Proceedings of the 8th International Conference on Computing and Data Science
DOI: 10.54254/2755-2721/2026.37609

Designing Chinese Tokenizers under Data Constraints: A White-Box Empirical Study of Character Fragmentation

Luofu Gong

Beijing Aidi School, Beijing, China
13359226905@163.com

Abstract. Chinese text requires substantially more tokens than English in large language models (LLMs), increasing API costs and context-window consumption. Most prior work measures this premium using off-the-shelf tokenizers, but the underlying mechanism—where the additional tokens actually arise—remains poorly understood, particularly under data-constrained training, which is common in domain-specific and small-scale models. We implement BPE, WordPiece, and Unigram from scratch (approximately 1,000 lines of standard-library code), train them on 20,000 Chinese news documents, and quantify per-character tokenization across vocabulary sizes from 8K to 128K. Characters are grouped by frequency, and evaluation uses a held-out test set with clustered-bootstrap confidence intervals. The central result is a low-frequency fragmentation effect: at an 8K vocabulary, byte-level BPE splits characters appearing fewer than 100 times in training into an average of 2.04 tokens per character, compared with 1.00 for high-frequency characters; enlarging the vocabulary 16-fold reduces this value to 1.10 at a fixed training scale. Comparison with production tokenizers reveals a spectrum from 2.24 (OpenAI's cl100k) and 2.04 (our 8K implementation) to 1.02 (DeepSeek's Chinese-optimized 128K vocabulary), indicating that the effect depends on vocabulary allocation and training scale rather than being an inherent property of Chinese. A semantic-equivalence experiment further supports this conclusion: for equivalent Chinese-English sentence pairs, an English-centric tokenizer uses 65% more tokens for Chinese, whereas a balanced byte-level BPE trained on 20,000 documents (10K Chinese + 10K English) uses 27% fewer. The paper concludes with configuration guidance for Chinese-focused tokenizers under data constraints.

Keywords: Chinese Tokenization, Byte Pair Encoding, Character Fragmentation, Data-Constrained Training

1. Introduction

The same query can be more expensive to tokenize in Chinese. For a 100-character Chinese prompt, GPT-style tokenizers produce roughly 130-300 tokens—four to five times the 25-30 tokens generated for a comparable English character count. Because API pricing and context windows are denominated in tokens, this premium translates directly into higher costs and reduced effective capacity [1]. At the same time, an increasing number of practitioners train or adapt language models on limited, domain-specific corpora, such as medical records, classical texts, dialects, or specialized

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2026.37609

news archives, where tokenizer training data may be measured in hundreds of megabytes rather than trillions of tokens. In this data-constrained regime, tokenizer design choices can substantially affect cost and efficiency, yet they receive far less attention than model architecture or training data.

Tokenization operates at the entry point of every LLM pipeline: a tokenizer (分词器) divides text into token units, each of which is mapped to an embedding. A dominant design is byte-level Byte Pair Encoding (BPE) [2, 3], as used by GPT-style models. It begins with 256 UTF-8 bytes and iteratively merges the most frequent adjacent pairs. For Chinese, this process interacts with a distinctive property of the script: a Chinese character occupies three bytes in UTF-8 and can therefore be represented by up to three byte tokens before merging. Whether these bytes are merged into a single token depends on their frequency in the training corpus; this frequency-driven allocation of merges is the mechanism examined in this paper.

The higher tokenization cost of Chinese relative to English is well established. Ahia et al. [1] quantified this disparity across commercial APIs, showing that CJK languages require substantially more tokens per unit of meaning. Lee et al. [4] further traced the gap to tokenizer choice in an eleven-language multilingual benchmark. Other work directly addresses structural limitations of byte-level BPE: Bostrom and Durrett [5] showed that greedy merging can populate the vocabulary with low-frequency "junk" tokens, while several pruning methods [6-8] seek to remove such residues. However, these studies do not provide a frequency-bucketed, character-level account of where the additional Chinese tokens arise, particularly in the data-constrained regime faced by many practitioners. The analysis of Lee et al. [4] comes closest, but Chinese is explicitly excluded from their script-level analysis, and no character-level measurements are reported.

In this paper, we present a white-box empirical study of Chinese tokenizer design under data-constrained training. We implement BPE, WordPiece, and Unigram from scratch (approximately 1,000 lines of standard-library Python), train them on 20,000 Chinese news documents, and quantify per-character tokenization using held-out test sets, frequency-bucketed analysis, and clustered-bootstrap confidence intervals. Our contributions are as follows:

1. A character-level characterization of byte-level BPE fragmentation in Chinese: low-frequency characters are systematically split into an average of 2.04 tokens per character at an 8K vocabulary; increasing the vocabulary from 8K to 128K substantially reduces this effect at a fixed training scale (from 2.04 to 1.10), although a residual penalty remains; and the fragmentation distribution is strongly frequency-dependent.

2. A production-tokenizer comparison: the same frequency-banded pattern appears in OpenAI's tokenizers (cl100k: 2.24, slightly above our 8K implementation at 2.04) and is nearly eliminated by DeepSeek's Chinese-optimized vocabulary (1.02), establishing a spectrum from 2.24 to 1.02 that is associated with vocabulary allocation and training scale.

3. A semantic-equivalence experiment showing that the "Chinese premium" reflects tokenizer bias rather than an inherent language property: on equivalent Chinese-English sentence pairs (WMT19 news), an English-centric tokenizer uses 65% more tokens for Chinese, whereas a balanced byte-level BPE trained on 20,000 documents (10K zh + 10K en) uses 27% fewer.

4. A from-scratch implementation, described in Section 3, together with configuration guidance for Chinese-focused tokenizer design under data constraints.

