Same Number, Cheaper Token: Exploiting Latent Tokenizer Asymmetry via Numeral Script Substitution

Anonymous ACL submission

Abstract

While analysing tokenization behaviour across twelve numeral scripts under both GPT-family BPE and Gemini-family SentencePiece tokenizers, we discover that Bengali digit sequences require fewer tokens than their English equivalents specifically in Gemini. A 4-digit number costs four tokens in English but two or three in Bengali. The effect compounds over numerically dense documents, yielding measurable cost reduction with no loss in task performance. We formalise this as the *Frequency Suppression Hypothesis*: high individual token frequency suppresses digit-pair merges via the unigram pruning objective, while low frequency combined with high numeric clustering enables them, a condition that Bengali alone appears to satisfy among the twelve scripts examined. This behavior can be explained by differences in training corpus composition and numeric clustering across scripts. We introduce a simple lossless substitution layer that replaces English digits with Bengali equivalents before any Gemini API call and reverses the substitution on output. Evaluated across seven numerical reasoning benchmarks, our method achieves up to 12.2% total token reduction and 4.6% cost reduction with no statistically significant change in task accuracy and negligible thinking token overhead. Code will be publicly released.

1 Introduction

Large language model inference is billed by the token, and token costs accumulate quickly at production scale. For numerically dense documents such as financial reports, scientific tables, and accounting records, the problem is compounded: numbers can account for a substantial fraction of every prompt, and unlike natural language text, they resist compression through paraphrasing or summarisation (Jiang et al., 2023). A single earnings table may contain hundreds of multi-digit figures that must be preserved exactly. The tokenization inequality literature has documented that non-English text costs more tokens than English equivalents (Petrov et al., 2023; Ahia et al., 2023), yet these studies aggregate over entire languages and do not ask a more precise question: why does the same number, written in two different numeral scripts, produce different token counts, and can that difference be exploited?

Digits occupy a special position in this question. Unlike words, numeral systems share a common semantic structure: each provides a bijective mapping from symbols to the digits zero through nine, with no translation ambiguity, no morphological variation, and no information loss when substituting one script for another. Any token count difference between scripts is therefore attributable entirely to tokenizer behaviour rather than linguistic content, making numerals a uniquely clean object of study.

We analyse twelve numeral systems under both GPT-family BPE and Gemini-family SentencePiece tokenizers, discovering a consistent asymmetry in the latter: Bengali digit sequences require fewer tokens than English equivalents. The prevailing narrative frames non-Latin scripts as categorically more expensive than English, with Bengali identified as carrying a tokenization penalty of three to four times that of English across popular models (Churchill and Skiena, 2026). Our finding inverts this for digit sequences. We show this is not arbitrary but aligns with the unigram pruning objective interacting with corpus-level frequency and numeric clustering, which we formalise as the *Frequency Suppression Hypothesis* (Section 4). Building on this, we introduce a lossless substitution method that exploits the asymmetry at the API level and evaluate it across seven numerical reasoning benchmarks (Section 6).

Our contributions are:

- A tokenization analysis of twelve numeral systems under two tokenizer architectures, revealing cross-script asymmetries that depend on architecture and corpus composition (Section 3).

- The Frequency Suppression Hypothesis, a mathematically grounded account of digit-pair merge survival under unigram pruning, derived and validated through analysis.

- A lossless digit substitution method requiring no model modification or prompt engineering, with up to 12.2% token reduction and preserved task accuracy across seven multilingual benchmarks (Sections 5 and 6).

2 Related Work

Subword tokenization foundations. The three dominant subword algorithms differ in how they build a vocabulary from corpus statistics. Byte Pair Encoding/BPE (Sennrich et al., 2016) greedily merges the most frequent adjacent pair at each step, producing a deterministic sequence driven by co-occurrence frequency. SentencePiece (Kudo and Richardson, 2018) is whitespace-agnostic and supports the unigram language model (Kudo, 2018), which prunes tokens whose removal minimally reduces corpus likelihood, an objective that operates on characters rather than byte n-grams. WordPiece (Devlin et al., 2019) maximises the LM likelihood directly. BPE-dropout (Provilkov et al., 2020) showed that stochastic resegmentation improves robustness.

Multilingual tokenization inequality. The systematic token-count penalty on non-English text is well established. Rust et al. (2021) showed that high fertility in mBERT directly predicts downstream underperformance. Petrov et al. (2023) measured up to $15\times$ disparity for the same passage across languages. Ahia et al. (2023) extended this to a commercial API audit, showing low-resource language users are overcharged and underserved together. Lundin et al. (2026) regressed accuracy against fertility across ten models, estimating an 8–18 percentage-point accuracy drop per additional token per word, and the $O(n^2)$ attention cost compounds this at sequence length. Ali et al. (2024) showed tokenizer choice explains 20–50% of cross-lingual accuracy variance and introduced parity as a complement to fertility. Chai et al. (2024) confirmed from a robustness angle that subword boundaries shift unpredictably under typographic perturbation and that scale only partially compensates. Nayeem et al. (2025) proposed the Single Token Retention Rate as a metric that captures whether high-frequency words survive as atomic tokens.

Script-specific and cross-architecture tokenization. The fairness literature mostly aggregates over languages without isolating script properties or comparing tokenizer architectures. Multi-byte UTF-8 characters of non-Latin scripts compete in BPE's merge race at far lower frequency than equivalent English byte sequences, making cascade merges structurally harder regardless of vocabulary size; unigram pruning is less sensitive to this because it initialises at the character level (Bostrom and Durrett, 2020). English digit dominance in large web crawls is well documented in corpus analyses such as OSCAR (Suárez et al., 2019), which show that English-language text saturates Common Crawl far beyond any single non-Latin script. The production gap between GPT-family tiktoken (BPE, 100k–200k vocabulary) and Gemini-family SentencePiece (unigram, 262k vocabulary, byte fallback (Team et al., 2024)) instantiates this architectural divergence at scale. Transliteration has been explored as a workaround for Ukrainian and Arabic (Maksymenko and Turuta, 2025), reducing token counts at the cost of a semantic mismatch penalty, a fix that treats symptoms rather than causes.

Numeral tokenization and numeric representation. Subword vocabularies are poorly matched to number structure because they assign tokens based on surface frequency rather than place value. Wallace et al. (2019) showed that subword embeddings capture only coarse numeric magnitude, and Nogueira et al. (2021) documented systematic arithmetic failures when operand tokenization varies across instances. Singh and Strouse (2024) established a direct link between digit tokenization granularity and arithmetic accuracy in frontier models, motivating closer attention to how number sequences are segmented. Schwartz et al. (2024) addressed this with NumeroLogic, prepending digit count to numbers for causal place-value access, while Golkar et al. (2023) took a more fundamental approach in xVal, embedding each number as a single continuous token scaled by its value.

Byte-level and character-level alternatives. ByT5 (Xue et al., 2022) and CANINE (Clark et al., 2022) demonstrated that byte- and character-level models match or exceed subword models on noise-sensitive and low-resource tasks. Charformer (Tay et al., 2021) learned soft token boundaries over character n-grams end-to-end, and the Byte Latent Transformer (Pagnoni et al., 2025) showed

that dynamic entropy-based patching can match tokenized LLM performance at scale.

3 Cross-Script Numeral Tokenization

3.1 Languages and Setup

We analyse twelve numeral systems spanning seven script families: Latin (English), Indic (Bengali, Hindi, Telugu, Tamil), Semitic (Arabic-Indic), Southeast Asian (Thai, Khmer, Myanmar), CJK (Chinese), Sino-Tibetan (Tibetan), and Mongolic (Mongolian). This selection covers the major non-Latin digit inventories present in large multilingual web corpora and includes scripts that encode digits as single bytes (Latin), two-byte codepoints (Arabic-Indic, Thai, Khmer, Myanmar, Chinese), and three-byte codepoints (Bengali, Hindi, Telugu, Tamil, Tibetan, Mongolian), providing coverage across all UTF-8 byte widths.

For each script we enumerate every integer from 0 to 9999 in its native numeral representation and record the token sequence produced by each tokenizer. BPE tokenization uses the `o200k_base` vocabulary via `tiktoken`, which is the encoding shared by GPT-5.x and o-series models (Singh et al., 2025). SentencePiece tokenization uses the built-in `countTokens` API of Gemini 2.5 Flash. Gemini $\geq$ 2.0 and Gemma 3 share an identical SentencePiece tokenizer (Kudo and Richardson, 2018) trained with the unigram language model algorithm (Kudo, 2018), with a vocabulary of 262,048 entries and byte-level fallback encodings for out-of-vocabulary characters (Team et al., 2023), so results generalise across both model families.

3.2 Why 9999 Is the Natural Ceiling

The range 0–9999 is not an arbitrary cutoff; it is the point beyond which further enumeration yields no new information about merge behaviour under either tokenizer family.

Under BPE, English achieves the deepest merges of any script: one-digit, two-digit, and three-digit sequences each tokenize to a single token, while four-digit sequences require exactly two tokens with no variance (Table 1). The transition at four digits establishes an upper bound for all scripts. Any n-digit number with $n \geq 5$ must contain a four-digit contiguous substring; since no four-digit sequence forms a single BPE token even for English, no five-digit sequence can either. Because English represents the ceiling for BPE and stalls at three digits, the analysis window 1–4 covers every distinct merge behaviour observable in this tokenizer family.

