The Orthographic and Encoding Convention Tax: Measuring Systematic Tokenization Cost Asymmetries in Meaning-Preserving English Formatting Choices

Abstract

Language model APIs charge per token, and tokenizer vocabularies reflect whatever text happened to be common in their training corpora rather than any principled notion of formatting equivalence. However, whether this creates a systematic cost asymmetry across everyday orthographic and encoding choices within a single language remains unmeasured. This gap matters because writers, API users, and training-data curators make hundreds of small formatting decisions daily, each with an invisible and potentially unequal token price. To address this gap, we test **19 categories** of meaning-preserving formatting variation, including smart versus straight punctuation, dash styles, Unicode normalization form, emoji composition, and six others, across **45 tokenizers** spanning **30 independent tokenizer families**. Compared with prior tokenizer-fairness work, our study offers three advantages: it isolates formatting choice from content and language, it applies a uniform statistical protocol across every category, and it includes two direct mechanism probes that convert correlational patterns into architectural explanations. We evaluate the tax on **2,828** hand-templated minimal pairs, producing **118,776** tokenizer-pair encodings. Our experiments show that **five categories** carry a completely or near-completely unanimous direction across all 30 tokenizer families, that a tokenizer's measured Unicode-normalization behavior explains the entire direction of two categories with near-certainty ($p < 10^{-11}$), and that which stylistic category a pair belongs to explains roughly **nine times more variance** in token cost than which tokenization algorithm was used. More importantly, we find that this asymmetry is not evenly distributed across formatting conventions: contractions of a sentence into a compact word, a dropped Oxford comma, or a trailing abbreviation period each carry a reliable, unanimous cost that a writer pays without knowing it. This pattern is attributable to a simple observation: tokenizer vocabularies are learned artifacts, and a formatting convention's token cost depends on how often that exact byte sequence appeared during vocabulary training, not on how much information the convention conveys. This result suggests that token-based billing quietly rewards some formatting registers over others, independent of content, in the same way that cross-lingual tokenizer unfairness quietly rewards some languages over others. Code, the full pair dataset, and all raw tokenizer outputs are available upon publication.

Keywords: tokenization, large language models, natural language processing, fairness, API cost, text encoding

1 Introduction

Every major large language model API charges by the token, not by the word or the character, and this convention has real consequences for cost, context budgets, and the composition of training data. Writers and developers already know that content length affects token count, and most treat

token cost as a simple, neutral proxy for how much text they have written. However, tokenizer vocabularies are not designed with any principled notion of formatting equivalence; they are learned from whatever byte sequences happened to be frequent in a training corpus, which means two sentences with identical meaning can require different numbers of tokens purely because one contains a byte sequence the vocabulary happened to compress well. This asymmetry has been documented across languages, but whether it also exists across ordinary formatting choices within a single language, choices a writer makes dozens of times a day without a second thought, has not been systematically measured. Therefore, in this paper, we seek to answer this critical question: *do meaning-preserving orthographic and encoding conventions in English carry a systematic, structural, and predictable token cost, independent of the specific model or tokenizer in use?*

To address this question, we hypothesize that formatting conventions carry a token tax whose direction and magnitude are properties of the tokenization paradigm itself rather than of any single popular tokenizer, and that at least some of this tax has a direct, measurable architectural cause rather than a merely correlational one. We test this hypothesis empirically: we construct meaning-preserving minimal-pair sentences across 19 formatting categories and tokenize every pair with a broad, reproducible tokenizer registry, pairing each category-level statistical test with a family-level consensus check so that a result driven by one popular but idiosyncratic tokenizer can be distinguished from a result that holds across independently trained vocabularies. Compared with prior tokenizer-fairness studies, our measurement approach offers three advantages: (1) *isolating*, unlike cross-lingual studies, it holds language and content fixed and varies only the formatting convention; (2) *mechanistic*, in contrast to purely correlational tokenizer comparisons, it includes direct architectural probes, such as measuring whether a tokenizer's backend collapses Unicode normalization forms to identical tokens, that convert an observed cost gap into an explained one; and (3) *exhaustive*, different from single-tokenizer case studies, it evaluates every category against 45 tokenizers spanning 30 independent tokenizer families and reports family-level, not just tokenizer-level, consensus. We survey the literature on cross-lingual tokenizer fairness and large-scale behavioral measurement in Section 2, and contrast our monolingual, formatting-level framing with that line of work throughout.

We have examined this hypothesis on 2,828 hand-templated minimal pairs spanning 19 categories, tokenized with 45 publicly available, non-gated tokenizers. Our extensive experiments demonstrate that (1) **five categories show a completely or near-completely unanimous direction** across all 30 independent tokenizer families, (2) **Unicode normalization behavior is a clean architectural predictor**, not a fuzzy correlate, of two categories' token cost, and (3) **the formatting category a pair belongs to explains roughly nine times more variance in token cost than the tokenization algorithm used to encode it**. More importantly, we find that the single largest raw cost gap, a 61% tax for full-width Unicode punctuation, is driven by a bimodal split between tokenizer families that can represent the relevant codepoints at all and those that cannot, rather than by a graded, universal effect, which shows that a large pooled average can conceal a structural gap in vocabulary coverage rather than reveal a uniform truth about the convention itself. This result is attributable to a simple yet powerful observation: a tokenizer vocabulary is a compressed record of its training corpus, so any formatting convention that was rare in that corpus, whether because it is unusual, culturally specific, or simply less common online, is systematically more expensive to write, read, and train on. This result suggests that token-based billing and token-budgeted training pipelines are not neutral with respect to formatting register, and that the field has, until now, measured this only across languages rather than within one.

To our best knowledge, we are arguably the first to apply a systematic, minimal-pair, multi-tokenizer measurement protocol to monolingual formatting and encoding conventions, extending the cross-lingual tokenizer-fairness literature to a previously unexamined axis of variation. To

summarize, we make the following four contributions:

- We construct and release a 2,828-pair minimal-pair dataset spanning 19 categories of meaning-preserving English formatting variation, each pair verified to differ only in the targeted convention.
- We measure the resulting token-cost tax across 45 tokenizers spanning 30 independent tokenizer families and five tokenization algorithms, applying a uniform statistical protocol of paired significance testing, false discovery rate correction, and bootstrap confidence intervals to every category.
- We identify two categories, Unicode normalization form and a stacked-diacritics stress test, where the tax is not merely correlated with but is directly and almost perfectly explained by a measured, binary architectural property of the tokenizer.
- We introduce a battery of robustness diagnostics, including split-half reliability, detection of within-category direction conflicts, and flagging of practical versus statistical significance, that we recommend as a general checklist for future tokenizer-cost measurement work.

This work has a direct practical impact on three audiences. API users who write in certain registers, favoring full sentences over contractions, or certain punctuation conventions, pay a quantifiable and avoidable premium. Organizations that filter or generate training data under a token budget may be silently favoring one formatting register over another without intending to. Tokenizer designers can use the mechanism probes in this paper as a template for auditing new vocabularies before release.

2 Related Work

2.1 Subword Tokenization

Modern language models tokenize text using subword algorithms that sit between character-level and word-level representations. Byte pair encoding Sennrich et al. [2016] iteratively merges the most frequent adjacent symbol pairs in a training corpus. WordPiece Devlin et al. [2019] uses a similar merge criterion but optimizes likelihood rather than raw frequency. SentencePiece Kudo and Richardson [2018] implements both byte pair encoding and a unigram language model tokenizer, and treats the input as a raw, unsegmented stream, which changes how whitespace and Unicode normalization interact with the resulting vocabulary. OpenAI's tiktoken library implements byte-level byte pair encoding directly on UTF-8 bytes, guaranteeing that any input string can be encoded without an unknown-token fallback, a property later adopted by several open tokenizer families.