Section 2 reviews related work; Section 3 describes our tokenizer implementations, metrics, and experimental setup; Section 4 presents the experiments and results; Section 5 discusses the mechanism, the role of scale, and design guidance; Section 6 concludes.

DOI: 10.54254/2755-2721/2026.37609

2. Related work

2.1. Subword tokenization foundations

Byte Pair Encoding was introduced as a data-compression technique by Gage [2] and later applied to neural machine translation by Sennrich et al. [3]. WordPiece [9, 10] selects merges using a normalized pair score, while Kudo's SentencePiece framework [11] provides a unified implementation of BPE and Unigram training; the latter is a probabilistic subword model with a principled objective [12].

2.2. Critiques of BPE

Bostrom and Durrett [5] showed that BPE is suboptimal for language-model pretraining relative to word-level and character-level baselines at the same vocabulary size, attributing the gap to greedy merges and the resulting low-frequency dead zone in the vocabulary. Rust et al. [13] demonstrated that tokenizer quality accounts for part of the performance gap in multilingual models. Mielke et al. [14] provide a comprehensive history of open-vocabulary modeling and tokenization. Recent methodological work addresses the same defect from an engineering perspective: PickyBPE [7] removes junk-producing merges during training, LiteToken [6] identifies that roughly 10% of tokens in widely used tokenizers are intermediate merge residues (including a Chinese idiom example), and Pruned BPE [8] applies post-training visibility pruning with token reallocation. Whereas these studies propose methods for removing such residues, we provide a Chinese-focused characterization of the underlying mechanism at character granularity.

2.3. Multilingual and Chinese token cost

Ahia et al. [1] measured token costs across commercial APIs and found that CJK and Indic languages pay substantially more per unit of meaning, proposing cost-normalized evaluation. On the Chinese side, a line of work expands or replaces English-centric vocabularies for Chinese LLMs — Cui et al. [15] enlarged LLaMA-style vocabularies with Chinese characters, and Si et al. [16] explored sub-character tokenization for Chinese pretrained models — while the Qwen, Baichuan, GLM, and CPM families all train Chinese-heavy tokenizers as part of their base models. A recent preprint by Ren et al. [17] tested whether Chinese prompts are more token-efficient than English for coding tasks, finding that the ratio depends on the model rather than the language (per-model ratios ranging from 0.98x to 1.28x). Closest to our study is Lee et al. [4], the first systematic benchmark of equitable tokenizers (byte-level BPE, parity-aware BPE, MYTE, and Byte Latent Transformer) across eleven Southeast Asian languages, evaluating compression, tokenizer parity, Gini coefficients, and downstream performance of 1.5B models trained from scratch. Their finding that tokenizer choice, not script type, dominates cross-lingual parity (partial $\eta^2 = 0.752$) establishes tokenization as a first-order design axis. However, their Chinese evidence is thin: Chinese comprises only 2.6% of their tokenizer training sentences and is explicitly excluded from their script analysis as the sole CJK-script language, and their metrics are language-level aggregates over aligned sentences — no character-level, frequency-bucketed analysis is performed. TREX [18] optimizes data mixtures for tokenizer training via regression, a complementary optimization objective; our focus is descriptive and design-oriented rather than prescriptive.

DOI: 10.54254/2755-2721/2026.37609

2.4. Tokenizer-free and byte-level models

A parallel line of work asks whether tokenizers are needed at all, operating directly on bytes or characters: CANINE [19] and ByT5 [20], followed by MEGABYTE [21], MambaByte [22], and the Byte Latent Transformer (BLT) [23], which matches token-based LLMs at scale using entropy-driven byte patching, and UTF8Tokenizer [24], which proposes a minimal byte-identity tokenizer. None of these studies analyze Chinese fragmentation specifically; our data quantifies what leaving subword vocabularies would cost for Chinese, informing this debate.

2.5. Token economics

Delétang et al. [25] established the formal equivalence between language modeling and lossless compression, providing the information-theoretic grounding for tokenizer efficiency metrics used in this paper.

The Chinese-specific, character-level, data-constrained design space examined here remains underexplored: existing work either measures aggregate token premiums using off-the-shelf tokenizers, proposes methods for removing merge residues, or benchmarks multilingual equity without a character-level analysis of Chinese.

3. Method

3.1. Problem definition and notation

Let T be a tokenizer mapping a text x to a sequence of token ids $T(x) = (t_1, \ldots, t_k)$. Every token id maps to a byte string (its piece), and for a lossless tokenizer the concatenation of these byte strings equals the UTF-8 encoding of x (up to the pre-tokenization normalization defined in Section 3.2). We define the fragmentation of a character as the number of tokens whose byte spans overlap the character's UTF-8 bytes. For a Chinese character, which occupies exactly three bytes in UTF-8, fragmentation ranges from 1 (a dedicated or merged token covering the whole character) to 3 (fully byte-split). Our primary metric is tokens per hanzi (TPH): the mean number of tokens covering a Chinese character, computed over a held-out corpus. We also report tokens per character (TPC) over all characters, and the compression ratio (bytes per token).

Because tokenizers are variable-length codes, TPH has an information-theoretic reading: with a vocabulary of size $|V|$, each token id carries at most $\log_2|V|$ bits under fixed-width indexing, so the per-character code length is bounded by $\mathrm{TPH} \cdot \log_2|V|$ bits. At $|V| = 128\mathrm{K}$ (17 bits per token), even a perfect 1:1 character mapping spends 17 bits per character — well above the entropy floor for Chinese text — and fragmentation inflates this further (e.g., $1.10 \times 17 \approx 18.7$ bits for low-frequency characters at 128K). This framing ties tokenizer efficiency to compression [25].

