.82**	**97.64**	**91.65**	**95.66**	91.91	**80.29**	**72.96**	**90.44**
Using SSLM as pretokenization: SSLM-BPE												
2M	76.95	59.89	58.89	88.05	61.76	91.28	56.18	83.37	76.61	50.99	53.62	72.38
12M	85.66	64.70	77.06	95.09	88.42	97.18	88.93	94.38	91.03	74.90	66.77	87.47
30M	86.01	64.70	78.01	**96.13**	92.78	97.45	91.59	**95.70**	**92.72**	80.20	70.09	89.88

Table 5: Downstream performance of BERT models at different scales trained with tokenizers BPE and SSLM-BPE. Best results for each task-language pair are in **bold**. Scores are averaged over three finetuning runs (seeds 42, 43, 44). We report accuracy for Sentiment Analysis, F1 for POS tagging and NER, and LAS for Dependency Parsing. Variants improved by SSLM pretokenization are underlined.

Hyperparameter	Value
Model Architecture	
Architecture Type	H-Net
Total Parameters	$\sim 3M$
Architecture Layout	$m_1 T_2$
Embedding Dimension	256
Feed-Forward Dimension	640
Attention Heads	4
Optimization Setup	
Optimizer	AdamW
Adam β_1, β_2	$(0.9, 0.95)$
Learning Rate	0.0003
Learning Rate Scheduler	Cosine
Warmup Ratio	10% of total steps
Weight Decay	0.01
Max Epochs	70
Early Stopping Patience	5
H-Net & Data Configuration	
Vocabulary Size (Bytes)	256
Max Sequence Length	4096
Batch Size	128
Ratio Loss Scale	1.0
Warmup Compression Epochs	0

Table 6: Hyperparameter configurations and approximate parameter count for the Hierarchical Network (H-Net) pre-training phase.

Hyperparameter	Value
Model Architecture	
Architecture Type	Transformer SSLM
Total Parameters	$\sim 2M$
Decoder Layers	3
Embedding Dimension	128
Feed-Forward Dimension	512
Attention Heads	4
Optimization Setup	
Optimizer	Adam
Adam β_1, β_2	$(0.9, 0.98)$
Learning Rate	0.0005
Learning Rate Scheduler	Inverse Square Root
Warmup Updates	1000
Initial Warmup Learning Rate	1×10^{-7}
Weight Decay	0.01
Gradient Clipping Norm	0.0
Dropout	0.1
Max Epochs	25
Early Stopping Patience	2
SSLM & Data Configuration	
Max Segment Length	5
Lexicon Max Size	10000
Tokens Per Sample	512
Max Tokens	16384

