# Token Cost Inequality: Measuring Tokenization Disparities Across Scripts in Roman Urdu and Urdu

**Waleed Jamil**
Edinburgh Napier University
Edinburgh, United Kingdom
waleed21195@gmail.com

**Saima Rafi**
Edinburgh Napier University
Edinburgh, United Kingdom
S.Rafi@napier.ac.uk

**Yanchao Yu**
Edinburgh Napier University
Edinburgh, United Kingdom
y.yu@napier.ac.uk

## Abstract

Tokenization is central to modern language models, yet its effects on cross-script efficiency, input cost, and truncation behavior remain underexplored. We study this issue through aligned comparisons of Urdu and Roman Urdu, asking whether semantically equivalent content incurs systematically different tokenization costs across scripts. We introduce Token Cost Inequality (TCI), a metric for quantifying relative tokenization efficiency under semantic alignment, and propose a multi-axis framework spanning token cost, fragmentation, and fixed-budget retention. Across three tokenizer families (`cl100k`, mT5, and ByT5), we find that tokenization disparities are strongly tokenizer-dependent, with substantial differences in token cost and segmentation behavior across scripts. We further identify an efficiency–retention paradox: token cost alone does not fully explain truncation behavior. Under fixed token budgets, Roman Urdu preserves more character-level content than native Urdu, reflecting differences in character-per-token density and fragmentation. Lightweight normalization yields minimal gains, suggesting that the observed disparities arise primarily from tokenizer design rather than superficial orthographic variation. These findings provide controlled evidence that fixed token budgets can produce unequal surface-coverage conditions across scripts, with implications for input-side cost estimation, benchmark design, and multilingual evaluation under constrained token budgets.

## 1 Introduction

Consider two users submitting semantically equivalent prompts to an LLM-powered system, one in native Urdu script and the other in Roman Urdu. Under `cl100k` tokenization, native Urdu consumes 56.1 tokens on average while Roman Urdu requires only 26.0. At a fixed budget of 16 tokens, native Urdu retains 44.9% of its surface content while Roman Urdu retains 74.1%. The linguistic content is aligned, but the choice of script changes how much text fits within the same token budget.

Tokenization is often treated as a background implementation choice, yet under fixed token budgets it determines how much text survives truncation, how inputs fragment, and what users effectively pay per query. In multilingual settings, this can create unequal surface-coverage conditions: equal token budgets do not necessarily correspond to equal retained content when semantically aligned content in different scripts tokenizes differently (Gehrmann et al., 2021).

This issue is especially salient for languages with multiple orthographic forms. Urdu is commonly written in a Perso-Arabic script but is also widely used in Romanized form, particularly in informal digital communication. Because these representations can express equivalent semantic content through different orthographic systems, they raise a consequential question: *does the same content incur different tokenization costs across scripts?* Under fixed context windows and token-based billing, such disparities can affect input-side cost estimation, retained context, and benchmark comparability (Liu et al., 2024; Petrov et al., 2023).

Building on work showing unequal token costs across languages and scripts, we isolate script-level effects by comparing Urdu and Roman Urdu under semantic alignment. We introduce **Token Cost Inequality (TCI)**, a measure of relative tokenization efficiency, within a **multi-axis evaluation framework** spanning token cost, fragmentation, and fixed-budget retention. Experiments cover three tokenizer families across 10,000-example subsamples of Roman-Urdu-Parl (Alam and Hussain, 2022), OPUS100, and Dakshina (Roark et al., 2020).

Our results show that disparities are strongly tokenizer-dependent, with TCI ranging from 0.47 to 0.97 on Roman-Urdu-Parl and from 0.29 to 1.03 overall. More consequentially, we identify



*Proceedings of the Fifth Workshop on Generation, Evaluation and Metrics (GEM)*, pages 563–573

an **efficiency–retention paradox**: token cost alone does not fully explain truncation behavior. Under `cl100k` at $B{=}16$, Roman Urdu achieves a retention ratio of $1.649$ over native Urdu; the English–Urdu baseline reaches $2.614$. This follows from character-per-token density: when a script fragments into short subword units, a fixed budget reaches less of the original text. Lightweight normalization yields negligible gains, with TCI reductions below $0.06\%$, suggesting that these disparities arise primarily from tokenizer design rather than superficial variation.

These findings provide controlled evidence that tokenization is a system-level variable in multilingual NLG evaluation (Gehrmann et al., 2021): fixed token budgets can standardize inputs only superficially, preserving measurable differences in surface coverage across scripts.

- **Token Cost Inequality (TCI):** A metric for quantifying relative tokenization efficiency under semantic alignment, isolating script-level disparities.
- **Multi-axis evaluation framework:** A unified analysis of token cost, fragmentation, and truncation retention as distinct dimensions of tokenization behavior.
- **Tokenizer dependence:** Cross-script disparities range from near parity to strong divergence, indicating that tokenization inequality is shaped by tokenizer design.
- **Efficiency–retention paradox:** Token cost alone does not determine truncation robustness; retention also depends on character-per-token density and fragmentation.
- **Implications for evaluation:** Fixed token budgets can induce unequal surface-coverage conditions across scripts, motivating tokenization-aware benchmark design.

## 2 Related Work

Recent work has shown that tokenization can introduce systematic inequalities across languages. Petrov et al. (2023) demonstrate that semantically equivalent content can yield substantially different token lengths across languages and tokenizer families, with implications for cost, latency, and effective context capacity. Building on this, Velayuthan and Sarveswaran (2025) show that such disparities arise not only from vocabulary allocation but also from pre-tokenization design, particularly for scripts with complex orthographic structure.