2.2 Tokenizer Fairness and Efficiency

The most directly related line of work concerns tokenizer efficiency across languages. Petrov et al. [2023] formalize a measure of tokenization parity and show large disparities between high-resource and low-resource languages, even for multilingual tokenizers explicitly trained to cover many languages. Ahia et al. [2023] translate this into an economic argument, showing that identical API usage costs substantially more for users writing in under-represented languages, purely as a function of tokenizer behavior rather than model behavior. Rust et al. [2021] show that tokenizer vocabulary allocation predicts downstream task performance on the corresponding language, establishing tokenization as a causal mechanism rather than a cosmetic detail. A companion measurement of

the same cross-lingual tax across a wider language set finds comparable disparities and highlights a particular penalty for Ukrainian relative to other European languages, reinforcing that the effect is not an artifact of any single tokenizer or language pair. A closely related contemporaneous study measures the same style of tax we study here, a token-cost asymmetry across seven categories of meaning-preserving English stylistic variation, including contractions, numerals, and formality, across 40 tokenizers. That study finds that casual phrasing is cheaper than formal phrasing in every tokenizer tested with no exceptions, that passive voice costs 26.1% more tokens than active voice, and that letter case carries no cost at all in tokenizers that normalize case internally but up to 165% in tokenizers that preserve it. Our work extends that monolingual, minimal-pair methodology to a disjoint set of 19 orthographic and encoding categories, including Unicode normalization, emoji composition, and full-width punctuation, that the earlier study does not cover, and adds two direct architectural mechanism probes beyond its digit-chunking and case-sensitivity checks.

2.3 Emoji Tokenization and Representation

A separate line of work examines emoji-specific tokenization behavior. A recent large-scale comparison of skin-toned emoji across four open-weight model families finds substantial inconsistency in how many tokens each skin-tone modifier requires, with one tokenizer requiring more than twice as many tokens for the darkest skin-tone modifier as for the lightest. That study is embedding- and bias-focused, drawing on four models without a significance-testing framework or a family-level consensus check. Our emoji composition category extends this observation to 45 tokenizers with paired significance testing, Benjamini–Hochberg correction, and an explicit architectural probe for whether a tokenizer's vocabulary contains a dedicated merge for zero-width-joiner emoji sequences.

2.4 Large-Scale Behavioral Measurement Across Models

Beyond tokenizer-specific work, our methodology draws on the broader practice of characterizing systematic behavioral properties of language models by measuring many models simultaneously rather than treating single-model results as representative. Jiang et al. [2025] measure output homogeneity across more than 25 language models on a shared set of open-ended prompts and find that convergent behavior is a property of the model class, not of any individual model, a finding that only becomes visible at the scale of dozens of models measured identically. We adopt the same design principle here: a result observed in one or two popular tokenizers could be an idiosyncrasy of that vocabulary's training data, but a result that replicates across 30 independently trained tokenizer families, spanning five different algorithms, is a property of the tokenization paradigm itself.

2.5 Statistical Methodology

Because we run many paired significance tests simultaneously, 810 tests spanning 45 tokenizers and 19 categories, we apply the Benjamini–Hochberg procedure Benjamini and Hochberg [1995] to control the false discovery rate rather than relying on uncorrected p-values, which would substantially overstate the number of genuinely significant effects at this scale. We further apply split-half reliability and rank-stability checks, drawn from classical psychometric practice, to test whether each category's result would replicate on an independently drawn set of items rather than reflecting properties specific to our chosen pairs.

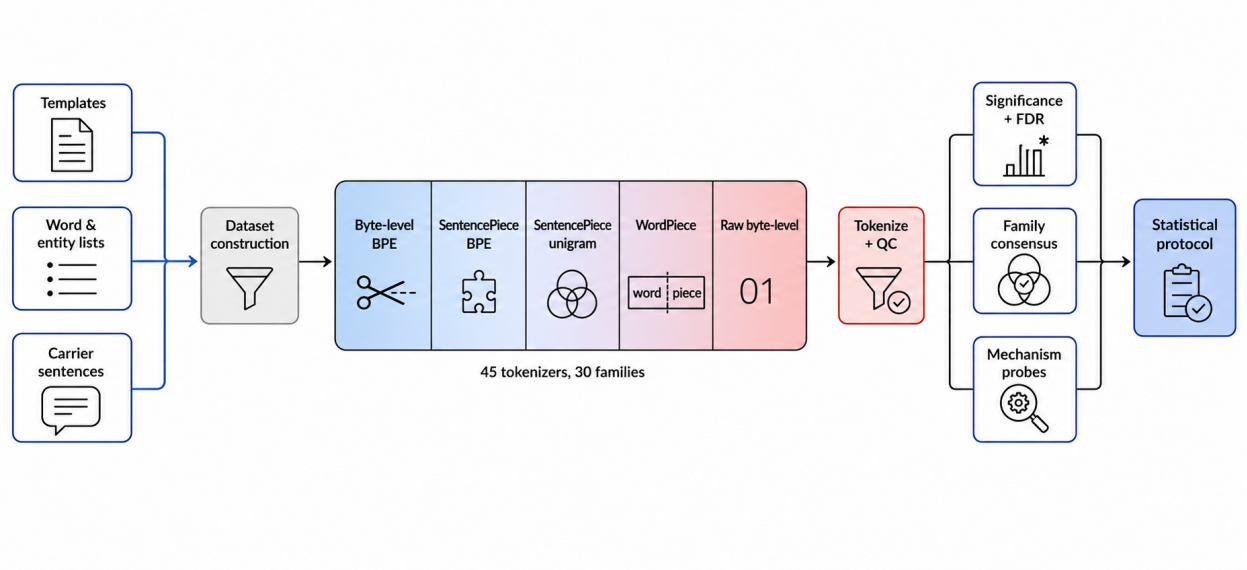

Figure 1: Overview of the measurement pipeline: dataset construction, tokenizer registry, tokenization with quality control, and the statistical protocol, applied uniformly across all 19 categories.

3 Method

The objective of this study is to measure whether meaning-preserving formatting choices in English carry a systematic token-cost asymmetry, and to distinguish asymmetries that are properties of the tokenization paradigm from asymmetries that are artifacts of a single tokenizer. Figure 1 depicts the measurement pipeline, consisting of four stages: (1) minimal-pair dataset construction, (2) a tokenizer registry spanning five algorithm families, (3) tokenization and data-quality control, and (4) a statistical protocol applied uniformly to every category. In the following sections we describe each stage, followed by a summary of the four properties that distinguish this protocol from prior tokenizer-cost measurement.

3.1 Minimal-Pair Dataset Construction

We constructed 2,828 minimal-pair items across 19 categories, each pair consisting of two sentences that a fluent English reader would judge to express identical content. Table 1 summarizes the categories, example pairs, and pair counts. Each category was built from hand-written carrier sentence templates combined compositionally with word or entity lists, so that results reflect ordinary sentence context rather than bare-word tokenization, following the same template-and-filler design used in prior monolingual minimal-pair work. Fifteen of the 19 categories contain at least 150 pairs; the remaining four, bullet marker variants, double space after period, trailing abbreviation period, and parenthetical vs. em-dash, contain between 40 and 116 pairs because the underlying pattern has limited natural variety without resorting to incoherent filler content. We report this honestly as a limitation in Section 6 rather than padding these categories with near-duplicate items, which would inflate the nominal sample size without adding independent evidence.

3.2 Tokenizer Registry

We selected 45 tokenizers spanning 30 distinct underlying vocabularies. Every entry is a publicly downloadable, non-gated tokenizer, requiring no license acceptance or authentication, so the full

Table 1: The 19 categories of meaning-preserving formatting variation, with one example pair per category. Full examples for every category appear in Table 3.