Under SentencePiece the argument is symmetric. Inspecting Table 1, every script reaches a minimum token count strictly greater than one for four-digit sequences, including Bengali, which has the lowest average at 2.85 tokens and a minimum of two. No four-digit numeral in any script is a single token.

3.3 Tokenization Patterns Across Scripts

Table 1 reports average token count and min–max range for sequence lengths one through four under both tokenizers. Several structural patterns emerge.

Under BPE, scripts divide into three tiers. English occupies the top tier alone, achieving single-token representation for all one-, two-, and three-digit sequences. The middle tier covers most Indic, Semitic, and Southeast Asian scripts, which achieve single-digit tokens but show fractional averages for longer sequences (few pairs, triplets merge). Telugu, Tamil, Tibetan, and Mongolian fragment even at the single-digit level due to multi-byte codepoints that do not survive BPE merges at any frequency.

Under SentencePiece the tier structure shifts notably. Tibetan recovers to single-token digits. Thai and Khmer resolve to perfectly deterministic counts across all lengths. Telugu, Tamil, and Mongolian remain fragmented throughout.

Bengali is the only script that consistently outperforms English under SentencePiece across all sequence lengths greater than one. At four digits, English averages 4.00 tokens with zero variance while Bengali averages 2.85 with a minimum of two. The same comparison under BPE runs in the opposite direction: Bengali averages 3.63 against English's 2.00. This cross-architecture inversion motivates the theoretical account in Section 4.

Scope: why Gemini. A natural question is whether other models using SentencePiece show the same Bengali advantage. We tested two representative alternatives and found neither replicates the effect. LLaMA 3.x (Grattafiori et al., 2024) migrated from SentencePiece to a BPE tokenizer via tiktoken with a 128k vocabulary. Mistral and Mixtral (Jiang et al., 2024) use a SentencePiece BPE tokenizer with a vocabulary of only 32,000 entries, which is insufficient to retain non-Latin pair tokens alongside the full Latin vocabulary. Bengali digits fall back to byte-level representation. The Bengali pair-merge advantage therefore requires three conditions simultaneously: a SentencePiece

Table 1: Average token count (and min–max range) per numeral sequence of length 1–4 under GPT-family BPE and Gemini-family SentencePiece (SP). Gray cells have zero variance (min = max) and tokenization is perfectly deterministic, indicating a structural rather than frequency-based mechanism. Teal cells mark Bengali's unique advantage under SP. It is the only script to produce noticably fewer tokens than English and other languages in that column. Orange marks anomalous cells: either sub-digit fragmentation (1-digit avg > 1) or SP performing strictly worse than BPE for the same script. See Appendix A for a byte-level analysis of individual script behaviours.

		GPT BPE				Gemini SentencePiece			
Language	**Family**	**1-dig**	**2-dig**	**3-dig**	**4-dig**	**1-dig**	**2-dig**	**3-dig**	**4-dig**
English	Latin	1.00 (1–1)	1.00 (1–1)	1.00 (1–1)	2.00 (2–2)	1.00 (1–1)	2.00 (2–2)	3.00 (3–3)	4.00 (4–4)
Bengali	**Indic**	1.00 (1–1)	1.79 (1–2)	2.63 (1–3)	3.63 (2–4)	**1.00 (1–1)**	**1.46 (1–2)**	**2.21 (1–3)**	**2.85 (2–4)**
Hindi	Indic	1.00 (1–1)	1.82 (1–2)	2.68 (1–3)	3.68 (2–4)	1.00 (1–1)	1.96 (1–2)	2.91 (1–3)	3.86 (2–4)
Telugu	Indic	2.00 (2–2)	4.00 (4–4)	6.00 (6–6)	8.00 (8–8)	2.80 (1–3)	5.58 (2–6)	8.38 (3–9)	11.18 (4–12)
Tamil	Indic	2.00 (2–2)	4.00 (4–4)	6.00 (6–6)	8.00 (8–8)	2.40 (1–3)	4.96 (2–6)	7.36 (3–9)	9.76 (4–12)
Arabic-Indic	Semitic	1.00 (1–1)	1.99 (1–2)	2.98 (2–3)	3.98 (3–4)	1.00 (1–1)	2.00 (2–2)	3.00 (3–3)	4.00 (4–4)
Thai	SE Asian	1.80 (1–2)	3.58 (2–4)	5.38 (3–6)	7.18 (4–8)	1.00 (1–1)	2.00 (2–2)	3.00 (3–3)	4.00 (4–4)
Khmer	SE Asian	1.00 (1–1)	1.98 (1–2)	2.96 (1–3)	3.96 (2–4)	1.00 (1–1)	2.00 (2–2)	2.99 (2–3)	3.98 (3–4)
Myanmar	SE Asian	1.00 (1–1)	1.99 (1–2)	2.98 (1–3)	3.98 (2–4)	1.00 (1–1)	2.00 (2–2)	2.99 (2–3)	3.98 (3–4)
Chinese	CJK	1.00 (1–1)	1.98 (1–2)	2.96 (2–3)	3.94 (2–4)	1.00 (1–1)	1.99 (1–2)	2.98 (2–3)	3.97 (2–4)
Tibetan	Sino-Tibetan	2.00 (2–2)	4.00 (4–4)	6.00 (6–6)	8.00 (8–8)	1.00 (1–1)	2.00 (2–2)	3.00 (3–3)	4.00 (4–4)
Mongolian	Mongolic	3.00 (3–3)	6.00 (6–6)	9.00 (9–9)	12.00 (12–12)	3.00 (3–3)	6.00 (6–6)	9.00 (9–9)	12.00 (12–12)

unigram tokenizer, a vocabulary large enough to retain non-Latin pair tokens after pruning, and sufficient Bengali numeric co-occurrence in the training corpus. Among widely deployed commercial API models, only the Gemini and Gemma families currently satisfy all three.

4 The Frequency Suppression Hypothesis

4.1 Unigram Tokenizer and Pruning Objective

Our analysis focuses on the unigram pruning objective, which determines which character sequences survive as single tokens and which are decomposed into smaller units or individual bytes.

The unigram model (Kudo, 2018) scores a sentence X by marginalising over all possible segmentations:

$$P(X) = \sum_{\mathbf{x} \in \mathcal{S}(X)} \prod_{i=1}^{|\mathbf{x}|} p(x_i) \tag{1}$$

where $\mathcal{S}(X)$ is the set of all segmentations and $p(x_i)$ is the unigram probability of token x_i. Training maximises log-likelihood over corpus $\mathcal{D}$:

$$\mathcal{L} = \sum_{s=1}^{|\mathcal{D}|} \log P\left(X^{(s)}\right) \tag{2}$$

For each candidate token t, the marginal loss from its removal is:

$$\Delta\mathcal{L}(t) = \mathcal{L} - \mathcal{L}_{\setminus t} \tag{3}$$

Tokens with $\Delta\mathcal{L}(t) < \eta$ are pruned; this repeats until the target vocabulary size is reached.

4.2 Inferring Corpus Statistics from Tokenizer Output

Since the Gemini training corpus is proprietary, we cannot measure token probabilities directly. We instead treat the observable tokenizer output as evidence: which tokens survived pruning constrains what the training distribution must have looked like. We introduce two quantities to formalise this inference.

Definition 1 (Inferred Individual Frequency Ordering)**.** *Let $\hat{p}(d)$ denote the inferred unigram probability of digit d, estimated from tokenizer output. From*

the cross-tokenizer contrast above and the universal presence of English digits across all language contexts in any large web crawl, we infer:

$$\hat{p}(e) \gg \hat{p}(d_{\mathcal{S}}) \qquad \forall\, e \in \mathcal{E},\ \mathcal{S} \in \{\mathcal{B}, \mathcal{H}, \mathcal{T}, \mathcal{K}, \ldots\} \tag{4}$$

where $\mathcal{E}$ denotes English (ASCII) digits and $\mathcal{B}$, $\mathcal{H}$, $\mathcal{T}$, $\mathcal{K}$ denote Bengali, Hindi, Thai, and Khmer digits respectively. Non-Latin digit tokens are individually weak because each script appears only in its own language context.

This low individual probability is what creates the opportunity for pair tokens to survive pruning: when $\hat{p}(d)$ is small, the product $\hat{p}(d_i) \cdot \hat{p}(d_j)$ is very small, making it easier for a pair token to appear often compared to independence predicts.

Definition 2 (Numeric Clustering Coefficient)**.** *We introduce the Numeric Clustering Coefficient (NCC) of script $\mathcal{S}$ as:*

$$\rho(\mathcal{S}) \;=\; P(c_{i+1} \in \mathcal{S} \mid c_i \in \mathcal{S}) \tag{5}$$

measuring how often a digit in $\mathcal{S}$ is immediately followed by another digit in $\mathcal{S}$ in the corpus. High ρ indicates digits appear predominantly in multi-digit runs; low ρ indicates digits appear mostly in isolation.

We cannot observe $\rho(\mathcal{S})$ from the proprietary corpus, but infer its relative ordering across scripts from which k-gram digit tokens survived pruning in the deployed tokenizer.

4.3 A Mechanistic Account of Merge Survival

We now develop a mechanistic account of the tokenization patterns observed in Table 1. The following lemma is a direct derivation from the unigram pruning objective of Kudo (2018); the hypothesis that follows uses it to explain the observed patterns across the twelve scripts.

