Tokenization Cost, Retention, and Orthography Robustness for Ladin and Italian Varieties

Alessio Staffini
Studio Strata
alessio.staffini@bocconialumni.it

Abstract

Tokenizer mismatch is a practical bottleneck for low-resource language varieties: when text is fragmented into disproportionately many subwords or bytes, it wastes context, increases truncation, and can be brittle to orthographic variation. We present a lightweight and reproducible audit centered on Ladin and evaluated on the Identification of Languages and Dialects of Italy benchmark of eleven Italian varieties. Our diagnostic suite combines tokenization cost measures (tokens per word, truncation pressure, bytes per token) with retention indicators (word split rate, continued-token rate, and type-level retention) and fragmentation proxies that reveal splitting patterns beyond fertility. We pair these diagnostics with a conservative orthography robustness protocol (diacritics, casing, punctuation and dash normalization) and assess how diagnostic changes relate to performance drops in lightweight baselines for sentence-level variety identification. We release code and derived statistics to support reproducible tokenizer audits in other low-resource settings.

1 Introduction

Low-resource language varieties are often discussed in terms of corpus availability and model coverage. However, representational bottlenecks can appear earlier in the pipeline. Modern language models (LMs) compute over discrete tokens: if a variety is systematically over-fragmented, it incurs extra "token budget" for the same content, increases truncation under fixed context windows, and may generalize poorly under spelling or diacritic variation. Ladin is a threatened Rhaeto-Romance language spoken in the Dolomite area of Northern Italy; its limited digital resources and orthographic variation make it a useful stress test for tokenizer robustness in low-resource settings (Hammarström et al., 2025).

Figure 1 illustrates this effect on a single Ladin sentence: across tokenizer families, the same content can require very different token budgets and exhibit different sensitivity to a minimal orthographic change.

This paper audits tokenization cost and orthography sensitivity for Italian language varieties with a focus on Ladin, using the Identification of Languages and Dialects of Italy benchmark (ITDI) (Aepli et al., 2022).[1] Our contribution is not a new tokenization metric; rather, we package a reproducible and lightweight diagnostic protocol for a realistic low-resource setting, and connect tokenization diagnostics to downstream robustness under controlled orthography perturbations. We aim for a method that practitioners can run *before* committing to a tokenizer/model for a low-resource variety.

We structure the study around three questions:

- **RQ1 (Cost):** How does tokenization cost vary across tokenizers and varieties (especially Ladin) in terms of fertility, truncation pressure, and byte/character cost?
- **RQ2 (Robustness):** Which tokenizers and feature representations are most brittle to realistic orthographic variation (diacritics/apostrophes/punctuation/dashes)?
- **RQ3 (Association):** Which diagnostic measures (especially retention and byte-cost metrics) are most strongly associated with robustness drops under orthographic perturbations in our evaluation baselines?

Our contributions are as follows: (i) A Ladin-centered audit slice on ITDI, with explicit split sanity checks and reproducible preprocessing; (ii) A diagnostic battery for tokenization *cost* and *retention* (fertility, char/byte ratios, fragmentation proxies, word split rate, continued-token rate, type re-

[1] Code and derived statistics (v1.0.0): https://github.com/staale92/ladin-itdi-tokenizer-audit/tree/v1.0.0.

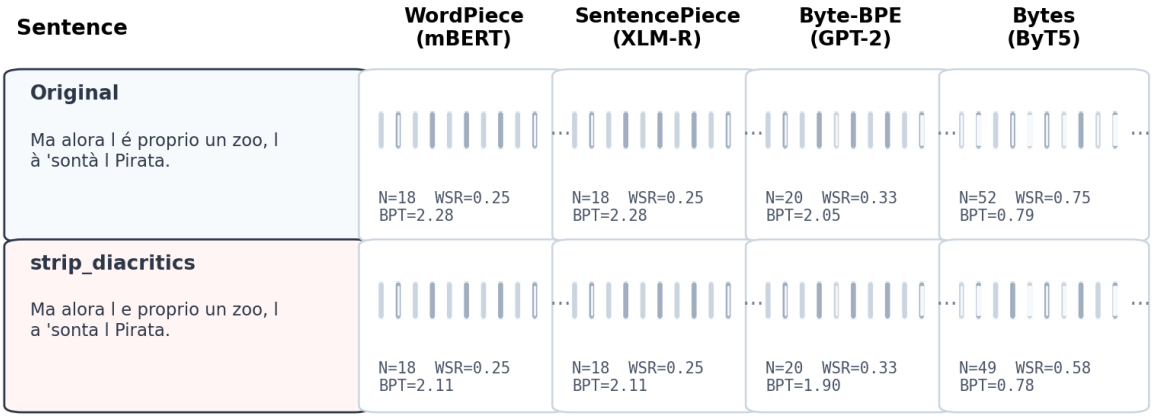

Figure 1: Same content, different token budgets and different sensitivity to spelling/diacritics. We show one Ladin test sentence and a minimally perturbed spelling (diacritics stripped), tokenized by four representative tokenizer families. For each tokenizer we report token count (N), word split rate (WSR), and bytes per token (BPT).

tention, token-length quantiles); and (iii) A simple orthography robustness protocol and lightweight baselines that link diagnostic shifts to performance drops under perturbations.

2 Related Work

Subword tokenizers behave unevenly across languages; measures such as fertility (tokens-per-word) and token-level continuation indicators have been used to diagnose performance degradation (Rust et al., 2021). Fertility alone can obscure whole-word retention effects; we therefore report word-level split rates, token-level continuation rates, and type-retention measures alongside fertility, including STRR (a type-retention-style metric) (Rust et al., 2021; Nayeem et al., 2025).

We compare widely used tokenizer families used in popular multilingual model ecosystems: WordPiece (Schuster and Nakajima, 2012; Devlin et al., 2019), SentencePiece (Kudo and Richardson, 2018; Conneau et al., 2020), byte-level byte-pair encoding (BPE) (Radford et al., 2019), and byte-level tokenization (Xue et al., 2022). We evaluate these tokenizers in the context of ITDI, a sentence-level benchmark of 11 Italian varieties designed to test generalization beyond Wikipedia-derived training data (Aepli et al., 2022); its domain shift makes it a useful testbed for orthography sensitivity.