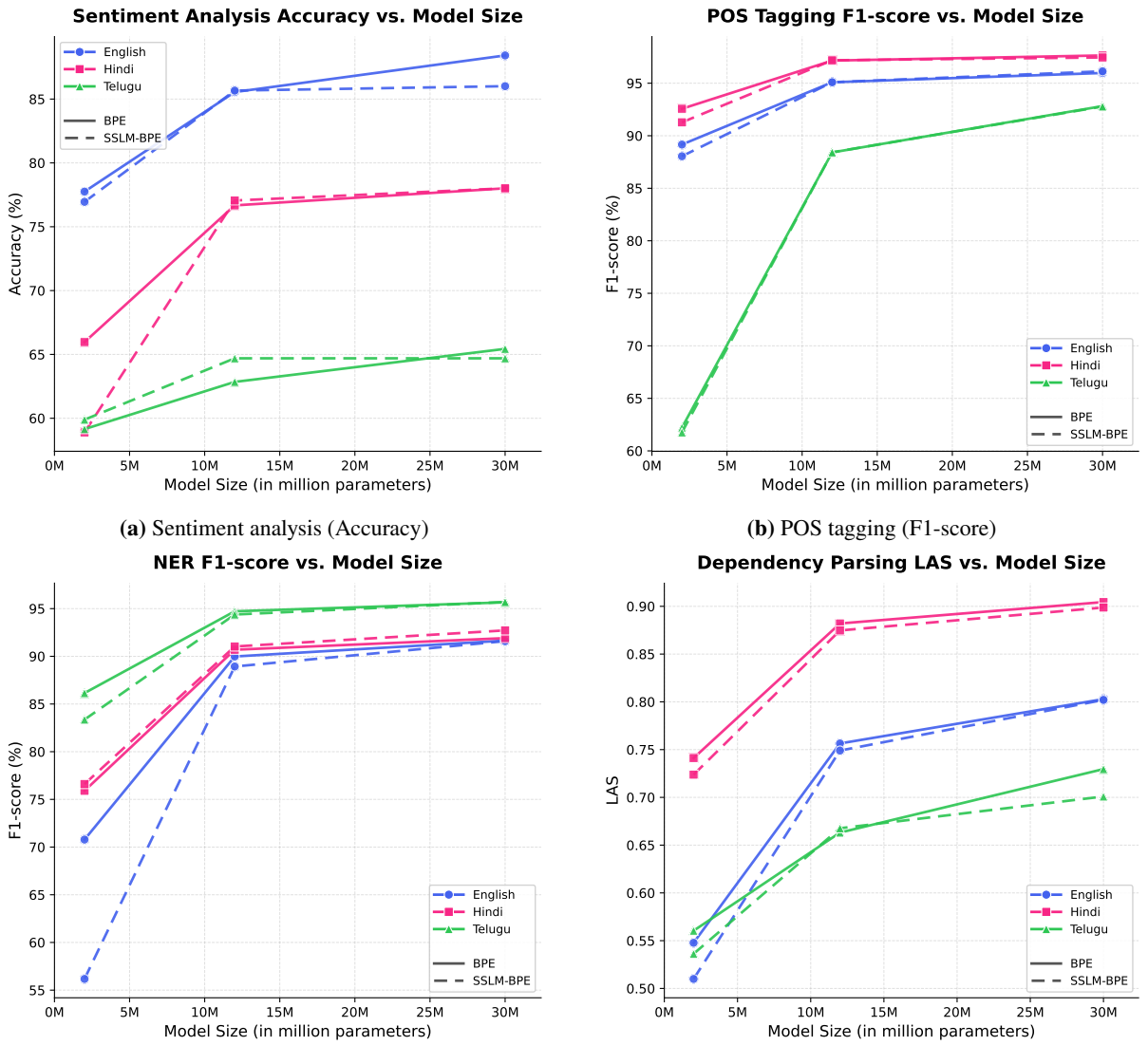
Table 7: Hyperparameter configurations and parameter count for the lightweight Subword Segmental Language Model (SSLM) pre-training phase.

B.3 Fixed-tokenizer Variants

To ensure a fair and comprehensive comparison across different tokenization strategies, we standardized the training hyperparameter configurations where applicable, while setting strategy-specific parameters to their best-known or recommended values.

Global Settings All tokenizers were trained across three final target vocabulary sizes: $T \in \{5000, 10000, 20000\}$. For tokenizers relying on the SentencePiece library (e.g., standard BPE, Unigram, and PathPiece), the character coverage parameter was explicitly set to 1.0 (100%) to prevent any heuristic character dropping and uniformly ensure complete representation of the datasets' alphabets. We consider fixed-tokenizer variants with final vocabulary size of $10,000$ for a fair comparison with SSLMs with initial lexicon of size $10,000$.

Standard Subword Architectures For **WordPiece**, we utilized the HuggingFace tokenizers library, configuring it with NFKC Unicode normalization and lowercasing. Pre-tokenization was handled via a standard whitespace split, and the continuation prefix was assigned as "##". The standard special tokens strictly enforced were `[UNK]`, `[CLS]`, `[SEP]`, `[PAD]`, and `[MASK]`. Standard **BPE** and **UnigramLM** models relied on SentencePiece de-



(a) Sentiment analysis (Accuracy)

(b) POS tagging (F1-score)

(c) NER (F1-score)

(d) Dependency parsing (LAS)

Figure 8: Performance of BERT models across model scales on sentiment analysis, POS tagging, NER, and dependency parsing.

faults beyond the mandatory complete character coverage.

SuperBPE Following the two-stage subword acquisition strategy of SuperBPE, the base vocabulary size (the transition limit t between stage 1 and stage 2) was configured at $t = 4000$ when final vocabulary size desired was $T = 10000$, and $t = 10000$ when $T = 25000$ (i.e., $T = t \times 2.5$, as found in Liu et al. (2025)).

BoundlessBPE For BoundlessBPE, the τ parameter, managing the trade-off threshold, was set to 0.9. The vocabulary pruning recalculation interval was configured to occur every 1000 iterations (`recalc = 1000`). We utilized the `ultimate2` regular expression split pattern constraints, taking advantage of the active vocabulary blowup tolerance (`blowup = 1`).