3.2. Pre-tokenization

We adopt SentencePiece-style pre-tokenization with three components: (i) whitespace normalization — whitespace runs are collapsed to single spaces; (ii) a dummy prefix — every word unit carries a leading space marker, including the first word of each sentence, so that space+word pieces can be learned uniformly (as in SentencePiece's ▁ convention); (iii) contiguous CJK runs are kept as single units, so that character-level tokenizers can learn multi-character words. We note that this is a design choice rather than a universal production convention: OpenAI's cl100k/o200k split Chinese per

DOI: 10.54254/2755-2721/2026.37609

character, and SentencePiece behavior depends on configuration; recent work proposes entropy-driven pre-tokenization to bias BPE toward more linguistically coherent Chinese boundaries [26]. Our cross-check against SentencePiece (token-count difference 2.21% on English, 5.78% on Chinese; Appendix A) verifies both sides of the pipeline against a production implementation.

3.3. Tokenizer implementations

All three tokenizers are implemented from scratch in Python using only the standard library (approximately 1,000 lines in total).

BPE

We implement byte-level BPE (base vocabulary: 256 UTF-8 bytes) and character-level BPE (base vocabulary: all distinct characters in the training corpus). Byte-level subwords were introduced by Wang et al. [27]; our implementation follows the same algorithmic family as production trainers. Training uses a linked-list corpus with incremental pair statistics and a lazy max-heap over pair frequencies, making 128K-vocabulary training on 10M+ characters feasible in pure Python. Encoding applies merges in canonical order (lowest merge rank first), identical to GPT-style tokenizers.

WordPiece

Merges are selected by the normalized score $\text{freq}(a,b)/(\text{freq}(a)\cdot\text{freq}(b))$ as in the original WordPiece [9, 10]; encoding is greedy longest-match-first against the vocabulary.

Unigram

We implement the Unigram language model [12] in byte-level mode: a seeded vocabulary of byte n-grams is trained with the EM algorithm in log space (expected counts via forward computation, score re-estimation, and pruning of the least useful pieces down to the target vocabulary size), and inference uses Viterbi decoding for the maximum-probability segmentation. A byte floor keeps never-seen bytes encodable, preserving the no-OOV guarantee.

3.4. Metrics

We report intrinsic tokenizer-level metrics, following Petrov et al. [28], who argue that tokenization effects are best isolated at the token level before downstream confounds are introduced, and the information-theoretic framework of Delétang et al. [25]. Aggregate fertility-style metrics can mask over-fragmentation, as recently noted by Nayeem et al. [29].

Table 1. Tokenizer-level evaluation metrics

Metric	Definition
Tokens per hanzi (TPH)	Mean number of tokens covering a Chinese character's UTF-8 bytes
Fragmentation distribution	Share of characters covered by 1 / 2 / 3 tokens
Tokens per character (TPC)	Total tokens / total characters
Compression ratio	Text bytes / tokens

DOI: 10.54254/2755-2721/2026.37609

Table 1. (continued)

Frequency bands	Characters grouped by frequency in the training corpus: 0–100, 100–1000, 1000–10000, and 10000+ (approximately decade-spaced; the 0–100 band isolates the long tail)
95% CI	Clustered bootstrap over documents (Section 3.6)

3.5. Corpus and experimental setup

We train on THUCNews [30], a public Chinese news corpus containing 50,000 documents. Of these, 20,000 documents are used for training and 2,000 held-out documents for testing, selected through a single seeded shuffle to ensure that the training and test sets are strictly disjoint. Documents are filtered to lengths of 20–1,000 characters. For the semantic-equivalence experiment (Section 4.4), we additionally use the WMT19 Chinese-English news test set (newstest2019; 3,738 clean pairs) [31] and Wikitext-103 [32]. The balanced tokenizer in that experiment is trained on 10K Chinese documents from THUCNews and 10K English documents from Wikitext-103 using the same cleaning pipeline.

The main experiment varies two design factors: tokenizer mode (byte-level versus character-level BPE) and vocabulary size (8K, 32K, or 128K), yielding six conditions. All conditions use the same corpus, cleaning pipeline, and metric code; only the tokenizer configuration differs. Random seeds are fixed to ensure reproducibility.

3.6. Statistical method

Characters within one document are not independent, so we use a clustered bootstrap: documents are resampled with replacement (1,000 iterations, fixed seed), and the bootstrap distribution of the document-level mean TPH yields the 95% confidence interval. With 2,000 test documents (approximately 840K character observations, before the per-condition lossless-encoding exclusions of Table 2), intervals are narrow. The production-anchoring run (Section 4.2) uses a separate 500-document test split (218K characters).

4. Experiments and results

4.1. E1: byte-level versus character-level BPE across vocabulary sizes

Table 2 reports TPH values with 95% confidence intervals for all six conditions.

Table 2. Mean tokens per hanzi (TPH) with clustered-bootstrap 95% CI

Condition	TPH	95% CI
Byte-level, 8K	1.028	[1.027, 1.028]
Byte-level, 32K	1.005	[1.004, 1.005]
Byte-level, 128K	1.001	[1.001, 1.001]
Character-level, 8K / 32K / 128K	1.000	[1.000, 1.000] †

† By construction: every character observed in training is a base vocabulary entry, so TPH is exactly 1.0 and has no variance across bootstrap resamples. Point estimates in this table are

DOI: 10.54254/2755-2721/2026.37609

unweighted means of per-document TPH (documents are the bootstrap clusters); Table 3 reports character-weighted statistics. Documents that fail lossless encoding are excluded from the per-document statistics (Section 3.6): 51 of 2,000 held-out documents contain characters absent from the character-level training vocabulary (char-level rows), while the byte-level rows exclude only 1 lossless failure. The identity floor applies to the fully encodable subset.

