Stop Taking Tokenizers for Granted: They Are Core Design Decisions in Large Language Models

Sawsan Alqahtani[†♠*] **Mir Tafseer Nayeem**[♣*]
Md Tahmid Rahman Laskar[♦♡] **Tasnim Mohiuddin**[◇] **M Saiful Bari**[♥]
[†]Princess Nourah Bint Abdulrahman University [♠]Saudi Data & AI Authority (SDAIA)
[♣]University of Alberta [♦]Dialpad [♡]York University
[◇]Qatar Computing Research Institute [♥]Amazon AGI
saalqhtani@pnu.edu.sa, mnayeem@ualberta.ca

Abstract

Tokenization underlies every large language model, yet it remains an under-theorized and inconsistently designed component. Common subword approaches such as Byte Pair Encoding (BPE) offer scalability but often misalign with linguistic structure, amplify bias, and waste capacity across languages and domains. This paper reframes tokenization as a core modeling decision rather than a preprocessing step. We argue for a context-aware framework that integrates tokenizer and model co-design, guided by linguistic, domain, and deployment considerations. Standardized evaluation and transparent reporting are essential to make tokenization choices accountable and comparable. Treating tokenization as a core design problem, not a technical afterthought, can yield language technologies that are fairer, more efficient, and more adaptable.

1 Introduction

Large Language Models (LLMs) have achieved remarkable success across tasks ranging from fluent text generation to advanced reasoning that now pushes the boundaries of agentic behavior (Laskar et al., 2023a; OpenAI, 2024; Grattafiori et al., 2024; Anil et al., 2025; Anthropic, 2025; DeepSeek-AI et al., 2025; Langford et al., 2025). Yet, amid the focus on architectural innovations, training methods, and alignment techniques, a fundamental component of LLM development has been overlooked: *tokenization*, the process of segmenting raw text into discrete units that serve as the interface between text and learned representations. This choice shapes a model's representation, efficiency, and generalization across tasks, languages, and domains (Ali et al., 2024; Goldman et al., 2024; Bommarito et al., 2025; Altıntaş et al., 2025).

The NLP community continues to rely on subword tokenization methods, particularly Byte Pair Encoding (BPE) (Gage, 1994; Sennrich et al., 2016), as the de facto standard for managing vocabulary and rare words. This reliance has produced uniform designs that mask trade-offs and ignore task-, language- or domain-specific needs. This fragmentation disrupts linguistic coherence[1] and undermines model performance on compositional tasks such as code generation, multi-step reasoning, and multi-hop question answering (Bostrom and Durrett, 2020; Ahia et al., 2023; Petrov et al., 2023; Dewangan et al., 2025; Ni et al., 2025).

These limitations have prompted a growing body of research to reexamine traditional tokenization practices, questioning long-held assumptions about the necessity of explicit segmentation and the impact of poorly aligned boundaries on model performance (Zouhar et al., 2023a; Goldman et al., 2024; Schmidt et al., 2024). Current practices also face methodological shortcomings: obscure design processes and poor documentation hinder reproducibility and adaptation to new domains and languages (Bostrom and Durrett, 2020; Dagan et al., 2024). These issues are compounded by the absence of standardized evaluation metrics and diagnostic tools, making it difficult to systematically assess tokenizer effectiveness (Beinborn and Pinter, 2023; Ali et al., 2024). Moreover, treating tokenization as a static, detached process prevents co-adaptation between segmentation and representation learning, limiting potential gains in efficiency and linguistic fidelity (Land and Bartolo, 2024; Zheng et al., 2024). Together, these gaps highlight that tokenization remains an open and undertheorized area in LLM development.

In practice, developers face consequential design decisions, including choosing token granularity, defining merge operations, and curating training corpora. As a result, many pipelines default to reusing pretrained tokenizers, a practice that often

[*]Equal contribution.

[1]See Appendix A

Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics
Volume 1: Long Papers, pages 8410–8432

misaligns with the linguistic characteristics or practical constraints of the target setting (Chen et al., 2023; Dagan et al., 2024; Nayeem and Rafiei, 2024; Seo et al., 2025; Bommarito et al., 2025). This mismatch reinforces inefficiencies and biases, highlighting the need for explicit, principled, context-aware decision-making around tokenizer auditing, reuse, adaptation, or retraining, grounded in systematic evaluation.

1.1 Our Perspective on Tokenization

In this paper, we argue that tokenization, despite its maturity, continues to be treated as an afterthought, implemented with insufficient scrutiny, contextual awareness, and methodological rigor. While subword tokenization was originally designed to balance the trade-offs between word- and character-level approaches, its reuse should not exempt it from renewed examination. We advocate for a shift from ad-hoc practices to a principled, context-aware design and evaluation process, aligned with model architecture and training objectives, elevating tokenization to an essential component of model development. We emphasize that this perspective neither assumes nor prioritizes retraining LLMs from scratch. Such retraining is often infeasible for many practitioners and language communities, particularly in low-resource settings, and even when feasible, may be unnecessary or suboptimal relative to principled reuse or adaptation.

Our position is supported by Dewangan et al. (2025), who show that even with fixed vocabulary sizes, optimizing the segmentation strategy alone can yield substantial gains. Yet, despite widespread reliance on standard tokenizers, the field lacks a unified and theoretically grounded approach to its selection, adaptation, and evaluation. To bridge this gap, we advocate a principled framework that integrates tokenizer–model co-design, standardized evaluation, and transparent documentation.

As illustrated in Figure 1, such a shift replaces rigid practices with a more adaptive and equitable foundation for language technology. Our paper advances this agenda through three contributions:

- Identifying the hidden design choices in standard tokenization: vocabulary construction, granularity, corpus selection, and reuse.
- Proposing a structured tokenizer-design framework that accounts for linguistic diversity, domain specificity, and task complexity.
- Defining evaluation dimensions for fairness, representational alignment, and coverage.

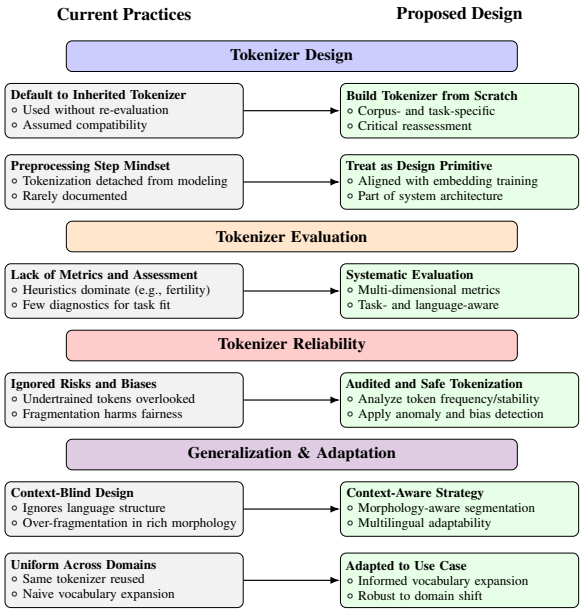

Figure 1: Thematic comparison of current practices and proposed design principles for tokenizer development.

By challenging the view of tokenization as an afterthought, we reposition it as a foundational area of empirical inquiry, essential for building language models that are not only efficient but also adaptable and equitable across domains and languages.

1.2 Alternative Perspectives on Tokenization

Despite growing skepticism toward the *one-size-fits-all* approach, several views continue to shape current tokenization practices (Schmidt et al., 2024; Reddy et al., 2025; Wegmann et al., 2025). We summarize and respond to five common perspectives, outlined in Table 1.

2 Tokenization and Alternatives

2.1 Subword Tokenization Methods

Subword tokenization addresses the drawbacks of word- and character-level models. Word-level approaches suffer from large vocabularies and out-of-vocabulary issues, while character-level models create long sequences that raise computational costs and obscure linguistic structure (Beinborn and Pinter, 2023). Subword methods (e.g., BPE, WordPiece, Unigram) balance these extremes by segmenting text into frequent, data-driven units, enabling efficient handling of rare words and manageable sequence lengths (Wolleb et al., 2023).

These methods differ in segmentation strategy.

[2] See Appendix B for examples from leading LLMs.