3 Data and Preprocessing

3.1 Dataset and Splits

The Identification of Languages and Dialects of Italy benchmark (ITDI) provides training data as Wikipedia-derived text for eleven Italian language varieties and dev/test splits drawn from non-Wikipedia sources (Aepli et al., 2022). We refer to varieties by their ITDI label codes when reporting per-label diagnostics and results (Table 1); in particular, Ladin (LLD) appears in test but not in the official dev split.

Code	Variety
LLD	Ladin
FUR	Friulian
VEC	Venetian
LMO	Lombard
LIJ	Ligurian
EML	Emilian-Romagnol
NAP	Neapolitan
PMS	Piedmontese
ROA_TARA	Tarantino
SC	Sardinian
SCN	Sicilian

Table 1: ITDI label codes used throughout the paper.

Split	#sentences	Notes
Train	215997	Wikipedia-derived text
Dev	10799	Official ITDI dev + train holdout
Test	11087	Official ITDI test (non-empty)

Table 2: Overall split sizes after preprocessing (total=237883).

Table 2 summarizes the resulting split sizes after preprocessing and split sanity checks.

We align our preprocessing with the shared-task evaluation by: (i) verifying processed dev/test per-label counts against published counts, (ii) removing empty lines from test, and (iii) recording which

labels appear in each split.

3.2 Preprocessing and Sampling

For practicality, we cap Wikipedia-derived training data at 20,000 sentences per variety and create dev holdouts (1,000 sentences each) for varieties missing an official dev split (LLD, EML, NAP, ROA_TARA). Holdouts are sampled per variety without replacement from the capped training pool after excluding any sentence strings present in official dev/test, to reduce leakage. These Wikipedia-derived holdouts are used only for model selection (e.g., hyperparameter tuning); all headline robustness results are reported on the official ITDI test split.

Training text is extracted from the provided Wikipedia-derived materials. Dev/test are loaded from the official released files. We apply minimal cleaning (e.g., stripping empty lines), preserving punctuation and diacritics to keep orthography effects observable.

To bound runtime, we cap examples per (split,label) using seeded reservoir sampling. The seed is stable per bucket to ensure reproducibility.

4 Tokenizers

We evaluate representative tokenizer families using Hugging Face tokenizers (no model training required). Table 3 lists the tokenizers used in this paper.

Our comparison is intentionally practical: we audit the exact tokenizer and pre-tokenization configurations shipped with popular pretrained multilingual models. As a result, differences reflect both tokenization family and design choices such as vocabulary/training data and internal normalization rules. Tokenizers can also apply internal normalization (e.g., lowercasing or accent stripping); Table 3 reports best-effort flags where exposed, and invariance to perturbations can reflect such internal normalization.

Byte-level tokenizers are not optimized for whole-word retention by design; we therefore interpret retention metrics (Section 5.2) primarily for subword tokenizers and rely on byte/character cost metrics to compare representational efficiency across tokenizer families.

5 Diagnostics

Overview. Our diagnostics fall into three groups: (i) cost and fragmentation (e.g., tokens per word, bytes per token, truncation pressure), (ii) retention (word split rate, continued-token rate, type retention), and (iii) robustness sensitivity (delta diagnostics between original and normalized variants). For readability, we use the following abbreviations throughout: TPW=tokens per word; TPC=tokens per character; CPT=characters per token; BPT=UTF-8 bytes per token (UTF-8 is a standard Unicode encoding); TP_L=truncation pressure at limit L; WSR=word split rate; CTR=continued-token rate; TypeRet=type retention (abbrev. TR in tables; TypeRet@K restricts to the top-K most frequent types and is written TR@K in tables). "UNK" denotes the tokenizer's unknown token (e.g., `[UNK]` or `<unk>`).

We compute diagnostics per tokenizer T, variety label V, split S, and normalization variant n. Tokenization excludes model special tokens (`add_special_tokens=False`). Words are extracted with a Unicode-aware regex; characters/bytes exclude whitespace.

The full diagnostic battery is released as machine-readable comma-separated values (CSV) (per tokenizer×label×split×variant) in the accompanying artifact; the paper surfaces a compact subset in Table 6 and uses both absolute and sensitivity diagnostics in the correlation analysis (Table 8).

Let x be an input sentence, and let $n(x)$ denote a deterministic normalization variant (possibly identity). Let $\tau_T(\cdot)$ map a string to a sequence of tokens under tokenizer T. Let $|\tau_T(n(x))|$ be token length. Let $w(x)$ be the number of words in the *original* string and $w(n(x))$ the number of words in the normalized string. Let $c(x)$ and $b(x)$ be non-whitespace character and UTF-8 byte counts, respectively.

5.1 Cost and Fragmentation Diagnostics

We report fertility and its inverses using *original-text denominators* by default to make sensitivity interpretable:

$$\text{TPW}(T,n) = \frac{\sum_x |\tau_T(n(x))|}{\sum_x w(x)} \tag{1}$$

$$\text{TPC}(T,n) = \frac{\sum_x |\tau_T(n(x))|}{\sum_x c(x)} \tag{2}$$

$$\text{CPT}(T,n) = \frac{\sum_x c(x)}{\sum_x |\tau_T(n(x))|} \tag{3}$$

$$\text{BPT}(T,n) = \frac{\sum_x b(x)}{\sum_x |\tau_T(n(x))|}. \tag{4}$$

Name	Vocab	Hugging Face ID	Unit	Lower	Accents
byt5_bytes	256	`google/byt5-small`	byte	unk	unk
gpt2_bytebpe	50257	`gpt2`	byte-BPE	unk	unk
mbert_wordpiece	119547	`bert-base-multilingual-cased`	subword	no	unk
xlmr_sentencepiece	250002	`xlm-roberta-base`	subword	unk	unk

Table 3: Tokenizers audited and selected internal normalization flags (best-effort). "Lower" and "Accents" report whether the tokenizer lowercases or strips accents internally (where exposed by the tokenizer implementation).

For reference, we also report `*_normdenom` variants that use $w(n(x))$ and $c(n(x))$. For bytes, we analogously report:

$$\text{BPT\_normdenom}(T, n) = \frac{\sum_x b(n(x))}{\sum_x |\tau_T(n(x))|}. \quad (5)$$

We report both views: BPT isolates token-count changes under normalization, while BPT_normdenom reflects effective bytes-per-token after normalization (useful when diacritics change UTF-8 byte length).