While these studies establish tokenization as a source of cross-lingual variation, they do not isolate script-level effects under controlled semantic alignment. It therefore remains difficult to separate disparities caused by language identity, script, tokenizer design, and dataset composition. Our work targets this distinction by studying Urdu and Roman Urdu as aligned orthographic variants, and by jointly analyzing token cost, fragmentation, and fixed-budget retention. While newer tokenizers such as `o200k_base` and Gemma3Tokenizer have since been released, we evaluate three distinct families—BPE, unigram SentencePiece, and byte-level tokenization—to compare structural strategies rather than benchmark specific systems. Truncation under fixed budgets has also been shown to affect downstream performance (Liu et al., 2024), motivating our analysis of retained surface coverage while leaving task-level validation to future work.

A related line of work examines Romanized South Asian languages and the resources required to model them. Roark et al. (2020) introduce the Dakshina dataset, which provides native-script text, romanization lexicons, and parallel sentence pairs for several languages, including Urdu. Kirov et al. (2024) show that Romanized text requires context-aware modeling due to its high variability and lack of standardization. While this literature treats Romanization primarily as a transliteration and modeling challenge, our focus is orthogonal: rather than improving modeling performance, we analyze how tokenizer design affects token cost, segmentation granularity, and retained surface coverage across native and Romanized forms.

Work specifically on Urdu has largely centered on segmentation and orthographic preprocessing. Durrani and Hussain (2010) highlight the difficulty of Urdu word segmentation due to inconsistent whitespace usage, while Bin Zia et al. (2018) model segmentation as a sequence labeling problem. These efforts aim to recover linguistically meaningful word boundaries. In contrast, we do not assume gold segmentation, but examine how subword and byte-level tokenizers interact with Urdu orthography, producing measurable differences in fragmentation, token efficiency, and fixed-budget retention.

Parallel resources for Urdu and Roman Urdu further enable controlled cross-script analysis. Alam

and Hussain (2022) introduce Roman-Urdu-Parl, a large parallel corpus widely used for transliteration and low-resource modeling (Butt et al., 2025). Rather than treating these datasets as end tasks, we use them as evaluation substrates to quantify tokenization disparities under aligned semantic content.

Taken together, prior work identifies tokenization as a source of cross-lingual variation and potential bias in multilingual systems (Petrov et al., 2023; Kanjirangat et al., 2025). However, less attention has been paid to controlled script-level comparisons and to how token cost, fragmentation, and fixed-budget retention relate to one another. We contribute a multi-axis evaluation protocol for aligned Urdu/Roman Urdu data, showing that script choice can affect input-side token cost and surface coverage while clarifying that downstream task-level effects remain future work.

## 3 Datasets and Alignment Setup

Our evaluation is designed to isolate tokenization disparities under controlled cross-script and cross-lingual conditions. We combine sentence-level parallel corpora with lexical transliteration resources to analyze token cost, fragmentation, and fixed-budget retention at multiple levels of granularity. To ensure comparable experimental conditions, all metrics are computed on uniformly sampled subsets of 10,000 examples per dataset (Table 1).

- **Roman-Urdu-Parl**: We use Roman-Urdu-Parl (Alam and Hussain, 2022), a large-scale parallel corpus of nearly 4M Roman Urdu–Urdu sentence pairs. This dataset forms the core of our analysis, enabling direct cross-script comparison under semantic alignment.

- **English–Urdu OPUS100**: We use the English–Urdu subset of OPUS100 (Zhang et al., 2020) as a cross-lingual baseline. This provides a reference point for interpreting Urdu/Roman Urdu script-level disparities relative to broader language-level differences.

- **Dakshina Urdu**: We use the Urdu portion of Dakshina (Roark et al., 2020), a lexical transliteration resource consisting of word-level Romanized–native pairs. This enables word-level analysis of tokenization behavior, helping separate lexical fragmentation effects from sentence-level context.

| DATASET | TYPE | FULL SIZE | ANALYZED |
|---|---|---|---|
| Roman-Urdu-Parl | Sent. | 3,981,709 | 10,000 |
| OPUS100 (EN–UR) | Sent. | 724,153 | 10,000 |
| Dakshina (UR) | Lex. | 101,235 | 10,000 |

Table 1: Datasets used in the study. All metrics are computed on uniformly sampled subsets of 10,000 pairs for controlled comparison across settings.

### 3.1 Sampling and Preprocessing

We uniformly sample 10,000 examples from each dataset to ensure computational tractability and balanced comparison across resources of different sizes. Sentence-level datasets (Roman-Urdu-Parl and OPUS100) are used with their original alignments, while Dakshina provides word-level pairs. These samples are intended to support controlled comparison rather than to exhaustively characterize each full corpus.

Preprocessing is intentionally minimal to preserve natural orthographic variation. In particular, we avoid normalization in the main experiments, allowing tokenization behavior to reflect realistic usage conditions. A separate normalization probe is used to assess whether observed disparities arise from surface variation or tokenizer structure.

### 3.2 Evaluation Scope

This setup supports three complementary evaluation regimes: (i) aligned cross-script comparisons (Roman Urdu vs. Urdu), (ii) cross-lingual comparisons (English vs. Urdu), and (iii) lexical-level analysis (Dakshina).

Together, these settings allow us to distinguish script-level effects from broader language-level and lexical effects. The analysis is therefore focused on controlled comparison across aligned representations, rather than on making exhaustive claims about all Urdu, Roman Urdu, or multilingual tokenization settings.

## 4 Measuring Tokenization Inequality: A Multi-Axis Framework

Modern language models operate over tokenized text, making tokenization a key determinant of input length, inference cost, and effective context utilization. In multilingual and cross-script settings, tokenization can introduce systematic disparities even when semantic content is aligned.