View 1 — Subword Tokenization Is Sufficient and Scalable	
Claim	Subword methods are considered a practical default, offering scalability, compression, and stable training. Their success in general-purpose models has led to the assumption that changing the tokenizer has a minimal impact on LLMs (Jimenez Gutierrez et al., 2023; Schmidt et al., 2024).
Response	This view downplays the importance of tokenization. In morphologically rich, domain-specific, or code-switched settings, subword tokenization can fragment meaningful units and degrade performance. Yet, tokenizers are often reused with minimal justification, hindering reproducibility. While prior work (e.g., Schmidt et al. 2024) reports minor performance differences across tokenizers, Schmidt et al. (2025) suggest this stems from high token overlap, where over 90% of common words are represented as single tokens, limiting variation to a small input fraction and diminishing any consistent advantage. Moreover, even large models suffer from inefficiencies due to poor tokenization, such as inflated sequence lengths.[2]
View 2 — Current Evaluation Metrics Accurately Reflect Tokenizer Quality	
Claim	In the context of LLMs, tokenization is often deprioritized because existing evaluations suggest minimal performance differences across tokenizers. It is assumed that model size and contextual depth compensate for poor segmentation, making tokenizer optimization unnecessary.
Response	This view stems from shallow evaluation practices that fail to capture tokenizer quality. Metrics like vocabulary size, fertility, and compression rate are easy to compute but poorly reflect linguistic alignment or downstream utility (§3.2.4). They often obscure issues such as over-fragmentation and poor generalization, especially in multilingual or domain-specific contexts, where similar fertility scores can mask divergent morphological behavior.
View 3 — Byte-Level Tokenization Removes the Need for Explicit Design	
Claim	This argument has led to claim that explicit tokenizer design is becoming obsolete (Feher et al., 2025). These methods promise universality, avoiding the need for language-specific preprocessing, and simplify the training pipeline by operating on raw text bytes.
Response	While appealing in their simplicity and universality, byte-level approaches do not eliminate tokenizer design so much as relocate it. By operating over raw bytes, they shift the burden of discovering linguistic, morphological, and semantic structure entirely onto the model. This shift introduces new trade-offs: substantially longer sequences, increased training and inference overhead under standard attention, and greater reliance on model capacity and inductive bias to recover structure (Moon et al., 2025). Although emerging architectures such as linear attention and patch-based processing may mitigate some of these costs, current evidence suggests that sequence length and computational overhead remain significant concerns in practical deployments (Pagnoni et al., 2024).
View 4 — Token-Free or Continuous Representations Will Replace Tokenization	
Claim	Emerging "token-free" paradigms argue that tokenization is an artificial bottleneck inherited from discrete symbolic modeling (Pagnoni et al., 2024; Hwang et al., 2025; Neitemeier et al., 2025). Such models aspire to bypass tokenization altogether, potentially enabling end-to-end differentiable pipelines and more fine-grained representational learning.
Response	While token-free modeling is a promising frontier, it is far from a resolved alternative. Continuous-input systems face challenges in scalability, interpretability, and transferability, particularly when training at web-scale. Moreover, the absence of discrete boundaries complicates linguistic analysis and data processing pipelines, which still rely heavily on symbolic granularity.
View 5 — Tokenizer–Model Joint Optimization Enables True Co-Design	
Claim	Recent work argues that genuine tokenizer–model co-design demands architectural coupling, treating the tokenizer as a learnable module co-optimized with model parameters (Hiraoka et al., 2021). In this view, token boundaries and vocabulary adapt dynamically to representational demands, aligning linguistic structure with learned embeddings to improve efficiency and adaptability.
Response	While conceptually appealing, integrating tokenization into the training loop poses challenges of stability, efficiency, and interpretability. Joint optimization risks entangling representation learning with unstable segmentation dynamics.

Table 1: Five alternative perspectives on tokenizer design and our responses to each.

BPE and WordPiece are merge-based: BPE joins frequent adjacent pairs by frequency, while WordPiece optimizes a likelihood-based objective tied to language modeling (Schmidt et al., 2024). Both yield deterministic outputs. In contrast, Unigram takes a probabilistic approach, pruning subwords with low data likelihood and allowing multiple valid segmentations. This often makes Unigram more flexible and linguistically aligned, especially for multilingual or morphologically rich languages (Kudo, 2018; Bostrom and Durrett, 2020).

BPE offers simplicity and speed, WordPiece emphasizes likelihood-based merge decisions informed by local token context, and Unigram provides cross-lingual adaptability at higher computational cost. These methods underpin models like GPT (Brown et al., 2020), BERT (Devlin et al., 2019), and LLaMA (Touvron et al., 2023a,b). Yet their reliance on frequency statistics exposes key limits (Meyer and Buys, 2022; Liu et al., 2024a; Chai et al., 2024) frequent forms dominate, while rare words, low-resource languages, and domain-specific terms are poorly captured. Token boundaries often mirror statistical artifacts rather than linguistic structure, impairing generalization in typologically complex languages (see Appendix A). Despite scalability, these methods often underperform in compression efficiency and cross-lingual representational alignment (Goldman et al., 2024).

2.2 Emerging Alternatives for the De Facto Tokenization Paradigm

Tokenization based on statistical frequency over large corpora has long dominated NLP workflows, but it struggles with rare languages, domain-specific terms, and structural linguistic nuances (Meyer and Buys, 2022; Liu et al., 2024a; Chai et al., 2024). In response, the research community has proposed several alternatives that challenge the subword-centric status quo. Without endorsing

a particular solution, we review these approaches to underscore a broader shift toward more adaptive, linguistically informed strategies and to highlight open questions that remain.

Learning Compositional Structure Explicitly: Traditional tokenization ignores linguistic composition, treating subwords as independent units (Bostrom and Durrett, 2020). Recent methods attempt to model internal structure more directly through sparse embeddings (Deiseroth et al., 2024) or morpheme-aware GRU frameworks (Singh et al., 2023), but these approaches add training complexity and remain underexplored in large-scale LLMs.

From Subwords to Raw Bytes: Subword tokenizers struggle with unseen characters and typographic variation (Xue et al., 2022). Byte-level methods address this by operating directly on raw input, avoiding fixed vocabularies. The Byte-Level Transformer (BLT) (Pagnoni et al., 2024) employs entropy-based patches to reduce inference cost and remove vocabulary constraints. While promising, these methods can increase sequence length and computational overhead.

Modeling Language at Semantic Level: Recent work explores higher-level abstractions by representing multi-word expressions or concepts as single units (Liu et al., 2025; Schmidt et al., 2025). Liu et al. (2024b) propose a dynamic phrase encoder with variable-length phrases to enhance generalization, while LCM team et al. (2024) introduce sentence-level embeddings that bypass token-level processing and view language as a hierarchy of meaning (Shao et al., 2025), benefiting multilingual tasks but facing data sparsity challenges.

3 Rethinking Tokenizer Design: From Pitfalls to Practice

Building on prior findings that highlight tokenization's influence on model behavior (Bostrom and Durrett, 2020; Ali et al., 2024), we identify four persistent pitfalls in current practice.

Design Integration: Tokenizer design should be integrated into model training, not inherited as a fixed preprocessing artifact. Reliance on legacy vocabularies, often optimized for different data or languages, limits control and reproducibility. Effective design begins with a core choice: train a tokenizer from scratch, or reuse an existing one for efficiency and interoperability (§3.2.1, §3.2.2).

Adaptation and Generalization: Tokenizers are frequently reused across domains without adaptation, resulting in representational gaps and inflated sequence lengths. As discussed in §3.2.2, domain- and language-aware tokenization is essential for fair and robust generalization across linguistic and structural variation.

Evaluation Practices, Reliability, and Bias: Tokenizer evaluation often relies on surface metrics such as fertility, which correlate weakly with performance (Zouhar et al., 2023a; Ali et al., 2024). As detailed in §3.2.4, systematic assessment across linguistic fit, task alignment, and cross-lingual robustness is needed. Current workflows overlook token fragmentation, undertrained embeddings, and biases against morphologically rich or low-resource languages. Responsible tokenization requires auditable workflows, including anomaly detection and vocabulary stability checks (§3.2.3).

3.1 A Structured Iterative Process

An effective tokenizer should meet empirically grounded desiderata, ensuring that vocabulary units occur frequently enough during training to support robust learning and practical utility:

Tokenizer Design Desiderata

- **Linguistically Faithful:** mirrors structure, morphology, syntax, and code-switching patterns.
- **Usage-Aligned:** matched to intended language coverage, domains, and downstream usage profiles.
- **Deployment-Ready:** efficient under latency and memory limits.
- **Architecturally Compatible:** fits model architecture and embeddings.

To move beyond ad hoc design, we propose a structured, iterative co-design process in which tokenization is periodically re-evaluated and refined alongside model architecture and training objectives. Tokenization is informed indirectly by model behavior, using representational analyses and downstream performance as feedback. The workflow integrates linguistic analysis and empirical evaluation, with each stage refining earlier choices to maintain coherence across linguistic, operational, and deployment constraints. Table 2 summarizes the core stages and outputs. While this process offers a practical roadmap, success hinges on how language is segmented and represented. The goal is not to over-specialize tokenization to a particular benchmark, but to avoid blindly inheriting a tokenizer misaligned with the target language or domain (Nayeem et al., 2025).

Stage	Key Activities	Expected Output	Illustrative Example
Initial Assessment	**(i)** Define the intended LLM purpose (e.g., general or domain-specific). **(ii)** Identify target languages, dialects, and cultural contexts. **(iii)** Specify required capabilities (e.g., reasoning). **(iv)** Establish model constraints (architecture, memory, latency).	A clear set of design objectives and operational constraints to guide tokenizer and vocabulary development.	Designing a biomedical-domain LLM focusing on English and Spanish medical notes, with strict latency constraints for on-device clinical use.
Reuse Decisions	**(i)** Identify existing tokenizers with appropriate licenses. **(ii)** Evaluate against linguistic fidelity, task alignment, and compatibility. **(iii)** Run diagnostics (e.g., vocabulary overlap, token length distribution). **(iv)** Decide to reuse without changes, adapt, or create a new tokenizer.	A diagnostic report and an evidence-based decision to reuse, adapt, or develop a new tokenizer.	Comparing GPT-2 and LLaMA tokenizers for biomedical abstracts, identifying the vocabulary overlap, and deciding to train a new tokenizer for domain-specific terminology.
Corpus Curation	**(i)** Build corpora aligned with the intended use case. **(ii)** Establish diagnostic criteria (coverage, diversity, task relevance).	A representative, task-aligned corpus.	Curating a multilingual biomedical corpus of clinical trial reports, medical guidelines, and radiology notes.
Pretokenization & Processing	**(i)** Apply pretokenization rules and preprocessing steps. **(ii)** Normalize encoding, clean text, set boundary rules. **(iii)** Handle code-switching, numbers, and mixed inputs. **(iv)** Align with application requirements.	A documented, reproducible set of pretokenization rules applied consistently.	Normalizing biomedical text by handling hyphenated terms (e.g., "HER-2/neu"), gene names, and units of measurement.
Tokenizer Training	**(i)** Select a segmentation strategy. **(ii)** Train tokenizer on the curated corpus to produce vocabulary.	A representative, task-aligned corpus and a trained tokenizer with an initial vocabulary.	Training a 60K-token BPE tokenizer on 25 GB of biomedical literature to capture domain-specific compounds (e.g., "immunohistochemistry").
Intrinsic & Extrinsic Evaluation	**(i)** Apply tokenizer to varied datasets. **(ii)** Compute metrics (e.g., token length, fragmentation). **(iii)** Audit vocabulary and visualize anomalies. **(iv)** Evaluate edge cases and small model performance.	A comprehensive evaluation report covering tokenizer-level diagnostics and model-based validation.	Measuring token length across English and Spanish biomedical texts; identifying over-segmentation in long clinical terms.
Iteration & Model-Guided Refinement	**(i)** Refine tokenizer using diagnostics. **(ii)** Treat tokenizer as co-evolving with the model. **(iii)** Re-run evaluations until stable performance.	A refined, model-aligned tokenizer with improved efficiency and quality.	After observing poor tokenization of biomedical abbreviations, re-merging tokens to preserve critical acronyms like "MRI" and "ICU."
Final Integration & Documentation	**(i)** Confirm stability and finalize configuration. **(ii)** Document known limitations and design rationale. **(iii)** Archive and publish tokenizer for reproducibility.	A production-ready tokenizer with comprehensive documentation.	Publishing biomedical tokenizer specifications with open access, including examples for drug names and clinical abbreviations.