Table 3 breaks down TPH by character-frequency band, which constitutes the central result of this paper.

Table 3. TPH by frequency band (training-corpus frequency). Band-level means with clustered-bootstrap 95% CIs (resampling documents, character-weighted statistic)

Vocabulary	0–100 (rare)	100–1000	1000–10000	10000+ (frequent)
Byte 8K	2.04 [2.04, 2.05]	1.23 [1.22, 1.24]	1.02 [1.01, 1.02]	1.00 [1.00, 1.00]
Byte 32K	1.41 [1.39, 1.44]	1.03 [1.02, 1.03]	1.00 [1.00, 1.00]	1.00 [1.00, 1.00]
Byte 128K	1.10 [1.08, 1.11]	1.00 [1.00, 1.01]	1.00 [1.00, 1.00]	1.00 [1.00, 1.00]
Character (all)	1.00	1.00	1.00	1.00

Table 3 highlights two findings. First, the low-frequency effect is pronounced and strongly frequency-dependent: characters appearing fewer than 100 times in the training corpus are split into an average of 2.04 tokens per character at an 8K vocabulary, more than twice the rate of frequent characters (1.00). Second, vocabulary size matters at a fixed training scale: increasing the vocabulary 16-fold, from 8K to 128K, reduces the rare-band mean from 2.04 to 1.10, an improvement of 0.94, although a residual penalty remains. The frequency gradient is monotonic: even the 100–1000 band shows a measurable penalty at smaller vocabularies (1.23 at 8K), while the fragmentation distribution in Table 4 shows that, at 8K, 2.5% of all character occurrences are split into two tokens and 0.1% into three.

Table 4. Fragmentation distribution at 8K / 32K / 128K vocabulary (byte-level), share of character occurrences

Vocab	1 token	2 tokens	3 tokens
8K	97.4%	2.5%	0.1%
32K	99.5%	0.4%	0.0%
128K	99.9%	0.1%	0.0%

Table 5 reports tokens per character and compression ratio. Byte-level BPE achieves compression comparable to that of character-level BPE (3.75 versus 3.51 at 8K, converging at larger vocabulary sizes), because its merge budget is largely allocated to genuine character-level merges rather than being wasted.

Table 5. Tokens per character and compression ratio

Condition	TPC	Compression (bytes/token)
Byte 8K	0.750	3.75
Byte 32K	0.596	4.71

DOI: 10.54254/2755-2721/2026.37609

Table 5. (continued)

Byte 128K	0.506	5.55
Character 8K	0.800	3.51
Character 32K	0.596	4.71
Character 128K	0.506	5.56

Figures 1–3 visualize these results (compression curves, frequency-band means, and fragmentation distribution).

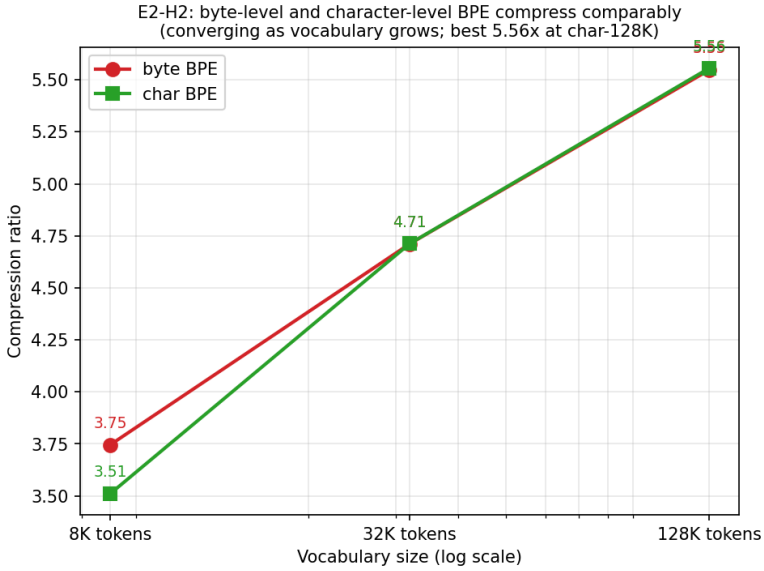

Figure 1. Vocabulary size vs. compression ratio (byte-level vs. character-level)