Fragmentation proxies. We compute (i) mean visible token length (after removing common subword markers; Appendix A, Appendix Table 9) and (ii) the proportion of visible-length-1 tokens. These proxies capture excessive fragmentation not reflected by fertility alone.

Truncation pressure. For typical LM limits $L \in \{128, 256, 512\}$, truncation pressure is:

$$\text{TP}_L(T, n) = \frac{1}{N} \sum_x \mathbb{I}\left[|\tau_T(n(x))| > L\right]. \quad (6)$$

We also report token-length quantiles $p50/p95/p99$ over sentence token lengths to summarize tail behavior. Although ITDI is sentence-level, truncation pressure is a useful diagnostic for downstream settings that impose fixed maximum lengths.

5.2 Retention Diagnostics

We measure whether words are preserved as single tokens or split into multiple tokens. For each word token u in the original sentence x, we compute the number of subtokens $m_T(u)$ by tokenizing a leading-space form to approximate mid-sentence behavior for whitespace-sensitive tokenizers. Concretely, we tokenize the string `" " + u`, where u is extracted by a Unicode-aware word regex (Appendix B); punctuation outside the regex is excluded from the word-level probe. If the tokenizer emits a standalone whitespace token or marker-only prefix due to the added leading space, we drop such prefixes so $m_T(u)$ reflects the word itself; Table 4 (and Appendix Table 15) provide qualitative examples. We then define:

$$\text{WSR}(T, n) = \frac{\sum_u \mathbb{I}[m_T(n_w(u)) \geq 2]}{\sum_u 1} \quad (7)$$

where $n_w(\cdot)$ applies only boundary-preserving operations (lowercasing/diacritics/apostrophe normalization) so that word boundaries remain interpretable.

We additionally report a token-level *continued-token rate* (CTR) that measures within-word continuation:

$$\text{CTR}(T, n) = \frac{\sum_u \max(0, m_T(n_w(u)) - 1)}{\sum_u m_T(n_w(u))}. \quad (8)$$

This definition is robust to the common edge case in whitespace-marker tokenizers where the sentence-initial word lacks a preceding-space marker; operationally, it counts all subtokens after the first as continuations. We exclude UNK-tokenized words from the CTR numerator and denominator.

We also compute a type retention rate (TypeRet), inspired by STRR-style measures:

$$\text{TypeRet}(T, n) = \frac{|\{u \in \mathcal{V} : m_T(n_w(u)) = 1\}|}{|\mathcal{V}|}, \quad (9)$$

where $\mathcal{V}$ is the set of observed word types. We report `TypeRet` over all observed types, and `TypeRet@K` over the top-K most frequent types ($K \in \{500, 1000\}$) to reduce sensitivity to rare types.

Worked example. In Table 4, mBERT tokenizes `mè` into two subtokens (`m` and `##U+00E8`); it therefore counts as split for WSR and contributes one continued token out of two for CTR (CTR=$1/2$ for that word).

Word	mBERT WordPiece	XLM-R SentencePiece	GPT-2 byte-BPE	ByT5 byte-level
sò	sU+00F2	U+2581s U+00F2	U+0120s U+00C3 U+00B2	s U+00C3 U+00B2
mè	m ##U+00E8	U+2581mU+00E8	U+0120m U+00C3 U+00A8	m U+00C3 U+00A8
stà	st ##U+00E0	U+2581s tU+00E0	U+0120st U+00C3 U+0142	s t U+00C3 U+00A0

Table 4: Qualitative tokenization examples for three frequent Ladin words (test split), illustrating marker conventions and non-ASCII handling across tokenizers (mBERT WordPiece, XLM-R SentencePiece, GPT-2 byte-BPE, and ByT5 byte-level; see Table 3). Spaces separate tokens; the WordPiece continuation marker is shown where emitted. We use Unicode codepoint renderings (e.g., U+2581 and U+0120) for readability; see Appendix Table 15 for more examples.

UNK handling. We distinguish single-token retention from collapse to [UNK]/<unk>: we report UNK word-token/type rates and exclude UNK-tokenized words from the retained sets when computing type retention.

6 Orthography Robustness Protocol

We evaluate sensitivity to realistic orthographic variation via deterministic transformations. We separate: (i) *normalizations* used to probe diagnostic sensitivity, and (ii) *perturbations* used as stress tests for downstream evaluation. These perturbations are conservative typographic canonicalizations and substitutions; we evaluate invariance to common alternative spellings and punctuation conventions rather than adversarial noise. For example, strip_diacritics maps accented letters to their base forms (e.g., é → e) rather than deleting letters. Models are trained on the unperturbed training split; perturbations are applied only at evaluation time (dev/test) to measure invariance. We intentionally prioritize language-agnostic, deterministic operations for reproducibility; we do not model language-specific substitution/confusion sets (e.g., s↔z) or phonetic misspellings, which could be incorporated via custom rules in our released configuration (Appendix C).

Table 5 lists the orthography operations used in our robustness protocol.

Coverage varies substantially by operation: punctuation spacing changes 99.5% of sentences, lowercasing 84.8%, and diacritics stripping 76.6%; apostrophe and dash normalization are no-ops (0%) on this ITDI slice (Appendix Table 10).

Name	Description / example
strip_diacritics	é → e; ü → u
apostrophe_normalize	U+2019 → U+0027
punctuation_spacing	"word ," → "word,"
dash_normalize	–/– → -
lowercase	A → a

Table 5: Orthography operations (deterministic canonicalizations/substitutions).

7 Baselines and Evaluation

We use sentence-level variety identification on ITDI dev/test splits. We use linear support vector machines (SVMs) because they are strong, widely used baselines for variety identification and help isolate representation effects.

- **Linear SVM (character TF-IDF):** character n-gram term frequency–inverse document frequency (TF-IDF) ($n = 1$–4) + LinearSVC.
- **Linear SVM (token TF-IDF; tokenizer tokens):** tokenizer-derived token n-gram TF-IDF ($n = 1$–2) + LinearSVC.
- **Frozen encoder baseline (lightweight neural probe):** we embed sentences with a frozen multilingual transformer encoder and train a linear SVM on the resulting sentence embeddings.