Lemma 3 (k-gram Survival Condition)**.** *For a digit sequence $\mathbf{d} = (d_1, \ldots, d_k)$ of length $k \geq 2$ from script $\mathcal{S}$, define its pointwise mutual information (PMI; Church and Hanks, 1990) against independent segmentation:*

$$\mathrm{I}(\mathbf{d}) \;=\; \log \hat{p}(\mathbf{d}) \;-\; \sum_{i=1}^{k} \log \hat{p}(d_i) \tag{6}$$

The marginal pruning loss for the k-gram token is:

$$\Delta\mathcal{L}(\mathbf{d}) \;\approx\; \widehat{\mathrm{freq}}(\mathbf{d}) \cdot \mathrm{I}(\mathbf{d}) \tag{7}$$

The token survives pruning when $\Delta\mathcal{L}(\mathbf{d}) \geq \eta$.

Derivation 4. *Removing $\mathbf{d}$ forces every occurrence to be re-segmented as the k-token sequence $(d_1) \cdots (d_k)$. The log-likelihood change per occurrence is:*

$$\delta \;=\; \log p(\mathbf{d}) \;-\; \sum_{i=1}^{k} \log p(d_i) \;=\; \mathrm{I}(\mathbf{d}) \tag{8}$$

Summing over $\widehat{\mathrm{freq}}(\mathbf{d})$ occurrences and substituting into Eq. (3) yields Eq. (7). The approximation follows Kudo (2018) in treating re-segmentation effects on surrounding context as negligible for short sequences.

Lemma 3 shows that PMI is the decisive quantity: a pair where the two digits co-occur no more than independence predicts has $\mathrm{I} \approx 0$ and is pruned regardless of absolute frequency. Survival requires the pair to appear more often compared to independence predicts.

Under the first-order Markov approximation, the k-gram probability factors as:

$$\hat{p}(d_1 \cdots d_k) \;\approx\; \rho(\mathcal{S})^{\,k-1} \cdot \hat{p}(d_1) \tag{9}$$

Substituting into Eq. (6) and using $\hat{p}(d_i) \approx \hat{p}(d)$ within the same script:

$$\mathrm{I}(\mathbf{d}) \;\approx\; (k-1)\, \log\left[\frac{\rho(\mathcal{S})}{\hat{p}(d)}\right] \tag{10}$$

The PMI of any digit k-gram thus depends on a single ratio: $\rho(\mathcal{S})$ in the numerator and $\hat{p}(d)$ in the denominator. High clustering raises PMI; high individual frequency suppresses it.

Hypothesis 5 (Frequency Suppression and Numeric Clustering)**.** *The observed tokenization patterns across all twelve scripts are consistent with a k-gram digit token surviving vocabulary pruning when:*

$$\widehat{\mathrm{freq}}(\mathbf{d}) \cdot (k-1) \cdot \log\left[\frac{\rho(\mathcal{S})}{\hat{p}(d)}\right] \;\geq\; \eta \tag{11}$$

Specifically, Eq. (11) is consistent with the following observations in Table 1:

1. ***English, all $k \geq 2$.*** *No English digit pair or longer sequence appears as a single token in Gemini. This is consistent with $\hat{p}(e) > \rho(\mathcal{E})$: English digits appear universally across all language contexts in any large web crawl, giving $\hat{p}(e)$ large enough that the log ratio in Eq. (11) is low for all $k \geq 2$.*

2. ***Bengali,*** $k = 2$. *$\hat{p}(b)$ is low (Definition 1) and $\rho(\mathcal{B})$ is inferred to be sufficiently high that the ratio $\rho(\mathcal{B})/\hat{p}(b)$ exceeds 1, making the log term high. This is consistent with the observed mean of 1.46 tokens per 2-digit sequence, where a substantial fraction of pairs survive pruning.*

3. ***Bengali,*** $k = 3$ ***(rare).*** *As k grows, $\widehat{\mathrm{freq}}(\mathbf{d})$ decreases faster than $(k-1)\log[\rho(\mathcal{B})/\hat{p}(b)]$ grows. This is consistent with only the highest-frequency Bengali triplets surviving, as observed empirically.*

4. ***Thai and Khmer,*** $k \geq 2$. *Although $\hat{p}(t)$ and $\hat{p}(k)$ are low (Definition 1), $\rho(\mathcal{T})$ and $\rho(\mathcal{K})$ are also inferred to be low. The ratio $\rho/\hat{p}$ does not exceed 1 and no pairs survive, consistent with each digit mapping to exactly one token.*

5. ***Hindi,*** $k = 2$ ***(marginal).*** *$\hat{p}(h)$ is low (Definition 1) and $\rho(\mathcal{H})$ is inferred to be intermediate between Bengali and Thai, consistent with the observed mean of 1.96 tokens per 2-digit sequence, between Bengali (1.46) and the fully unmerged case (2.00).*

Substituting Eq. (10) into Lemma 3 yields Eq. (11) directly. Case (i) is grounded in the empirical absence of English pairs from Gemini's vocabulary. The remaining cases rest on the inferred ordering $\rho(\mathcal{B}) > \rho(\mathcal{H}) > \rho(\mathcal{T}) \approx \rho(\mathcal{K})$ compared to their respective individual digits. We validate this ordering on Wikipedia as a proxy for web-scale numeral usage (Appendix B): measured $\rho(\mathcal{S})$ values are 0.651 for Bengali, 0.618 for Hindi, and 0.577 for Thai, matching the relative ordering implied by tokenizer output. Hypothesis 5 suggests that two corpus-level properties jointly determine whether a script benefits from multi-digit token merges: the individual frequency of its digit tokens (Definition 1) and the degree to which those digits cluster into multi-digit runs (Definition 2).

Alternative explanations. Several alternative mechanisms could also produce the observed pair merges, and we try to address each. A larger vocabulary budget is unlikely to be the cause: Hindi shares the same tokenizer and vocabulary yet shows near-zero pair merges, ruling out a size-driven explanation. Manual curation of Bengali digit pairs is inconsistent with the non-zero variance in merge rates across specific pairs, which a curated fixed set would not produce. Unicode normalisation artefacts are excluded by the fact that all Bengali digit codepoints are stable across normalisation forms. Corpus contamination from Bengali financial tables cannot be fully ruled out given the proprietary training data, but would not constitute a different mechanism. It would simply be a specific instance of high $\rho(\mathcal{B})$ in the Frequency Suppression Hypothesis, and the hypothesis holds regardless of the underlying source of that clustering.

5 Digit Substitution Method

Figure 1 illustrates the full substitution pipeline.

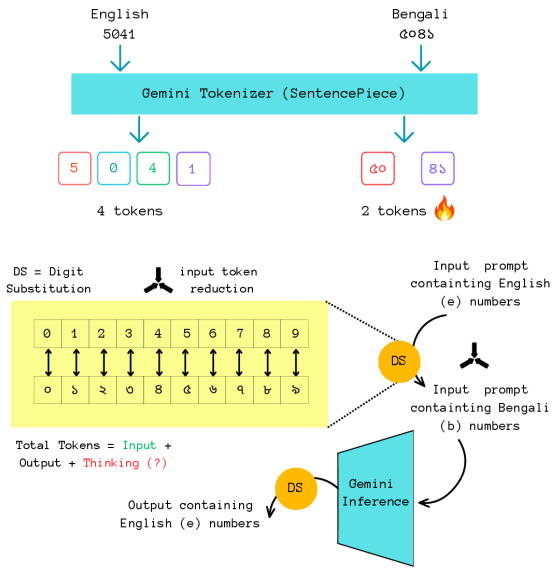

Figure 1: **Top:** The same numeral (5041) tokenizes to 4 tokens under Gemini's SentencePiece tokenizer when written in English digits, versus 2 tokens when written in Bengali digits. **Bottom:** The Digit Substitution (DS) layer intercepts both the input prompt and the generated output. Input prompts containing English numerals are converted to Bengali (a simple 4 line script) before submission to Gemini, reducing input token count. The model response, which retains Bengali numerals, is converted back to English digits before delivery to the user. Total token cost comprises input, output, and thinking tokens (we will evaluate the cost empirically).

Datasets. We evaluate on seven multilingual benchmarks covering financial, mathematical, and numerical reasoning. Table 3 summarises key statistics.

Evaluation. For free-form datasets, we use lenient numerical matching (absolute tolerance 0.5 or relative tolerance 5%). For MCQ datasets we compare predicted option letter against gold. Our goal

Table 2: Results across four models and seven benchmarks under other language and Bengali digit encodings (no document language change). A_{E} (other), A_{B} (Bengali): mean$_{\pm\mathrm{std}}$ accuracy over 5 seeds at $T = 0.2$. †: paired-TOST equivalence within $\varepsilon = 2$ pp ($p < 0.05$ both tails). ΔIn, ΔThk, ΔTot are percentage changes in input, thinking, and total tokens. ΔCost is total **cost reduction** percentage. $\bar{T}$ represents average token numbers per prompt.