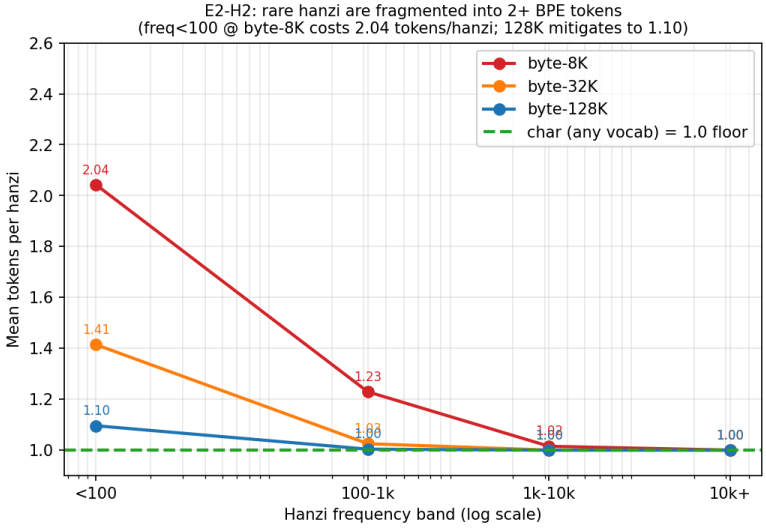

Figure 2. Tokens per hanzi by frequency band and vocabulary size

DOI: 10.54254/2755-2721/2026.37609

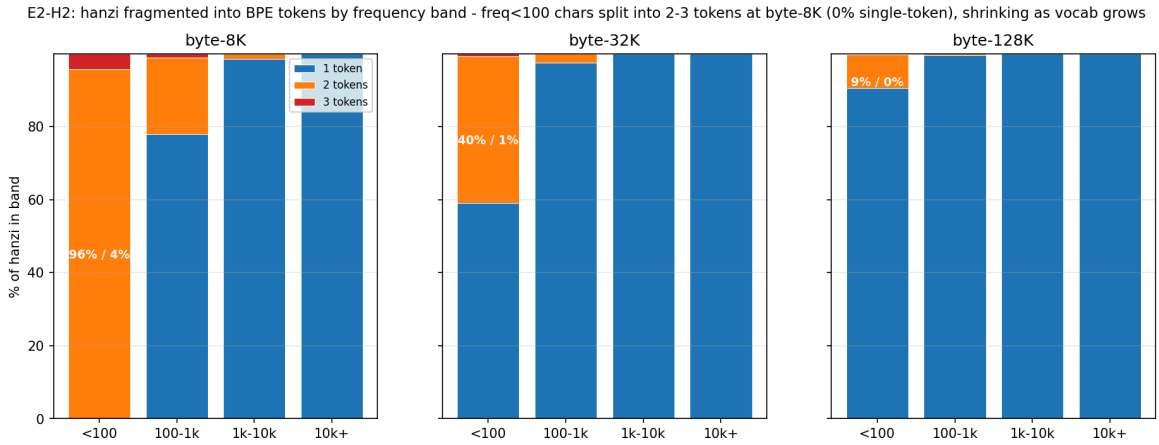

Figure 3. Fragmentation distribution (1/2/3 tokens per character)

4.2. E2: production anchoring

To rule out implementation artifacts, we measure the same frequency-banded statistics using production tokenizers on the same Chinese corpus: OpenAI's cl100k and o200k tokenizers (via tiktoken) and DeepSeek-V3's open-source tokenizer (vocabulary size: 129,283). Table 6 and Figure 4 present the full spectrum of rare-band means.

Table 6. Rare-character band (0–100) mean tokens per hanzi across tokenizers. Production encoders are deterministic; band means are exact statistics over the 218K-character anchoring split. Bands are defined by frequencies in our training corpus, so a character DeepSeek would treat as frequent is still labeled "rare" if it is rare in THUCNews; this is a deliberate, conservative choice for comparing tokenizers on identical content, but it does not reflect each tokenizer's own training distribution. The first row uses the 20K-document training frequencies; the other rows use a separate 3K-document frequency table drawn from the same corpus, so band assignments differ slightly across rows (rare-character sets are larger under the 3K table)

Tokenizer	Rare-band TPH
Ours, byte-level 8K	2.04
cl100k (English-centric)	2.24
o200k (balanced)	1.47
DeepSeek 128K (Chinese-optimized)	1.02

Production tokenizers exhibit the same frequency-dependent pattern (cl100k: 2.24), indicating that the effect is structural rather than an artifact of our implementation. Both cl100k and our 8K implementation fragment nearly all rare characters. The difference in their means (2.04 versus 2.24) reflects the proportion of rare characters split into three tokens rather than two; however, the two values are not directly comparable because they use different band definitions and test splits (Table 6). DeepSeek's tokenizer nearly eliminates the effect (1.02): among 218K test characters, only 272 (0.12%) are split. Of these, all 170 characters that fall within our frequency bands belong to the rare band, while the remaining 102 are absent from our training-frequency table and are therefore not assigned to any band. According to its technical report, DeepSeek-V3 uses standard byte-level BPE. Its low fragmentation is consistent with Chinese-heavy training data at massive scale, together with

DOI: 10.54254/2755-2721/2026.37609

a large vocabulary, although these factors cannot be separated for this external system. These results indicate that the token tax—the additional tokens incurred by Chinese text—is governed by vocabulary allocation and scale interacting with the script's byte encoding, rather than by an inherent property of Chinese.

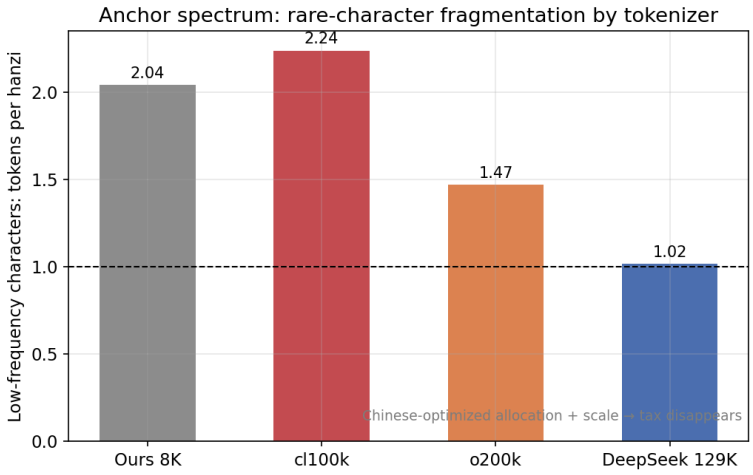

Figure 4. Production anchoring spectrum — rare-character fragmentation by tokenizer

4.3. E3: three-algorithm comparison

Table 7 compares BPE, WordPiece, and Unigram on the Chinese corpus at an 8K vocabulary (5,000 training / 500 test documents — a smaller scale than the main experiment to keep WordPiece's training time tractable, so Table 7 and Table 2 are not directly comparable).