For token-based baselines, we use the raw tokenizer tokens (including boundary markers such as /U+2581/U+0120 where applicable) as features; we do not add model special tokens. All TF-IDF vectorizers use lowercase=False so casing effects are evaluated only via explicit perturbations. For the frozen encoder, the tokenizer and encoder are coupled by the pretrained model; we interpret results as a robustness check rather than a tokenizer-only ablation. In our experiments, the frozen encoder is distilbert-base-multilingual-cased with mean pooling and max length 96 (Appendix D, Appendix Table 11).

Model hyperparameters are fixed or tuned on dev; full settings are reported in Appendix D (Appendix Table 11).

We emphasize that these baselines are used as controlled probes of representational stability under tokenizer-induced variation, not as a proxy for end-to-end fine-tuning. Full-parameter fine-tuning can in principle learn to compensate for some segmentation differences, whereas tokenization cost (sequence length, truncation risk, and compute) remains regardless of downstream adaptation; we therefore treat downstream experiments as lightweight validation of diagnostic signal and discuss generalization limits in the Limitations section. Token TF-IDF is a particularly direct probe of surface-form stability: when perturbations or tokenization changes alter token strings, train/test feature overlap decreases and performance can drop sharply. This makes it a useful early warning signal for tokenizer-induced instability even when end-to-end fine-tuning may later recover some robustness.

ITDI dev/test contain subsets of labels. We report: (i) macro-F1 over labels present in the gold split (primary), and (ii) macro-F1 over all training labels (diagnostic; labels absent from a split contribute 0 under `zero_division=0`), as well as per-label F1. We also report robustness drops under perturbations (original test vs perturbed test). For transparency and reuse, we provide CSV artifacts in the accompanying repository; Table 7 and Appendix Table 12 summarize the macro-$F1_{\text{present}}$ robustness drops used in this paper.

For the linear SVM baselines, we compute paired stratified bootstrap 95% confidence intervals (B=1000) for macro-F1 drops by resampling test sentences within gold labels (Appendix E, Appendix Table 14). For diagnostic–drop correlations we bootstrap over (tokenizer,label) pairs (Table 8).

8 Results

8.1 RQ1: Tokenization cost and truncation

Table 6 summarizes cost diagnostics for Ladin and comparator varieties. Figure 2 visualizes tokenization cost across tokenizers.

Token budget differs sharply across tokenizer families. For Ladin (LLD), subword tokenizers (WordPiece/SentencePiece) require ∼1.6 tokens/word, GPT-2 byte-BPE ∼1.9, while byte-level tokenization (ByT5) requires ∼4.6 (Table 6). Among the comparator varieties, LIJ shows consistently higher cost than LLD under subword tokenizers, suggesting that tokenization burden is not uniform even across closely related varieties. Retention diagnostics follow the same ordering (higher WSR/CTR and lower TR@500 for more fragmented tokenizations), with UNK word rates at 0.0 in these slices, indicating that differences stem from segmentation rather than collapse to `[UNK]`. Truncation pressure is negligible on ITDI (sentence-level), but we retain it as a practical diagnostic for longer-context LM settings.

8.2 RQ2: Robustness under orthography perturbations

We evaluate robustness by applying perturbations to the test set and measuring performance drops. Table 7 reports macro-F1 and per-label drops for each representation/tokenizer.

On the unperturbed ITDI test set, the character TF-IDF SVM is strongest (macro-$F1_{\text{present}}$=0.737), token TF-IDF baselines are competitive (0.642–0.693), and the frozen encoder baseline reaches 0.585 (Table 7). Under diacritics stripping, all models degrade; among token-based baselines, byte-BPE shows the smallest macro drop (0.036) and ByT5 tokens the largest (0.093), while the frozen encoder drops by 0.111. For Ladin, the ByT5 token baseline shows a larger per-label drop (0.172) than the subword/byte-BPE token baselines (Table 7). `apostrophe_normalize` is a no-op on this slice (Appendix Table 10), hence its drop is 0.0 and the combined perturbation matches diacritics stripping. Selected macro/per-label drops are in Table 7; the full perturbation grid is in Appendix Table 12, with a visualization of selected drops in Appendix Figure 4. We emphasize that the "byte-level brittleness" finding is specific to token-TF-IDF features built from byte-level tokens; it does not imply that byte-level *neural* models are intrinsically brittle.

8.3 RQ3: Which diagnostics are associated with robustness drops?

We analyze correlations between diagnostic measures and robustness drops. Figure 3 shows an illustrative focus-label relationship by tokenizer; Table 8 summarizes correlations computed over all (tokenizer,label) pairs (labels present in ITDI test) and aggregated across perturbations. We include both absolute diagnostics and sensitivity diagnostics (delta metrics between original and perturbed variants). Appendix Table 13 fits a simple regression modeling per-label F1 drops using sensitivity diagnostics while controlling for baseline F1.

Tokenizer	Label	TPW ↓	BPT ↑	WSR ↓	CTR ↓	TR@500 ↑	UNK_w ↓	TP_{256} ↓
byt5_bytes	LLD	4.626	0.80	0.836	0.718	0.034	0.000	0.000
	FUR	5.509	0.83	0.900	0.773	0.016	0.000	0.002
	VEC	5.424	0.83	0.907	0.771	0.014	0.000	0.000
	LMO	5.543	0.83	0.931	0.775	0.014	0.000	0.000
	LIJ	5.546	0.83	0.827	0.774	0.022	0.000	0.001
gpt2_bytebpe	LLD	1.864	1.99	0.484	0.413	0.264	0.000	0.000
	FUR	2.124	2.15	0.552	0.496	0.238	0.000	0.000
	VEC	2.092	2.15	0.629	0.493	0.188	0.000	0.000
	LMO	2.253	2.04	0.596	0.526	0.216	0.000	0.000
	LIJ	2.533	1.83	0.641	0.574	0.152	0.000	0.000
mbert_wordpiece	LLD	1.624	2.28	0.368	0.320	0.436	0.000	0.000
	FUR	1.877	2.43	0.455	0.425	0.376	0.000	0.000
	VEC	1.708	2.63	0.413	0.372	0.468	0.000	0.000
	LMO	1.948	2.36	0.488	0.444	0.360	0.000	0.000
	LIJ	2.214	2.09	0.563	0.505	0.282	0.000	0.000
xlmr_sentencepiece	LLD	1.621	2.28	0.358	0.317	0.464	0.000	0.000
	FUR	1.808	2.52	0.454	0.403	0.390	0.000	0.000
	VEC	1.685	2.66	0.406	0.363	0.470	0.000	0.000
	LMO	1.946	2.37	0.497	0.446	0.344	0.000	0.000
	LIJ	2.123	2.18	0.547	0.486	0.308	0.000	0.000