Table 2: Core Stages in Tokenizer Development Process, extended with illustrative examples.

3.2 Critical Decisions within the Workflow

Each of the pitfalls identified earlier maps to design decisions that shape tokenizer quality. We unpack these decisions, showing how principled choices throughout tokenizer development can systematically address the deficiencies outlined above. Each subsection concludes with guiding questions, consolidated in Table 7 in the Appendix.

3.2.1 Training Tokenizers from Scratch

Designing a tokenizer from scratch enables deliberate control over segmentation, corpus, vocabulary size, and preprocessing, aligning it with linguistic, functional, and deployment needs (Dagan et al., 2024). In contrast, reusing a pretrained model's tokenizer (e.g., LLaMA) constrains these choices, often limiting performance and generalization. This section outlines key design factors and open questions for principled tokenizer development.

Choosing the Right Tokenizer Algorithm. Selecting a tokenization algorithm is a crucial early step but often defaults to convenience over task or language needs (Ali et al., 2024). Unigram better captures morpheme-level structure in agglutinative languages (Bostrom and Durrett, 2020), while frequency-based BPE remains dominant. These choices influence representation quality, sequence length, and generalization (Ahia et al., 2023). Recent work shows that tokenizers tailored to linguistic and operational contexts outperform generic ones, especially in domain-specific and multilingual settings (Bommarito et al., 2025). Evidence also suggests that multilingual tokenizers preserve morphology better in complex languages than in simpler ones (Arnett and Bergen, 2025), though the benefits of morphologically aligned tokenization remain debated (Arnett et al., 2025).

Training Data Matters for Tokenizer Effectiveness. Training corpus composition critically shapes tokenizer performance (Reddy et al., 2025). Reliance on large-scale web crawls introduces representational biases, particularly for low-resource languages and specialized domains, where vocabularies built from unrelated or noisy data degrade performance (Rust et al., 2021; Goldman et al., 2024). Yet metrics for corpus quality remain limited. Empirical studies show that compression quality, driven by corpus composition and segmentation strategy, correlates strongly with model utility, especially for rare or domain-specific terms (Goldman et al., 2024). Evidence also suggests tokenizer quality plateaus beyond roughly 150–180GB of

training data, indicating diminishing returns from further scaling (Reddy et al., 2025).

Pre-tokenization and Normalization Are Foundational. Pre-tokenization and normalization encode linguistic assumptions that shape token consistency, sequence length, and vocabulary fragmentation (Dagan et al., 2024). Even minor choices (e.g., Unicode normalization or diacritic removal) can markedly alter segmentation.[3] They also guide subword merges via boundary cues: incorporating multiword expressions or local context yields more coherent merges, especially in agglutinative languages, often requiring BPE adjustments (Schmidt et al., 2024). Removing pre-tokenization consistently degrades performance (Dagan et al., 2024; Velayuthan and Sarveswaran, 2025), sometimes more than changing vocabulary size or corpus composition (Wegmann et al., 2025), and can create indivisible units that limit gains from larger corpora (Reddy et al., 2025).

Vocabulary Size Is a Strategic Design Choice. Vocabulary size should be co-optimized with model architecture and training goals. Larger vocabularies reduce fragmentation and sequence length but raise memory and embedding costs (Tao et al., 2024). These trade-offs intensify in multilingual or morphologically rich settings (Ali et al., 2024). Studies show that scaling vocabulary can improve compression and efficiency, though gains depend on corpus, task, and model capacity (Tao et al., 2024; Huang et al., 2025). Yet, Schmidt et al. (2025) warns that adding low-frequency tokens may increase redundancy with minimal benefit. Despite this, vocabulary size is often set heuristically (Salesky et al., 2020; Gowda and May, 2020). A more principled approach would treat it as a tunable parameter informed by token frequencies, learning dynamics, and compute limits, though key questions remain.

Open Trade-offs. Table 3 summarizes approaches that learn tokenizers directly from raw text instead of reusing legacy vocabularies (Sennrich et al., 2016; Kudo, 2018; Kudo and Richardson, 2018; Tay et al., 2022). When data and budget are controllable, first-principles design is often preferable (Gowda and May, 2020; Zouhar et al., 2023b). Vocabulary size should be treated as a computational lever, not a sign of model "purity": smaller vocabularies inflate sequence length and attention cost, while larger ones increase parameters and memory with diminishing returns (Gowda and May, 2020; Zouhar et al., 2023b). Classic subword methods (BPE, WordPiece, Unigram) remain strong baselines for minimizing tokens per sample (Sennrich et al., 2016; Kudo, 2018; Kudo and Richardson, 2018). Morphology-aware or lexically grounded variants are worthwhile for agglutinative or templatic languages (Libovický and Helcl, 2024; Chizhov et al., 2024; Asgari et al., 2025), while block-level schemes such as Charformer downsampling retain character-level robustness without excessive sequence inflation (Tay et al., 2022).

[3] E.g., normalizing "coöperate" to "cooperate" avoids inconsistent splits in OCR or historical texts.

3.2.2 Adapting Tokenizers: Reuse Requires Verification and Documentation

Reuse of mature, pretrained tokenizers is often a reasonable and practical default, particularly when compatibility, distillation from existing models, and engineering efficiency are primary concerns. Whether a pretrained tokenizer can serve as-is or requires adaptation depends on its alignment with the target language or domain. When LLMs are extended to new settings, work typically centers on weight fine-tuning, yet performance often degrades because English-optimized tokenizers fail to capture domain terms and morphology in low-resource contexts (Sha et al., 2025). Vocabulary expansion is a common but ad hoc remedy, with limited understanding of when it helps (Wang et al., 2019; Kiulian et al., 2024), often yielding short-term gains that fail to generalize.

Robust adaptation instead requires treating tokenization as a core design task, guided by systematic protocols for token addition, embedding initialization, and adaptation staging. While reuse improves compatibility and cost, unexamined reuse can introduce long-term drawbacks, including loss of linguistic fidelity, coverage gaps, and fairness issues, particularly when the tokenizer's original training data or design assumptions diverge from the new setting. Crucially, although training or adapting a tokenizer incurs a one-time engineering cost, inheriting a suboptimal tokenizer imposes a persistent inference-time penalty: inflated sequence lengths increase latency and compute per query over the lifetime of a deployed model, especially in non-English and domain-specific scenarios (see Appendix A).

Base Model Choice: *Tokenizer Compatibility Is Not Guaranteed.* Adaptation often starts with

choosing a base LLM and reusing its tokenizer by default, yet the suitability of this choice for a new domain or language is rarely explicitly tested or documented. Even when vocabularies are expanded, such as extending the LLaMA 2 tokenizer from 32K to 61K tokens for Arabic (Bari et al., 2025), the linguistic rationale behind token boundaries remains unclear. Without evaluating what the original tokenizer captures or omits, expansion risks becoming a workaround.

Vocabulary Size: ***Incremental Growth Without Clear Guidance.*** In adaptation, expanding vocabulary means adding new tokens to an existing tokenizer. This can shorten sequences and improve representation for new domains or languages, but also raises issues of compatibility, memory, and convergence (Tejaswi et al., 2024). Unlike tokenizers built from scratch, expansion operates within the constraints of an inherited scheme. Evidence suggests that targeted additions (e.g., 5–10K tokens) yield better returns than aggressive scaling (Liu et al., 2024a), yet consistent methods for selecting, validating, and integrating tokens remain lacking. Without a grounded approach, expansions risk inefficiency and misalignment with the base model's learned representations. Because expansion preserves the original token inventory, it is often preferred in settings where distillation from an existing model, checkpoint reuse, or interoperability with deployed systems is required.

Initialization: ***A Crucial but Understudied Factor.*** How new token embeddings are initialized strongly affects downstream performance. Approaches include random initialization, subtoken averaging, and using external word vectors (Yao et al., 2021; Dobler and de Melo, 2023), but comparative evidence remains limited and context-dependent. Stopping criteria such as fertility scores are often borrowed from unrelated tasks and rarely validated across languages. The field needs benchmarks and reporting standards to identify which strategies generalize and under what conditions.

Embedding Warm-Up: ***A Temporary Fix or Best Practice?*** Embedding warm-up (training new token embeddings in isolation before unfreezing the full model) has emerged as a practical compromise (Zhao et al., 2024). Some workflows add phased unfreezing of input or output layers (Kim et al., 2024), but these remain empirical heuristics rather than principled frameworks. Without systematic evaluation, it remains unclear whether such strategies are robust solutions or fragile stopgaps that conceal deeper misalignments.

Open Trade-offs. Table 4 summarizes tokenizer adaptations across languages and domains. Blind reuse of legacy tokenizers often inflates sequence length through over-fragmentation, wasting context and compute. In contrast, retokenization, vocabulary expansion, and merge refinement are offline interventions that typically repay their cost with shorter sequences and better context use (Rust et al., 2021; Yehezkel and Pinter, 2023; Chizhov et al., 2024). Large-scale studies show that tokenizer choice materially affects both quality and training cost, and that naïve reuse can increase training steps across languages (Ali et al., 2024; Schmidt et al., 2024). Adaptations in Arabic and Southeast Asian settings demonstrate that targeted expansions or dedicated tokenizers close token-length gaps while preserving English performance (Nguyen et al., 2023; Bari et al., 2025; Fanar et al., 2025). For unseen scripts, compact in-language tokenizers with aligned embeddings reduce UNKs and avoid costly character fallback (Pfeiffer et al., 2021). Taken together, these findings suggest that reuse should be treated as a default hypothesis to be tested, rather than an unquestioned assumption.