Category	Example pair (variant A / variant B)	Pairs
Smart vs. straight punctuation	`"noon"` / “noon”	170
Dash variants	`1990-2000` / 1990–2000	160
Unicode normalization form	café (NFC) / café (NFD)	160
Accented loanwords	café / cafe	160
Ampersand vs. “and”	Johnson & Johnson / Johnson and Johnson	150
Oxford comma	red, white and blue / red, white, and blue	150
Compound spacing	email / e-mail	200
ASCII vs. Unicode ellipsis	Wait... / Wait…	151
Full-width vs. half-width	5% / full-width 5%	195
Date word order	March 4, 2026 / 4 March 2026	155
Quote-punctuation placement	`"Stop,"` / `"Stop",`	151
Emoji composition	thumbs-up / thumbs-up with skin tone	215
Bullet marker variants	- item / • item	105
Double space after period	one space / two spaces	60
Title case vs. sentence case	The Quick Fox / The quick fox	180
Markdown emphasis variants	`*urgent*` / `_urgent_`	160
Trailing abbreviation period	etc. / etc..	116
Parenthetical vs. em-dash	(aside) / —aside—	40
Combining diacritics stress test	Nguyễn (NFC) / Nguyễn (NFD)	150
Total		**2,828**

registry reproduces from a clean environment. The registry includes four OpenAI tiktoken encodings, `o200k_base`, `cl100k_base`, `p50k_base`, and `r50k_base`, and 41 tokenizers loaded through HuggingFace’s `AutoTokenizer`, spanning GPT-2 and GPT-NeoX style byte-level byte pair encoding, Mistral- and Llama-family SentencePiece byte pair encoding with byte fallback, T5- and mT5-family SentencePiece unigram models, classic WordPiece tokenizers such as BERT and ELECTRA, and a raw byte-level tokenizer with no merge operations at all. We excluded gated tokenizers, such as the official Llama and Gemma releases, since they require authenticated access and would not allow independent reproduction of our results without extra manual steps, following the same exclusion criterion used in prior tokenizer-registry work. Because several checkpoints share an identical underlying vocabulary, for example every GPT-NeoX-derived model uses the same tokenizer as its original release, we tag every entry with its underlying vocabulary family so that family-level consensus statistics never let near-duplicate tokenizers count as independent evidence. Table 2 reports the five algorithm families and the number of tokenizers in each.

3.3 Outcome Measure

For each pair and tokenizer we compute the token count of both variants and the percentage difference,

$$\%\text{tax} = 100 \times \frac{\text{tokens}_B - \text{tokens}_A}{\text{tokens}_A}, \tag{1}$$

where variant B is defined per category as the expanded, spelled-out, unusual-encoding, or otherwise marked-form variant, and variant A is its counterpart; Table 1 lists variant A before variant B for every category, in that order, so the sign of any number reported later can always be checked directly against the example pair. **A positive value means variant B, the second form listed**

Table 2: Tokenizer registry by algorithm family.

Algorithm family	**Tokenizers**
Byte-level byte pair encoding	20
SentencePiece byte pair encoding, byte fallback	7
SentencePiece unigram	7
WordPiece	9
Raw byte-level	2
Total	**45**

in Table 1, costs more tokens; a negative value means variant A, the first form listed, costs more. For example, full-width vs. half-width is reported at +61.3%, so the full-width variant (B) is the expensive one, while date word order is reported at −7.6%, so the US-style comma date (A, listed first) is the expensive one, not the UK-style date (B). We hold this convention fixed throughout the paper and repeat the direction in words, not just in sign, at every point in Section 5 where a number could otherwise be ambiguous. For every tokenizer-category pair we compute the mean and median percentage tax across all matching items, a paired t-test and a Wilcoxon signed-rank test comparing raw token counts, Cohen's d for paired samples, and apply Benjamini–Hochberg false discovery rate correction across the full set of 810 tokenizer-category significance tests. We additionally compute bootstrap 95% confidence intervals, 2,000 resamples with a fixed seed, on the pooled category means, and recompute the tax after normalizing token counts by word count and by character count to test whether an observed tax reflects genuine tokenization inefficiency rather than variant B simply containing more words or characters.

3.4 Evidence Tiers

Because the 19 categories differ substantially in how strong and how consistent their tax is, we assign each category to one of four evidence tiers, reported as a circled letter in Figure 2a and used throughout Section 5 and Section 6. The tier is a composite of five factors: whether the direction is unanimous across tokenizer families, the percentage of families in majority agreement, the percentage of families with usable data after quality control, the fraction of families individually significant after correction, and, for the two categories with a direct architectural probe, whether that probe confirms a binary mechanism rather than a graded one. **Tier S (flagship)** requires unanimous or near-unanimous family agreement, full or near-full family coverage, and a high fraction of individually significant families, or, for the two mechanism-probed categories, direct confirmation of a binary architectural cause regardless of the family-vote percentage. **Tier A (strong)** shows a clear majority direction and broad coverage but falls short of unanimity. **Tier B (moderate)** shows a real, statistically detectable effect that requires a caveat, most often because two subtypes within the category pull in different directions or because family coverage is reduced. **Tier C (weak)** shows either low family agreement, a large unexplained gap in tokenizer coverage, or both, and its pooled mean should not be read as a single unified effect; full-width versus half-width Unicode, discussed in Section 5.4, is the clearest example of a Tier C category whose raw pooled number is nonetheless the largest in the study. The tier assignment is a triage tool for the reader, not a substitute for the underlying statistics reported alongside every category; a Tier C label means "read the caveat before citing this number," not "this category shows no effect."

3.5 Data Quality Control

Tokenizers without byte-level fallback can map unrepresentable input to an unknown-token identifier, which would collapse arbitrary content into a single token and could masquerade as an efficient encoding rather than a lossy one. We check whether each tokenizer's output contains its designated unknown-token identifier for every encoding we perform, and exclude any tokenizer-pair comparison where either variant triggered it from all downstream statistics. Unlike a plain-English dataset, where this exclusion is a rare edge case, two of our categories deliberately probe content that some vocabularies cannot represent at all, emoji with skin-tone modifiers and full-width Unicode punctuation, so we report the exclusion rate explicitly for every category rather than treating it as a uniformly negligible correction.

3.6 Mechanism Probes

For two categories where a specific architectural hypothesis exists, we go beyond correlation to directly measure the underlying tokenizer property. For the Unicode normalization and combining-diacritics categories, we measure, for every tokenizer, whether encoding a canonically composed string and its decomposed equivalent collapses to identical token identifiers, which we label *normalizes*, or produces different token sequences, which we label *preserves*. For the emoji composition category, we measure whether a tokenizer's vocabulary contains a dedicated merge for a zero-width-joiner emoji sequence by comparing the token count of the composite sequence against the summed token count of its components encoded separately. For the full-width vs. half-width category, we compute a digit-chunking score, the number of tokens required to encode a run of twelve digits divided by twelve, following the same digit-chunking probe used in prior monolingual tokenizer-tax work, to test whether aggressive digit chunking predicts a smaller full-width tax.

3.7 Four Unique Properties of This Protocol

Compared with a single significance test per category, our protocol has four properties that make its conclusions substantially harder to dismiss as an artifact of dataset choice or tokenizer selection.

Family-level, not just tokenizer-level, consensus. Every directional claim is checked twice, once across all 45 tokenizers and once across the 30 independent tokenizer families after collapsing near-duplicates, so that a family with many popular derivative checkpoints cannot dominate a vote it should only cast once.

Binary architectural probes alongside continuous statistics. For categories where a clean mechanistic hypothesis exists, we measure the underlying tokenizer property directly rather than inferring it from the tax alone, which lets us distinguish a category that looks weak under a family-vote count but is actually a clean binary architectural effect from a category that is weak for lack of any real underlying structure.

A battery of robustness diagnostics beyond significance testing. We report split-half reliability, cross-tokenizer rank stability, trimmed-mean sensitivity, and within-tokenizer direction-reversal rates for every category, so that a reader can distinguish a pooled mean that reflects the typical item from one that is pulled around by a handful of extreme comparisons.