Table 7. Three-algorithm comparison (Chinese, 8K vocabulary; 5,000 training / 500 test documents — smaller than the main experiment)

Algorithm	TPH	Rare band (0–100)	Compression (bytes/token)	Training time
BPE (byte-level)	1.029	1.66 [1.63, 1.68]	3.72	16 s
WordPiece	1.001	1.00 [1.00, 1.00]	2.35	2,411 s
Unigram (byte-level)	1.025	1.81 [1.78, 1.83]	3.47	304 s

WordPiece's TPH (1.001) lies at the identity floor for the encodable subset because its character-level base vocabulary covers all hanzi in the training vocabulary. The 71 out-of-vocabulary test characters (0.03% of 205,640) fall back to one placeholder token per UTF-8 byte (three per character), slightly increasing the measured fragmentation. This advantage comes with the poorest compression (2.35 bytes/token) and the longest training time (2,411 s in our implementation, which rescans the corpus after every merge; an incremental implementation would be faster). BPE's rare-band TPH (1.66) is lower than Unigram's (1.81): greedy, frequency-driven merging allocates the budget directly to the highest-frequency pairs, so the marginal-likelihood objective of Unigram provides no rare-band advantage on this corpus. Unigram's compression ratio (3.47) also remains below that of BPE (3.72).

DOI: 10.54254/2755-2721/2026.37609

4.4. E4: semantic equivalence — is Chinese really more expensive

To test the claim that "Chinese costs more" directly, we encode equivalent Chinese and English sentences (WMT19 newstest2019, 3,738 aligned pairs) with the *same* tokenizer on both sides, and compute $r = \text{tokens(zh)}/\text{tokens(en)}$ per pair.

Table 8. Chinese/English token ratio on equivalent content (same tokenizer, both languages). The two tokenizers differ in vocabulary size and training scale as well as allocation; the contrast isolates the direction of the effect, not its decomposition

Tokenizer	Mean ratio r	95% CI
Ours, balanced byte-level BPE 32K (trained on 10K zh + 10K en)	0.725	[0.719, 0.732]
cl100k (English-centric)	1.652	[1.638, 1.667]

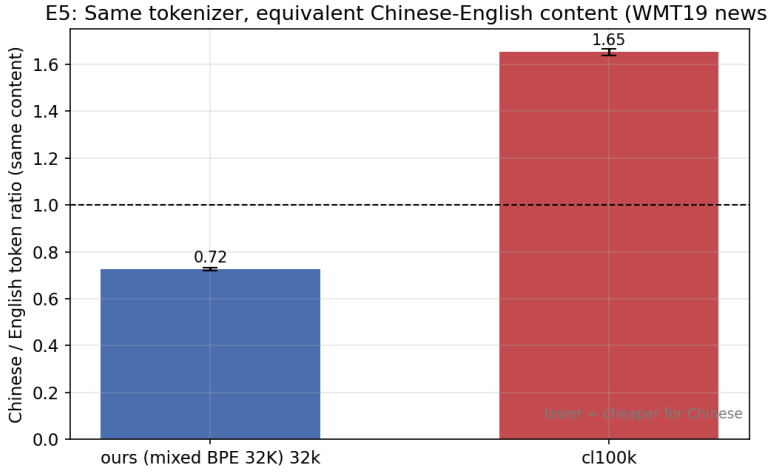

Figure 5. Chinese/English token ratio — the premium is tokenizer bias

The result directly addresses the central question: the premium depends on the tokenizer. An English-centric production tokenizer uses 65% more tokens for Chinese equivalents, consistent with the range reported by Ahia et al. [1], whereas a balanced byte-level BPE trained on 20,000 documents (10K zh from THUCNews + 10K en from Wikitext-103) uses 27% fewer tokens for Chinese. Two caveats qualify this reversal. First, the two tokenizers differ simultaneously in pre-tokenization (our CJK-run-keeping convention versus cl100k's per-character splits; Section 3.2), data balance (10K zh + 10K en versus English-centric training), vocabulary size (32K versus 100K+), and training scale (20K documents versus trillions of tokens). The comparison therefore establishes the direction of the effect rather than attributing it to any single factor; a controlled comparison that varies one factor at a time is left to future work. Second, the English side of our balanced tokenizer is trained on Wikitext-103, whereas the test sentences are drawn from news. Part of the reduction may therefore reflect English-side domain mismatch rather than Chinese information density alone. The reversal would disappear only if news-domain English training reduced English token counts by more than 27.5% relative to Wikitext-103; this was not measured directly. Moreover, the Chinese side is trained and tested within the same domain (news), whereas the English side is not, creating an asymmetry that may bias the ratio in our favor. The "Chinese premium" is therefore better interpreted as a consequence of tokenizer training bias rather than a

DOI: 10.54254/2755-2721/2026.37609

fixed property of the language, consistent with the model-dependent token ratios reported in [17] and more pronounced in our byte-level setting.

Idioms. As an exploratory check, we tested whether four-character idioms receive whole-token treatment. Across 40 idioms (20 common and 20 rare), none of the tested tokenizers (DeepSeek, cl100k, or ours) encodes them as single tokens. Instead, idioms are segmented into character- or word-level pieces, with a modest rare-idiom penalty of 0.15–0.40 tokens per idiom across tokenizers (point estimates without confidence intervals; the common/rare split follows frequency in our training corpus). For these vocabularies, frequency-driven merging therefore tends to stop at approximately the two-character-word level, leaving the model to compose idiomatic meaning across multiple tokens.