3.2.3 Responsible Tokenization: Confronting Risks to Fairness, Stability, and Security

Most tokenizer discussions emphasize performance, but robust and responsible design is equally critical. Tokenization influences fairness, stability, and security, shaping language coverage, equity, and susceptibility to attacks. Fairness refers to avoiding unequal segmentation quality, representational accuracy, or computational burden across languages, scripts, or demographic groups, ensuring parity in token lengths and downstream performance. Stability refers to how consistently and reliably a tokenizer segments text across noise, scripts, and domains, while avoiding low-quality or undertrained units. As LLMs enter real-world deployment (Laskar et al., 2023b; Fu et al., 2024; Nayeem and Rafiei, 2025), tokenization must be treated as a potential source of bias for transparent design.

Undertrained Tokens: ***Silent Failures in the Vocabulary.*** Undertrained tokens, units that occur infrequently in the pretraining corpus, receive too few updates during training. They often stem from rare words, domain-specific terms, or overly frag-

mented vocabularies. Their effects are subtle but harmful, causing instability and poor generalization (Yang et al., 2024). The notorious SolidGoldMagikarp token, which emerged in GPT models due to skewed data distributions, illustrates how poorly trained tokens can trigger erratic behavior (Rumbelow and Watkins, 2024). Yet such tokens are seldom identified. Embedding audits, token pruning, vocabulary refinement, and frequency analysis should be standard in tokenizer evaluation (Chizhov et al., 2024; Cognetta et al., 2024a; Bauwens and Delobelle, 2024; Lian et al., 2025).

Anomalous Tokens: ***Artifacts That Disrupt Learning.*** Tokenizer vocabularies often contain anomalous tokens, units introduced by data noise, encoding errors, or preprocessing inconsistencies. These may capture HTML fragments, URL encodings, or control characters that add no linguistic value and instead degrade embedding quality (Jiang et al., 2024; Chai et al., 2024). Their presence injects noise into training, causing unpredictable behavior in generation and classification. More broadly, such artifacts pose challenges for reproducibility, auditing, and verification across machine learning pipelines, where undocumented preprocessing decisions can silently affect downstream behavior. While this lack of validation is a broader issue in model reproduction and auditing, tokenizers are especially affected because vocabulary artifacts become fixed and propagate throughout training and deployment. Clearer standards for corpus preparation, token quality control, and automated anomaly detection are therefore essential not only for tokenizer quality, but for reproducible, auditable, and trustworthy model development.

Tokenization Bias: ***Unequal Representation by Design.*** Tokenization may not treat all languages or identity groups equally. Morphologically rich languages often suffer excessive fragmentation, lengthening sequences and degrading performance (Toraman et al., 2023) (see Appendix A). Similarly, identity-related terms from underrepresented regions are frequently over-segmented, impairing tasks like named entity recognition (Ahia et al., 2024). These are not random artifacts but outcomes of design choices that allocate vocabulary space unequally across languages and demographics. Responsible tokenizer development must therefore ensure balanced language representation, introduce fairness constraints, and systematically analyze segmentation across diverse groups. Without such efforts, tokenization remains a structural source of bias in downstream applications.

Security and Adversarial Risks: ***A Hidden Attack Surface.*** Tokenizer design can introduce overlooked security vulnerabilities. Adversarial token sequences, prompt injection, and homograph attacks exploit weaknesses in tokenization rules and preprocessing (Wang et al., 2024; Jang et al., 2025). Poor handling of visually similar characters across scripts further enables such exploits. Security should not be an afterthought: tokenization must be included in adversarial testing, with controlled preprocessing, careful treatment of special tokens, and rigorous auditing and versioning. As LLMs enter sensitive domains, securing tokenization is essential for both performance and trust.

Open Trade-offs. As shown in Table 5, tokenizer design is a governance decision, not incidental plumbing. It determines sequence length, training cost, and robustness to noise and cross-language parity (Rust et al., 2021; Petrov et al., 2023; Ahia et al., 2023). Minor orthographic noise or typos can destabilize segmentation (Chai et al., 2024), while morphology-aware constraints and merge refinement improve stability and parity with minimal runtime overhead (Chizhov et al., 2024; Asgari et al., 2025). Recent studies show that targeted adaptations reduce token-length gaps in Southeast Asian and Arabic settings without harming English performance (Nguyen et al., 2023; Bari et al., 2025).

3.2.4 Evaluation Framework and Gaps

Despite tokenization's central role in model performance, evaluation remains limited and fragmented. Most work relies on surface metrics (compression rate, vocabulary size, fertility) that are easy to compute but reveal little about linguistic alignment or generalizable learning (Ali et al., 2024; Laskar et al., 2024; Uzan et al., 2024; Nayeem et al., 2025). This gap is most pronounced in multilingual and domain-specific settings, where similar fertility scores can hide large differences in morphological preservation or rare-construction handling (Bostrom and Durrett, 2020). These metrics are also sensitive to corpus composition, language typology, and preprocessing, limiting cross-domain validity (Rust et al., 2021). Without a principled evaluation framework, trade-offs among linguistic fidelity, efficiency, and robustness remain poorly understood. Tokenization choices are seldom tied to downstream performance (Dagan et al., 2024).

Notably, many of these diagnostics (such as token-per-character statistics by language, fragmentation analysis, anomaly detection, and undertrained-token audits) can be applied directly to existing pretrained models without retraining, making them practical even under tight compute or data constraints. We do not argue that tokenizer evaluation requires training large models to convergence, but instead advocate low-cost intrinsic and model-based probes (e.g., token-per-character parity and information-theoretic efficiency (Tsvetkov and Kipnis, 2024)) that correlate with downstream multilingual performance.

Toward Multi-Dimensional Evaluation. Recent work has questioned the validity of common intrinsic metrics. Early studies hypothesized that reducing corpus token counts (CTC) would improve downstream performance (Gallé, 2019; Goldman et al., 2024), but later findings show no consistent correlation between CTC, vocabulary size, and task accuracy (Schmidt et al., 2024; Ali et al., 2024). In response, researchers have proposed richer measures of token quality. Zouhar et al. (2023a) introduce Rényi efficiency, which penalizes distributions skewed toward frequent, low-content tokens and correlates more strongly with downstream results, though its generality remains debated (Cognetta et al., 2024b). Other studies note that standard pre-tokenization pipelines often over-map pretokens to high-frequency words, reducing representational depth (Reddy et al., 2025). Building on these efforts, Bommarito et al. (2025) propose a framework encompassing token-per-character efficiency, domain coverage, and token size distribution. Collectively, these approaches point to a multidimensional view of tokenizer quality that integrates linguistic structure, statistical balance, and utility. ***We distill this view into five core dimensions*** (see Appendix C for examples). Each dimension emphasizes different evaluation levels: corpus-level diagnostics for fragmentation and alignment, and model-based probes for generalization and embedding use. Efficiency often trades off with alignment or interpretability, and priorities vary across applications. Developing benchmarks to quantify and balance these trade-offs remains an essential direction for future work.

Toward Practical Evaluation Methodologies. Translating these dimensions into practice requires consistent methodologies. Promising directions include unified corpus diagnostics, model-level robustness tests under distribution shift, and cross-lingual benchmarks linking tokenization quality to downstream performance. Standardized protocols would enable reproducibility and move the field beyond ad hoc, dataset-specific evaluation.

Key Evaluation Dimensions for Tokenizer Quality

- **Coverage:** Does the tokenizer represent the linguistic units relevant to the target domain or language?
- **Generalizability:** How well does it handle unseen forms, rare constructions, and morphologically complex variants?
- **Linguistic Alignment:** Do token boundaries align with morphemes, affixes, or syntax?
- **Robustness:** Is it resilient to variation, noise, and distributional shift (e.g., dialects, OCR errors)?
- **Representation Utilization:** Are embeddings meaningfully activated during inference, or is much of the vocabulary underused?

Open Trade-offs. As summarized in Table 6, evaluating tokenizers by only dev-set perplexity or vocabulary size obscures the real drivers of cost and quality. Report metrics that matter like tokens per 1k characters by language, parity ratios, compression–structure balance, and estimated pretrain FLOPs at fixed data budgets (Goldman et al., 2024; Ali et al., 2024; Schmidt et al., 2024). Performance should be summarized across all data categories and shards used for training. Intrinsic metrics can triage candidates but need quick downstream probes to prevent gaming and preserve structural integrity (Zouhar et al., 2023a; Cognetta et al., 2024b; Uzan et al., 2024). Formal analyses and sizing analyses (explicit evaluations of how vocabulary size, sequence length, and token granularity affect memory use, throughput, and total training FLOPs) offer practical standards for selecting tokenizers that minimize total cost while maintaining target quality across languages (Gowda and May, 2020; Zouhar et al., 2023b; Nayeem et al., 2025).

4 Conclusion

The dominance of subword tokenization has fostered a one-size-fits-all mindset. Yet evidence from multilingual, low-resource, and domain-specific settings exposes its limits. Tokenization should be reimagined as a context-aware design choice, shaped by structure, data, task, and deployment constraints. This shift is vital for models that are efficient, scalable, fair, and linguistically grounded. Tokenization is not a preprocessing afterthought but a core component of model development, designed, evaluated, and refined alongside model behavior.

Limitations

Our argument centers on reframing tokenization as a modeling decision, but it remains primarily conceptual. We do not provide new empirical results or large-scale ablations, and thus our claims should be viewed as hypotheses for future validation and research. Implementing our proposed co-design and evaluation framework also assumes access to multilingual data, compute, and tooling that may not be universally available. Moreover, identifying standardized proxy probes that robustly predict large-scale downstream performance remains an open empirical problem, and we view this as a key direction for the future works. Finally, while we gesture toward token-free and byte-level approaches, these alternatives evolve too quickly for comprehensive inclusion here; our discussion focuses on guiding principles rather than definitive solutions.