Table 6: Main tokenization cost and retention diagnostics on ITDI test (variant=original). TPW=tokens/word, BPT=UTF-8 bytes/token (original denominator), WSR=word split rate (fraction of word tokens split into $\geq$2 subtokens), CTR=continued-token rate (fraction of word subtokens that are continuations, i.e., not the first subtoken; UNK-tokenized words excluded), TR@500=type retention over top-500 frequent word types, UNK_w=UNK word-token rate, TP_{256}=fraction of sentences over 256 tokens.

Across token-based baselines, absolute fertility alone (TPW) is only weakly associated with robustness drops (median Spearman $\rho = 0.139$; Table 8). In contrast, bytes per token (BPT) shows a weak, borderline negative association with drops ($\rho = -0.312$; 95% CI upper bound close to 0), suggesting that tokenizations that cover more UTF-8 bytes per token (i.e., fewer tokens per byte) may be slightly more robust under orthographic perturbations in this setting. Sensitivity diagnostics provide stronger signal: ΔTPW correlates with drops (median $\rho = -0.374$), and ΔWSR remains associated after controlling for baseline F1 (Appendix Table 13). We treat these relationships as diagnostic rather than definitive given the small tokenizer/perturbation set.

9 Discussion and Practical Recommendations

9.1 What the audit can tell you before training

We summarize practical, model-agnostic lessons for choosing tokenizers in low-resource varieties:

- Do not rely on fertility alone; include retention diagnostics (word split/continuation rates and type retention) and byte-cost measures.
- Measure truncation pressure at realistic limits (128/256/512) and report tail quantiles.
- Evaluate robustness under conservative orthography perturbations when evaluation domain differs from training.

A simple audit recipe.

1. Pick candidate tokenizers you would realistically deploy.
2. Compute cost and retention diagnostics (TPW/BPT/TP_L; WSR/CTR/TypeRet@K) on dev-like text.
3. Choose 2–3 plausible perturbations (e.g., diacritics stripping, lowercasing, punctuation spacing) and verify non-trivial coverage.
4. Recompute diagnostics on perturbed text and compare delta metrics (ΔTPW, ΔWSR, etc.).
5. Decide whether to switch tokenizers, add normalization/augmentation, or accept the trade-offs and verify end-to-end.

9.2 What requires end-to-end verification

We recommend running a small audit on dev/test-like text before committing to a tokenizer: high

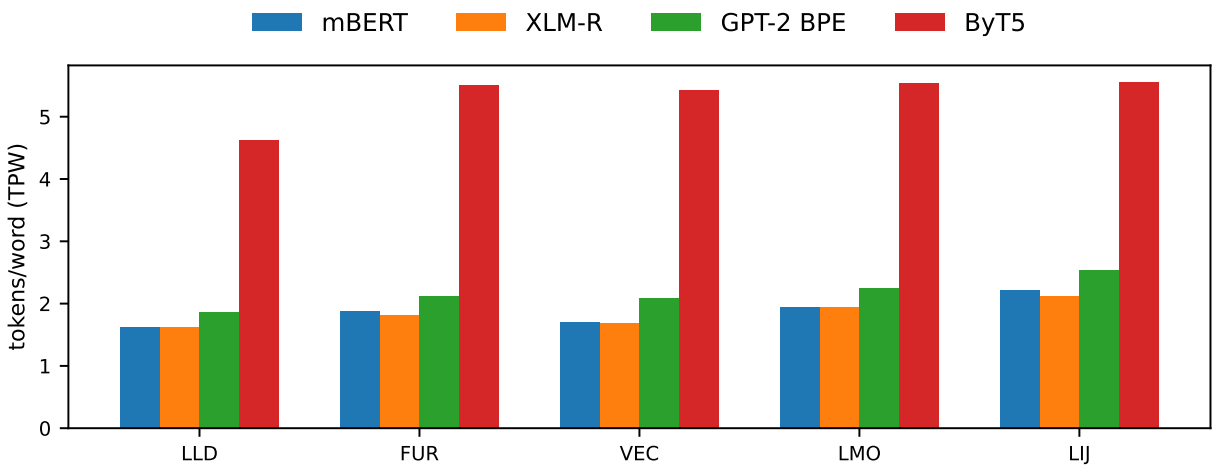

Figure 2: Tokenization cost (tokens/word) by variety and tokenizer on ITDI test (variant=original).

Model	Perturbation	F1 (test)	F1 (pert)	Drop ↓	LLD drop ↓
char_tfidf	apostrophe_normalize	0.737	0.737	0.000	0.000
token_tfidf.mbert_wordpiece	apostrophe_normalize	0.691	0.691	0.000	0.000
token_tfidf.xlmr_sentencepiece	apostrophe_normalize	0.693	0.693	0.000	0.000
token_tfidf.gpt2_bytebpe	apostrophe_normalize	0.685	0.685	0.000	0.000
token_tfidf.byt5_bytes	apostrophe_normalize	0.642	0.642	0.000	0.000
frozen_encoder.distilbert-base-multilingual-cased	apostrophe_normalize	0.585	0.585	0.000	0.000
frozen_encoder.distilbert-base-multilingual-cased	strip_diacritics	0.585	0.474	0.111	0.053
token_tfidf.byt5_bytes	strip_diacritics	0.642	0.549	0.093	0.172
char_tfidf	strip_diacritics	0.737	0.688	0.049	0.066
token_tfidf.xlmr_sentencepiece	strip_diacritics	0.693	0.650	0.043	0.049
token_tfidf.mbert_wordpiece	strip_diacritics	0.691	0.648	0.043	0.067
token_tfidf.gpt2_bytebpe	strip_diacritics	0.685	0.648	0.036	0.062
frozen_encoder.distilbert-base-multilingual-cased	strip_diacritics+apostrophe	0.585	0.474	0.111	0.053
token_tfidf.byt5_bytes	strip_diacritics+apostrophe	0.642	0.549	0.093	0.172
char_tfidf	strip_diacritics+apostrophe	0.737	0.688	0.049	0.066
token_tfidf.xlmr_sentencepiece	strip_diacritics+apostrophe	0.693	0.650	0.043	0.049
token_tfidf.mbert_wordpiece	strip_diacritics+apostrophe	0.691	0.648	0.043	0.067
token_tfidf.gpt2_bytebpe	strip_diacritics+apostrophe	0.685	0.648	0.036	0.062