PickyBPE The intersection-over-sequence threshold regulating token acceptance in PickyBPE was strictly confined to 0.9.

Morphologically-Informed Tokenizers (MorphBPE, MorphULM, MorphWP) To create tokenizers aligned with linguistic boundaries, an unsupervised **Morfessor** Baseline model was first trained on the raw corpus to dictate morphological decisions. Texts were pre-tokenized along the predicted morphological seams, creating a pseudo-corpus separated by whitespace. The subsequent BPE, Unigram, and WordPiece counterparts (**MorphBPE**, **MorphULM**, and **MorphWP**) were exclusively trained on this constrained configuration, forcing the statistically induced vocabularies

to respect Morfessor's proposed boundaries.

B.4 Downstream Evaluation

As discussed, we perform downstream evaluation by pretraining BERT (Devlin et al., 2019) models and later finetuning them on each task. Table 8 lists the hyperparameters we used while pretraining BERT models. Similarly, Table 9 lists the hyperparameters while finetuning for each task.

We describe our downstream tasks as follows:

POS tagging: Involves assigning a grammatical category such as noun, verb, adjective, to each word in a sentence. It involves reasoning through both the definition of the word and its context. We consider the class of word as the class corresponding to its first subword. We use Universal Dependency (UD) treebanks (Nivre et al., 2020) via HuggingFace universal dependency dataset: `en_ewt` for English, `te_mtg` for Telugu, and `hi_hdtb` for Hindi. Labels are drawn from the 17 universal POS tags (UPOS). Performance is measured with weighted-average F1-score.

Sentiment Analysis: This tasks involves determining the sentiment of a sentence, e.g., positive, negative, or neutral. We use ACTSA (Annotated Corpus for Telugu Sentiment Analysis) dataset for Telugu (Mukku and Mamidi, 2017), IITP Product Review dataset (Akhtar et al., 2016), and SST-2 dataset from GLUE benchmark (Wang et al., 2018).

Named Entity Recognition (NER): This tasks involves identifying and classifying named entities into predefined categories such as persons, organizations, locations, etc. We use CoNLL NER dataset (Tjong Kim Sang and De Meulder, 2003) for English, and WikiAnn dataset (Pan et al., 2017; Doddapaneni et al., 2023). WikiAnn consists of coarse grained labels as follows: Person (POS), Organization (ORG), and Location (LOC). Whereas, CoNLL NER contains an additional Miscellaneous (MISC) tag.

Dependency Parsing (DP): This task involves analyzing the grammatical structure of a sentence by identifying the relationship between head words and their dependents. We again use Universal Dependency (UD) treebank dataset (Nivre et al., 2020). We report Labeled Attachment Score (LAS) (Nivre and Fang, 2017).

Hyperparameter	Value
Model Architecture	
Architecture Type	BERT
Total Parameters	~2M / ~12M / ~30M
Hidden Layers	3 / 6 / 8
Hidden Dimension	128 / 384 / 512
Feed-Forward Dimension	512 / 1024 / 2048
Attention Heads	4 / 6 / 8
Max Position Embeddings	128
Activation Function	`gelu`
Regularization	
Attention Dropout	0.1
Hidden Dropout	0.1
Initializer Range	0.01 / 0.02 / 0.02
Layer Norm ϵ	1×10^{-12}
Pre-Training Configuration	
Pre-Training Objective	Masked LM
MLM Masking Probability	0.15
Sequence Length	128
Max Epochs	5
Early Stopping Patience	1

Table 8: Architecture and pre-training configurations for the three BERT model scales (2M / 12M / 30M). Where values differ across scales, they are listed in order from smallest to largest model.

Hyperparameter	Value
Train batch size	32
Eval batch size	32
Epochs	3 for GLUE 10 for ACTSA-TE, Dep. Parsing, IITP-PR, NER (CoNLL), POS, Wiki-NER
Learning rate	2e-5
LR schedule	Linear warmup
Warmup ratio	10% of steps
Weight decay	0.01
Adam ϵ	1e-8
Adam β_1	0.9
Adam β_2	0.999
Max sequence length	128
Precision	FP16

Table 9: Hyperparameters used for fine-tuning BERT across all downstream tasks.