Acknowledgements

We thank all the anonymous reviewers and the meta-reviewer for their valuable feedback and constructive suggestions for improving this work. Additionally, Mir Tafseer Nayeem acknowledges support from the Huawei PhD Fellowship, and Md Tahmid Rahman Laskar acknowledges the support from the CUPE 3903 Research Grant Fund, York University.

References

Orevaoghene Ahia, Sachin Kumar, Hila Gonen, Valentin Hofmann, Tomasz Limisiewicz, Yulia Tsvetkov, and Noah A. Smith. 2024. MAGNET: Improving the multilingual fairness of language models with adaptive gradient-based tokenization. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Orevaoghene Ahia, Sachin Kumar, Hila Gonen, Jungo Kasai, David Mortensen, Noah Smith, and Yulia Tsvetkov. 2023. Do all languages cost the same? tokenization in the era of commercial language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9904–9923, 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.

Gül Sena Altıntaş, Malikeh Ehghaghi, Brian Lester, Fengyuan Liu, Wanru Zhao, Marco Ciccone, and Colin Raffel. 2025. Toksuite: Measuring the impact of tokenizer choice on language model behavior. *arXiv preprint arXiv:2512.20757*.

Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy Lillicrap, Angeliki Lazaridou, and 1 others. 2025. Gemini: A family of highly capable multimodal models. *Preprint*, arXiv:2312.11805.

Anthropic. 2025. Claude 3.7 sonnet system card. https://assets.anthropic.com/m/785e231869ea8b3b/original/claude-3-7-sonnet-system-card.pdf. Accessed: 2025-05-22.

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

Ehsaneddin Asgari, Yassine El Kheir, and Mohammad Ali Sadraei Javaheri. 2025. Morphbpe: A morpho-aware tokenizer bridging linguistic complexity for efficient llm training across morphologies. *arXiv preprint arXiv:2502.00894*.

M Saiful Bari, Yazeed Alnumay, Norah A. Alzahrani, Nouf M. Alotaibi, Hisham Abdullah Alyahya, Sultan AlRashed, Faisal Abdulrahman Mirza, Shaykhah Z. Alsubaie, Hassan A. Alahmed, Ghadah Alabduljabbar, Raghad Alkhathran, Yousef Almushayqih, Raneem Alnajim, Salman Alsubaihi, Maryam Al Mansour, Saad Amin Hassan, Dr. Majed Alrubaian, Ali Alammari, Zaki Alawami, and 7 others. 2025. ALLam: Large language models for arabic and english. In *The Thirteenth International Conference on Learning Representations*.

Thomas Bauwens and Pieter Delobelle. 2024. BPE-knockout: Pruning pre-existing BPE tokenisers with backwards-compatible morphological semi-supervision. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5810–5832, Mexico City, Mexico. Association for Computational Linguistics.

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

Michael J Bommarito, Daniel Martin Katz, and Jillian Bommarito. 2025. Kl3m tokenizers: A family of domain-specific and character-level tokenizers for legal, financial, and preprocessing applications. *Preprint*, arXiv:2503.17247.

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.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Yekun Chai, Yewei Fang, Qiwei Peng, and Xuhong Li. 2024. Tokenization falling short: On subword robustness in large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1582–1599, Miami, Florida, USA. Association for Computational Linguistics.

Zeming Chen, Alejandro Hernández-Cano, Angelika Romanou, Antoine Bonnet, Kyle Matoba, Francesco Salvi, Matteo Pagliardini, Simin Fan, Andreas Köpf, Amirkeivan Mohtashami, Alexandre Sallinen, Alireza Sakhaeirad, Vinitra Swamy, Igor Krawczuk, Deniz Bayazit, Axel Marmet, Syrielle Montariol, Mary-Anne Hartley, Martin Jaggi, and Antoine Bosselut. 2023. Meditron-70b: Scaling medical pretraining for large language models. *Preprint*, arXiv:2311.16079.

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.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, and 1 others. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Marco Cognetta, Tatsuya Hiraoka, Rico Sennrich, Yuval Pinter, and Naoaki Okazaki. 2024a. 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.

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

Gautier Dagan, Gabriel Synnaeve, and Baptiste Rozière. 2024. Getting the most out of your tokenizer for pre-training and domain adaptation. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Björn Deiseroth, Manuel Brack, Patrick Schramowski, Kristian Kersting, and Samuel Weinbach. 2024. T-FREE: Subword tokenizer-free generative LLMs via sparse representations for memory-efficient embeddings. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 21829–21851, Miami, Florida, USA. 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.

Vikrant Dewangan, Bharath Raj S, Garvit Suri, and Raghav Sonavane. 2025. When every token counts: Optimal segmentation for low-resource language models. In *Proceedings of the First Workshop on Language Models for Low-Resource Languages*, pages 294–308, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Konstantin Dobler and Gerard de Melo. 2023. FOCUS: Effective embedding initialization for monolingual specialization of multilingual models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13440–13454, Singapore. Association for Computational Linguistics.

Fanar Fanar, Ummar Abbas, Mohammad Shahmeer Ahmad, Firoj Alam, Enes Altinisik, Ehsannedin Asgari, Yazan Boshmaf, Sabri Boughorbel, Sanjay Chawla, Shammur Chowdhury, and 1 others. 2025. Fanar: An arabic-centric multimodal generative ai platform. *arXiv preprint arXiv:2501.13944*.

Darius Feher, Ivan Vulić, and Benjamin Minixhofer. 2025. Retrofitting large language models with dynamic tokenization. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 29866–29883, Vienna, Austria. Association for Computational Linguistics.

Xue-Yong Fu, Md Tahmid Rahman Laskar, Elena Khasanova, Cheng Chen, and Shashi Tn. 2024. Tiny titans: Can smaller large language models punch above their weight in the real world for meeting summarization? In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 6: Industry Track)*, pages 387–394, Mexico City, Mexico. Association for Computational Linguistics.

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

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

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

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*, pages 3955–3964, Online. Association for Computational Linguistics.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 1 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Tatsuya Hiraoka, Sho Takase, Kei Uchiumi, Atsushi Keyaki, and Naoaki Okazaki. 2021. Joint optimization of tokenization and downstream model. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 244–255.

Hongzhi Huang, Defa Zhu, Banggu Wu, Yutao Zeng, Ya Wang, Qiyang Min, and Xun Zhou. 2025. Over-tokenized transformer: Vocabulary is generally worth scaling. *Preprint*, arXiv:2501.16975.

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

Eugene Jang, Kimin Lee, Jin-Woo Chung, Keuntae Park, and Seungwon Shin. 2025. Improbable bigrams expose vulnerabilities of incomplete tokens in byte-level tokenizers. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 18209–18216, Suzhou, China. Association for Computational Linguistics.

Minhao Jiang, Ken Ziyu Liu, Ming Zhong, Rylan Schaeffer, Siru Ouyang, Jiawei Han, and Sanmi Koyejo. 2024. Investigating data contamination for pre-training language models. *arXiv preprint arXiv:2401.06059*.

Bernal Jimenez Gutierrez, Huan Sun, and Yu Su. 2023. Biomedical language models are robust to sub-optimal tokenization. In *The 22nd Workshop on Biomedical Natural Language Processing and BioNLP Shared Tasks*, pages 350–362, Toronto, Canada. Association for Computational Linguistics.

Seungduk Kim, Seungtaek Choi, and Myeongho Jeong. 2024. Efficient and effective vocabulary expansion towards multilingual large language models. *Preprint*, arXiv:2402.14714.

Artur Kiulian, Anton Polishko, Mykola Khandoga, Yevhen Kostiuk, Guillermo Gabrielli, Łukasz Gagała, Fadi Zaraket, Qusai Abu Obaida, Hrishikesh Garud, Wendy Wing Yee Mak, Dmytro Chaplynskyi, Selma Belhadj Amor, and Grigol Peradze. 2024. From english-centric to effective bilingual: Llms with custom tokenizers for underrepresented languages. *Preprint*, arXiv:2410.18836.

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.

Sander Land and Max Bartolo. 2024. Fishing for magikarp: Automatically detecting under-trained tokens in large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11631–11646, Miami, Florida, USA. Association for Computational Linguistics.

Aaron Langford, Aayush Shah, Abhanshu Gupta, Abhimanyu Bhatter, Abhinav Goyal, Abhinav Mathur, Abhinav Mohanty, Abhishek Kumar, Abhishek Sethi,

Abi Komma, and 1 others. 2025. The amazon nova family of models: Technical report and model card. *arXiv preprint arXiv:2506.12103*.

Md Tahmid Rahman Laskar, Sawsan Alqahtani, M Saiful Bari, Mizanur Rahman, Mohammad Abdullah Matin Khan, Haidar Khan, Israt Jahan, Amran Bhuiyan, Chee Wei Tan, Md Rizwan Parvez, and 1 others. 2024. A systematic survey and critical review on evaluating large language models: Challenges, limitations, and recommendations. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 13785–13816.

Md Tahmid Rahman Laskar, M Saiful Bari, Mizanur Rahman, Md Amran Hossen Bhuiyan, Shafiq Joty, and Jimmy Huang. 2023a. A systematic study and comprehensive evaluation of chatgpt on benchmark datasets. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 431–469.

Md Tahmid Rahman Laskar, Xue-Yong Fu, Cheng Chen, and Shashi Bhushan Tn. 2023b. Building real-world meeting summarization systems using large language models: A practical perspective. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 343–352.