	Gemini 2.5 Flash $0.30 / $2.50 per 1M						**Gemini 2.5 Pro** $1.25 / $10.00 per 1M					
Dataset	A_{E}	A_{B}	ΔIn%	ΔThk%	ΔTot%	ΔCost%	A_{E}	A_{B}	ΔIn%	ΔThk%	ΔTot%	ΔCost%
FinQA (EN)	$74.3_{\pm 0.2}$	$74.7^{\dagger}_{\pm 0.5}$	−16.2	+0.2	−12.0	+4.1	$81.2_{\pm 0.3}$	$81.5^{\dagger}_{\pm 0.2}$	−16.2	+0.1	−12.2	+4.6
TAT-QA (EN)	$75.1_{\pm 0.4}$	$74.6^{\dagger}_{\pm 0.5}$	−13.7	+3.8	−8.5	+2.0	$83.8_{\pm 0.3}$	$83.2^{\dagger}_{\pm 0.4}$	−13.7	+4.2	−8.2	+1.6
MathQA (EN)	$56.6_{\pm 0.3}$	$57.1^{\dagger}_{\pm 0.4}$	−13.9	−0.2	−2.9	+0.8	$76.3_{\pm 0.2}$	$75.9^{\dagger}_{\pm 0.2}$	−13.9	+0.3	−2.6	+1.0
IM-TQA (ZH)	$86.6_{\pm 1.6}$	$89.5_{\pm 1.4}$	−13.5	+1.5	−9.8	+1.9	$91.3_{\pm 0.6}$	$90.4^{\dagger}_{\pm 0.6}$	−13.5	+2.8	−9.3	+1.5
SAHM (AR)	$51.6_{\pm 0.6}$	$54.0_{\pm 0.7}$	−13.5	−0.2	−6.4	+2.7	$65.8_{\pm 0.4}$	$66.4^{\dagger}_{\pm 0.3}$	−13.5	−1.4	−6.1	+2.4
MULTITAT (TH)	$47.6_{\pm 1.0}$	$46.4^{\dagger}_{\pm 1.1}$	−9.2	+0.7	−6.4	+3.9	$59.6_{\pm 0.8}$	$58.8^{\dagger}_{\pm 0.9}$	−9.2	−0.8	−6.6	+4.3
MULTITAT (TE)	$46.0_{\pm 0.9}$	$44.0_{\pm 1.2}$	−8.2	−1.9	−6.5	+3.8	$60.1_{\pm 0.8}$	$58.9^{\dagger}_{\pm 0.6}$	−8.2	−1.1	−6.1	+3.6

	Gemini 3.1 Pro Preview $2.00 / $12.00 per 1M						**Gemma 3 27B** open weights‡					
Dataset	A_{E}	A_{B}	ΔIn%	ΔThk%	ΔTot%	ΔCost%	A_{E}	A_{B}	ΔIn%	ΔThk%	ΔTot%	ΔCost%
FinQA (EN)	$84.9_{\pm 0.3}$	$83.7^{\dagger}_{\pm 0.6}$	−16.2	+0.6	−11.7	+4.0	$64.9_{\pm 0.6}$	$64.7^{\dagger}_{\pm 0.7}$	−16.2	+0.0	−16.2	–
TAT-QA (EN)	$85.4_{\pm 0.3}$	$85.7^{\dagger}_{\pm 0.3}$	−13.7	+1.5	−9.3	+2.5	$61.3_{\pm 0.5}$	$62.2^{\dagger}_{\pm 0.6}$	−13.7	+0.0	−13.7	–
MathQA (EN)	$76.8_{\pm 0.2}$	$75.9^{\dagger}_{\pm 0.3}$	−13.9	+0.4	−2.7	+0.8	$44.1_{\pm 0.5}$	$43.3^{\dagger}_{\pm 0.6}$	−13.9	+0.0	−13.9	–
IM-TQA (ZH)	$93.3_{\pm 0.7}$	$93.1^{\dagger}_{\pm 0.5}$	−13.5	+2.6	−9.5	+1.8	$72.4_{\pm 0.8}$	$72.6^{\dagger}_{\pm 0.3}$	−13.5	+0.0	−13.5	–
SAHM (AR)	$68.4_{\pm 0.3}$	$67.8^{\dagger}_{\pm 0.5}$	−13.5	−0.5	−7.3	+3.3	$40.8_{\pm 0.9}$	$40.2^{\dagger}_{\pm 0.6}$	−13.5	+0.0	−13.5	–
MULTITAT (TH)	$63.6_{\pm 0.4}$	$62.4^{\dagger}_{\pm 0.7}$	−9.2	+3.6	−6.0	+3.2	$28.4_{\pm 1.2}$	$28.8^{\dagger}_{\pm 1.3}$	−9.2	+0.0	−9.2	–
MULTITAT (TE)	$66.0_{\pm 0.7}$	$66.4^{\dagger}_{\pm 0.8}$	−8.2	−0.6	−6.2	+3.9	$26.8_{\pm 1.1}$	$25.2^{\dagger}_{\pm 1.2}$	−8.2	+0.0	−8.2	–

$\Delta\mathrm{Tot\%} = (\bar{T}_{\mathrm{E}} - \bar{T}_{\mathrm{B}}) / \bar{T}_{\mathrm{E}} \times 100$. $\Delta\mathrm{Cost\%} = (\bar{C}_{\mathrm{E}} - \bar{C}_{\mathrm{B}}) / \bar{C}_{\mathrm{E}} \times 100$ where $\bar{C} = \bar{T}_{\mathrm{in}} \times P_{\mathrm{in}} + \bar{T}_{\mathrm{thk}} \times P_{\mathrm{thk}}$.

Dataset	**Lang**	**N**	**Avg**	**Format**
FinQA (Chen et al., 2021)	EN	1,147	high	Free-form
TAT-QA (Zhu et al., 2021)	EN	1,669	high	Free-form
MathQA (Amini et al., 2019)	EN	2,985	low	MCQ
IM-TQA (Zheng et al., 2023)	ZH	627	high	Free-form
MULTITAT (Zhang et al., 2025)	TE, TH	500	med	MCQ
SAHM (Elbadry et al., 2026)	AR	1,979	med	MCQ

Table 3: Evaluation datasets. Avg refers to relative digit density per question (high $>$ 50 digits/prompt, med $=$ 20–50). N is the number of test examples used.

is not to achieve the best accuracy but to show any difference in the exact same setting. We control the output via prompt to specifically provide the *answer only* which makes this token cost negligible compared to input and thinking tokens. We fix $T = 0.2$ and run 5 seeds per configuration (Gemini models do not produce fully deterministic outputs), reporting mean±std. Per-cell equivalence uses paired TOST (Schuirmann, 1981) on seed-paired differences with $\varepsilon = 2$ pp, over 3× the median within-condition seed std (0.55 pp) across all 56 cells.

6 Results

Table 2 reports accuracy, input token count, thinking token count, and cost saved under English (or other language) and Bengali digit representations across all benchmarks. Gemma 3 does not have reasoning support hence no variation in thinking tokens. Cost saved is not directly proportional to numeral density as thinking tokens have different cost models.

6.1 Token Reduction and Cost Saving

Input token reduction scales directly with prompt numeral density. FinQA, the most numerically dense benchmark tested, yields the largest per-prompt saving across all seven datasets, while MathQA, where numbers are sparse relative to reasoning text, yields a smaller but consistent reduction. Figure 2 shows this relationship on the left panel, where the regression line remains above zero across the observed density range for FinQA but stays near zero for MathQA, confirming that the benefit compounds only when digit characters are a substantial fraction of the prompt. The right panel projects these per-prompt savings to monthly API cost at scale: FinQA-style workloads exceed $50 in monthly savings at roughly 10,000 calls per day, while MathQA-style workloads require higher volume to reach the same threshold.

6.2 Thinking Token Overhead

A key concern with mixed-script prompts is whether Bengali digit substitution increases model reasoning cost by introducing an unfamiliar script.

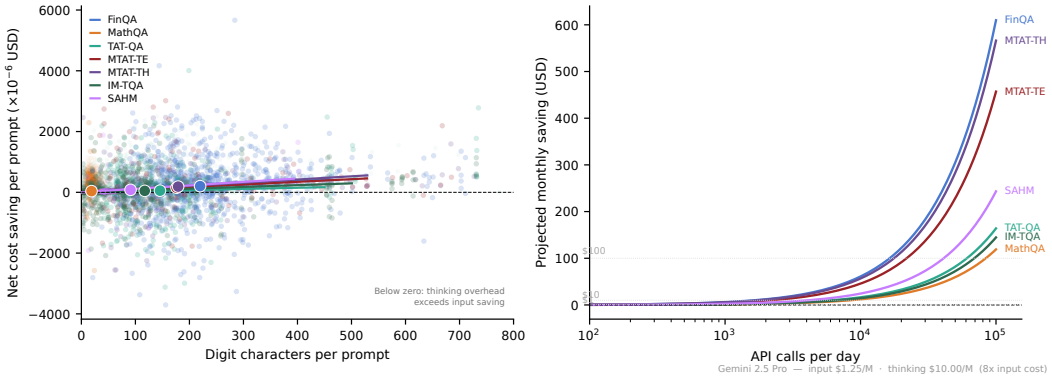

Figure 2: Left: net cost saving per prompt ($\times 10^{-6}$ USD, after accounting for thinking token overhead at Gemini 2.5 Pro pricing) against digit character density. Each point is one prompt, filled markers show dataset means. Right: projected monthly saving in USD as a function of daily API call volume. Prompts in these benchmarks hover around 200 to 1200 input tokens. Organisations processing longer requests and hundreds of millions of tokens per month will see savings scale proportionally faster than the per-call curves suggest.