To capture these effects, we propose a **multi-axis evaluation framework** with four axes: *token cost*,

*fragmentation*, *fixed-budget retention*, and *cost multipliers*, complemented by a normalization diagnostic (§4.5). These axes are related but distinct: token cost measures how many tokens are used, fragmentation measures how tokens are distributed within words, retention measures what survives under a fixed token budget, and cost multipliers summarize average input-side cost. Each therefore captures a different aspect of tokenization behavior rather than replacing the others.

### 4.1 Token Cost Inequality (TCI)

Our primary metric is **Token Cost Inequality (TCI)**, defined as the ratio of token counts between aligned representations under tokenizer $\tau$:

$$\mathrm{TCI}_\tau(x^{(L)}, x^{(R)}) = \frac{T_\tau(x^{(L)})}{T_\tau(x^{(R)})}, \tag{1}$$

where $T_\tau(x)$ denotes the number of tokens assigned to sequence $x$.

TCI quantifies relative token cost under aligned semantic content. Values greater than 1 indicate higher tokenization cost for the left-side representation. We report mean, median, and upper-tail statistics (P95, P99) to capture both typical behavior and extreme per-example disparities.

### 4.2 Fragmentation

To characterize segmentation granularity, we measure **fragmentation** using tokens per word (TPW):

$$\mathrm{TPW}_\tau(x) = \frac{T_\tau(x)}{\#\mathrm{words}(x)}, \tag{2}$$

where words are defined via whitespace tokenization.

While TCI captures sequence-level token cost, TPW reveals how tokens are distributed within words. We report mean TPW, tail statistics, and the proportion of words exceeding fixed thresholds (e.g., $\geq 5$, $\geq 8$ tokens), capturing heavy-tail effects not visible from averages alone. Fragmentation also affects retention: when a representation is split into more tokens per word, a fixed budget may cover fewer characters of the original input.

### 4.3 Fixed-Budget Retention

To analyze behavior under constrained token budgets, we truncate tokenized sequences to a budget $B$ and decode the retained prefix. We define **character-level retention** as:

$$\mathrm{Retain}_\tau(x, B) = \frac{|\mathrm{decode}(\tau(x)_{1:B})|}{|x|}. \tag{3}$$

Here, decode denotes tokenizer-specific decoding, and $|\cdot|$ denotes character length.

This metric approximates how much surface content is preserved after truncation, serving as a tokenizer-agnostic proxy for usable input coverage. It does not directly measure semantic preservation, but provides a consistent way to compare retained context under the same token budget.

### 4.4 Cost Multipliers

To provide an operational summary, we compute **cost multipliers** as the ratio of expected token counts:

$$\mathrm{CostMult}_\tau = \frac{E[T_\tau(x^{(L)})]}{E[T_\tau(x^{(R)})]}, \tag{4}$$

where the expectation is taken over aligned pairs.

Cost multipliers differ from mean TCI: they are ratios of expected token counts, whereas mean TCI is the expectation of per-pair ratios. The two coincide only under restrictive conditions. Thus, TCI captures pairwise inequality, while CostMult provides an aggregate input-side cost estimate for settings where cost scales with token count.

### 4.5 Normalization Probe

Finally, we introduce a **normalization diagnostic** to assess whether observed disparities arise from superficial orthographic variation or deeper tokenizer-induced effects. We apply lightweight normalization to Roman Urdu, including repeated-character collapse and whitespace normalization, and compare token counts and TCI before and after preprocessing.

This analysis distinguishes disparities attributable to input variability from those more closely associated with tokenizer design.

## 5 Experimental Setup

We evaluate three tokenizer families:

- **`cl100k_base`**: a BPE tokenizer used in tiktoken-based systems (OpenAI, 2022), with a 100,277-token vocabulary optimized primarily for English and code.
- **mT5**: a SentencePiece unigram tokenizer trained on mC4 across 101 languages (Xue et al., 2021), with a 250,000-token vocabulary providing broad multilingual coverage.

- **ByT5**: a byte-level tokenizer operating over raw UTF-8 bytes without a learned vocabulary (Xue et al., 2022), with 256 token types plus special tokens.

These tokenizers differ in vocabulary construction, segmentation granularity, and sensitivity to script variation. We compare structural tokenization strategies rather than exhaustively benchmarking current tokenizers.

### 5.1 Evaluation Protocol

Experiments follow the datasets and alignment settings in Section 3, covering three regimes: (i) aligned cross-script comparisons (Roman Urdu vs. Urdu), (ii) cross-lingual comparisons (English vs. Urdu), and (iii) lexical-level analysis (Dakshina). For aligned comparisons, the left-hand representation corresponds to Roman Urdu or English and the right-hand to native Urdu; all pairwise metrics are computed over aligned inputs.

### 5.2 Preprocessing and Metrics

Preprocessing follows Section 3: minimal cleaning with no normalization or script conversion in main experiments. We evaluate using the multi-axis framework in Section 4: TCI, fragmentation (TPW), character-level retention under budgets $B \in \{16, 32, 64, 128, 256\}$, and cost multipliers, computed independently per tokenizer and dataset. These budgets are controlled stress tests, not full context-window sizes, modeling constrained snippets, prompt components, and truncated benchmark inputs. For the normalization diagnostic, we apply repeated-character collapse and whitespace normalization to Roman Urdu and report results separately.

### 5.3 Reporting

We report mean, median, and upper-tail statistics (P95, P99) over each sampled dataset. All ratios are computed relative to native Urdu. This setup enables controlled comparison of tokenization efficiency, segmentation behavior, and fixed-budget retention across tokenizers, datasets, and evaluation regimes.

## 6 Results & Discussion

### 6.1 Results

Table 2 and Figure 1 present Token Cost Inequality (TCI) across aligned datasets and tokenizer families. Together, they quantify tokenization efficiency between left-side representations (Roman Urdu or English) and native Urdu. The table reports central tendency and upper-tail statistics, while the figure summarizes mean TCI relative to the equal-cost baseline (TCI $= 1.0$).