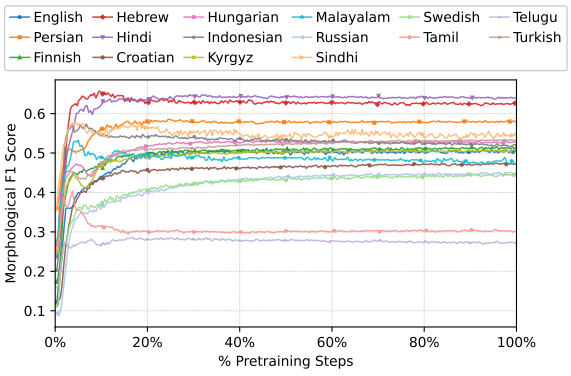


Figure 9: Evolution of Morphological Alignment (F1-score) in Transformer-based Subword Segmental Language Models (SSLM).

English		Hebrew		Hindi		Tamil		Hungarian		Indonesian	
Bangladeshis		ריבוזיים		डिस्ट्रीब्यूटरों		தொடுக்க		zeneszerzőzseniket		kehabisan	
Bangladeshi s		ריבוזי ים		डिस्ट्रीब्यूटर ों		தொடு க்க		zeneszerzőzseni ket		ke habis an	
Segmentation	*F1*	*Segmentation*	*F1*	*Segmentation*	*F1*	*Segmentation*	*F1*	*Segmentation*	*F1*	*Segmentation*	*F1*
Ba ngl ades his	0	ריבו ז יים	0	डिस् ट्री ब्यू टरों	0	தொ டுக்க	0	zene szerz őz sen iket	0	keha bisa n	0
Ba ngla deshi s	0.5	ריב וזי ים	0.67	डिस् ट्री ब्यू ट र ों	0.33	தொட ு க்க	0.67	zene szerz ő zs en i ket	0.29	ke habi s an	0.8
Ba ngla deshi s	0.5	רי בוז י ים	0.5	डिस् ट्री ब्यू ट र ों	0.33	தொ டுக்க	0	zene szerz ő zs en ik et	0	keha bis an	0.5
Ba ngla deshi s	0.5	רי בוז י ים	0.5	डिस् ट्री ब्यू टर ों	0.4	தொ டுக்க	0	zene szerz ő zs eni ket	0.33	keha bis an	0.5
Ba ngla deshi s	0.5	רי בוז י ים	0.5	डिस् ट्री ब्यू टर ों	0.4	தொ டுக்க	0	zene szerz ő zs eni ket	0.33	keha bis an	0.5
meadowlarks		נבשושיים		रजिस्ट्रारों		வசித்து		megtörténhetésének		disucikan	
meadowlark s		גבשושי ים		रजिस्ट्रार ों		வசி த்து		megtörténhetés ének		di suci kan	
Segmentation	*F1*	*Segmentation*	*F1*	*Segmentation*	*F1*	*Segmentation*	*F1*	*Segmentation*	*F1*	*Segmentation*	*F1*
mea do wl arks	0	ג בשו שיים	0	रजि स्ट्र ारों	0	வசி த்து	1	megt örtén he té sének	0	dis uci kan	0.5
mea dow lark s	0.5	ג בש ושי ים	0.5	रजिस् ट्रा र ों	0.5	வசி த்து	1	meg törté nhet és ének	0.4	di s uci kan	0.80
mea dow lark s	0.5	ג בש ושי ים	0.67	रजिस् ट्रा रों	0	வசி த்து	1	meg törté nhet ésé nek	0	dis uci kan	0.5
mea dow lark s	0.5	ג בש ו ש י ים	0.4	रजिस् ट्रा र ों	0.5	வசி த்த ு	0.67	meg törté n het és ének	0.33	dis uci kan	0.5
mea dow lark s	0.5	ג בש ו ש י ים	0.4	रजिस् ट्रा र ों	0.5	வசி த்த ு	0.67	meg törté n het és ének	0.33	dis uci kan	0.5
eschewing		מאושפזות		पुरातत्वविदों		எல்லையய்		szavazatából		kegigihan	
eschew ing		מ אושפז ות		पुरातत्वविद ों		எல்லை ய்		szavazat ából		ke gigih an	
Segmentation	*F1*	*Segmentation*	*F1*	*Segmentation*	*F1*	*Segmentation*	*F1*	*Segmentation*	*F1*	*Segmentation*	*F1*
es che wing	0	מא ושפ זות	0	पुरा तत् ववि दों	0	எல்லை ய்	1	szav azat ából	0.67	keg igi han	0