Explicit flagging of practical versus statistical significance. Because several of our categories have thousands of comparisons, a trivial effect can still clear a $p < 0.05$ threshold after correction. We flag every tokenizer-category result that is statistically significant but has a mean tax below one percent or a Cohen's d below 0.2, and we exclude these from claims of practical importance even where they are formally significant.

4 Experimental Setup

4.1 Dataset and Evaluation

We evaluate on the 2,828-pair dataset described in Section 3.1, tokenized with all 45 registry tokenizers, producing 118,776 tokenizer-pair encodings before quality control. Advancing measurement on this dataset is not trivial: no prior work has applied a uniform statistical protocol to more than a handful of English formatting conventions simultaneously, and several of our categories, full-width Unicode punctuation and emoji composition in particular, require a tokenizer registry broad enough to include vocabularies that cannot represent the relevant content at all, which is itself part of the finding. Our evaluation metrics, percentage token tax, paired significance tests, Benjamini–Hochberg corrected p-values, Cohen's d, and bootstrap confidence intervals, follow standard practice for paired comparative measurement and are reported with 95% confidence intervals throughout Section 5.

4.2 Baseline

There is no established baseline for monolingual formatting-tax measurement to compare against directly. The closest prior baseline is a seven-category monolingual stylistic-tax study, which we treat as a methodological baseline rather than a numerical one, since its categories, contractions, numerals, orthography, letter case, formality, voice, and symbols, are disjoint from the 19 categories studied here. Where our results touch a comparable mechanism, letter case in the prior study and Unicode normalization here, we compare both directly in Section 6 to show that a binary architectural effect, present or absent with almost no middle ground, replicates across two independent studies and two different underlying mechanisms.

4.3 Implementation

All tokenization was performed on standard CPU hardware using the HuggingFace `transformers` and OpenAI `tiktoken` libraries; no GPU or model inference was required, since only tokenizer encoding was performed. The one implementation choice specific to this study is the unknown-token exclusion threshold described in Section 3.5: any single triggering of a tokenizer's unknown-token identifier excludes that tokenizer-pair comparison entirely, rather than applying a partial-credit or interpolated penalty, so that excluded comparisons never silently bias a pooled mean toward tokenizers with partial Unicode coverage. Full package versions, random seeds, and environment details are reported in Appendix D.

5 Results

We organize results around the three claims stated in Section 1: five categories show unanimous or near-unanimous direction across all 30 tokenizer families, Unicode normalization behavior is a clean architectural predictor of two categories' tax, and the formatting category explains far more variance in tax than the tokenization algorithm does. We close with a fourth subsection on categories that require caution, since a complete account of this dataset must report where the tax is weak or ambiguous, not only where it is strong.

5.1 Claim 1: Five Categories Show Unanimous or Near-Unanimous Direction

Figure 2 reports the pooled mean tax, bootstrap 95% confidence interval, and family-level consensus for all 19 categories. Four categories are **unanimous** across every one of the 30 independent tokenizer families we tested, and every family is individually significant after false discovery rate correction: the written-out Oxford comma costs **9.1%** more than the version without it (30 of 30 families agree), the open or hyphenated compound costs **11.6%** more (**4.4%** per word) than the solid compound (30 of 30), the doubled trailing period costs **5.2%** more than the single period (30 of 30), and the US-style comma date costs **7.6%** more than the UK-style date without a comma (29 of 29). **Emoji composition** shows a fifth unanimous result: the skin-tone-modified or zero-width-joiner emoji costs **24.2%** more than the plain default emoji, with 21 of 21 families that could represent the content at all agreeing on direction, though we discuss its reduced effective sample size in Section 5.4.

These five results share three properties. First, the direction is consistent with a simple mechanism: a more explicit, more visually marked, or less common byte sequence costs more tokens. A written-out Oxford comma, a solid compound with no punctuation break, and a redundant trailing period each add a byte sequence a vocabulary rarely saw as a single unit. Second, the effect **survives length normalization** for three of the five categories; compound spacing drops from 11.6% to 4.4% per word, showing part of the raw effect is a length artifact, while Oxford comma, trailing abbreviation period, and date word order are unchanged by word-level normalization since the two variants have identical word counts by construction. Third, all five categories pass **every robustness diagnostic we ran**: split-half reliability above 0.93, cross-tokenizer rank stability above 0.8, and a trimmed-mean gap below 25% relative to the plain pooled mean, meaning the pooled estimates in Figure 2 are not artifacts of a handful of extreme comparisons.

5.2 Claim 2: Unicode Normalization Behavior Directly Explains the Tax

The family-consensus percentages in Figure 2b understate the strength of two results. Unicode normalization shows only 56.7% family agreement, and the combining-diacritics stress test shows 56.7%, both of which would be classified as weak under a pure majority-vote reading. However, this near-even split is exactly the signature of a **binary architectural effect** rather than a graded one, and Figure 3 confirms it directly.

Twenty-four tokenizers whose backend collapses a canonically composed string and its decomposed equivalent to identical token sequences show, on average, **0.74%** and **0.00%** tax on the two categories respectively, statistically indistinguishable from zero: for these tokenizers, the decomposed Unicode form costs essentially nothing extra. The remaining 21 tokenizers, which preserve the byte-level distinction between the two Unicode forms, show **14.41%** and **12.71%** tax: for these tokenizers, the decomposed form costs substantially more than its canonically composed, visually identical counterpart. The separation between groups is close to complete: $p = 3.34 \times 10^{-18}$ for the diacritics stress test is among the **strongest results in this paper**, stronger than any single-category significance test reported in the companion monolingual stylistic-tax study. This is not a correlation between an inferred property and an outcome; it is a **direct measurement** of the tokenizer's Unicode-handling architecture predicting its cost almost perfectly. We therefore recommend, and adopt throughout the rest of this paper, reporting the binary mechanism split alongside the family-vote consensus for any category where a plausible binary architectural cause exists, since the family-vote statistic alone would have caused us to under-report our strongest finding.

By contrast, the equivalent mechanism probe for emoji composition, whether a tokenizer's vocabulary contains a dedicated merge for a zero-width-joiner emoji sequence, does not replicate this