Table 3 reports tokens-per-word (TPW) statistics for left-side and native Urdu representations across datasets and tokenizer families. It summarizes both average segmentation granularity and tail behavior, including P99 TPW and the proportion of highly fragmented words ($\geq 5$ tokens). These statistics show how different tokenizers distribute tokens within words and expose heavy-tail fragmentation effects.

Table 4 reports retention ratios under fixed token budgets, comparing the amount of surface content preserved by left-side representations and native Urdu. By varying the token budget $B$, the table shows how tokenization differences affect retained input coverage under constrained settings, relative to the equal-retention baseline.

Table 5 reports average token counts for left-side and native Urdu representations, along with their corresponding cost multipliers. These values provide an operational view of tokenization inequality by quantifying relative input-side token usage across tokenizer families and datasets under aligned comparisons.

Table 6 reports the effect of lightweight normalization on TCI, comparing raw and normalized values across datasets and tokenizer families. It also includes the percentage reduction in TCI and the proportion of examples affected, providing a direct assessment of whether simple surface-level preprocessing reduces the observed disparities.

### 6.2 Discussion

**Token Cost Inequality.** Table 2 and Figure 1 show that tokenization disparities are strongly tokenizer-dependent: both their direction and magnitude vary across tokenizer families rather than being determined by the language pair alone. On Roman-Urdu-Parl, Roman Urdu is more token-efficient than native Urdu under `cl100k` (mean TCI $= 0.473$) and ByT5 ($0.669$), while mT5 yields near parity ($0.967$). Dakshina exhibits the same pattern, with Roman Urdu cheaper under `cl100k` ($0.580$) and ByT5 ($0.654$), but slightly more expensive under mT5 ($1.027$). The English–Urdu baseline further reinforces this behavior, with English consistently more token-efficient than Urdu across all tokenizers. The near-parity under mT5 is

| DATASET | TOKENIZER | Mean | Med. | P95 | P99 |
|---|---|---|---|---|---|
| *Roman-Urdu-Parl* | cl100k | 0.473 | 0.463 | 0.625 | 0.750 |
| | mT5 | 0.967 | 0.956 | 1.217 | 1.400 |
| | ByT5 | 0.669 | 0.667 | 0.755 | 0.821 |
| *Dakshina Urdu* | cl100k | 0.580 | 0.571 | 1.000 | 1.000 |
| | mT5 | 1.027 | 1.000 | 1.667 | 2.000 |
| | ByT5 | 0.654 | 0.643 | 0.875 | 1.000 |
| *OPUS100 EN–UR* | cl100k | 0.289 | 0.267 | 0.459 | 0.742 |
| | mT5 | 0.796 | 0.778 | 1.188 | 1.462 |
| | ByT5 | 0.619 | 0.602 | 0.790 | 0.945 |

Table 2: Token Cost Inequality (TCI) across aligned datasets and tokenizer families.

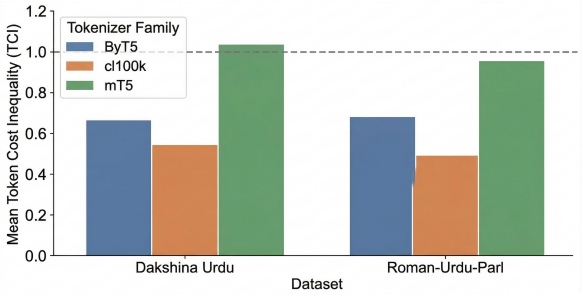


Figure 1: Mean Token Cost Inequality (TCI) across aligned datasets.

consistent with its multilingual vocabulary design: its 250,000-token SentencePiece model, trained on mC4 across 101 languages, allocates subword capacity across many languages and scripts, unlike the more English- and code-oriented `cl100k` vocabulary. Upper-tail behavior (Table 2) amplifies this effect. On Roman-Urdu-Parl, P99 TCI remains below 1 for `cl100k` (0.750) and ByT5 (0.821), but rises above 1 under mT5 (1.400). Dakshina shows a stronger divergence, with P99 TCI reaching 2.0 under mT5, indicating that extreme per-example disparities can be substantial even when mean TCI is near parity. Taken together, these results show that tokenization disparities are not uniformly directional and do not generalize across tokenizer families. Consequently, conclusions about token budget comparability drawn under one tokenizer may not transfer to another. For multilingual NLG evaluation (Gehrmann et al., 2021), this suggests that fixed token budgets can create unequal surface-coverage conditions across scripts unless tokenizer effects are explicitly controlled.

**Fragmentation Analysis.** As shown in Table 3, fragmentation reveals structural differences in tokenization that are not captured by aggregate token cost. Under `cl100k`, native Urdu is substantially more fragmented than Roman Urdu: on Roman-Urdu-Parl, mean TPW is 3.655 for Urdu versus 1.838 for Roman Urdu, and 26.3% of Urdu words exceed 5 tokens compared with only 0.3% for Roman Urdu. ByT5 produces the heaviest fragmentation overall, with native Urdu reaching a P99 of 14 tokens per word on Roman-Urdu-Parl and 18 on Dakshina. By contrast, mT5 yields near-parity in average fragmentation. Tail statistics reveal an important nuance. On Dakshina under mT5, Roman Urdu reaches a P99 of 5 tokens per word compared with 4 for Urdu, and shows a higher proportion of words exceeding the 5-token threshold (1.3% vs. 0.7%). Thus, even when average fragmentation is near parity, a subset of Roman Urdu forms is disproportionately segmented, indicating that mean TPW alone can obscure tail-level disparities. These results show why fragmentation is complementary to token-cost analysis. Under fixed token budgets, more fragmented representations consume more token positions for comparable surface content, often reducing the number of characters reached by the budget. This pattern is reflected in the retention ratios in Table 4 and helps explain why token cost alone does not fully predict fixed-budget retention. Fragmentation therefore provides a useful view of disparities that are not captured by average TCI alone. Additional tail statistics are provided in Appendix Table 9.