esc hew ing	0.67	מאו שפ ז ות	0.4	पुरा तत् वविद ों	0.5	எல்லை ய ்	0.67	szava zat ából	0.67	ke gi g ih an	0.67
esc hew ing	0.67	מ א ושפ ז ות	0.67	पु रात त्व विद ों	0.4	எ ல்லைய ்	0	sz avaz atá ból	0	ke g igi h an	0.67
esc hew ing	0.67	מא ושפ ז ות	0.4	पु रात त्व विद ों	0.4	எ ல்லைய ்	0	sz avaz at ából	0.5	ke g igi h an	0.67
esc hew ing	0.67	מא ושפ ז ות	0.4	पु रात त्व विद ों	0.4	எல்லை ய ்	0.67	sz avaz at ából	0.5	keg igi h an	0.4
Veterans		אידיאלים		प्रतिष्ठानों		படையைச்		tartozásának		perumahan	
Veteran s		אידיאל ים		प्रतिष्ठान ों		படை யைச்		tartozás ának		pe rumah an	
Segmentation	*F1*	*Segmentation*	*F1*	*Segmentation*	*F1*	*Segmentation*	*F1*	*Segmentation*	*F1*	*Segmentation*	*F1*
Ve ter ans	0	אידי אלים	0	प्रति ष्ठा नों	0	படை யைச்	1	tar tozás ának	0.67	per uma han	0
V et eran s	0.5	אי דיאל ים	0.67	प्रति ष्ठ ान ों	0.5	படை யைச்	1	tarto zás ának	0.67	pe rumah an	1
Vet eran s	0.67	אי דיאל ים	0.67	प्रति ष्ठ ान ों	0.5	படை யைச்	1	tarto zásá nak	0	per umah an	0.5
Vet eran s	0.67	אי דיאל ים	0.67	प्र तिष्ठ ान ों	0.5	ப டை யைச்	0.67	tarto zás ának	0.67	per umah an	0.5
Vet eran s	0.67	אי דיאל י ם	0.5	प्र तिष्ठ ान ों	0.5	ப டை யைச்	0.67	tarto zás ának	0.67	per umah an	0.5
roasted		עיתונאית		स्लाइड्स		முடிவெடுக்கப்		atrocitások		bergabung	
roast ed		עיתונא ית		स्लाइड ्स		முடிவெடு க்கப்		atrocitás ok		ber gabung	
Segmentation	*F1*	*Segmentation*	*F1*	*Segmentation*	*F1*	*Segmentation*	*F1*	*Segmentation*	*F1*	*Segmentation*	*F1*
roa sted	0	עיתונ אית	0	स्ला इड ्स	0.67	முடிவ ெடுக் கப்	0	atr oci tások	0	berg abung	0
roast ed	1.0	עיתונ אי ת	0.67	स्ल ाइड ्स	0.67	முடிவ ெடு க்க ப்	0.5	a tro c itás ok	0.4	ber gab ung	0.67
roast ed	1.0	עיתונ אי ת	0.67	स ्लाइ ड ्स	0.5	முடிவ ெடு க்க ப்	0.5	a tro c itás ok	0.4	ber gab ung	0.67
roast ed	1.0	ע יתונא י ת	0.5	स ्लाइ ड ्स	0.5	முடிவ ெடுக்க ப்	0	a troc itás ok	0.5	ber gab ung	0.67
roast ed	1.0	ע יתונא י ת	0.5	स्ल ाइड ्स	0.67	முடிவ ெடுக்க ப்	0	atroc itás ok	0.67	b ergab ung	0.5

Table 10: Evolution of segmentation of few more candidate wordforms produced by *tokenizer-free* approach SSLM with morphological alignment F1-scores. English, Hebrew, and Hindi are fusional or analytical languages, while Tamil, Hungarian, and Indonesian are agglutinative languages.