5. Discussion

5.1. Mechanism

Our results are consistent with a frequency-driven merge-competition account of the rare-band effect. Starting from 256 byte tokens, byte-level BPE spends its merge budget on the most frequent adjacent pairs; a Chinese character's three bytes merge into a single token only if their pairwise frequencies clear the competition. High-frequency characters do this reliably, while low-frequency characters — rare in the training data — remain partially or fully byte-split. This mirrors the long-tail corollary of Bostrom and Durrett's dead-zone analysis [5]: greedy, frequency-driven vocabulary construction starves the characters that need dedicated tokens most. The same competition plausibly extends beyond characters to rare words and proper nouns, which also compete for merge budget.

5.2. Scale masks, but does not solve

The DeepSeek anchor (1.02) may suggest that the problem has been solved. We argue instead that it is largely masked by scale: (i) even DeepSeek splits genuinely rare characters (Section 4.2), and any domain shift, such as toward classical texts, dialects, or specialized terminology, introduces new low-frequency units that again receive insufficient merge allocation; (ii) this performance is associated with training on trillions of Chinese tokens, a scale unavailable to most practitioners; and (iii) enlarging the vocabulary at a fixed training scale is helpful but costly. A 16-fold increase reduces the rare-band TPH by 0.94 (from 2.04 to 1.10; Section 4.1) while increasing the embedding-table size by a factor of 16, and a residual penalty of 1.10 remains. Algorithm choice is not a stronger lever in our measurements: at the same 8K vocabulary and reduced data budget (5,000 documents; Section 4.3), BPE already outperforms Unigram on the rare band (1.66 versus 1.81). Scale can obscure structural limitations, but understanding the underlying mechanism remains important in settings where large-scale training is not feasible.

5.3. Design guidance for data-constrained Chinese tokenizers

Vocabulary allocation appears to be more effective than vocabulary size in reducing Chinese character fragmentation under data-constrained training. At a fixed training scale, allocating more vocabulary capacity to Chinese—through Chinese-heavy training data or explicit character seeding—may reduce fragmentation more efficiently than simply enlarging the vocabulary, although this remains a hypothesis because the comparisons in Sections 4.2 and 4.4 vary allocation, vocabulary size, and training scale simultaneously. This interpretation is consistent with both DeepSeek's

DOI: 10.54254/2755-2721/2026.37609

Chinese-optimized tokenizer and the E4 results, which show that the Chinese token premium depends strongly on tokenizer design. For Chinese-only applications, character-level or Chinese-seeded BPE therefore provides a robust default, as its 1.0 TPH identity floor eliminates fragmentation for characters represented in the training vocabulary, albeit at the cost of byte-level out-of-vocabulary coverage. When byte-level tokenization is required for multilingual or open-vocabulary settings, vocabulary size becomes an important design parameter: increasing the vocabulary from 8K to 128K reduces rare-band TPH from 2.04 to 1.10, though this improvement comes with a 16-fold increase in embedding-table parameters at a fixed embedding width, and the remaining penalty is concentrated in rare characters and proper nouns. At the smaller scale tested in this study (8K vocabulary and 5,000 training documents), byte-level BPE also performs better than Unigram in both rare-band TPH and compression while training substantially faster in our implementation, whereas WordPiece reaches the identity floor at the expense of poorer compression and substantially longer training time. Taken together, these results suggest that tokenizer design should balance vocabulary allocation, vocabulary size, coverage requirements, and computational cost rather than optimizing any single factor in isolation.

5.4. Limitations

Our study is intrinsic: we do not train downstream models, and we rely on compression- and tokenization-level metrics, which are directly observable and independent of downstream quality. We leave downstream validation to future work. The main experiments use a single domain (news); the semantic-equivalence result is likewise domain-conditioned. Our Unigram comparison is at a single vocabulary size and training scale. Three further caveats apply. First, our pre-tokenization (Section 3.2) is not ablated: the CJK-run-keeping convention affects fragmentation, and different pre-tokenizers could shift the numbers, though the frequency-ordering of the effect is unlikely to change. Second, our from-scratch tokenizers are cross-checked against SentencePiece on both English (2.21% token-count difference) and Chinese (5.78%, Appendix A); the Chinese-side check uses the same pre-tokenization convention as the main experiments. All main experiments use a single corpus (THUCNews news) and seed=1; the confidence intervals reflect within-corpus sampling and do not cover corpus- or seed-level variation. Third, the frequency-band boundaries (Section 3.4) are a choice; we report them roughly decade-spaced, and the rare band is the smallest sample (1,322 of 1,999 test documents contain rare characters, after the byte-level rows' single lossless exclusion), so its confidence interval is the widest of the four bands on unrounded values.

6. Conclusion

This paper presents a white-box empirical study of Chinese tokenizer design under data-constrained training. From-scratch implementations of BPE, WordPiece, and Unigram quantify the low-frequency fragmentation effect: rare characters require 2.04 tokens per character at an 8K vocabulary, falling to 1.10 at 128K under the same training scale. Comparison with production tokenizers places this effect on a spectrum from 2.24 to 1.02 (with our 8K implementation at 2.04), while equivalent-content measurements show that the "Chinese premium" depends on tokenizer design rather than being an inherent property of the language (0.73 versus 1.65). These findings provide practical configuration guidance for practitioners who cannot rely on trillion-token training runs. Because scale can mask but does not fully eliminate the underlying limitation, understanding the mechanism is especially important in data-constrained settings.