LCM team, Loïc Barrault, Paul-Ambroise Duquenne, Maha Elbayad, Artyom Kozhevnikov, Belen Alastruey, Pierre Andrews, Mariano Coria, Guillaume Couairon, Marta R. Costa-jussà, David Dale, Hady Elsahar, Kevin Heffernan, João Maria Janeiro, Tuan Tran, Christophe Ropers, Eduardo Sánchez, Robin San Roman, Alexandre Mourachko, and 2 others. 2024. Large concept models: Language modeling in a sentence representation space. *Preprint*, arXiv:2412.08821.

Haoran Lian, Yizhe Xiong, Jianwei Niu, Shasha Mo, Zhenpeng Su, Zijia Lin, Hui Chen, Jungong Han, and Guiguang Ding. 2025. Scaffold-bpe: Enhancing byte pair encoding for large language models with simple and effective scaffold token removal. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 24539–24548.

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.

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

Chengyuan Liu, Shihang Wang, Lizhi Qing, Kun Kuang, Yangyang Kang, Changlong Sun, and Fei Wu. 2024a. Gold panning in vocabulary: An adaptive method for vocabulary expansion of domain-specific LLMs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7442–7459, Miami, Florida, USA. Association for Computational Linguistics.

Yanting Liu, Tao Ji, Changzhi Sun, Yuanbin Wu, and Xiaoling Wang. 2024b. Generation with dynamic vocabulary. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 18931–18948, Miami, Florida, USA. Association for Computational Linguistics.

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.

Sangwhan Moon, Tatsuya Hiraoka, and Naoaki Okazaki. 2025. Bit-level bpe: Below the byte boundary. *Preprint*, arXiv:2506.07541.

Mir Tafseer Nayeem, Sawsan Alqahtani, Md Tahmid Rahman Laskar, Tasnim Mohiuddin, and M Saiful Bari. 2025. Beyond fertility: Analyzing strr as a metric for multilingual tokenization evaluation. *Preprint*, arXiv:2510.09947.

Mir Tafseer Nayeem and Davood Rafiei. 2024. KidLM: Advancing language models for children – early insights and future directions. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 4813–4836, Miami, Florida, USA. Association for Computational Linguistics.

Mir Tafseer Nayeem and Davood Rafiei. 2025. OpinioRAG: Towards generating user-centric opinion highlights from large-scale online reviews. In *Second Conference on Language Modeling*.

Pit Neitemeier, Björn Deiseroth, Constantin Eichenberg, and Lukas Balles. 2025. Hierarchical autoregressive transformers: Combining byte-and word-level processing for robust, adaptable language models. *arXiv preprint arXiv:2501.10322*.

Xuan-Phi Nguyen, Wenxuan Zhang, Xin Li, Mahani Aljunied, Zhiqiang Hu, Chenhui Shen, Yew Ken Chia, Xingxuan Li, Jianyu Wang, Qingyu Tan, and 1 others. 2023. Seallms–large language models for southeast asia. *arXiv preprint arXiv:2312.00738*.

Ruikang Ni, Da Xiao, Qingye Meng, Xiangyu Li, Shihui Zheng, and Hongliang Liang. 2025. Benchmarking and understanding compositional relational reasoning of llms. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(18):19703–19711.

OpenAI. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Artidoro Pagnoni, Ram Pasunuru, Pedro Rodriguez, John Nguyen, Benjamin Muller, Margaret Li, Chunting Zhou, Lili Yu, Jason Weston, Luke Zettlemoyer, and 1 others. 2024. Byte latent transformer: Patches scale better than tokens. *arXiv preprint arXiv:2412.09871*.

Aleksandar Petrov, Emanuele La Malfa, Philip Torr, and Adel Bibi. 2023. Language model tokenizers introduce unfairness between languages. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2021. UNKs everywhere: Adapting multilingual language models to new scripts. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10186–10203, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI*. Accessed: 2024-11-15.

Varshini Reddy, Craig W. Schmidt, Yuval Pinter, and Chris Tanner. 2025. How much is enough? the diminishing returns of tokenization training data. *Preprint*, arXiv:2502.20273.

Jessica Rumbelow and Matthew Watkins. 2024. Solidgoldmagikarp (plus, prompt generation). *lesswrong.com*.

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.

Elizabeth Salesky, Andrew Runge, Alex Coda, Jan Niehues, and Graham Neubig. 2020. Optimizing segmentation granularity for neural machine translation. *Machine Translation*, 34(1):41–59.

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.

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.

Jean Seo, Jaeyoon Kim, SungJoo Byun, and Hyopil Shin. 2025. How does a language-specific tokenizer affect llms? *Preprint*, arXiv:2502.12560.

Jiu Sha, Mengxiao Zhu, Chong Feng, and Yuming Shang. 2025. VEEF-multi-LLM: Effective vocabulary expansion and parameter efficient finetuning towards multilingual large language models. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 7963–7981, Abu Dhabi, UAE. Association for Computational Linguistics.

Chenze Shao, Fandong Meng, and Jie Zhou. 2025. Beyond next token prediction: Patch-level training for large language models. In *The Thirteenth International Conference on Learning Representations*.

Faisal Tareque Shohan, Mir Tafseer Nayeem, Samsul Islam, Abu Ubaida Akash, and Shafiq Joty. 2024. XL-HeadTags: Leveraging multimodal retrieval augmentation for the multilingual generation of news headlines and tags. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 12991–13024, Bangkok, Thailand. Association for Computational Linguistics.

Telem Joyson Singh, Sanasam Ranbir Singh, and Priyankoo Sarmah. 2023. Subwords to word back composition for morphologically rich languages in neural machine translation. In *Proceedings of the 37th Pacific Asia Conference on Language, Information and Computation*, pages 691–700, Hong Kong, China. Association for Computational Linguistics.

Chaofan Tao, Qian Liu, Longxu Dou, Niklas Muennighoff, Zhongwei Wan, Ping Luo, Min Lin, and Ngai Wong. 2024. Scaling laws with vocabulary: Larger models deserve larger vocabularies. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Yi Tay, Vinh Q. Tran, Sebastian Ruder, Jai Gupta, Hyung Won Chung, Dara Bahri, Zhen Qin, Simon Baumgartner, Cong Yu, and Donald Metzler. 2022. Charformer: Fast character transformers via gradient-based subword tokenization. In *International Conference on Learning Representations*.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, and 1 others. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Atula Tejaswi, Nilesh Gupta, and Eunsol Choi. 2024. Exploring design choices for building language-specific LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10485–10500, Miami, Florida, USA. 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 Transactions on Asian and Low-Resource Language Information Processing*, 22(4):1–21.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Alexander Tsvetkov and Alon Kipnis. 2024. Information parity: Measuring and predicting the multilingual capabilities of language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7971–7989, Miami, Florida, USA. Association for Computational Linguistics.

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.

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.

Dixuan Wang, Yanda Li, Junyuan Jiang, Zepeng Ding, Guochao Jiang, Jiaqing Liang, and Deqing Yang. 2024. Tokenization matters! degrading large language models through challenging their tokenization. *arXiv preprint arXiv:2405.17067*.

Hai Wang, Dian Yu, Kai Sun, Jianshu Chen, and Dong Yu. 2019. Improving pre-trained multilingual model with vocabulary expansion. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 316–327, Hong Kong, China. Association for Computational Linguistics.

Anna Wegmann, Dong Nguyen, and David Jurgens. 2025. Tokenization is sensitive to language variation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 10958–10983, Vienna, Austria. Association for Computational Linguistics.

Benoist Wolleb, Romain Silvestri, Georgios Vernikos, Ljiljana Dolamic, and Andrei Popescu-Belis. 2023. Assessing the importance of frequency versus compositionality for subword-based tokenization in NMT. In *Proceedings of the 24th Annual Conference of the European Association for Machine Translation*, pages 137–146, Tampere, Finland. European Association for Machine Translation.

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.

Jin Yang, Zhiqiang Wang, Yanbin Lin, and Zunduo Zhao. 2024. Problematic tokens: Tokenizer bias in large language models. *2024 IEEE International Conference on Big Data (BigData)*, pages 6387–6393.

Yunzhi Yao, Shaohan Huang, Wenhui Wang, Li Dong, and Furu Wei. 2021. Adapt-and-distill: Developing small, fast and effective pretrained language models for domains. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 460–470, Online. 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.

Jun Zhao, Zhihao Zhang, Luhui Gao, Qi Zhang, Tao Gui, and Xuanjing Huang. 2024. Llama beyond english: An empirical study on language capability transfer. *Preprint*, arXiv:2401.01055.

Mengyu Zheng, Hanting Chen, Tianyu Guo, Chong Zhu, Binfan Zheng, Chang Xu, and Yunhe Wang. 2024. Enhancing large language models through adaptive tokenizers. *Advances in Neural Information Processing Systems*, 37:113545–113568.

Vilém Zouhar, Clara Meister, Juan Gastaldi, Li Du, Mrinmaya Sachan, and Ryan Cotterell. 2023a. 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.

Vilém Zouhar, Clara Meister, Juan Gastaldi, Li Du, Tim Vieira, Mrinmaya Sachan, and Ryan Cotterell. 2023b. A formal perspective on byte-pair encoding. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 598–614, Toronto, Canada. Association for Computational Linguistics.

Supplementary Material: Appendices

A Illustrative Example for Token Fragmentation

A simple illustration of inefficient token fragmentation can be seen when applying a standard English-trained BPE tokenizer to morphologically rich languages such as Arabic. We emphasize that such tokenizers were not designed to support Arabic or other morphologically rich non-European languages, and are used here solely to illustrate the consequences of language–tokenizer misalignment rather than to imply a design failure. Consider the Arabic word in Figure 2 ("and they will recognize it"). This word combines a conjunction, a verb root, a subject marker, and an object pronoun, all within a single surface form. An English-centric tokenizer typically fails to capture this structure and instead fragments the word into arbitrary subword units.