Table 7: Robustness under selected perturbations on ITDI test. Macro-F1 is over labels present in test (macro_f1_present). Drop = F1(test) − F1(pert) (difference in macro-F1 points).

word-split/continuation rates and low type retention are warning signs for surface-form sensitivity, while higher token costs (e.g., more tokens per word/byte) directly imply higher truncation risk and compute. For the ITDI varieties studied here, subword tokenizers (WordPiece/SentencePiece) provide a strong trade-off between token budget, retention, and robustness in our token-TF-IDF probes; byte-BPE is slightly more robust in macro drop but also more token-expensive, while byte-level tokenization is substantially more token-expensive and correspondingly more sensitive in token-based linear classifiers. We do not claim that the robustness ranking among tokenizer families will necessarily match full fine-tuning; because our downstream validation uses lightweight probes (linear models and a frozen encoder), we treat the ranking as diagnostic signal and recommend end-to-end verification in the intended deployment setting. More generally, when a perturbation is plausible at test time (e.g., missing diacritics), we recommend aligning the training pipeline through normalization or augmentation rather than assuming invariance, and verifying that each proposed perturbation actually changes the evaluation text (some canonicalizations may be no-ops in a given dataset).

10 Conclusion

We presented a Ladin-centered audit of tokenization and orthography bias on ITDI. Our main contribution is a reproducible protocol combining cost/retention diagnostics with an orthography robustness evaluation, and showing that several diagnostics are consistently associated with down-

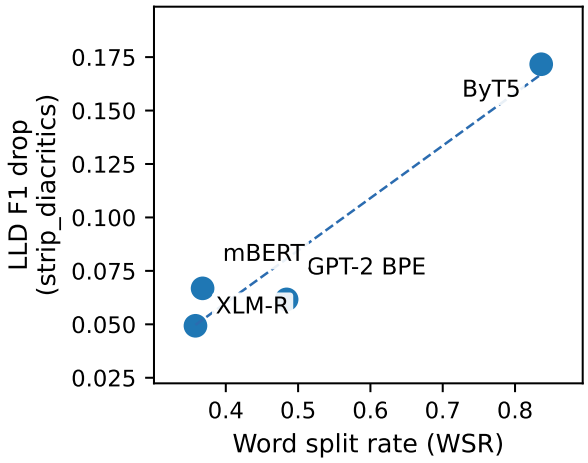

Figure 3: Diagnostics vs robustness for the focus label (LLD): word split rate (WSR) vs per-label F1 drop under diacritics stripping (token-based baselines).

Diagnostic	Median ρ	95% CI
ΔTPW	-0.374	[-0.568, 0.041]
ΔWSR	-0.186	[-0.520, 0.220]
ΔCTR	-0.247	[-0.556, 0.148]
ΔBPT	0.141	[-0.227, 0.398]
ΔTR@500	0.150	[-0.236, 0.498]
TPW	0.139	[-0.111, 0.396]
WSR	0.067	[-0.187, 0.347]
CTR	0.119	[-0.118, 0.372]
BPT	-0.312	[-0.502, -0.003]
TR@500	-0.148	[-0.399, 0.133]

Table 8: Median Spearman correlation between diagnostics and per-label F1 drops under orthography perturbations for token-based baselines (token_tfidf). Correlations are computed within each perturbation over (tokenizer,label) pairs (labels present in ITDI test), and aggregated by median ρ with bootstrap CIs over pairs.

stream robustness drops in lightweight probes. Raw ITDI text is obtained from the shared-task distribution; we release code and derived statistics.

Limitations

Our downstream validation uses token TF-IDF linear classifiers and a frozen encoder probe. These are intended as controlled probes of surface-form stability under tokenization and orthographic variation, not as a proxy for end-to-end fine-tuning; as a result, robustness rankings may not directly transfer to fully fine-tuned neural models. Adding fully fine-tuned multilingual transformer baselines (e.g., XLM-R/ByT5) is left to future work. At the same time, several parts of the audit are model-agnostic: tokenization cost diagnostics (sequence length, truncation pressure, bytes per token) affect compute and context budget regardless of downstream adaptation, even if fine-tuning can compensate for some segmentation differences.

Our robustness protocol focuses on deterministic, reproducible canonicalizations/substitutions; it does not capture language-specific confusion sets, phonetic spellings, or human-in-the-loop misspelling models.

Finally, we focus on ITDI as a realistic, low-resource variety identification benchmark; applying the protocol to additional languages and tasks is future work. The diagnostic suite itself is dataset-agnostic and can be rerun on other corpora by providing text, tokenizers, and (optionally) custom perturbation rules to the released pipeline. Differences between Wikipedia-derived training text and dev/test sources may also interact with tokenization effects; we mitigate this by focusing evaluation on official ITDI dev/test and reporting per-label results.

Ethical Considerations

We avoid deficit framing: disparities are treated as tooling and modeling gaps, not language deficits. We follow dataset licensing and recommend redistributing scripts and derived statistics rather than raw text if redistribution rights are unclear. We use respectful terminology; community preferences may vary between "language" and "dialect."

Acknowledgments

We thank the organizers and contributors of the VarDial 2022 ITDI shared task for releasing the benchmark and documentation used in this study. We also thank the developers and maintainers of the open-source tools we relied on (e.g., Hugging Face tokenizers/transformers and scikit-learn). Any remaining errors are our own.

References

Noëmi Aepli, Antonios Anastasopoulos, Adrian-Gabriel Chifu, William Domingues, Fahim Faisal, Mihaela Găman, Radu Tudor Ionescu, and Yves Scherrer. 2022. Findings of the VarDial evaluation campaign 2022. In *Proceedings of the Ninth Workshop on NLP for Similar Languages, Varieties and Dialects*, pages 1–13, Gyeongju, Republic of Korea. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised

cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

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

Harald Hammarström, Robert Forkel, Martin Haspelmath, and Sebastian Bank. 2025. Glottolog 5.2. Accessed 2026-01-31.