| | | Mean TPW | | P99 TPW | | % ≥ 5 | |
|---|---|---|---|---|---|---|---|
| **Dataset** | **Tokenizer** | **L** | **R** | **L** | **R** | **L** | **R** |
| Roman-Urdu-Parl | cl100k | 1.838 | 3.655 | 4 | 8 | 0.3 | 26.3 |
| | mT5 | 1.602 | 1.670 | 4 | 4 | 0.2 | 0.1 |
| | ByT5 | 4.062 | 6.578 | 9 | 14 | 36.4 | 65.5 |
| Dakshina Urdu | cl100k | 2.796 | 5.092 | 5 | 10 | 3.8 | 60.9 |
| | mT5 | 2.488 | 2.548 | 5 | 4 | 1.3 | 0.7 |
| | ByT5 | 6.502 | 10.086 | 11 | 18 | 76.5 | 88.0 |
| OPUS100 EN–UR | cl100k | 1.409 | 3.699 | 4 | 8 | 0.2 | 27.5 |
| | mT5 | 1.548 | 1.701 | 4 | 4 | 0.3 | 0.1 |
| | ByT5 | 4.402 | 6.545 | 11 | 14 | 41.5 | 64.9 |

Table 3: Fragmentation (TPW) for Left (L) vs. Urdu (R).

**Truncation Effects and the Efficiency–Retention Paradox.** Table 4 reveals a key result: truncation behavior is not determined by average token cost alone. Token-efficient representations may preserve more surface content under fixed budgets, but the degree of retention depends on how characters are distributed across tokens. On Roman-Urdu-Parl under `cl100k` at $B$=16, Roman Urdu retains 74.1% of its surface content compared with 44.9% for native Urdu (retention ratio = 1.649), while also using fewer tokens on average. ByT5 exhibits

| Dataset | Tokenizer | Retention Ratio (L/R) | | |
|---|---|---|---|---|
| | | $B = 16$ | $B = 32$ | $B = 64$ |
| Roman-Urdu-Parl | cl100k | 1.649 | 1.267 | 1.097 |
| | mT5 | 1.018 | 1.006 | 1.001 |
| | ByT5 | 1.514 | 1.379 | 1.159 |
| OPUS100 EN–UR | cl100k | 2.614 | 1.969 | 1.437 |
| | mT5 | 1.206 | 1.126 | 1.045 |
| | ByT5 | 1.648 | 1.582 | 1.405 |

Table 4: Retention ratio under fixed token budgets ($B$)

| Dataset | Tokenizer | L Mean | R Mean | Multiplier |
|---|---|---|---|---|
| Roman-Urdu-Parl | cl100k | 25.960 | 56.111 | 0.463 |
| | mT5 | 23.293 | 24.236 | 0.961 |
| | ByT5 | 72.625 | 108.970 | 0.666 |
| Dakshina Urdu | cl100k | 2.796 | 5.092 | 0.549 |
| | mT5 | 2.488 | 2.548 | 0.976 |
| | ByT5 | 6.502 | 10.086 | 0.645 |
| OPUS100 EN–UR | cl100k | 30.861 | 116.716 | 0.264 |
| | mT5 | 38.078 | 50.717 | 0.751 |
| | ByT5 | 131.847 | 223.995 | 0.589 |

Table 5: Average token counts (L = left-side, R = native Urdu) and cost multipliers (L/R).

a similar asymmetry (retention ratio $= 1.514$ at $B$=16), whereas mT5 remains close to parity, consistent with its near-equal TCI values. Dakshina lexical pairs are uniformly retained across all budgets tested and are therefore excluded, indicating that fixed-budget retention differences primarily arise at the sentence level rather than in isolated lexical forms. The English–Urdu baseline reinforces this effect. Under `cl100k` at $B$=16, English achieves a retention ratio of $2.614$ over Urdu, retaining more than twice as much surface content at the same budget. Although this disparity decreases as $B$ increases, it remains visible under constrained budgets, indicating that tokenization differences matter most when only a limited prefix or chunk is retained. This efficiency–retention mismatch arises from character-per-token density. When a representation is split into many short subword or byte units, a fixed token budget reaches fewer characters of the original text. Conversely, representations with higher character-per-token density preserve more surface content under the same budget. This explains why token cost alone is insufficient to predict usable input coverage.

For multilingual NLG evaluation under fixed token budgets (Gehrmann et al., 2021), this suggests that equal budgets may not correspond to equal retained surface coverage across representations. Character-level retention is a surface-form proxy rather than a direct measure of semantic preservation or downstream quality, but it provides a consistent tokenizer-agnostic indicator of how tokenization differences affect input coverage in practice.

**Cost Multipliers.** Table 5 shows that cost multipliers closely track TCI and remain strongly tokenizer-dependent. On Roman-Urdu-Parl, Roman Urdu requires substantially fewer input tokens than native Urdu under `cl100k` and ByT5, with cost multipliers of $0.463$ and $0.666$, respectively, while mT5 remains close to parity at $0.961$. Dakshina lexical pairs exhibit the same pattern: Roman Urdu is cheaper under `cl100k` ($0.549$) and ByT5 ($0.645$), but nearly identical in cost under mT5 ($0.976$). These results indicate that relative input-side cost differences arise from tokenizer-specific segmentation behavior rather than being fixed properties of script.