In this example, the English-trained BPE tokenizer (GPT-2) produces eleven tokens, heavily fragmenting the Arabic word into individual characters and ignoring its internal morphological structure. The multilingual XLM-R tokenizer and the more recent LLaMA 3.1 model perform significantly better, segmenting the word into larger, semantically meaningful units. Interestingly, the morpheme-aware tokenizer results in more tokens than LLaMA 3.1, raising the question of whether fewer tokens always indicate better linguistic alignment or modeling efficiency. In contrast, the earlier LLaMA 1 tokenizer exhibits even greater over-fragmentation, treating spaces and letters as separate tokens and producing eleven fragments. Such over-fragmentation increases sequence length, reduces modeling efficiency, and disrupts semantic representation, effects that become especially pronounced in morphologically rich languages like Arabic, where single surface forms often encode multiple grammatical and semantic elements (Toraman et al., 2023; Ali et al., 2024). This over-fragmentation leads to higher tokenization costs and reduced model efficiency, particularly for non-English users who encounter faster context length limits and increased API costs.

B Evolution of Tokenization in Leading Models

As discussed, the tokenizer is a fundamental component of model training, not merely an implementation detail. As LLMs evolve, future models must prioritize tokenization strategies to strengthen linguistic and logical capabilities. The following examples demonstrate that improving a language model often begins with revisiting and refining its vocabulary.

B.1 GPT Series Tokenizer Advancements

OpenAI's early GPT models, including GPT-2, used BPE tokenizer with a vocabulary of approximately 50k tokens (Radford et al., 2019). This tokenizer worked well for English but led to over-fragmentation of text in many other languages, causing significantly longer token sequences for non-English content (See §A). To address these issues, OpenAI improved the tokenizer in later models like ChatGPT (GPT-3.5) and GPT-4, introducing the cl100k_base tokenizer with around 100k tokens, doubling the vocabulary size of GPT-2 (Yang et al., 2024). This reduction in token count led to significant improvements. OpenAI continued refining the tokenizer with the 2024 "GPT-4o" update, which expanded the vocabulary to roughly 200k tokens. This change further reduced token usage, especially for non-English languages; however, this expansion came with a trade-off: the inclusion of rare tokens that were under-trained, leading to occasional issues like hallucinations or unexpected behavior, particularly with complex words in languages such as Chinese (See §3.2.3).

B.2 Tokenization Strategies in Google's Multilingual Models: PaLM, Gemma, and ByT5

Google's PaLM (Pathways Language Model) (Chowdhery et al., 2023) employs a SentencePiece tokenizer with a 256k vocabulary to handle a broad range of languages, reducing token fragmentation in languages like Vietnamese and Tamil. This strategy enhanced PaLM's multilingual performance, particularly in tasks like translation. While the larger vocabulary increased memory usage, it provided improved language coverage. In 2024, Google released Gemma (Team et al., 2024), an open-access LLM with a redesigned tokenizer featuring a 262k vocabulary, also used in the Gemini models (Anil et al., 2025). This improved tokenizer mitigated over-tokenization and mis-splitting issues, enabling Gemma to

Tokenizer	Tokens Produced for وسيعرفونها	Sequence Length
GPT-2 (English BPE)	[و, س, ي, ع, ر, ف, و, ن, ه, ا]	11
XLM-R (Multilingual Unigram)	[وسي, عرف, ونها]	3
LLaMA 1	[, و, س, ي, ع, ر, ف, و, ن, ه, ا]	11
LLaMA 3.1 / LLaMA 3.2	[وسي, عرف, ونها]	3
Morpheme-Aware Tokenizer	[و, س, يعرف, ون, ها]	5

Figure 2: Tokenization of the Arabic word using different tokenization strategies.

manage over 100 languages and perform well in multilingual benchmarks, underscoring the key role of the tokenizer in its success, albeit with increased memory usage.

Google's ByT5 (Xue et al., 2022) takes a different approach by eliminating tokenization altogether, working directly on raw bytes to address challenges like out-of-vocabulary words, misspellings, and non-Latin scripts that subword tokenizers struggle with. ByT5 uses a 256-byte "vocabulary" (1 byte = 1 token), simplifying the process and ensuring comprehensive text coverage at the expense of longer sequences. While this increases sequence length, ByT5 reduces the model size for embeddings and improves robustness, particularly for noisy inputs. It outperformed tokenized models on tasks such as GLUE and QA, especially for smaller models, demonstrating the potential benefits of eliminating tokenization in specific scenarios. However, byte-level models, such as ByT5, face the trade-off of slower training and inference times due to longer sequence lengths. Despite this, Transformers remain computationally efficient, though ByT5 is slower during training and significantly slower during inference.

B.3 LLaMA Series Tokenizer Advancements

Meta's LLaMA (Touvron et al., 2023a) and LLaMA-2 (Touvron et al., 2023b) used a 32k BPE vocabulary primarily trained on English and European languages, leading to longer sequences, wasted context space, and poor performance on underrepresented languages (See §A). Meta did not substantially redesign the tokenizer between LLaMA 1 and LLaMA 2, but improving multilingual performance in LLaMA 3 required changes beyond simple reuse, underscoring how tokenizer choices can necessitate retraining or substantial adaptation. With the release of LLaMA 3 and LLaMA 3.1 (Grattafiori et al., 2024), Meta addressed these issues by introducing a new tokenizer with a larger vocabulary and broader language coverage. The improved tokenizer reduces over-fragmentation in underrepresented languages and enhances performance across a wider range of languages. This update allows LLaMA 3 models to handle multilingual inputs more effectively, providing a more balanced representation of diverse languages and increasing model efficiency. However, these improvements come with the trade-off of higher memory usage and computation, which Meta optimized through architectural adjustments in LLaMA 3.1.

C Core Evaluation Dimensions

(i) Coverage: Evaluation must include coverage of rare words, domain-specific terminology, and named entities. Excessive fragmentation often indicates poor vocabulary alignment and leads to inefficiencies during modeling. For example, in the sentence "The patient was diagnosed with pheochromocytoma and referred to Dr. Al-Rahmani at the Hospital," a poorly designed tokenizer may oversegment the rare medical term pheochromocytoma or unnaturally split named entities like Al-Rahmani. High-coverage tokenizers should preserve such expressions with minimal degradation. As another example, for morphologically rich languages, tokenizer coverage can also be evaluated by comparing its output to that of a morphological analyzer. Measuring the divergence between tokenized segments and known morphemes can reveal how well the tokenizer captures meaningful subword units, contributing to both the depth (internal structure) and breadth (lexical variety) of vocabulary representation.

(ii) Generalizability: For instance, in the sentence "She re-nationalized the industry after years of pri-

vatization," a well-generalizing tokenizer should segment re-nationalized in a way that reflects its compositional structure, even if the full form was absent from training. Tokenizers that treat related forms inconsistently can harm representation learning and impair downstream generalization.

(iii) Linguistic Alignment: Poor alignment obscures grammatical structure, reduces interpretability, and degrades performance in syntax-sensitive tasks. For example, Arabic verbs like yaktubūn ("they write") and yarsimūn ("they draw") share morphological patterns that should be preserved to support better generalization and cross-linguistic representation learning.

(iv) Robustness: Robust tokenizers should manage spelling errors, informal usage, dialectal forms, and code-switching without excessive fragmentation. In the mixed-language sentence "We discussed the results en la reunión this morning," a resilient tokenizer should treat the embedded Spanish phrase as a cohesive unit rather than fragmenting it unnaturally. Robustness is especially important in multilingual, real-world deployments (Shohan et al., 2024).

(v) Representation Utilization: Evaluation must also consider which parts of the vocabulary and embedding space are actually used during inference. If a benchmark activates only 30% of token embeddings, the majority of the learned vocabulary remains untested. Tokenizer evaluation should include coverage diagnostics that quantify how well a benchmark exercises the full representational capacity of the model.

D Usage of Large Language Models

We disclose that large language models were used in limited, assistive roles. Specifically, they supported **text polishing**, including improvements to grammar, spelling, phrasing, and word choice, with all suggestions reviewed by the authors. All outputs were manually verified, and the authors remain fully responsible for the research content and conclusions.

Paper (year)	Efficiency	Linguistic fidelity	Fairness	Computational cost
Sennrich et al. (2016)	High compression compared to characters, shorter sequences than characters or bytes.	Greedy merges can split morphemes, alignment varies by language.	English-centric corpora bias split patterns across languages.	Fewer tokens lower attention cost, tokenizer training is offline, embedding size grows with vocabulary.
Kudo (2018)	Compression similar to BPE with improved robustness from sampling.	Unigram often respects morphemes better than greedy BPE.	Less brittle across scripts than word level.	Training adds sampling on the fly, inference similar to BPE, FLOPs comparable at a fixed sequence length.
Kudo and Richardson (2018)	Fast raw-text training and deployment.	Supports BPE or Unigram with normalization that affects morphology.	Behavior depends on corpus balance and normalization choices.	Minimal runtime overhead, compute impact is via vocabulary size and sequence length.
Tay et al. (2022)	Learns blocks from characters, downsampling yields near-subword compression.	Improved compositionality from learned subwords over raw chars.	More uniform across scripts than fixed subword vocabularies.	Light module overhead but shorter effective sequences reduce FLOPs versus char or byte inputs.
Libovický and Helcl (2024)	Uses morphological analyzers to guide merges at training time.	Better morpheme alignment than frequency-only segmentation.	Benefits morph-rich and low-resource languages.	Heavier tokenizer training, similar inference token counts to BPE, can reduce wasted tokens where over-segmentation existed.
Chizhov et al. (2024)	Prunes low-value merges to improve compression and utility.	Cleaner boundaries and fewer undertrained tokens.	Reduces fragmentation disparities in multilingual settings.	Small extra tokenizer-training cost, shorter sequences give modest FLOP savings.
Zouhar et al. (2023b)	Faster implementations and complexity insights.	Theoretical guidance for better merge schedules.	***Not Available***	Lower tokenizer build cost, model compute unchanged except via sequence length.
Gowda and May (2020)	Identifies a sweet spot balancing compression and generalization.	Very small vocabularies harm morphology, very large vocabularies memorize idiosyncrasies.	Dominant-language tuning can harm others.	Too small vocabularies increase sequence length, too large increase embedding parameters, choose V to minimize total compute.
Asgari et al. (2025)	Slightly less compression than aggressive BPE with better utility.	Blocks merges across morpheme boundaries for higher consistency.	Improves parity for morph-rich languages by reducing fragmentation.	Minimal integration cost, sequence length can rise slightly but faster convergence often offsets compute.