Language	ISO 639-3	ISO 15924	Typology	Family	Group
Agglutinative					
Finnish	fin	latn	Agglutinative (with some Fusion)	Uralic	Finno-Ugric
Hungarian	hun	latn	Agglutinative (Suffixing)	Uralic	Finno-Ugric
Malayalam	mal	mlym	Agglutinative (Highly Synthetic)	Dravidian	Dravidian
Tamil	tam	taml	Agglutinative (Highly Synthetic)	Dravidian	Dravidian
Telugu	tel	telu	Agglutinative (Highly Synthetic)	Dravidian	Dravidian
Kyrgyz	kir	cyrl	Agglutinative (Suffixing)	Turkic	Turkic
Turkish	tur	latn	Agglutinative (Highly Productive Suffixing)	Turkic	Turkic
Mongolian	mon	cyrl	Agglutinative (Suffixing)	Mongolic	Mongolic
Indonesian	ind	latn	Agglutinative (Reduplication-Heavy)	Austronesian	Malayo-Polynesian
Fusional					
Sanskrit	san	deva	Fusional (Polysynthetic Tendencies)	Indo-European	Indo-Aryan
Hindi	hin	deva	Fusional / Analytic (Split-Ergative)	Indo-European	Indo-Aryan
Sindhi	snd	arab	Fusional (Moderate)	Indo-European	Indo-Aryan
Croatian	hrv	latn	Fusional (Highly Synthetic)	Indo-European	Slavic
Russian	rus	cyrl	Fusional (Highly Synthetic)	Indo-European	Slavic
Persian	fas	arab	Fusional (Weakly Synthetic)	Indo-European	Indo-Iranian
Analytic & Introflexive					
English	eng	latn	Analytic / Weakly Fusional	Indo-European	Germanic
Swedish	swe	latn	Analytic / Weakly Fusional	Indo-European	Germanic
Hebrew	heb	hebr	Introflexive (Non-concatenative)	Afroasiatic	Semitic

Table 11: Languages categorized by morphological typology. Grouping highlights the structural similarities in word formation across different language families.

Language	ISO 15924	BP	#sentences	#words	ASL	#MS	#MN	Data Source
Agglutinative								
Finnish	latn	1.0589051	432,422	4,532,824	10.48	10172	3,745,139	NewsCrawl
Hungarian	latn	1.0199851	280,276	4,883,416	17.42	6350	1,044,996	NewsCrawl
Malayalam	mlym	2.8852389	438,501	4,053,859	9.24	131	-	NewsCrawl
Tamil	taml	2.7292892	397,206	4,216,313	10.61	1179	-	NewsCrawl
Telugu	telu	2.6198705	475,463	4,582,098	9.64	8092	-	NewsCrawl
Kyrgyz	cyrl	1.963557	410,075	5,154,290	12.57	4221	-	NewsCrawl
Turkish	latn	1.0444815	351,292	4,733,457	13.47	30076	-	NewsCrawl
Mongolian	cyrl	1.8046135	623,147	5,719,249	9.18	-	16,221	NLLB
Indonesian	latn	1.1788023	371,710	6,252,116	16.82	2785	-	NewsCrawl
Fusional								
Sanskrit	deva	2.5428913	857,298	5,159,696	6.02	16184	-	NLLB
Hindi	deva	2.3701629	427,015	6,936,484	16.24	1301	-	NewsCrawl
Sindhi	arab	1.5880165	614,055	7,258,014	11.82	3874	-	NLLB
Croatian	latn	0.9897218	309,090	5,652,252	18.29	7749	1,765,011	NewsCrawl
Russian	cyrl	1.8228284	341,500	5,296,132	15.51	21569	1,414,063	NewsCrawl
Persian	arab	1.6790492	444,478	7,197,903	16.19	11859	-	NewsCrawl
Analytic & Introflexive								
English	latn	1	250,000	6,249,150	25	3688	874,725	NewsCrawl
Swedish	latn	1.0210256	492,345	6,002,549	12.19	6223	140,843	NLLB
Hebrew	hebr	1.3555346	508,250	5,248,164	10.33	4641	-	NLLB

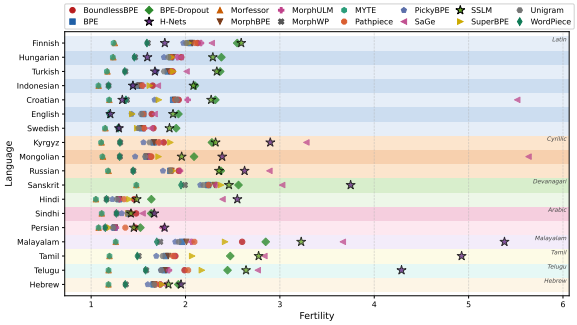
Table 12: Comparative Statistics after byte-premium (BP) (Arnett et al., 2024) adjustments and Dataset Sources for Languages Categorized by Morphological Typology. Abbreviations are BP: Byte-premiums, ASL: Average Sentence Length (in number of words), #MS: Number of items in MorphScore (Arnett and Bergen, 2025; Arnett et al., 2025), #MN: Number of items in MorphyNet (Batsuren et al., 2021), NLLB: No Language Left Behind (Team et al., 2022). "-" indicates the language is not covered in MorphyNet.