The English–Urdu baseline is more pronounced. On OPUS100, English requires fewer tokens than Urdu across all tokenizers, with cost multipliers of $0.264$ for `cl100k`, $0.751$ for mT5, and $0.589$ for ByT5. Under `cl100k`, this implies that an aligned Urdu input may require nearly four times as many tokens as its English counterpart. A system switching from `cl100k` to mT5 would see the Roman Urdu–Urdu cost ratio shift from $0.463$ to $0.961$, reducing the relative input-side cost advantage from roughly $54\%$ to under $4\%$. This illustrates that multilingual cost estimates depend critically on the tokenizer and are not stable across systems.

Taken together, these results show that tokenization disparities are not only descriptive but also operational: differences in segmentation translate directly into differences in input-side token usage under token-billed systems. Total generation cost may also depend on output length and output script, so these multipliers should be interpreted as input-side cost estimates rather than complete deployment costs. This still has practical implications for deployment, where tokenizer choice can materially affect system cost and retained context across languages.

**Normalization Probe.** As shown in Table 6, normalization yields negligible reductions in tokenization cost. On Roman-Urdu-Parl, normalization modifies only $1.4\%$ of examples, with token-count reductions ranging from $0.027\%$ to $0.045\%$ and TCI reductions from $0.036\%$ to $0.055\%$ across tokenizers. On Dakshina Urdu, the effect is smaller still: only $0.05\%$ of lexical pairs are modified, with

| Dataset | Tokenizer | Raw TCI | Norm TCI | Red. (%) | Chg. (%) |
|---|---|---|---|---|---|
| Roman-Urdu-Parl | cl100k | 0.473 | 0.473 | 0.055 | 1.40 |
| | mT5 | 0.967 | 0.967 | 0.044 | 1.40 |
| | ByT5 | 0.669 | 0.669 | 0.036 | 1.40 |
| Dakshina Urdu | cl100k | 0.580 | 0.579 | 0.009 | 0.05 |
| | mT5 | 1.027 | 1.027 | 0.013 | 0.05 |
| | ByT5 | 0.654 | 0.654 | 0.007 | 0.05 |

Table 6: Effect of lightweight Roman Urdu normalization on TCI.

both token-count and TCI reductions near zero. Additional token-count reduction statistics are provided in Appendix Table 11. This result has an important interpretive implication. If the observed disparities were primarily driven by surface orthographic variation, such as inconsistent spelling, repeated characters, or irregular spacing, lightweight normalization would be expected to reduce them more substantially. The limited effect observed here suggests that these disparities are more closely associated with structural properties of tokenizer design, in particular how tokenizer vocabularies and segmentation mechanisms interact with Perso-Arabic versus Latin script, rather than with incidental input variability. This suggests that lightweight preprocessing alone is unlikely to mitigate cross-script tokenization disparities, and that tokenizer-level interventions may be needed. We emphasize that this is a **lightweight normalization probe**, not a full normalization study. The results do not rule out stronger, script-aware preprocessing strategies, but they indicate that simple surface-level cleanup is insufficient in this setting.

## 7 Conclusion and Future Work

This paper shows that tokenization is a system-level factor shaping input-side cost, fragmentation, and retained surface coverage in multilingual NLP. Using a controlled cross-script comparison of Urdu and Roman Urdu, we introduce Token Cost Inequality (TCI) and a multi-axis framework spanning token cost, fragmentation, and fixed-budget retention. Across tokenizer families, we show that cross-script disparities vary from near parity to substantial divergence and are strongly shaped by tokenizer design rather than script alone. A central finding is the efficiency–retention paradox: token cost alone does not determine retention under fixed token budgets. Retention also depends on character-per-token density, revealing a disconnect between token efficiency and effective input coverage. Lightweight normalization has negligible impact, suggesting that these disparities arise primarily from structural tokenizer behavior rather than surface variation.

These findings suggest that tokenization warrants explicit treatment in multilingual evaluation and deployment. Fixed token budgets can impose unequal surface-coverage conditions across scripts, even for aligned content, suggesting that evaluation should account for tokenization-sensitive measures of usable input in addition to raw token counts. Future work should extend this analysis to other native/Romanized language pairs, newer tokenizer families, and downstream tasks to test whether retention disparities translate into measurable performance differences. It should also examine combined input-output cost models for deployed generation systems. More broadly, we argue that tokenization should be treated as an explicit object of analysis in multilingual NLP, rather than an invisible implementation choice.

## Limitations

This study has several limitations. All experiments use uniformly sampled subsets of 10,000 examples per dataset, improving comparability but not capturing the full variation of the underlying corpora. The datasets also serve different analytic roles: Roman-Urdu-Parl and OPUS100 provide sentence-level comparisons, while Dakshina is lexical, making results not fully comparable across all axes. Fixed-budget retention is measured using **character-level retention**, which captures surface coverage but does not directly measure semantic preservation, generation quality, or downstream task performance. We also evaluate a limited set of tokenizer families. Although these represent distinct segmentation strategies, they do not cover all multilingual or newer LLM-specific tokenizers used in practice. Finally, the normalization analysis is deliberately lightweight: it shows that simple surface-level preprocessing has minimal impact, but does not rule out stronger script-aware normalization or transliteration approaches. Accordingly, our results should be interpreted as controlled evidence of cross-script tokenization disparities, not a comprehensive account of multilingual tokenization or downstream effects.

## Ethical Statement

This work analyses tokenization disparities across scripts using publicly available datasets, including

Roman-Urdu-Parl, OPUS100, and Dakshina. No human subjects were involved, and no personal or sensitive data were collected or processed. The main ethical consideration concerns fairness in multilingual NLP systems. Our findings show that tokenization can introduce systematic differences in input-side cost, fragmentation, and retained surface coverage across scripts, even when content is semantically aligned. Such differences may affect how much text users can include under the same token budget. We present these findings to highlight a source of inequity in token-based systems and to motivate tokenization-aware evaluation and fairer tokenizer design. We emphasize that the observed disparities arise from tokenizer design choices rather than inherent properties of languages or scripts. The work poses minimal risk of misuse, but misinterpretation could lead to incorrect claims about language efficiency. Overall, this study supports more equitable and transparent multilingual AI systems.