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

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

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. Technical report.

Phillip Rust, Jonas Pfeiffer, Ivan Vulić, Sebastian Ruder, and Iryna Gurevych. 2021. How good is your tokenizer? on the monolingual performance of multilingual language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3118–3135, Online. Association for Computational Linguistics.

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

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.

Appendix overview. This appendix expands on implementation details and analysis choices that support reproducibility and interpretation. We document marker handling (Appendix A), our word-extraction heuristic (Appendix B), baseline settings (Appendix D), and additional statistical results (Appendix E).

A Tokenizer marker handling

Visible-length markers. For fragmentation proxies (mean visible token length and single-character token rates) we compute a marker-stripped "visible length" for each token. Appendix Table 9 summarizes the marker conventions used across tokenizers.

Leading-space word probe. For WSR/CTR we estimate per-word subtoken counts by tokenizing `" " + word` and dropping any leading whitespace-only or marker-only prefix tokens introduced by the added space. This avoids inflating counts for whitespace-sensitive tokenizers and ensures the first word behaves like a mid-sentence word.

Tokenizer family	Marker	Handling
WordPiece		stripped (prefix)
SentencePiece	U+2581	stripped (prefix)
Byte-level BPE	U+0120	stripped (prefix)
Byte tokens	<0x..>	visible length = 1

Table 9: Marker handling for visible token length (used for mean token length and single-character token rates). For word-level retention measures we tokenize words as `" " + word` and drop leading whitespace-only and marker-only prefix tokens introduced by the added space (e.g., <0x20> or `" "`).

B Word extraction regex

Heuristic "word" definition. Word-level retention metrics require a stable, language-agnostic notion of "word". We use a Unicode-aware regex that retains letters, marks, digits, and common intra-word punctuation such as apostrophes and hyphens. This heuristic supports the Ladin/Italian-varieties setting (frequent clitics and diacritics), but it does not aim to match any specific linguistic tokenization standard.

```
WORD_RE = \p{L}[\p{L}\p{M}\p{N}'\u2019-]*
```

We treat words as non-overlapping regex matches on the (possibly normalized) Unicode string. This heuristic includes ASCII apostrophes (', U+0027) and U+2019 inside tokens, but does not attempt full morphological tokenization.

C Perturbation specifications

We use the perturbations listed in `configs/perturbations.yaml`, which mirror the normalization variants used for diagnostic sensitivity. All perturbations are deterministic canonicalizations/substitutions and are applied only at evaluation time to measure invariance. Appendix Table 10 reports perturbation coverage (percent of sentences whose surface form changes) to guard against no-op operations on a given evaluation slice.

D Baseline hyperparameters

Vectorizer comparability. To make representation comparisons meaningful, we keep vectorizer feature caps and document TF-IDF settings (ngram ranges, analyzer choices, and case sensitivity). Appendix Table 11 reports the complete configuration used for the linear SVM baselines and the frozen encoder baseline.

E Additional results

Robustness across all perturbations. Appendix Table 12 reports macro-F1 drops for every perturbation evaluated (Table 5), complementing the coverage statistics in Table 10.

Regression. Appendix Table 13 reports an ordinary least squares (OLS) regression over (tokenizer,label,perturbation) points that complements the correlation analysis in Table 8. We standardize predictors to make coefficients comparable and include baseline F1 as a control for label difficulty.

Predictor	β (std.)	95% CI
ΔTPW	-0.041	[-0.088, 0.015]
ΔWSR	-0.049	[-0.068, -0.022]
ΔBPT	-0.048	[-0.092, 0.003]
Baseline F1	0.017	[0.006, 0.026]

Table 13: OLS regression modeling per-label F1 drop using standardized sensitivity diagnostics and baseline F1, over points (tokenizer,label,perturbation) for token-based baselines (N=192). Coefficients are on standardized predictors. R^2=0.382 with bootstrap 95% CI [0.245, 0.622] over points (B=2000).

Bootstrap confidence intervals. Appendix Table 14 reports paired, stratified bootstrap confidence intervals for macro-F1 drops of the linear SVM baselines. We compute CIs only for models

Perturbation	% changed (all)	% changed (LLD)
strip_diacritics	76.6%	77.3%
apostrophe_normalize	0.0%	0.0%
dash_normalize	0.0%	0.0%
lowercase	84.8%	89.0%
punctuation_spacing	99.5%	99.5%
strip_diacritics + apostrophe	76.6%	77.3%

Table 10: Perturbation coverage on the ITDI test split: percentage of sentences whose surface form changes under each deterministic operation (overall and for Ladin (LLD)). This guards against no-op perturbations (e.g., if U+2019 is absent, apostrophe normalization changes 0% of sentences).

Component	Setting
Linear SVM (char TF-IDF)	analyzer=char; ngram_range=(1,4); lowercase=False; min_df=5; max_features=200000
Linear SVM (token TF-IDF)	analyzer=word + HF tokenizer; ngram_range=(1,2); lowercase=False; min_df=5; max_features=200000
Frozen encoder baseline	`distilbert-base-multilingual-cased`; pooling=mean; max_length=96; batch_size=32; train_per_label=1000
Classifier	LinearSVC (default C=1.0); max_iter=1000
Seed	13
Evaluation	macro_f1_present, macro_f1_all, and per-label F1

Table 11: Baseline hyperparameters and evaluation conventions (`scripts/train_eval_clf.py`).

with saved per-sentence predictions, ensuring pairing between original and perturbed predictions on each bootstrap resample.

Qualitative tokenization examples. Appendix Table 15 provides tokenizations for frequent Ladin words with diacritics/apostrophes under each tokenizer. These examples help sanity-check marker conventions and illustrate how byte/byte-BPE tokenizers represent non-ASCII characters.