Method	Description
Standard Algorithms	
Byte-pair Encoding (BPE) (Gage, 1994; Sennrich et al., 2016)	Greedy frequency-based merge algorithm that iteratively combines the most frequent symbol pairs to construct a fixed subword vocabulary.
UnigramLM (Kudo, 2018)	Probabilistic subword model trained via EM that selects a vocabulary maximizing corpus likelihood under a mixture model.
WordPiece (Schuster and Nakajima, 2012)	Likelihood-based subword segmentation used in BERT; greedily selects merges that maximize training data likelihood.
Algorithmic Modifications	
SaGe (Yehezkel and Pinter, 2023)	Subword segmentation algorithm preferring units that occur in fewer distinct contexts, encouraging morphological coherence.
Morfessor (Smit et al., 2014)	Unsupervised morphology-inspired tokenizer that segments words into likely morpheme-like units using a probabilistic model of subword structure.
Inference Modifications	
BPE-dropout (Provilkov et al., 2020)	Subword regularization method that introduces stochasticity into the deterministic Byte Pair Encoding process by randomly dropping merge operations during training, allowing a model to observe multiple possible segmentations of the same word while remaining fully compatible with standard BPE at inference time.
PathPiece (Schmidt et al., 2024)	Lossless subword tokenizer that utilizes a directed acyclic graph (DAG) to find the shortest path of tokens from a given vocabulary, thereby segmenting a document into the minimum possible number of tokens.
Vocabulary Modifications	
PickyBPE (Chizhov et al., 2024)	Performs vocabulary refinement during training by using the Intersection over Self (IoS) metric to identify and remove intermediate "junk" tokens, thereby improving vocabulary efficiency and reducing under-trained tokens without compromising text compression.
Pre-tokenization Modifications	
MorphBPE (Banerjee and Bhattacharyya, 2018), MorphULM, MorphWP	Incorporates morphological boundary detection into the pre-tokenization stage prior to BPE learning, and similarly to UnigramLM (i.e., MorphULM) and WordPiece learning (i.e., MorphWP).
Encoding Modifications	
MYTE (Limisiewicz et al., 2024)	Alternative byte-level encoding replacing standard UTF-8 handling to better structure multilingual character representation.
Cross-token Algorithms	
BoundlessBPE (Schmidt et al., 2025)	Removes explicit word-boundary constraints, allowing merges across whitespace-defined token limits.
SuperBPE (Liu et al., 2025)	Extends BPE to permit cross-boundary merges and improved vocabulary utilization across token splits.

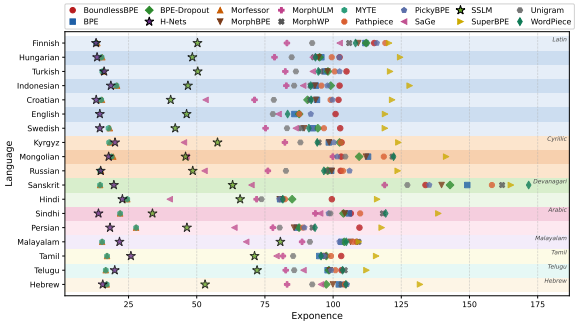
Table 13: List of Fixed-tokenizer approaches included in this study, organized by the stage at which they modify the tokenization pipeline.