## References


Mehreen Alam and Sibt Ul Hussain. 2022. Roman-urdu-parl: Roman-urdu and urdu parallel corpus for urdu language understanding. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 21(1).

Haris Bin Zia, Agha Ali Raza, and Awais Athar. 2018. Urdu word segmentation using conditional random fields (CRFs). In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2562–2569, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Umer Butt, Stalin Varanasi, and Günter Neumann. 2025. Low-resource transliteration for roman-urdu and urdu using transformer-based models. In *Proceedings of the Eighth Workshop on Technologies for Machine Translation of Low-Resource Languages*, pages 144–153.

Nadir Durrani and Sarmad Hussain. 2010. Urdu word segmentation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 528–536.

Sebastian Gehrmann, Tosin Adewumi, Karmanya Aggarwal, Pawan Sasanka Ammanamanchi, Aremu Anuoluwapo, Antoine Bosselut, Khyathi Raghavi Chandu, Miruna Clinciu, Dipanjan Das, Kaustubh D. Dhole, Wanyu Du, Esin Durmus, Ondřej Dušek, Chris Emezue, Varun Gangal, Cristina Garbacea, Tatsunori Hashimoto, Yufang Hou, Yacine Jernite, and 37 others. 2021. The GEM benchmark: Natural language generation, its evaluation and metrics. In *Proceedings of the 1st Workshop on Natural Language Generation, Evaluation, and Metrics*, pages 96–120, Online. Association for Computational Linguistics.

Vani Kanjirangat, Tanja Samardžić, Ljiljana Dolamic, and Fabio Rinaldi. 2025. Tokenization and representation biases in multilingual models on dialectal NLP tasks. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*.

Christo Kirov, Cibu Johny, Anna Katanova, Alexander Gutkin, and Brian Roark. 2024. Context-aware transliteration of romanized south asian languages. *Computational Linguistics*, 50(2):475–534.

Nelson F. Liu, Kevin Lin, Peter Chen, Rohan Taori, Alane Yang, Xin Chen, Christopher Potts, Christopher D. Manning, and Danqi Chen. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12.

OpenAI. 2022. tiktoken: A fast BPE tokenizer for use with OpenAI's models. `https://github.com/openai/tiktoken`.

Aleksandar Petrov, Emanuele La Malfa, Philip H. S. Torr, and Adel Bibi. 2023. Language model tokenizers introduce unfairness between languages. In *Advances in Neural Information Processing Systems*.

Brian Roark, Lawrence Wolf-Sonkin, Christo Kirov, Sabrina J. Mielke, Cibu Johny, Isin Demirsahin, and Keith Hall. 2020. Processing south asian languages written in the latin script: The dakshina dataset. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 2413–2423.

Menan Velayuthan and Kengatharaiyer Sarveswaran. 2025. Egalitarian language representation in language models: It all begins with tokenizers. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 5987–5996, Abu Dhabi, UAE. Association for Computational Linguistics.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Mihaylov, 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.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

Biao Zhang, Philip Williams, Jörg Tiedemann, and Rico Sennrich. 2020. OPUS-100: A free multi-lingual machine translation corpus. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5461–5470, Online. Association for Computational Linguistics.

# A Additional Results

This appendix provides additional distributional and robustness analyses that support the main findings in Section 6. In particular, we examine (i) directionality of token cost disparities, (ii) retention behavior across the full range of token budgets, (iii) heavy-tail fragmentation effects, and (iv) the magnitude of normalization-induced changes. These analyses complement the aggregate statistics in the main paper by revealing how tokenization disparities manifest across distributions and operating regimes.

## A.1 TCI Exceedance Rates

Table 7 reports the proportion of aligned pairs for which the left-side representation is more token-expensive than native Urdu (TCI $> 1.0$). This provides a distributional view of directionality beyond mean TCI.

Under `cl100k` and ByT5, exceedance rates are near zero across all datasets, indicating a consistent left-side efficiency advantage. In contrast, under mT5, $28.8\%$ of Roman-Urdu-Parl pairs and $23.6\%$ of Dakshina pairs exceed TCI $= 1.0$. This is consistent with the near-parity mean TCI values reported in Section 6, and suggests that mT5 exhibits bidirectional cost variation rather than a consistent directional disparity.

| Dataset | Tokenizer | % pairs TCI $> 1.0$ |
|---|---|---|
| Roman-Urdu-Parl | cl100k | 0.06 |
| | mT5 | 28.79 |
| | ByT5 | 0.10 |
| Dakshina Urdu | cl100k | 0.40 |
| | mT5 | 23.58 |
| | ByT5 | 0.22 |
| OPUS100 EN–UR | cl100k | 0.29 |
| | mT5 | 14.45 |
| | ByT5 | 3.54 |

Table 7: Proportion of aligned pairs where the left-side representation (Roman Urdu or English) is more token-expensive than native Urdu (TCI $> 1.0$). Values near zero indicate consistent directional efficiency; values closer to 50% indicate near-parity with bidirectional variation.

## A.2 Extended Retention Ratios

Table 8 extends Table 4 by reporting retention ratios across all token budgets ($B \in \{16, 32, 64, 128, 256\}$). This analysis provides a more complete view of truncation behavior under varying constraints.