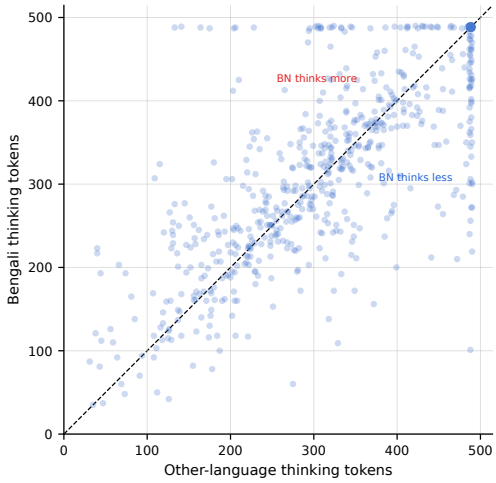

Figure 3: Thinking token counts for a random sample of matched prompt pairs across all benchmarks. Each point represents the same problem submitted in its original script (x-axis) and with Bengali digit substitution (y-axis). Points scatter on both sides of the diagonal with no systematic upward shift.

Figure 3 addresses this directly. Thinking token counts for matched prompt pairs scatter symmetrically around the diagonal, indicating that Bengali digit substitution does not systematically increase thinking depth. Across 2000 matched pairs, the median Bengali-minus-other thinking-token difference is a mere 1 token (mean +2.8; Wilcoxon signed-rank p = 0.33, effect size r = 0.03), supporting the symmetry visible in Figure 3 (Wilcoxon, 1945).

6.3 Accuracy

Accuracy is mostly preserved across all benchmarks. Per-cell paired TOST accepts equivalence at $\varepsilon = 2$ pp in 25 of 28 cells, and the pooled $A_{\mathrm{B}} - A_{\mathrm{E}}$ across all 28 paired comparisons is -0.20 pp (90% CI $[-0.51, +0.11]$, pooled TOST $p < 0.001$). All three per-cell exceptions occur on Gemini 2.5 Flash and split in direction: IM-TQA ($+2.9$) and SAHM ($+2.4$) favour Bengali while MULTITAT-TE (-2.0, at margin) favours English. Flash's pooled delta of $+0.36$ pp confirms no systematic encoding effect.The concentration of exceptions on the smallest model with mixed direction is consistent with sampling variance rather than encoding-induced degradation.

7 Conclusion

We identified a tokenization asymmetry in Gemini's SentencePiece unigram model. Bengali digit sequences require consistently fewer tokens than their English and ten other language equivalents, a consequence of corpus-level frequency suppression that we formalise as the Frequency Suppression Hypothesis. Exploiting it requires only a lossless substitution layer, which achieves up to 12% total token reduction across seven numerical reasoning benchmarks with no statistically significant accuracy change and negligible thinking token overhead. The savings scale with prompt numeral density and are most pronounced for the financial and tabular document processing tasks. The Bengali advantage is best read as an empirical consequence of that mechanism rather than a general-purpose technique. Future work should focus on whether similar corpus-statistical asymmetries exist in other deployed tokenizer families.

Limitations

The primary limitation of this work is that the observed token reduction is not a general property of language model tokenizers but a result of a specific combination: Gemini's SentencePiece unigram algorithm, its vocabulary size, and the particular composition of its training corpus that caused Bengali numeric content to cluster sufficiently for digit-pair merges to survive pruning. No other script among the twelve we examined replicates this combination under the same tokenizer. GPT-family models using BPE show no similar savings for Bengali, confirming that the effect is architecture and corpus-specific rather than universal.

The phenomenon is stable for all currently deployed Gemini and Gemma model versions, since production models are not retrained after release and their tokenizers are fixed. However, future Gemini versions may use a different tokenizer, a rebalanced training corpus, or a larger vocabulary, any of which could eliminate or shift the beneficiary script. The method should be re-validated against each new model release before deployment in cost-sensitive pipelines.

Token savings scale directly with numeral density, which limits the practical benefit to specific document types. Financial reports, accounting records, and statistical tables with dense multi-digit figures yield the largest savings. Mathematical word problems, where numbers are sparse and prompts are short, yield modest savings as shown. Applications involving primarily natural language with occasional numbers will see little to no benefit. The substitution layer adds no cost when savings are zero, so it carries no downside beyond the negligible overhead of the character translation itself.

We measure thinking token count as a proxy for reasoning overhead but do not analyse the content of generated reasoning traces. It is possible that Bengali digit representation shifts intermediate chain-of-thought steps in ways that our token-count and accuracy metrics do not capture, particularly for multi-step arithmetic problems (Wei et al., 2022). A qualitative analysis of reasoning chain content is left for future work.

For non-English datasets, the substitution replaces ASCII digits in otherwise Arabic or Chinese text, producing mixed-script prompts. While our accuracy results suggest the models handle this correctly, multilingual LLMs have been shown to underperform on code-switched inputs relative to monolingual ones, and this behaviour is not yet well understood (Zhang et al., 2023), leaving edge cases in our multilingual evaluation underexplored.

Finally, we cannot directly verify the Numeric Clustering or individual token probability values inferred from tokenizer output, as Gemini's training corpus is proprietary. The Frequency Suppression Hypothesis rests on indirect evidence and the consistency of observed tokenization patterns with the unigram pruning framework. A direct confirmation would require access to training corpus statistics that are not publicly available.

Ethical Considerations

The substitution layer operates transparently as a pre- and post-processing wrapper. No content is masked or misrepresented to the model or the end user, and the transformation is fully reversible with no effect on semantic content. The method is consistent with standard API usage. It submits valid text and receives standard responses. Bengali numerals are used strictly for token-efficient representation and carry no claim over the linguistic or cultural context of the script.

Use of AI Writing Tools. Claude Opus 4.6 (Anthropic) was used for minor formatting and code scaffolding during paper preparation.

References

Orevaoghene Ahia, Sachin Kumar, Hila Gonen, Jungo Kasai, David R Mortensen, Noah A 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.

Mehdi Ali, Michael Fromm, Klaudia Thellmann, Richard Rutmann, Max Lübbering, Johannes Leveling, Katrin Klug, Jan Ebert, Niclas Doll, Jasper Buschhoff, and 1 others. 2024. Tokenizer choice for llm training: Negligible or crucial? In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3907–3924.

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. 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 2357–2367.

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.

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.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan R Routledge, and 1 others. 2021. Finqa: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3697–3711.

Kenneth Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29.

Geoffrey Churchill and Steven Skiena. 2026. Reducing tokenization premiums for low-resource languages. *arXiv preprint arXiv:2601.13328*.

Jonathan H Clark, Dan Garrette, Iulia Turc, and John Wieting. 2022. Canine: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, 10:73–91.

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.

Rania Elbadry, Sarfraz Ahmad, Ahmed Heakl, Dani Bouch, Momina Ahsan, Muhra AlMahri, Yuxia Wang, Salem Lahlou, Sophia Ananiadou, Veselin Stoyanov, and 1 others. 2026. Sahm: A benchmark for arabic financial and shari'ah-compliant reasoning. *arXiv preprint arXiv:2604.19098*.

Siavash Golkar, Mariel Pettee, Michael Eickenberg, Alberto Bietti, Miles Cranmer, Geraud Krawezik, Francois Lanusse, Michael McCabe, Ruben Ohana, Liam Parker, and 1 others. 2023. xval: a continuous numerical tokenization for scientific language models. *arXiv e-prints*, pages arXiv–2310.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, and 1 others. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. Llmlingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 13358–13376.

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.

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.

Jessica M Lundin, Ada Zhang, Nihal Karim, Hamza Louzan, Guohao Wei, David Ifeoluwa Adelani, and Cody Carroll. 2026. The token tax: Systematic bias in multilingual tokenization. In *Proceedings of the 7th Workshop on African Natural Language Processing (AfricaNLP 2026)*, pages 103–112.

Daniil Maksymenko and Oleksii Turuta. 2025. Tokenization efficiency of current foundational large language models for the ukrainian language. *Frontiers in Artificial Intelligence*, 8:1538165.

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. *arXiv preprint arXiv:2510.09947*.

Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. 2021. Investigating the limitations of transformers with simple arithmetic tasks. *arXiv preprint arXiv:2102.13019*.

Artidoro Pagnoni, Ramakanth Pasunuru, Pedro Rodriguez, John Nguyen, Benjamin Muller, Margaret Li, Chunting Zhou, Lili Yu, Jason E Weston, Luke Zettlemoyer, and 1 others. 2025. Byte latent transformer: Patches scale better than tokens. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9238–9258.

Aleksandar Petrov, Emanuele La Malfa, Philip Torr, and Adel Bibi. 2023. Language model tokenizers introduce unfairness between languages. *Advances in neural information processing systems*, 36:36963–36990.

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

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.

DL Schuirmann. 1981. On hypothesis-testing to determine if the mean of a normal-distribution is contained in a known interval. In *Biometrics*, volume 37, pages 617–617. INTERNATIONAL BIOMETRIC SOC 808 17TH ST NW SUITE 200, WASHINGTON, DC 20006-3910.

Eli Schwartz, Leshem Choshen, Joseph Shtok, Sivan Doveh, Leonid Karlinsky, and Assaf Arbelle. 2024. Numerologic: Number encoding for enhanced llms' numerical reasoning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 206–212.

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.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, and 1 others. 2025. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*.

Aaditya K Singh and DJ Strouse. 2024. Tokenization counts: the impact of tokenization on arithmetic in frontier llms. *arXiv preprint arXiv:2402.14903*.

Pedro Javier Ortiz Suárez, Benoît Sagot, and Laurent Romary. 2019. Asynchronous pipeline for processing huge corpora on medium to low resource infrastructures. In *7th Workshop on the Challenges in the Management of Large Corpora (CMLC-7)*. Leibniz-Institut für Deutsche Sprache.