Perturbation	Char SVM	mBERT	XLM-R	GPT-2 BPE	ByT5	Frozen Enc.
strip_diacritics	0.049	0.043	0.043	0.036	0.093	0.111
apostrophe_normalize	0.000	0.000	0.000	0.000	0.000	0.000
dash_normalize	0.000	0.000	0.000	0.000	0.000	0.000
lowercase	0.002	-0.002	0.001	-0.003	-0.001	-0.005
punctuation_spacing	0.013	0.000	0.012	0.009	0.036	0.000
strip_diacritics+apostrophe	0.049	0.043	0.043	0.036	0.093	0.111

Table 12: Macro-F1 drops for all perturbations evaluated on ITDI test (macro_f1_present). Drops are differences from the unperturbed test set: F1(test) − F1(perturbed) (negative values indicate higher F1 on perturbed text). Values near 0 can reflect invariance or a no-op perturbation (see Table 10).

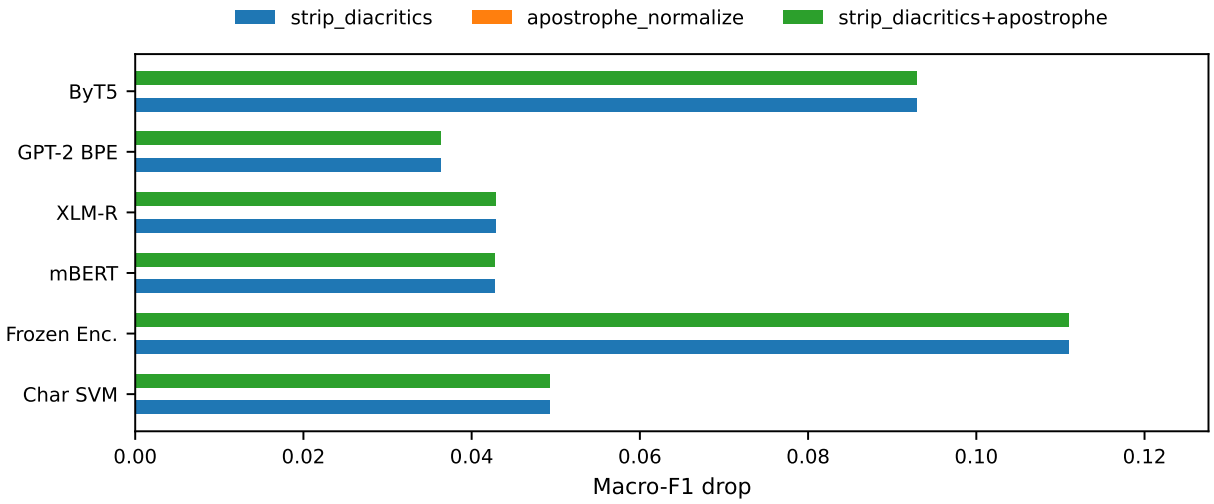

Figure 4: Macro-F1 drop under selected perturbations (macro_f1_present), including character and frozen-encoder baselines.

Model	Perturbation	Drop	Paired 95% CI
char_tfidf	apostrophe_normalize	0.000	[0.000, 0.000] (n=11087)
token_tfidf.byt5_bytes	apostrophe_normalize	0.000	[0.000, 0.000] (n=11087)
token_tfidf.gpt2_bytebpe	apostrophe_normalize	0.000	[0.000, 0.000] (n=11087)
token_tfidf.mbert_wordpiece	apostrophe_normalize	0.000	[0.000, 0.000] (n=11087)
token_tfidf.xlmr_sentencepiece	apostrophe_normalize	0.000	[0.000, 0.000] (n=11087)
char_tfidf	strip_diacritics	0.049	[0.042, 0.057] (n=11087)
token_tfidf.byt5_bytes	strip_diacritics	0.093	[0.084, 0.101] (n=11087)
token_tfidf.gpt2_bytebpe	strip_diacritics	0.036	[0.029, 0.044] (n=11087)
token_tfidf.mbert_wordpiece	strip_diacritics	0.043	[0.035, 0.050] (n=11087)
token_tfidf.xlmr_sentencepiece	strip_diacritics	0.043	[0.035, 0.050] (n=11087)
char_tfidf	strip_diacritics+apostrophe	0.049	[0.042, 0.056] (n=11087)
token_tfidf.byt5_bytes	strip_diacritics+apostrophe	0.093	[0.085, 0.101] (n=11087)
token_tfidf.gpt2_bytebpe	strip_diacritics+apostrophe	0.036	[0.028, 0.044] (n=11087)
token_tfidf.mbert_wordpiece	strip_diacritics+apostrophe	0.043	[0.035, 0.050] (n=11087)
token_tfidf.xlmr_sentencepiece	strip_diacritics+apostrophe	0.043	[0.036, 0.050] (n=11087)

Table 14: Paired stratified bootstrap confidence intervals (B=1000) for macro-F1 drops (macro_f1_present) between unperturbed ITDI test and perturbations. Resampling is performed over test sentences within gold labels (paired between original and perturbed predictions).

Word	**mbert_wordpiece**	**xlmr_sentencepiece**	**gpt2_bytebpe**	**byt5_bytes**
à	U+00E0	U+2581U+00E0	U+0120U+00C3U+0142	U+00C3 U+00A0
é	U+00E9	U+2581U+00E9	U+0120U+00C3U+00A9	U+00C3 U+00A9
sò	sU+00F2	U+2581s U+00F2	U+0120s U+00C3 U+00B2	s U+00C3 U+00B2
dì	dU+00EC	U+2581dU+00EC	U+0120d U+00C3 U+00AC	d U+00C3 U+00AC
stà	st ##U+00E0	U+2581s tU+00E0	U+0120st U+00C3U+0142	s t U+00C3 U+00A0
mè	m ##U+00E8	U+2581mU+00E8	U+0120m U+00C3U+00A8	m U+00C3 U+00A8
là	lU+00E0	U+2581lU+00E0	U+0120l U+00C3U+0142	l U+00C3 U+00A0
ió	i ##U+00F3	U+2581i U+00F3	U+0120i U+00C3U+00B3	i U+00C3 U+00B3
dà	dU+00E0	U+2581dU+00E0	U+0120d U+00C3U+0142	d U+00C3 U+00A0
sì	sU+00EC	U+2581sU+00EC	U+0120s U+00C3 U+00AC	s U+00C3 U+00AC

Table 15: Qualitative tokenization examples for frequent Ladin words with apostrophes/diacritics (test split). For readability, long tokenizations are truncated.