Dakshina entries are uniformly $1.000$, as lexical pairs are short enough to be fully retained under all budgets. This supports the interpretation in Section 6 that truncation effects primarily arise at the sentence level. For Roman-Urdu-Parl and OPUS100, retention ratios consistently decrease toward parity as $B$ increases, but disparities persist under ByT5 even at $B$=128 and $B$=256. This is consistent with the fragmentation patterns discussed in Section 6, suggesting that byte-level segmentation can sustain truncation differences even at larger budgets.

| | | Retention Ratio (L/R) | | | | |
|---|---|---|---|---|---|---|
| Dataset | Tok. | $B = 16$ | $B = 32$ | $B = 64$ | $B = 128$ | $B = 256$ |
| Roman-Urdu-Parl | cl100k | 1.649 | 1.267 | 1.097 | 1.017 | 1.000 |
| | mT5 | 1.018 | 1.006 | 1.001 | 1.000 | 1.000 |
| | ByT5 | 1.514 | 1.379 | 1.159 | 1.066 | 1.016 |
| Dakshina Urdu | cl100k | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| | mT5 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| | ByT5 | 1.002 | 1.000 | 1.000 | 1.000 | 1.000 |
| OPUS100 EN–UR | cl100k | 2.614 | 1.969 | 1.437 | 1.119 | 1.015 |
| | mT5 | 1.206 | 1.126 | 1.045 | 1.003 | 0.997 |
| | ByT5 | 1.648 | 1.582 | 1.405 | 1.242 | 1.087 |

Table 8: Retention ratios across token budgets. Values above 1.0 indicate that the left-side representation preserves more surface content under the same budget.

## A.3 Extended Fragmentation Statistics

Table 9 extends Table 3 by reporting P95 TPW, maximum TPW, and the proportion of words exceeding 8 tokens per word. These statistics provide a more detailed view of heavy-tail fragmentation behavior.

Under ByT5, extreme fragmentation is observed, with native Urdu reaching maximum values of 28 tokens per word on Roman-Urdu-Parl, 26 on Dakshina, and 52 on OPUS100. This supports the interpretation in Section 6 that byte-level tokenization produces substantially heavier tails, which in turn contribute to the persistence of truncation disparities at larger budgets.

## A.4 Fragmentation Ratios (L/R)

Table 10 reports TPW ratios between left-side and native Urdu representations. Values below 1.0 indicate lower fragmentation on the left side.

Under `cl100k`, mean ratios of $0.503$ (Roman-Urdu-Parl) and $0.381$ (OPUS100) indicate substantially lower fragmentation for Roman Urdu and English. In contrast, mT5 ratios cluster near $1.0$,

| | | P95 TPW | | Max TPW | | % ≥ 8 | |
|---|---|---|---|---|---|---|---|
| **Dataset** | **Tok.** | **L** | **R** | **L** | **R** | **L** | **R** |
| Roman-Urdu-Parl | cl100k | 3 | 6 | 8 | 15 | 0.0 | 1.6 |
| | mT5 | 3 | 3 | 8 | 6 | 0.0 | 0.0 |
| | ByT5 | 7 | 12 | 15 | 28 | 4.9 | 39.8 |
| Dakshina Urdu | cl100k | 4 | 8 | 10 | 13 | 0.0 | 8.1 |
| | mT5 | 4 | 4 | 8 | 7 | 0.0 | 0.0 |
| | ByT5 | 10 | 14 | 20 | 26 | 27.1 | 88.4 |
| OPUS100 EN–UR | cl100k | 3 | 7 | 10 | 30 | 0.0 | 2.2 |
| | mT5 | 3 | 3 | 12 | 14 | 0.0 | 0.0 |
| | ByT5 | 9 | 12 | 22 | 52 | 10.7 | 36.6 |

Table 9: Extended fragmentation statistics.

consistent with its near-parity behavior in both TCI and fragmentation reported in Section 6.

| **Dataset** | **Tok.** | **Mean ratio** | **P95 ratio** | **P99 ratio** |
|---|---|---|---|---|
| Roman-Urdu-Parl | cl100k | 0.503 | 0.500 | 0.500 |
| | mT5 | 0.959 | 1.000 | 1.000 |
| | ByT5 | 0.618 | 0.583 | 0.643 |
| Dakshina Urdu | cl100k | 0.549 | 0.500 | 0.500 |
| | mT5 | 0.976 | 1.000 | 1.250 |
| | ByT5 | 0.645 | 0.714 | 0.611 |
| OPUS100 EN–UR | cl100k | 0.381 | 0.429 | 0.500 |
| | mT5 | 0.910 | 1.000 | 1.000 |
| | ByT5 | 0.673 | 0.750 | 0.786 |

Table 10: Fragmentation ratios (L/R).

### A.5 Normalization: Token Count Reductions

Table 11 reports raw and normalized token counts alongside proportional reductions. Token-count reductions are extremely small across all settings (below $0.05\%$), consistent with the negligible TCI changes reported in Section 6. This supports the interpretation that lightweight normalization has minimal impact on tokenization disparities in this setting.

| **Dataset** | **Tok.** | **Raw** | **Norm** | **Tok. red. (%)** | **TCI red. (%)** |
|---|---|---|---|---|---|
| Roman-Urdu-Parl | cl100k | 25.960 | 25.948 | 0.045 | 0.055 |
| | mT5 | 23.293 | 23.284 | 0.036 | 0.044 |
| | ByT5 | 72.625 | 72.605 | 0.027 | 0.036 |
| Dakshina Urdu | cl100k | 2.796 | 2.796 | 0.007 | 0.009 |
| | mT5 | 2.488 | 2.487 | 0.012 | 0.013 |
| | ByT5 | 6.502 | 6.501 | 0.008 | 0.007 |

Table 11: Token count reductions from lightweight normalization.

**Summary.** Across these analyses, tokenization disparities are not only visible in averages but also in distributional asymmetries, tail behavior, and robustness across token budgets. These findings complement the main results by showing that the observed effects persist across multiple analytical views and operating regimes.