Yi Tay, Vinh Q Tran, Sebastian Ruder, Jai Gupta, Hyung Won Chung, Dara Bahri, Zhen Qin, Simon Baumgartner, Cong Yu, and Donald Metzler. 2021. Charformer: Fast character transformers via gradient-based subword tokenization. *arXiv preprint arXiv:2106.12672*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, and 1 others. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

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*.

Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. 2019. Do nlp models know numbers? probing numeracy in embeddings. 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 5307–5315.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Frank Wilcoxon. 1945. Individual comparisons by ranking methods. *Biometrics bulletin*, 1(6):80–83.

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.

Ruochen Zhang, Samuel Cahyawijaya, Jan Christian Blaise Cruz, Genta Indra Winata, and Alham Fikri Aji. 2023. Multilingual large language models are not (yet) code-switchers. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 12567–12582.

Xuanliang Zhang, Dingzirui Wang, Keyan Xu, Qingfu Zhu, and Wanxiang Che. 2025. Multitat: Benchmarking multilingual table-and-text question answering. *arXiv preprint arXiv:2502.17253*.

Mingyu Zheng, Yang Hao, Wenbin Jiang, Zheng Lin, Yajuan Lyu, QiaoQiao She, and Weiping Wang. 2023. Im-tqa: A chinese table question answering dataset with implicit and multi-type table structures. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5074–5094.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021. Tat-qa: A question answering benchmark on a hybrid of tabular and textual content in finance. 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 3277–3287.

A Script-Level Analysis

The token counts in Table 1 follow directly from two properties: a script's UTF-8 byte structure and its estimated presence in the training corpus. This section traces each anomalous case. Observations attributed to *GPT's BPE tokenizer* reflect what can be inferred about the GPT 5.x and o-series training corpus. Those attributed to *Gemini's SP tokenizer* reflect the Gemini 2.x/Gemini 3.x/Gemma 3 training distribution.

Byte structure. All scripts except English and Arabic-Indic encode digits as three-byte UTF-8 sequences. Because Unicode places each script's ten digits in consecutive codepoints within a larger block, all ten digits share the same leading two bytes. Table 4 lists these shared prefixes. For GPT's tokenizer, achieving a single-token digit requires two successive merge steps, first the two-byte prefix pair and then the prefix merged with the distinguishing third byte. Arabic-Indic digits are two-byte and require only one step.

Script	Range	Bytes	Shared prefix
English	U+0030–U+0039	1	—
Arabic-Indic	U+0660–U+0669	2	`D9`
Bengali	U+09E6–U+09EF	3	`E0 A7`
Hindi	U+0966–U+096F	3	`E0 A5`
Thai	U+0E50–U+0E59	3	`E0 B9`
Khmer	U+17E0–U+17E9	3	`E1 9F`
Myanmar	U+1040–U+1049	3	`E1 81`
Tibetan	U+0F20–U+0F29	3	`E0 BC`
Mongolian	U+1810–U+1819	3	`E1 A0`

Table 4: UTF-8 byte ranges and shared two-byte prefixes. GPT's tokenizer must complete both merge steps to achieve 1 token/digit for three-byte scripts.

Tibetan (GPT: 2/digit; Gemini: 1/digit). The prefix `E0 BC` is shared by the entire Tibetan Unicode block and accumulates enough frequency from all Tibetan text to merge in GPT's tokenizer. The second step requires each of the ten specific sequences `[E0BC]+A0` through `[E0BC]+A9` to be frequent enough individually, but Tibetan numeral text in the GPT training corpus is too sparse for any of them to win a merge slot. Every Tibetan digit therefore decomposes as `[E0BC][Ax]`, giving exactly 2 tokens with zero variance. Gemini's tokenizer initialises at the Unicode codepoint level, so even modest Tibetan presence suffices to retain each digit character above the pruning threshold. GPT stalls at the halfway point; Gemini succeeds from the start.

Myanmar (GPT: 1/digit; Gemini: 1/digit). Myanmar shares the same three-byte structure and shared prefix (`E1 81`) as Tibetan, yet both tokenizers produce 1 token/digit. Both merge steps complete in GPT's tokenizer, implying substantially more Myanmar numeral text in the GPT training corpus than Tibetan.

Mongolian (GPT: 3/digit; Gemini: 3/digit). Even the first prefix merge (`E1+A0`) does not succeed in GPT's tokenizer. In Gemini's tokenizer the character-level candidate still falls below the pruning threshold and the byte-level fallback (Team et al., 2023) applies, decomposing each digit into three individual byte tokens. Mongolian's 3-per-digit behaviour with zero variance in both tokenizers indicates the numeral script was very rare/effectively absent in both training distributions.

Arabic-Indic (GPT: 1/digit; Gemini: 1/digit; no pairs). Arabic-Indic digits are two-byte characters requiring only one merge step in GPT's tokenizer, which they achieve easily given Arabic's substantial web presence. In Gemini's tokenizer, each digit character retains its vocabulary slot without difficulty. Neither tokenizer produces pairs. Under the frequency suppression hypothesis for SP, this indicates that the ratio $\rho(\mathcal{A})/\hat{p}(a)$ does not exceed 1 which suppresses pair PMI.

Chinese (GPT: 1/digit; Gemini: 1/digit; no pairs). Chinese logographic numerals are three-byte CJK characters with very high training corpus presence in both tokenizer families. Individual character merges succeed trivially, yet no pairs survive. High individual token probability suppresses pair PMI regardless of clustering density, mirroring English exactly. High resource and no pairs is the pattern for both.

Corpus density from tokenizer output. The cases above establish four tiers readable from Table 1 without corpus access. Scripts where GPT single-digit equals 1 and Gemini shows pair merges have moderate-high corpus presence with sufficient numeric clustering, a condition only Bengali satisfies. Scripts where both tokenizers produce 1 token per digit but no pairs split into two mechanistically distinct sub-cases: those where individual digit probability is high enough to suppress any pair regardless of clustering (Chinese, Arabic-Indic, and English), and those where clustering is genuinely insufficient relative to individual frequency (Myanmar, Khmer). The suppression is scale-relative, not

absolute. Scripts where GPT single-digit exceeds 1 but Gemini single-digit equals 1 indicate low presence in the GPT training corpus and marginal presence in Gemini's, covering Tibetan and Thai. Scripts where both tokenizers show single-digit greater than 1 were effectively absent from both training distributions, which applies only to Mongolian.

B Corpus Validation of the NCC Ordering

The Frequency Suppression Hypothesis infers the ordering $\rho(\mathcal{B}) > \rho(\mathcal{H}) > \rho(\mathcal{T})$ from tokenizer output alone. We validate this on Wikipedia as a proxy for web-scale content in each language, processing all available articles from the Bengali (302,166),[1] Hindi (212,978),[2] and Thai (338,554)[3] editions, covering 438M, 236M, and 361M characters respectively. Table 5 reports the results.

Metric	Bengali	Hindi	Thai
NCC $\rho(\mathcal{S})$	0.651	0.618	0.577
Pair formation rate	94.0%	90.4%	88.3%
Avg. run length	2.86	2.62	2.36
Native digit total	7.89M	0.69M	0.15M
English / native ratio	0.35×	6.26×	68.9×
SP pair merges (Table 1)	yes	marginal	no

Table 5: Corpus statistics computed on Wikipedia latest editions as a proxy for web-scale numeral usage patterns. Native digit total counts script-specific digit characters. English/native ratio measures ASCII digit occurrences relative to native digit occurrences within each wiki.

The NCC ordering $\rho(\mathcal{B}) > \rho(\mathcal{H}) > \rho(\mathcal{T})$ is confirmed and aligns precisely with the tokenizer pair merge rates observed in Table 1 under both tokenizer families. Under Gemini's SentencePiece, Bengali shows substantial pair merges (mean 1.46 for 2-digit sequences), Hindi shows marginal pair merges (1.96), and Thai shows none (2.00). Under GPT's BPE tokenizer the same relative ordering holds and explains BPE's characteristics: Bengali achieves slightly more pair merges than Hindi (1.79 vs 1.82 for 2-digit sequences), while Thai fragments even at the single-digit level (mean 1.80 tokens per single digit), reflecting its near-absence from GPT's training corpus. The NCC ordering thus predicts cross-architecture tokenization behaviour across both tokenizer families.

[1] https://dumps.wikimedia.org/bnwiki/
[2] https://dumps.wikimedia.org/hiwiki/
[3] https://dumps.wikimedia.org/thwiki/

The English-to-native digit ratio reveals a complementary factor. Bengali Wikipedia uses Bengali digits as the primary numeral representation, with ASCII digits appearing at only 0.35 times the rate of Bengali digits. Hindi Wikipedia reverses this: ASCII digits appear 6.26 times more often than Hindi digits, indicating that Hindi authors default to English numerals. Thai Wikipedia shows this effect at an extreme: ASCII digits outnumber Thai digits 68.9 to 1, making Thai digit characters near-absent from Thai web content. The survival condition Eq. (11) has three factors: pair co-occurrence frequency, the PMI log ratio, and the sequence length term. Bengali satisfies all three: adequate absolute frequency (7.89M native digit occurrences), higher NCC (0.651), and low individual token probability. Thai fails primarily on absolute frequency: with only 149K native digit occurrences across 338K articles, the product $\widehat{\text{freq}}(\mathbf{d}) \cdot \text{I}(\mathbf{d})$ remains below threshold η regardless of NCC, because Thai digits are near-absent from web content in favour of ASCII.