Table 3: Training Tokenizers from Scratch: core algorithms, learned tokenizers, and theory that shape efficiency, fidelity, and compute from the ground up.

Paper (year)	Efficiency	Linguistic fidelity	Fairness	Computational cost
Pfeiffer et al. (2021)	Reduces UNK rates on unseen scripts, dedicated in-language tokenizer yields shorter sequences than character fallback.	Captures script-specific orthography, aligned embeddings preserve word-level semantics.	Targets under-served languages with unseen scripts, narrows cross-language token-length disparities.	Offline tokenizer build plus small added embeddings, no full model retraining, inference cost similar while fewer tokens reduce attention and cache compute.
Rust et al. (2021)	Monolingual tokenizers improve efficiency for target languages.	Better alignment and accuracy for the intended language.	Multilingual vocabularies show monolingual bias and parity issues.	Retokenizing and re-embedding have low overhead compared to pretraining, shorter sequences reduce FLOPs.
Ali et al. (2024)	Tokenizer choice materially shifts compression and utility.	Fertility and parity alone are weak quality predictors.	English-centric designs raise costs for many languages.	Inefficient vocabularies lengthen sequences and steps, better multilingual tokenizers lower training and inference compute.
Yehezkel and Pinter (2023)	Slightly less compression than pure BPE with better utility.	Context-aware subwords improve cohesion and morphology.	Reduces over-segmentation in agglutinative languages.	Heavier offline build only, runtime similar to subword, potential token reductions yield modest savings.
Chizhov et al. (2024)	Merge pruning improves compression and quality.	Cleaner boundaries and fewer under-trained tokens.	Less fragmentation across languages.	Small tokenizer-training overhead, shorter sequences save compute.
Nguyen et al. (2023)	Vocabulary expansion merges NLLB tokens into LLaMA for SEA languages.	Fewer unnatural splits in non-Latin scripts with better word coverage.	Directly reduces cross-language token length disparities while preserving English efficiency.	Expanded vocabularies increase embedding size, fewer tokens cut attention and cache cost, later versions reuse large Gemma vocabularies.
Bari et al. (2025)	Tokenizer augmentation improves Arabic token economy.	Arabic-aware subwords better match morphemes while preserving English skill.	Narrows parity gaps for Arabic versus English.	Embedding and softmax grow with expansion, fewer Arabic tokens reduce FLOPs per step, tokenizer work is offline.
Fanar et al. (2025)	From-scratch and continued-pretrain models show practical adaptation.	Arabic-centric training improves segmentation for Arabic forms.	Improves parity for Arabic in a bilingual setup.	Reusing Gemma tokenizer avoids costly remapping, larger vocab raises memory, fewer tokens reduce attention cost.
Uzan et al. (2024)	Intrinsic measures reduce expensive full model comparisons.	Segmentation quality can be assessed without retraining models.	Better selection reduces bias from ad-hoc choices.	Cuts search compute by avoiding multiple retrains during adaptation.
Schmidt et al. (2024)	Compression alone is not a sufficient design signal.	Emphasizes morphology and semantics beyond length.	Warns that compression-driven reuse can harm parity.	Guidance avoids wasted pretraining and encourages retokenization where it reduces steps.

Table 4: Adapting Tokenizers: rethinking reuse through language expansion, refinement, and context-aware construction to cut tokens and cost in new domains.

Paper (year)	Efficiency	Linguistic fidelity	Fairness	Computational cost
Petrov et al. (2023)	Documents large cross-language token count gaps for the same content.	Splits often ignore morphology outside Latin scripts.	Up to large disparities with pricing and latency inequities.	Over-tokenized languages pay higher FLOPs per word and lose effective context, fairer tokenizers reduce compute.
Ahia et al. (2023)	API cost varies with tokenizer-induced length differences.	Over-fragmentation lowers utility for some languages.	Identifies systematic overcharging of certain scripts.	Parity-aware tokenizers reduce steps and tokens, inference cost scales with tokens so parity improves efficiency.
Chai et al. (2024)	Segmentation is fragile under typos and format shifts.	LLMs are sensitive to token boundaries, dropout improves robustness.	Variants can disproportionately affect morph-rich languages.	Robustness training adds small CPU overhead, inference unchanged, avoids token length blowups in the wild.
Asgari et al. (2025)	Competitive compression with better utility in difficult scripts.	Prevents merges across morpheme boundaries.	Improves parity for morph-rich languages by reducing fragmentation.	Slight sequence length increase can be offset by faster convergence, overall compute does not rise materially.
Chizhov et al. (2024)	Pruning merges improves compression and quality.	Cleaner boundaries reduce pathological splits.	Less fragmentation in multilingual settings.	Token counts drop slightly, giving modest FLOP savings.
Nguyen et al. (2023)	Vocabulary expansion improves token economy in SEA languages.	Fewer unnatural splits for non-Latin scripts.	Shrinks token length disparities while preserving English performance.	Embedding growth trades for fewer tokens, net reduction in attention compute for SEA text.
Bari et al. (2025)	Arabic tokenization improved through expansion and augmentation.	Arabic-aware units better match morphology.	Narrowed parity gap for Arabic.	Fewer Arabic tokens reduce step compute, expansion cost is offline.
Rust et al. (2021)	Monolingual tokenizers are more efficient for specific languages.	Better alignment for the target language.	Multilingual vocabularies show parity issues.	Retokenization overhead is small compared to the compute saved by shorter sequences.

Table 5: Responsible tokenization addressing parity, robustness, and stability to reduce unequal costs and failures across languages and noisy inputs.

Paper (year)	Efficiency	Linguistic fidelity	Fairness	Computational cost
Goldman et al. (2024)	Higher compression often correlates with better generation quality at smaller scales.	Compression is helpful but structure also matters.	Over-compressed languages can lose linguistic structure.	Fewer tokens reduce FLOPs, gains saturate at large scale, report both compression and structure.
Zouhar et al. (2023a)	Intrinsic metric correlates with downstream in some settings.	Penalizes overly common or overly rare units to encourage balance.	Can guide parity-aware design if used cautiously.	No model-time cost, helps pick vocabularies that reduce sequence length or maintain quality at same compute.
Cognetta et al. (2024b)	Shows intrinsic metrics can be gamed without real gains.	High Rényi scores do not always imply better segmentation.	Warns against proxy metrics.	Avoids wasted pretraining on misleading metrics, no runtime change.
Uzan et al. (2024)	Compares intrinsic measures to avoid repeated full retrains.	Assesses segmentation quality without a model in the loop.	Selection procedures can reduce bias from ad-hoc choices.	Cuts search compute during tokenizer selection, enables faster iteration.
Schmidt et al. (2024)	Compression alone is not a sufficient signal for design.	Emphasizes morphology and semantics in evaluation.	Warns that compression-driven design can harm parity.	Guidance reduces wasted compute and improves selection efficiency.
Zouhar et al. (2023b)	Complexity and implementation analysis for BPE.	Theoretical guidance for constraints and schedules.	***Not Available***	Tokenizer build cost reduced, model compute mediated by sequence length only.
Gowda and May (2020)	Locates vocabulary size that balances compression and generalization.	Very small or very large vocabularies degrade linguistic structure.	***Not Available***	Minimizes total compute by trading embedding size against sequence length, report chosen V with rationale.
Ali et al. (2024)	Shows tokenizer choice materially affects cost and quality.	Finds weak correlation of some popular intrinsic indicators with final quality.	Highlights cross-language disparities from English-centric design.	Encourages reporting token counts, length parity, and training cost impacts alongside quality.

Table 6: Evaluation Framework and Missing Standards of metrics, analyses, and protocols that make tokenizer selection measurable, comparable, and compute-aware.

Stage	Guiding Questions
Training from Scratch	(i) Do target languages over-fragment under English-centric vocabularies? (ii) Which vocabulary size minimizes end-to-end FLOPs rather than dev-set perplexity? (iii) Can morphology-aware constraints reduce token counts in dominant languages? (iv) Could a character/byte backbone with learned downsampling add robustness without higher compute cost?
Adapting Existing Tokenizers	(i) Where is token inflation worst, and what parity ratio is acceptable in production? (ii) Is a small vocabulary expansion sufficient, or is a dedicated tokenizer needed? (iii) Can checkpoint compatibility be preserved when modifying the tokenizer (e.g., mapped or appended embeddings)? (iv) What intrinsic or downstream probes will you run before a full retrain? (v) How will you attribute compute savings to shorter sequences versus parameter changes?
Responsible Tokenization	(i) What token-length disparity across top languages is acceptable, and how will you enforce it? (ii) Do evaluations include noisy, code-switched, or adversarial text to test segmentation stability? (iii) Which mitigation (dropout, merge refinement, morphology-aware rules, or language-specific expansion) yields the largest token reduction for the weakest languages? (iv) How will you track pricing and latency equity linked to tokenizer-induced length differences? (v) What rollout and monitoring plan will detect regressions in parity or robustness after deployment?
Evaluation Framework and Missing Standards	(i) Which metrics act as pass/fail gates for selection—tokens per 1k characters, parity ratios, compression–structure measures, or estimated FLOPs deltas? (ii) What small, repeatable downstream tasks will validate candidates beyond intrinsic metrics? (iii) How will you disentangle and report the effects of sequence length versus embedding size on step time and memory? (iv) What decision rule determines when to stop iterating and lock the tokenizer for production? (v) How will you document tokenizer choices to ensure cost and quality claims are reproducible?

Table 7: Consolidated guiding questions for principled tokenizer development and adaptation. Each stage highlights key diagnostic prompts for evaluating design trade-offs, resource efficiency, and cross-lingual robustness.