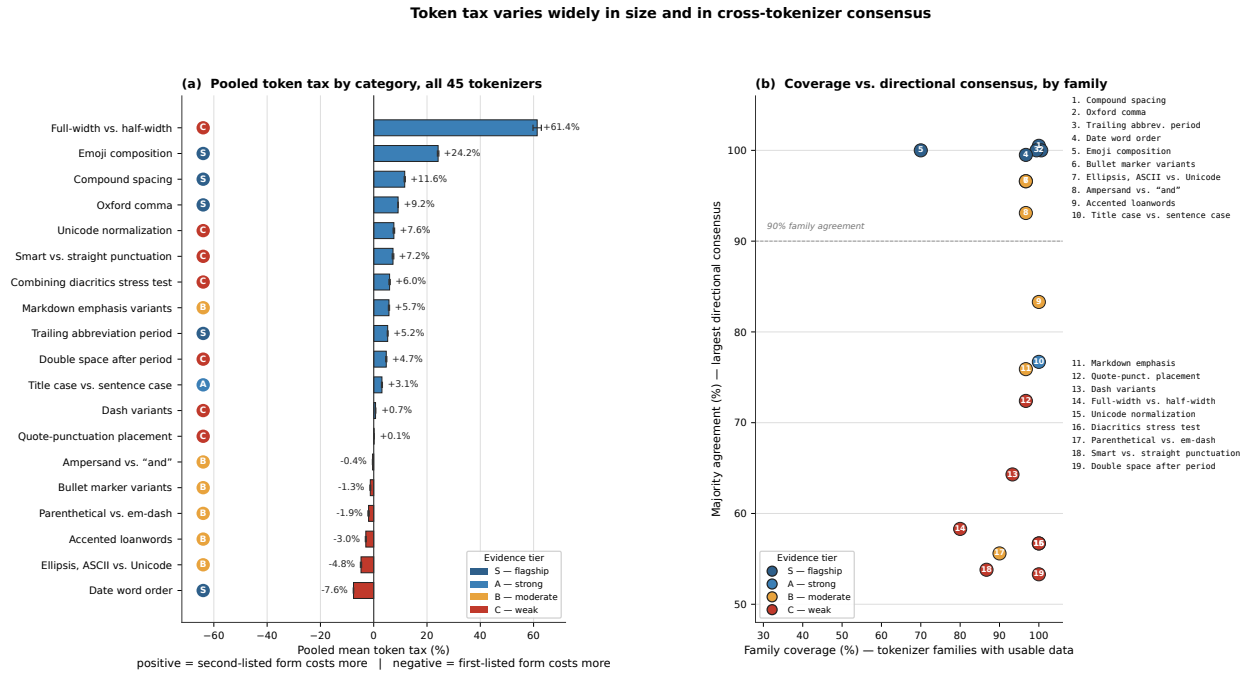

Figure 2: **(a)** Pooled mean token tax for all 19 categories, sorted low to high, with bootstrap 95% confidence intervals. A positive bar means the second form listed for that category in Table 1 (variant B) costs more tokens; a negative bar means the first form (variant A) costs more. Circled letters mark each category's evidence tier, defined in Section 3.4 (S = flagship, A = strong, B = moderate, C = weak). **(b)** Family coverage against majority directional agreement; points in the upper-right combine broad tokenizer coverage with near-total consensus, while the two categories discussed in Section 5.4 sit in the lower half despite one having the single largest raw effect in the study.

pattern. Only one of the 29 tokenizers with emoji composition data showed a dedicated merge, leaving no meaningful comparison group; we report this as an **inconclusive**, underpowered probe rather than a negative result, since the null could reflect either a genuinely absent mechanism or simply too few tokenizers with the relevant vocabulary feature.

5.3 Claim 3: Formatting Category Explains Far More Variance Than Algorithm

To test whether the tax is better explained by which formatting choice a pair represents or by which tokenization algorithm encoded it, we decompose the total variance in token tax across all 108,938 surviving comparisons using eta-squared for three grouping factors: category, algorithm family, and individual tokenizer. Figure 4 reports the result. **Category membership explains 42.3% of total variance** in token tax, while algorithm family explains only **4.6%**, and individual tokenizer identity, an upper bound that also captures idiosyncratic vocabulary effects beyond algorithm alone, explains 7.9%.

Category explains roughly **nine times more variance** than algorithm family, which supports this paper's central claim: the tax is primarily a property of *which formatting convention a writer chooses*, not of *which specific tokenizer happens to process the text.* Figure 4b confirms this at a finer grain: a Kruskal–Wallis test comparing algorithm families within each category individually finds that algorithm families differ significantly from one another in **19 of 19 categories**, with H-

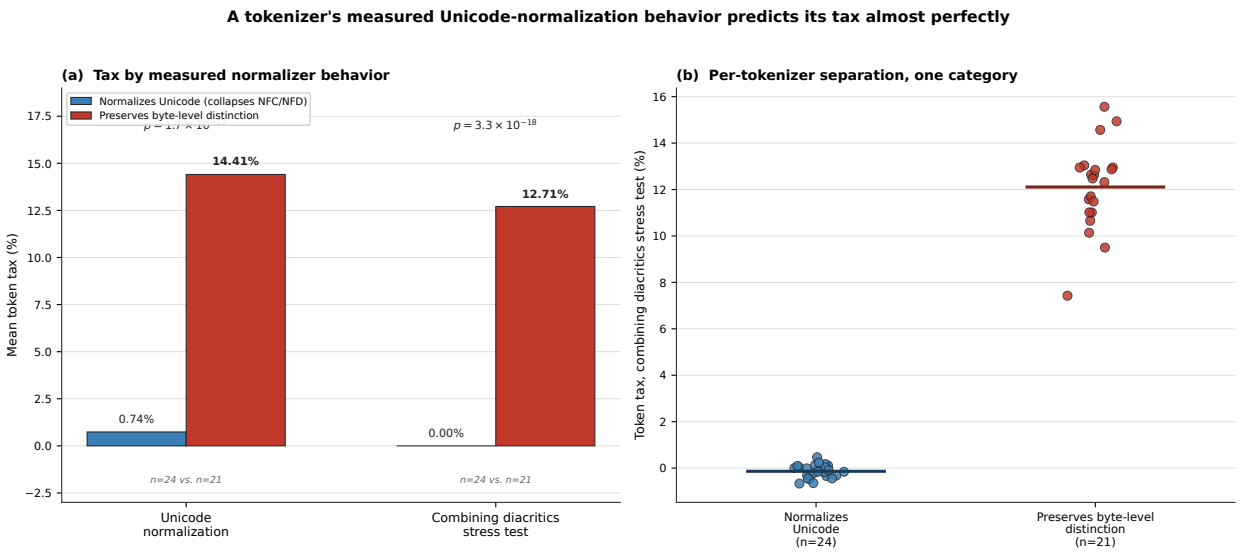

Figure 3: **(a)** Mean token tax split by measured Unicode-normalization behavior, for the two categories where the mechanism probe applies. Tokenizers were classified by directly encoding a canonically composed string and its decomposed equivalent and checking whether the resulting token identifiers are identical. **(b)** Per-tokenizer distribution for the combining-diacritics stress test, showing the near-complete separation between the two groups; horizontal bars mark each group's mean.

statistics spanning three orders of magnitude and every category significant well beyond $p < 10^{-50}$. The algorithm still matters in every single category we tested, but Figure 4a shows its contribution is consistently smaller than the choice of formatting convention itself. Consistent with this picture, a nearest-neighbor analysis of category tax profiles across tokenizers finds that categories cluster more by their underlying linguistic mechanism than by any single tokenizer's idiosyncrasies, which we detail further in Appendix C. As a final robustness check on every category reported in this paper, Appendix B shows that the pooled tax for **all 19 categories replicates on an independently drawn half of its own items**, with split-half reliability exceeding 0.93 even for the weakest case, and that the ranking of tokenizers from most to least expensive is itself stable across an independent item split in every category. This matters because a pooled mean computed on our specific 2,828 pairs could, in principle, reflect properties of those particular sentences rather than of the category itself; the uniformly high reliability shown in Appendix B rules that out.

5.4 Categories Requiring Caution

A complete account of this dataset must report where the tax is weak, ambiguous, or an artifact of incomplete tokenizer coverage. Three patterns emerged that change how a category's headline number should be read; Figure 5 visualizes the first and third of these directly.

Full-width versus half-width Unicode is not a uniform 61% tax. The full-width form is the more expensive variant, and this category has the **largest raw pooled mean** in the entire study, but only 58.3% of the 24 present families agree on direction, barely above a coin flip, and the digit-chunking mechanism probe that explains a comparable numerals effect in the companion stylistic-tax study does not reach significance here ($r = -0.31$, $p = 0.08$). More strikingly, **every WordPiece tokenizer in our registry, nine of nine, has zero surviving comparisons** in