DOI: 10.54254/2755-2721/2026.37609

References

[1] Ahia, O., Kumar, S., Gonen, H., Kasai, J., *et al.* (2023). Do all languages cost the same? Tokenization in the era of commercial language models. In *EMNLP*.
[2] Gage, P. (1994). A new algorithm for data compression. *C Users Journal, 12*(2), 23–38.
[3] Sennrich, R., Haddow, B., & Birch, A. (2016). Neural machine translation of rare words with subword units. In *ACL*.
[4] Lee, K. S. J. W., Qorib, M. R., Soegeng, A. I., & Ng, H. T. (2026). Equity with efficiency: An empirical study of tokenizers for multilingual large language models.
[5] Bostrom, K., & Durrett, G. (2020). Byte pair encoding is suboptimal for language model pretraining. In *Findings of EMNLP*.
[6] Sun, Y., Yang, H., Lin, Z., & Zhang, M. (2026). LiteToken: Removing intermediate merge residues from BPE tokenizers.
[7] Chizhov, P., Arnett, C., Korotkova, E., & Yamshchikov, I. P. (2024). BPE gets picky: Efficient vocabulary refinement during tokenizer training. In *EMNLP*.
[8] Shao, K. (2026). Pruned BPE: Post-training visibility pruning and token reallocation for byte pair encoding.
[9] Schuster, M., & Nakajima, K. (2012). Japanese and Korean voice search. In *ICASSP*.
[10] Wu, Y., Schuster, M., Chen, Z., *et al.* (2016). Google's neural machine translation system.
[11] Kudo, T., & Richardson, J. (2018). SentencePiece: A simple and language independent subword tokenizer. In *EMNLP (system demonstrations)*.
[12] Kudo, T. (2018). Subword regularization: Improving neural network translation models with multiple subword candidates. In *ACL*.
[13] Rust, P., Pfeiffer, J., Vulić, I., Ruder, S., & Gurevych, I. (2021). How good is your tokenizer? On the monolingual performance of multilingual language models.
[14] Mielke, S. J., Alyafeai, Z., Salesky, E., *et al.* (2021). Between words and characters: A brief history of open-vocabulary modeling and tokenization in NLP. *Computational Linguistics*.
[15] Cui, Y., Yang, Z., & Yao, X. (2023). Efficient and effective text encoding for Chinese LLaMA and Alpaca.
[16] Si, C., Zhang, Z., Chen, Y., *et al.* (2021). Sub-character tokenization for Chinese pretrained language models.
[17] Ren, S., Shen, X., Zhou, Y., Ng, D., & Raj, A. (2026). Chinese language is not more efficient than English in vibe coding.
[18] Won, I., Yoo, H., Cho, M., Park, J., Song, H., & Lim, K. (2026). TREX: Tokenizer regression for optimal data mixture.
[19] Clark, J. H., Garrette, D., Turc, I., & Wieting, J. (2022). CANINE: Pre-training an efficient tokenization-free encoder for language representation. *TACL, 10*, 73–91.
[20] Xue, L., *et al.* (2022). ByT5: Towards a token-free future with pre-trained byte-to-byte models. *TACL, 10*, 291–306.
[21] Yu, L., Simig, D., Flaherty, C., Aghajanyan, A., Zettlemoyer, L., & Lewis, M. (2023). MEGABYTE: Predicting million-byte sequences with multiscale transformers. In *NeurIPS*.
[22] Wang, J., Gangavarapu, T., Yan, J. N., & Rush, A. M. (2024). MambaByte: Token-free selective state space model.
[23] Pagnoni, A., Pasunuru, R., Rodriguez, P., *et al.* (2024). Byte latent transformer: Patches scale better than tokens.
[24] Moryossef, A., Meister, C., Stepachev, P., & Elliott, D. (2025). Back to bytes: Revisiting tokenization through UTF-8.
[25] Delétang, G., Ruoss, A., Duquenne, P.-A., *et al.* (2023). Language modeling is compression.
[26] Hu, Y., *et al.* (2025). Entropy-driven pre-tokenization for byte-pair encoding. In *ICML 2025 Tokenization Workshop*.
[27] Wang, C., Cho, K., & Gu, J. (2020). Neural machine translation with byte-level subwords. In *AAAI*.
[28] Petrov, A., Torr, P. H. S., & Bibi, A. (2024). Tokenization is more than compression.
[29] Nayeem, M. T., Alqahtani, S., Laskar, M. T. R., Mohiuddin, T., & Bari, M. S. (2025). Beyond fertility: Analyzing STRR as a metric for multilingual tokenization evaluation.
[30] Sun, M., *et al.* (2016). THUCTC: An efficient Chinese text classifier.
[31] Barrault, L., et al. (2019). Findings of the 2019 conference on machine translation (WMT19). In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)* (pp. 1–61).
[32] Merity, S., Xiong, C., Bradbury, J., & Socher, R. (2016). Pointer sentinel mixture models.

DOI: 10.54254/2755-2721/2026.37609

Appendix A. Sentencepiece cross-check

To verify that our from-scratch implementations are consistent with a production reference, we compare token counts with SentencePiece BPE on English (Wikitext-103) and Chinese (THUCNews) using 5,000 training documents, 1,000 test documents, an 8K vocabulary, and character-level BPE with byte fallback and nmt_nfkc normalization. The token-count differences are 2.21% for English (SentencePiece: 68,321 tokens; ours: 69,831) and 5.78% for Chinese (SentencePiece: 377,142; ours: 398,958), with a lossless round trip on 200 sampled English documents. The Chinese test set contains characters absent from the training vocabulary, so lossless coverage applies only to the encodable subset, consistent with the character-level exclusion used in the main experiments. This cross-check is performed at the token-count level; the reference and our training samples are drawn from the same corpus using different seeds rather than from a strictly disjoint split.