Metric	Description	Requirements	What it Addresses
Coverage			
Subword Fertility	Average number of tokens produced per word.	Tokenizer and representative corpus.	Sequence length and basic storage efficiency.
Compression Rate	Ratio of raw text size to tokenized sequence length.	Tokenizer and representative corpus.	Storage and context window efficiency.
Token Frequency Distribution	Distributional profile of token occurrences (e.g., long-tail vs. concentrated usage).	Token counts from a representative corpus.	Statistical balance of vocabulary usage and fragmentation patterns.
Token Length Distribution	Distributional profile of tokens grouped by lengths from a representative corpus	Vocabulary of tokenizer	Granularity balance between short and long tokens, which affects efficiency and segmentation behavior.
Contextual Exponence	Number of distinct neighbours each token encounters (Yehezkel and Pinter, 2023). We consider only immediate neighbourhoods in our evaluation.	Representative corpus and its corresponding tokens	Degree to which the tokenizer optimizes tokens' contextual soundness
Effective Vocabulary Size	Total number of unique tokens available in the tokenizer vocabulary. For *tokenizer-free* approaches, we consider number of unique tokens when a corpus is segmented using it.	Trained tokenizer specification or model.	Memory footprint, embedding-table size, and granularity trade-offs.
Generalizability			
Type-Token Ratio (TTR)	Ratio of unique token types to total token occurrences in a corpus sample.	Tokenized corpus with fixed sample size (or moving-window normalization).	Lexical diversity and degree of repetition; helps compare whether a tokenizer yields broader vs. concentrated token usage.
Linguistic Alignment			
Morphological Alignment	Divergence between token segments and known morphemes.	Morphological analyzer and gold-standard data.	Linguistic fidelity and grammatical preservation.
Robustness			
Rényi Efficiency (Zouhar et al., 2023)	Information-theoretic metric penalizing skewed distributions.	Token frequency data from a corpus.	Statistical balance and information density.

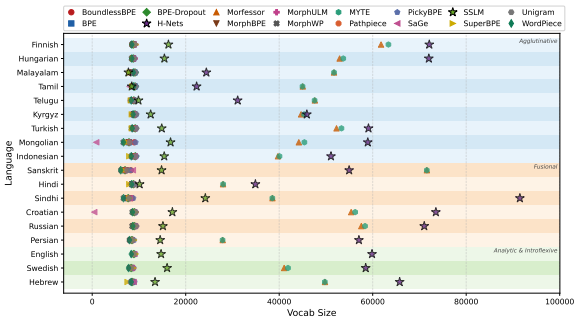
Table 14: Summary of Tokenizer Evaluation Metrics grouped by the proposed multidimensional framework in Alqahtani et al. (2026)



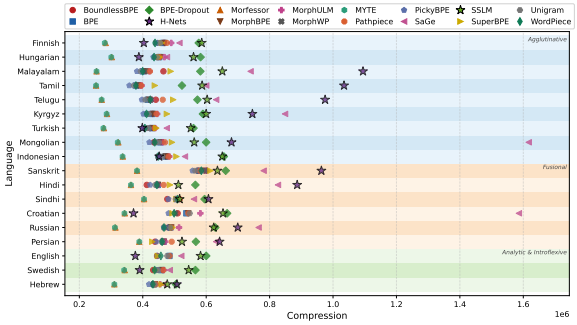
(a) Fertility by script



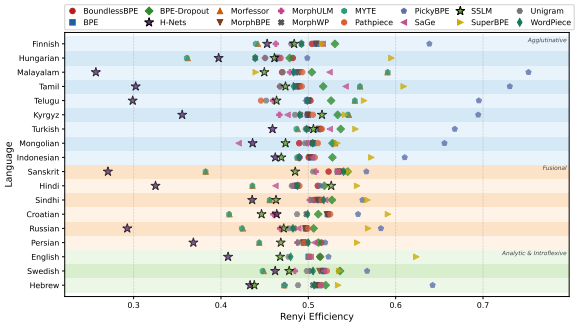
(b) Contextual exponence by script



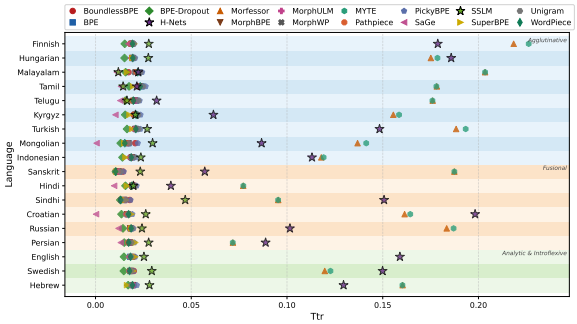
(c) Effective vocabulary size by typology



(d) Compression by typology



(e) Rényi efficiency by typology



(f) Type-token ratio by typology

Figure 10: Intrinsic properties of *tokenizer-free* approaches (H-Nets and SSLMs) compared to fixed-tokenizer approaches across scripts and language typologies.