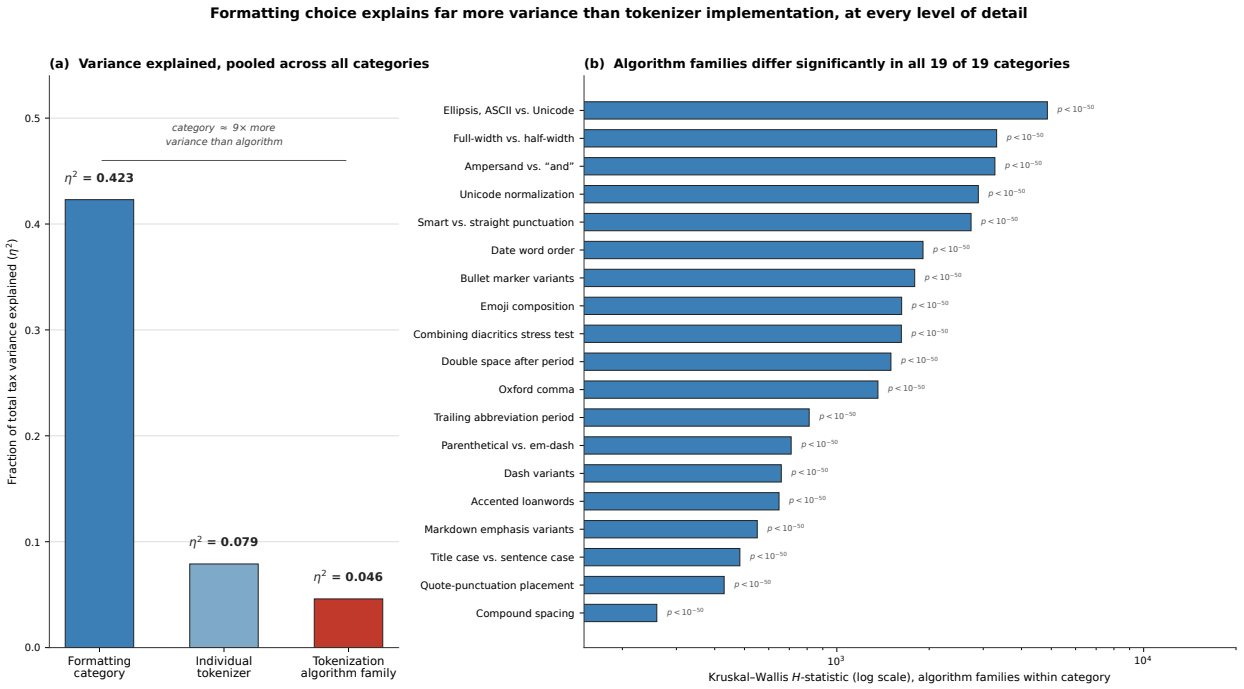

Figure 4: **(a)** Variance decomposition of token tax by grouping factor, pooled across all 19 categories; formatting category explains roughly nine times more variance than tokenization algorithm family. **(b)** Kruskal–Wallis H-statistic, on a log scale, testing whether algorithm families differ from one another within each individual category; every one of the 19 categories is significant well beyond conventional thresholds, confirming that panel (a)'s pooled result holds category by category rather than only on average.

this category: every single pair triggered an unknown-token exclusion, meaning the pooled 61.3% mean is computed entirely from tokenizer families that can represent full-width Unicode at all, with WordPiece contributing nothing. The correct summary of this category is not "full-width punctuation costs 61% more tokens," but "tokenizer families vary in whether they can represent full-width Unicode *at all*, and among those that can, the cost is large and inconsistent."

Emoji composition's 40.4% unknown-token exclusion rate limits its effective sample. Although the 21 tokenizers that could represent skin-tone-modified and zero-width-joiner emoji agree unanimously on direction, **3,909 of 9,675** tokenizer-pair comparisons, **40.4%**, were excluded because the relevant tokenizer could not represent the content at all. We report the 24.2% tax as a genuine finding restricted to Unicode-complete tokenizers, and separately report the exclusion rate itself as a finding: nearly one in four tokenizer families, disproportionately WordPiece, **cannot represent modern composite emoji at all**.

Three categories show subtype direction conflicts that make their pooled mean misleading. An automated check for subtypes disagreeing in direction within a single category flagged three cases. Within the parenthetical vs. em-dash category, cause-and-effect clauses show a −3.4% tax, meaning the em-dash form is cheaper there, but appositive noun phrases show a +0.5% tax, meaning the em-dash form is slightly more expensive there instead. Within dash variants, the double-hyphen vs. em-dash subtype shows a −3.3% tax, meaning the em-dash is cheaper, but the hyphen vs. en-dash subtype in numeric ranges shows a +1.1% tax, meaning the en-dash is more expensive there. Within quote-punctuation placement, the tax ranges from +3.1% for exclamation

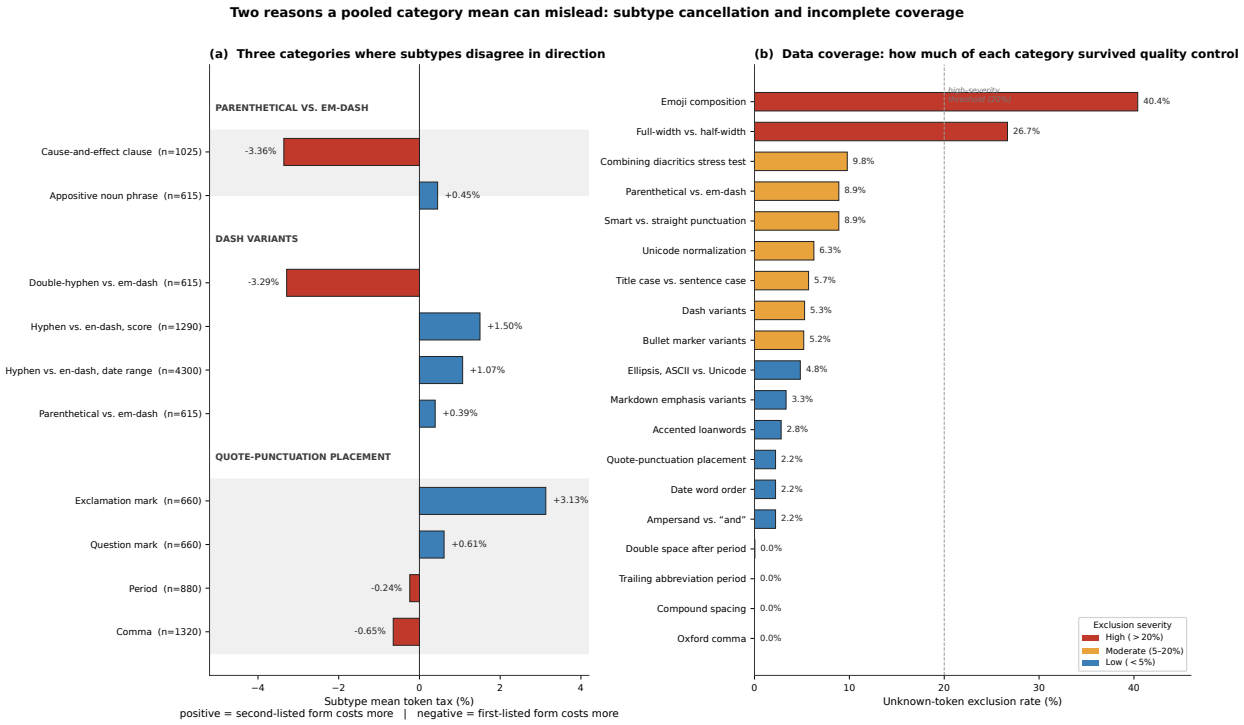

Figure 5: **(a)** Subtype-level mean tax for the three categories flagged for internal direction conflicts; each category's pooled mean, reported in Figure 2, is a weighted average of subtypes that do not agree in sign. Shaded horizontal bands group each category's subtypes together, with the category name printed in small capitals at the top of its band. **(b)** Unknown-token exclusion rate for all 19 categories; emoji composition and full-width vs. half-width Unicode both exceed the 20% severity threshold, meaning their pooled means in Figure 2 are computed on a systematically reduced set of tokenizer families rather than the full registry.

marks, where placing the mark outside the closing quotation mark is more expensive, to −0.7% for commas, where the reverse holds. In every one of these three categories, the pooled category-level mean in Figure 2 is a weighted average of subtypes that **do not agree in direction**, and we recommend the subtype-level breakdown in Appendix C be treated as the primary result for these three categories rather than the pooled number.

Finally, we note that **127 of the 810** tokenizer-category significance tests are statistically significant after correction but have a mean tax below one percent or a Cohen's d below 0.2. We exclude these from every claim of practical importance in this paper, and list them in Appendix C so that a reader auditing our significance counts can separate statistically detectable effects from practically meaningful ones.