These measurements are a proxy rather than a direct observation of Gemini's or GPT's training distribution. Wikipedia is indexed by most large-scale web crawls and is plausibly represented in training data, though we cannot confirm this directly. The consistency of the corpus ordering with the tokenizer's observed behaviour nonetheless supports the Frequency Suppression Hypothesis as an explanatory account of the cross-script asymmetry.

C Simulating Unigram Pruning on Demo Corpora

To make the Frequency Suppression Hypothesis visible under circumstances, we trace the unigram pruning decision for the pair **35** (English) and its Bengali equivalent through a small demo corpus. This is only for simulation purposes and should not be treated as a real training example. The corpus is deliberately minimal and constructed so that both pairs appear with *identical frequency*. The only difference is how often each script's individual digit tokens appear in non-paired contexts, which is the corpus-level property we infer from tokenizer output.

From Lemma 3, the marginal log-likelihood loss from removing a token t is:

$$\Delta\mathcal{L}(t) = \text{freq}(t) \cdot \underbrace{\log\left[\frac{\hat{p}(t)}{\prod_i \hat{p}(d_i)}\right]}_{\text{PMI}(t)} \qquad (12)$$

A token survives pruning when $\Delta\mathcal{L}(t) \geq \eta$. PMI is the decisive quantity: a pair whose components are individually common has PMI ≈ 0 regardless of how often the pair itself appears, and is pruned. A pair whose components are individually rare has high PMI even at modest pair frequency, and survives.

Bengali demo corpus

Bengali digits appear only in Bengali-language text, so their individual frequency is low relative to total corpus tokens. Each line below lists the digits it contributes in brackets.

আয় ৩৫ টাকা।	[৩][৫]
ব্যয় ২০ টাকা।	[২][০]
মোট ৪৫ টাকা।	[৪][৫]
আয় ৩২০ টাকা।	[৩][২][০]
ব্যয় ২০০ টাকা।	[২][০][০]
মোট ৩৫ টাকা।	[৩][৫]
জনসংখ্যা ৭৮।	[৭][৮]
দাম ৪৬ টাকা।	[৪][৬]

Total digits $N = 18$.
৩: freq = 3 ৫: freq = 3 ৩৫ pair: freq = 3
(The pair ৩৫ accounts for all three occurrences of ৩; the digit never appears alone.)

English demo corpus

English digits appear across all language contexts in any web crawl: in dates, postal codes, phone extensions, and numeric identifiers.

income 35 dollars	[3] [5]
expense 23 dollars	[2] [3]
total 55 dollars	[5] [5]
income 350 dollars	[3] [5] [0]
expense 200 dollars	[2] [0] [0]
total 35 dollars	[3] [5]
Mar 3, page 13	[3] [1] [3]
ZIP 55035, ext 3	[5] [5] [0] [3] [5] [3]

Total digits $N = 23$.
3: freq = 8 5: freq = 8 35 pair: freq = 3
(Digits 3 and 5 each appear 5 additional times outside the pair, in dates, postal codes, and identifiers.)

Both corpora contain the pair exactly **three times** out of $N = 18$ and $N = 23$ total digits respectively.. Marginal probability of each component digit also differs. In Bengali, digit ৩ appears only inside the pair itself, giving $\hat{p}() = 3/18$. In English, digit 3 appears in dates, postal codes, and identifiers as well, giving $\hat{p}(3) = 8/23$. The same asymmetry holds for ৫ and 5. Table 6 applies Eq. 12 to both cases.

English digit 3 appears 8 times in the corpus but only 3 of those are inside the pair. The remaining 5 are solo appearances in dates and codes. This solo frequency makes the pair's co-occurrence unremarkable, PMI drops low. Bengali pair ৩৫, by contrast, gives a sense that the pair is not just common but *expected*. PMI is large, and the pruner has strong incentive to keep it. The mechanism generalises: any script whose digits cluster tightly into multi-digit runs and appear rarely in isolation will exhibit the same survival advantage under unigram pruning.

Quantity	Bengali (৩৫)	English (35)
$\hat{p}(\text{pair})$	$3/18 = 0.167$	$3/23 = 0.130$
$\hat{p}(d_1)$	$3/18 = 0.167$	$8/23 = 0.347$
$\hat{p}(d_2)$	$3/18 = 0.167$	$8/23 = 0.347$
$\hat{p}(d_1) \cdot \hat{p}(d_2)$	0.028	0.120
PMI	$\log(0.167/0.028)$	$\log(0.130/0.120)$
$\Delta\mathcal{L} = \text{freq} \times \text{PMI}$	$3 \times 0.77 = \mathbf{2.33}$	$3 \times 0.034 = \mathbf{0.104}$
Threshold $\eta = 1$	$2.33 \geq \eta$ **SURVIVES**	$0.104 < \eta$ **PRUNED**

Table 6: Unigram pruning computation for the digit pair **35** in Bengali and English under the demo corpus. Pair frequency is identical in both cases. Assuming a threshold η, the Bengali pair survives because its component digits are rare outside the pair, producing high PMI. The English pair is pruned because its components appear frequently in cross-lingual contexts, driving PMI lower regardless of pair frequency.

D Edge Cases in Digit Substitution

Bengali digit substitution operates on ASCII digit characters only, leaving all other tokens unchanged. Several non-standard numeric representations appear across the benchmarks and are worth noting.

Decimal points and separators. Expressions such as `1,496.5` or `3.7%` are substituted as ১,৪৯৬.৫ and ৩.৭%. The comma and period are not digits and pass through untouched, preserving the numeric structure.

Powers and scientific notation. MathQA contains expressions such as `10^3` or `2e6`. The digit characters within these are substituted while the exponent symbols (^, e) remain in ASCII, producing ১০^৩ and ২e৬. Gemini parses these correctly.

Negative numbers and currency. Signed values (`-19.17`) and currency strings (`$206,587`) substitute their digit portions only. The minus sign and currency symbol are unaffected.

Observed impact. These mixed-representation forms were present throughout MathQA, FinQA, and the MULTITAT benchmarks. Accuracy differences between both conditions were negligible across all datasets, suggesting the model handles substituted edge cases robustly. We do not conduct

a per-category breakdown of edge case types versus accuracy, as noted in Limitations.

E Thinking Token Accounting

Gemini 2.5 Flash and Pro expose thinking tokens separately via `usage_metadata.thoughts_token_count`; we record them alongside input and output counts per call. Thinking content is excluded from the visible response by filtering parts where `part.thought = True`. We set `thinking_budget=5000` for Flash and Pro. The default (`-1`) enables dynamic allocation up to these caps.

Gemini 3 series replaces `thinkingBudget` with a discrete `thinkingLevel` parameter (`minimal` / `low` / `medium` / `high`); we used `high` (the default) for 2.0 Pro Preview runs.[4]

The substitution layer modifies only user prompt content. System prompts, function declarations, and tool payloads are passed through unchanged, so thinking token counts are not confounded by tool-use overhead in any benchmark evaluated here.

F On the First-Order Markov Approximation

Lemma 3 reduces the PMI of a digit k-gram to $(k-1)\log[\rho(\mathcal{S})/\hat{p}(d)]$ via the first-order Markov factorisation $\hat{p}(d_1 \cdots d_k) \approx \rho(\mathcal{S})^{k-1}\hat{p}(d_1)$. This is a deliberate simplification; we discuss its scope.

Why it holds well enough. The pruning decision is a binary threshold comparison, not a continuous quantity, so moderate PMI estimation error does not change the survival outcome unless the true PMI lies near η. The Bengali advantage is inferred to be large, making the conclusion robust to approximation noise. A badly wrong approximation would produce anomalies in Table 1, and none are observed.

Oversimplification case. Comma-separated numerals interrupt digit runs and reduce effective NCC. Defining $f_{\mathcal{S}}$ as the fraction of consecutive digit positions within a comma group, English 3-digit grouping gives $f_{\mathcal{E}} \approx 2/3$, while Bengali lakh–crore grouping (sizes 1,2,2,…,3) gives $f_{\mathcal{B}} \approx 1/2$, since 2-digit groups contribute fewer within-group pairs per digit than 3-digit groups. Bengali therefore has *more* comma boundaries per digit. In practice, however, both Bengali and English numerals in raw corpora and web-scale datasets are often written without adherence to conventional comma-grouping rules, which alleviates the effective impact of these formatting differences. $\rho_{\text{eff}}(\mathcal{B}) > \rho_{\text{eff}}(\mathcal{E})$ is mainly preserved because $\rho_{\text{within,B}} \gg \rho_{\text{within,E}}$: English digits saturate all language contexts in web corpora, suppressing their individual PMI regardless of grouping, while Bengali digits appear almost exclusively in Bengali text. This cross-lingual frequency gap dominates the $f_{\mathcal{E}}/f_{\mathcal{B}} \approx 1.3\times$ penalty. NCC estimates from Wikipedia may not fully represent domain-specific or low-resource script distributions, though the observed tokenizer ordering is consistent with the Wikipedia-based estimates.

G Substitution Layer Latency

The substitution layer is a single Python `str.translate` call, which operates as a C-level character scan in $O(n)$ time where n is the number of characters in the prompt. For the largest prompts in our benchmarks (FinQA, up to ≈10,000 characters), this completes in under 0.1 ms on commodity hardware. API round-trip latency for Gemini 2.5 Flash and Pro ranges from 3 to 18 seconds depending on prompt length and thinking budget, making the substitution overhead less than 0.005% of total wall-clock time.