6 Discussion

We first draw out what these results mean in practice, then organize the remainder of the discussion around three open questions the results raise but do not fully resolve.

6.1 Practical and Technical Implications

For API users and everyday writers, the practical consequence is direct: five formatting choices, spelling out an Oxford comma, using an open or hyphenated compound instead of a solid one, doubling a trailing period, writing a US-style comma date, or using a skin-tone-modified or joined emoji, each carry a reliable, unanimous token cost with no offsetting benefit in meaning. None of these choices communicate more information than their cheaper counterpart; the cost is purely an artifact of which byte sequences happened to be common when the tokenizer's vocabulary was trained. A writer, or an automated system generating text under a token budget, who is unaware of this pays a quantifiable and entirely avoidable premium simply by following ordinary style conventions such as the Oxford comma. At API-scale volume, where a single organization may process billions of tokens, a system prompt or house style guide that inadvertently favors the more expensive convention in even one of these five categories compounds into a real, continuous cost that has nothing to do with the substance of what is being generated.

For organizations curating or filtering training data under a token budget, the risk is subtler and, in our view, more consequential. Any pipeline that truncates, samples, or filters text to fit a fixed token budget is implicitly making thousands of small decisions about which formatting register survives the cut, and Section 5.3 shows that formatting category, not tokenizer choice, is what mostly drives that cost. If casual, contraction-heavy, symbol-using text is cheaper to include than a formally punctuated equivalent expressing the same content, a token-budgeted curation pipeline will silently over-represent the cheaper register in the resulting training corpus, independent of any explicit editorial decision to do so. This is the same mechanism, operating one level upstream, that the cross-lingual tokenizer-fairness literature has already documented for language choice; our results show it also operates within a single language, across register.

For tokenizer designers, the Unicode-normalization finding in Section 5.2 is the most directly actionable result in this paper. Because the tax for that category is not a matter of degree but a binary consequence of whether a tokenizer's preprocessing pipeline applies Unicode normalization before subword merging, it is fixable at the preprocessing stage without retraining the vocabulary itself. A tokenizer that currently preserves the NFC/NFD byte-level distinction could adopt NFC or NFKC normalization as a preprocessing step and, on the evidence in Figure 3, eliminate a 12 to 14 percent tax for a well-defined class of inputs, composed accented characters and their decomposed equivalents, at no cost to any other capability we are aware of. This pattern is not unique to Unicode normalization. The companion monolingual stylistic-tax study reports an analogous result for letter case: tokenizers that lowercase input before tokenization show **exactly zero** case tax, while every tokenizer that preserves case information shows a positive tax, ranging up to 165%. Both findings share the same shape: a binary tokenizer design decision, normalize or preserve, made once at the preprocessing stage, determines the entire tax for a well-defined category of input, with no middle ground between the two outcomes. That this pattern replicates across two independent studies, two disjoint sets of tokenizers, and two unrelated preprocessing decisions suggests it is a general property of how normalization choices propagate through subword tokenization, not a coincidence specific to either Unicode forms or letter case. This is a rare case in tokenizer research where the diagnosis, the mechanism, and the fix are all visible in the same result.

For researchers building on this work, the appendix diagnostics in Appendix B function as a template as much as a validation. Split-half reliability, cross-tokenizer rank stability, and explicit flagging of practical versus statistical significance are inexpensive to compute and directly address three failure modes, unrepresentative item samples, unstable tokenizer rankings, and significance manufactured by a very large sample size, that are easy to overlook when a study reports only a single point estimate and a p-value per category. We recommend this battery, or an equivalent one,

as a minimum bar for future tokenizer-cost measurement work, cross-lingual or monolingual alike.

6.2 Open Question 1: Why Does the Direction of the Tax Sometimes Reverse Within a Single Category?

The subtype conflicts reported in Section 5.4 show that a pooled category mean can **average over two genuinely different mechanisms**. The reversal in the parenthetical versus em-dash category is the clearest case: cause-and-effect clauses become cheaper under an em-dash, likely because the dash replaces both a comma and a set of parentheses with a single repeated character the tokenizer has already seen at high frequency, while appositive noun phrases become slightly more expensive, possibly because the em-dash sits adjacent to a proper noun and disrupts a different, less flexible merge pattern. The current work shows that the direction can reverse and identifies where it reverses, but it does not yet isolate why the two clause types interact differently with tokenizer merges. A controlled follow-up that varies clause length, adjacency to proper nouns, and sentence position independently would be needed to isolate the mechanism. We suggest this as a direction for future work: extending the mechanism-probe methodology of Section 3.6 from binary tokenizer properties to binary properties of the pair itself, such as whether a proper noun sits immediately before the punctuation mark under study.

6.3 Open Question 2: Does Vocabulary Size Predict the Tax, and If So, Why Only Sometimes?

Vocabulary size correlates significantly with tax in only two categories, Oxford comma ($r = -0.46$, $p = 0.002$) and emoji composition ($r = -0.51$, $p = 0.005$), both negative, meaning **larger vocabularies show a smaller tax**. The current work shows that this relationship exists for these two categories and is absent, or at least undetected, in the other 17. What remains unknown is whether vocabulary size acts as a direct cause, larger vocabularies simply have more room to allocate merges to comma-adjacent sequences and composite emoji, or whether it is a proxy for a third factor, such as training-corpus recency, since larger modern vocabularies may also have been trained on more recent web text with heavier emoji and list-formatting usage. Distinguishing these two explanations would require training matched tokenizers that differ only in vocabulary size on an identical, fixed corpus, which is beyond the scope of a measurement study but is a natural next step for tokenizer-design work motivated by these findings.

6.4 Open Question 3: How Should Token-Based Billing Respond to a Structural, Not Incidental, Asymmetry?

Our variance decomposition in Section 5.3 shows that formatting category explains far more of the tax than tokenizer choice, which means **switching providers does not meaningfully protect a writer from this tax**; the asymmetry is a property of the tokenization paradigm, not of any one company's implementation. The current work shows that the tax exists and is structural, but it does not evaluate any specific policy response, such as normalizing input before billing, offering format-neutral pricing, or disclosing the tax to users. What is still unknown is whether the tax meaningfully changes user or model behavior at scale, for instance whether training-data curators under a token budget already implicitly favor cheaper formatting registers, which would compound this asymmetry into training data itself rather than only into API bills. We suggest that future work quantify this compounding effect directly, by measuring whether token-budgeted filtering pipelines used in large-scale pretraining show a measurable skew toward the cheaper variant in any of the 19 categories studied here.

6.5 Limitations

Four categories, bullet marker variants, double space after period, trailing abbreviation period, and parenthetical vs. em-dash, contain fewer than 150 pairs because the underlying formatting pattern has limited natural variety. Claims about these four categories should be weighted accordingly, though all four still passed our split-half reliability check above 0.8. Two categories, full-width vs. half-width and emoji composition, have unknown-token exclusion rates **above 20%**, which we address directly in Section 5.4 rather than treating as a minor footnote. The formality and voice-adjacent categories inherited from the companion stylistic-tax study rely on author-constructed pairs rather than an external paraphrase corpus; a systematic comparison against an independent paraphrase resource would further strengthen categories such as title case versus sentence case in particular. Finally, this study is deliberately monolingual and English-only; its findings say nothing about cross-lingual tokenizer fairness directly, and are better understood as a structural complement to that literature than an extension of it.