Table 7 summarises measurements across benchmark prompt sizes.

Benchmark	Avg Chars	Subst. (ms)	API Latency (s)
MathQA	172	<0.01	3–10
TAT-QA	1388	<0.05	3–15
IM-TQA	1426	<0.05	3–15
MULTITAT	2712	<0.05	5–18
SAHM	1631	<0.05	5–18
FinQA	2680	<0.05	5–18

Table 7: Substitution layer timing vs. API round-trip latency. Substitution times are averaged over 5 runs on a standard 8 core CPU and a rounded span is reported. API latency can vary with problem difficulty, thinking budget, even server load.

The reverse substitution on the output is equally negligible since model responses to numeric queries are short (typically under 10-20 tokens). The substitution layer therefore adds no measurable latency to any production pipeline.

[4] https://ai.google.dev/gemini-api/docs/thinking

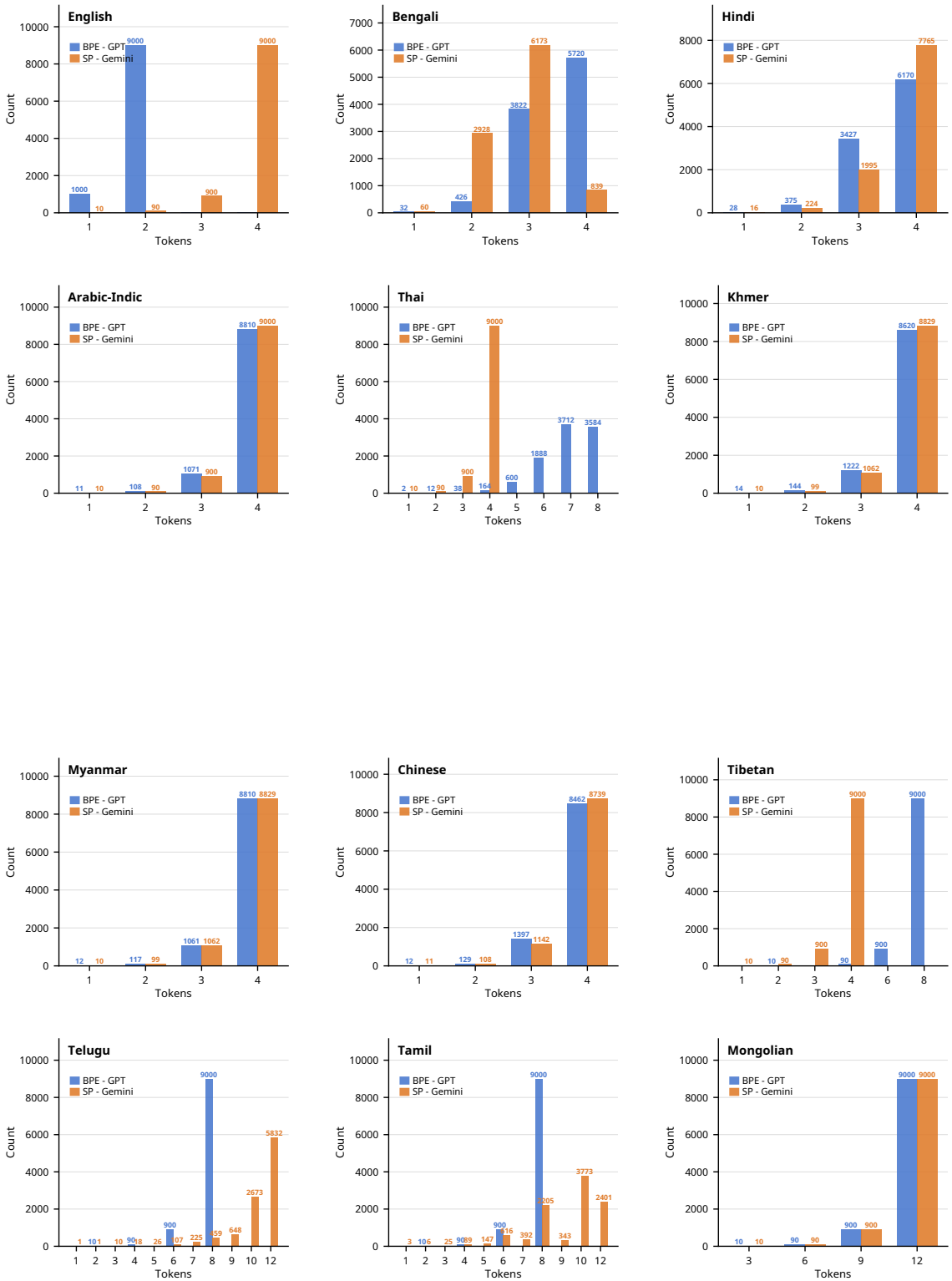

Figure 4: Token count distributions for all 12 numeral scripts over 10,000 numerals (0–9999) under GPT-family BPE and Gemini-family SentencePiece (SP).

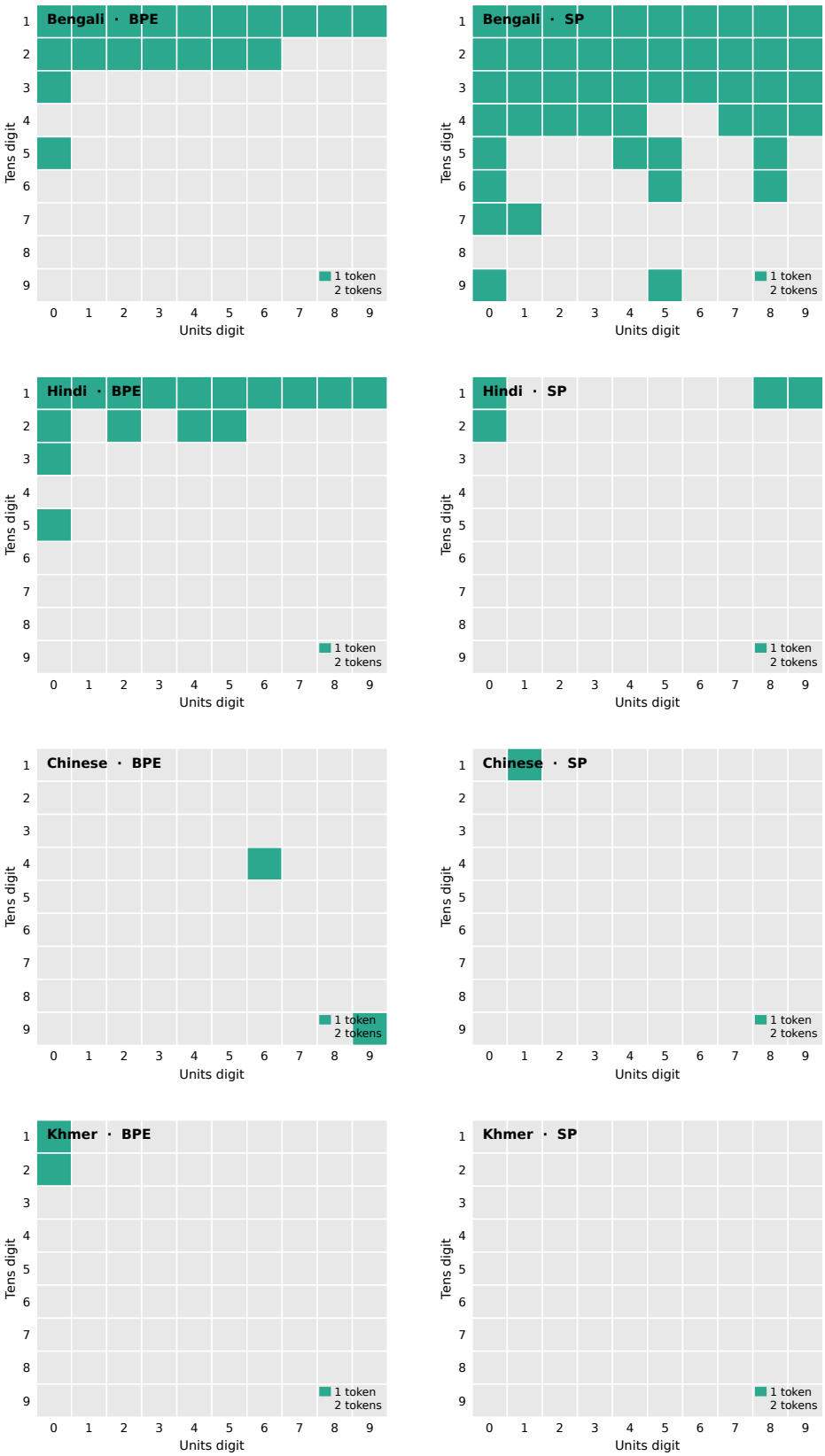

Figure 5: Token allocation maps for two-digit numerals (10–99) under GPT-family BPE and Gemini-family SentencePiece (SP). Teal cells are merged into a single token; gray cells require two. Under BPE, Bengali and Hindi show 1-token merges only for lower-valued pairs, reflecting frequency-driven merge limits. Under SP, Bengali retains substantially more single-token pairs than Hindi. For Chinese and Khmer, merged pairs are sparse and show simple digit patterns (like 10, 11, 20, 99 are merged) regardless of the tokenizer.