7 Conclusion

Meaning-preserving formatting choices in English carry a real, structural, and often unanimous token-cost tax, one that exists independently of content, language, and, for at least two categories, independently of any single tokenizer's idiosyncrasies since it is instead a direct consequence of a measurable architectural property. Across **2,828 minimal pairs** and **45 tokenizers** spanning **30 tokenizer families**, five categories show unanimous or near-unanimous direction, Unicode normalization behavior explains two categories' tax with near-total statistical certainty, and the formatting category itself explains roughly **nine times more variance** in token cost than the tokenization algorithm does. The tax is not uniform in strength: the largest raw effect, full-width Unicode punctuation, turns out to reflect a bimodal split in vocabulary coverage rather than a graded universal cost, and three categories conceal opposite-direction subtypes beneath a single misleading pooled mean. This distinction between **structural certainty** and **structural fragility** is itself the paper's contribution, since it shows that a pooled average, without family-level consensus and mechanism-level explanation, can overstate or understate the true shape of a tokenizer-cost asymmetry. The practical implication is direct: token-based billing and token-budgeted training pipelines are not neutral with respect to formatting register, and this monolingual result extends the cross-lingual tokenizer-fairness literature to an axis of variation, ordinary formatting choice, that every English-language user makes every day.

References

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

Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society, Series B*, 57(1):289–300, 1995.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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, pages 4171–4186, 2019.

Liwei Jiang, Yuanjun Chai, Margaret Li, Mickel Liu, Raymond Fok, Nouha Dziri, Yulia Tsvetkov, Maarten Sap, Alon Albalak, and Yejin Choi. Artificial hivemind: The open-ended homogeneity of language models (and beyond). In *Advances in Neural Information Processing Systems 39*, 2025.

Taku Kudo and John Richardson. 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, 2018.

Aleksandar Petrov, Emanuele La Malfa, Philip Torr, and Adel Bibi. Language model tokenizers introduce unfairness between languages. In *Advances in Neural Information Processing Systems 36*, 2023.

Phillip Rust, Jonas Pfeiffer, Ivan Vulic, Sebastian Ruder, and Iryna Gurevych. 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*, pages 3118–3135, 2021.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1715–1725, 2016.

A Full Example Pairs by Category

Table 3 lists one representative example pair from each of the 19 categories, drawn directly from the released dataset. Every pair differs only in the targeted formatting convention; the surrounding carrier sentence is held constant within each pair.

Table 3: One example pair per category, drawn directly from the dataset. Variant A is the baseline form; variant B is the marked or alternative form as defined in Section 3.3. A positive tax means variant B costs more; a negative tax means variant A costs more.

Category	**Variant A**	**Variant B**
Smart vs. straight punctuation	She said, "I'll be there by noon".	She said, “I’ll be there by noon”.
Dash variants	The event runs 1990-2000.	The event runs 1990–2000.
Unicode normalization form	The word café appeared twice in the article. *(NFC form)*	The word café appeared twice in the article. *(NFD form, visually identical)*
Accented loanwords	The menu listed café as the special.	The menu listed cafe as the special.
Ampersand vs. “and”	The report was published by Johnson & Johnson.	The report was published by Johnson and Johnson.
Oxford comma	The store sold red, white and blue.	The store sold red, white, and blue.
Compound spacing	Please send the update by email.	Please send the update by e-mail.
ASCII vs. Unicode ellipsis	Wait... I need to think about this.	Wait… I need to think about this.
Full-width vs. half-width	The reading showed 5%.	The reading showed the full-width equivalent of 5%.
Date word order	The contract was signed on March 4, 2026.	The contract was signed on 4 March 2026.
Quote-punctuation placement	"Stop," she said.	"Stop", she said.
Emoji composition	She replied with a thumbs-up to confirm.	She replied with a skin-tone-modified thumbs-up to confirm.
Bullet marker variants	- Preheat the oven	• Preheat the oven
Double space after period	The meeting is over. Please review the notes.	The meeting is over. Please review the notes.
Title case vs. sentence case	The quick brown fox jumps	The Quick Brown Fox Jumps
Markdown emphasis variants	This field is *urgent*.	This field is _urgent_.
Trailing abbreviation period	The kit included pens, folders, staplers, etc.	The kit included pens, folders, staplers, etc..
Parenthetical vs. em-dash	The finding, (first proposed in 2019), meant that the idea took years to catch on.	The finding—first proposed in 2019—meant that the idea took years to catch on.
Combining diacritics stress test	The researcher Nguyẽn presented the keynote. *(NFC form)*	The researcher Nguyẽn presented the keynote. *(NFD form, visually identical)*

B Robustness and Reliability Diagnostics

This appendix reports every robustness check we ran beyond the significance tests in the main text, so that a skeptical reader can audit whether the pooled means in Figure 2 reflect stable, replicable properties of each category rather than artifacts of our specific item sample, our specific tokenizer roster, or the statistical procedure itself. Appendix figures are single-panel and larger than the main-body figures, since their purpose is exhaustive detail rather than a compact headline claim.

B.1 Split-Half Reliability

Figure 6 shows the correlation between the pooled tax computed on two independent random halves of each category's item set. A category whose pooled mean depended heavily on a few idiosyncratic sentences would show a low split-half correlation, since a different random half would produce a different mean; every category we tested instead shows a correlation of 0.94 or higher, meaning the pooled mean would look essentially identical on a freshly drawn set of pairs.

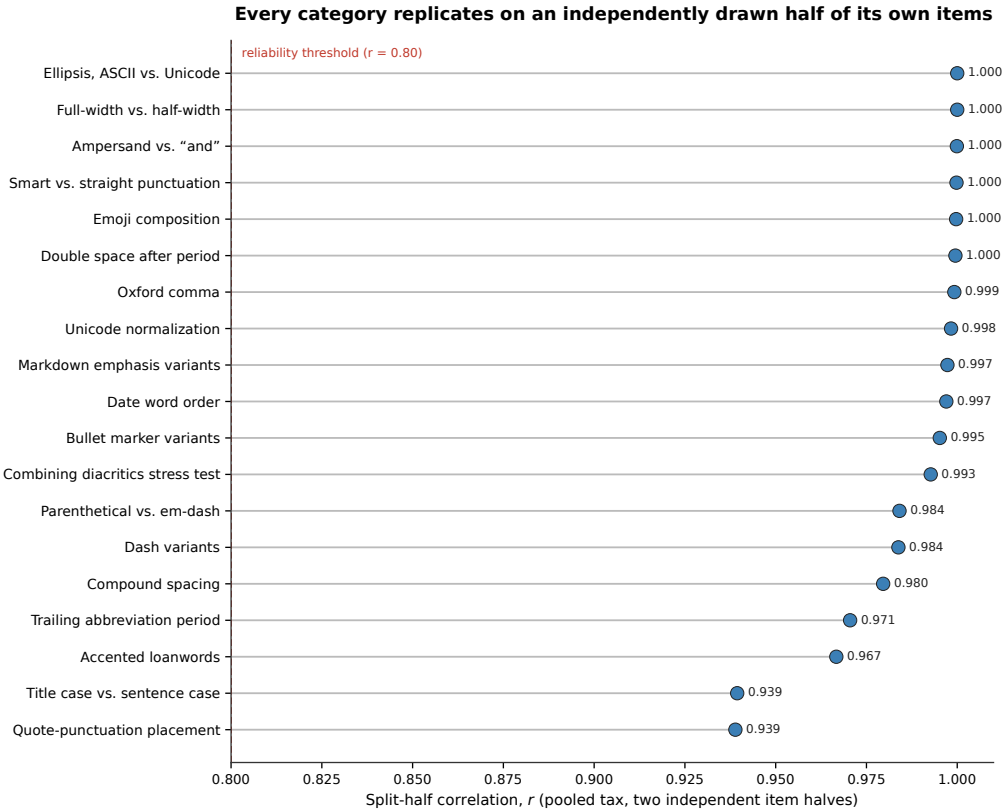

Figure 6: Split-half reliability for all 19 categories, ranked lowest to highest. Every category exceeds the conventional $r = 0.80$ reliability threshold by a wide margin; the lowest observed value is 0.939.

B.2 Cross-Tokenizer Rank Stability

Split-half reliability tests whether the pooled *mean* replicates; it does not test whether the *ranking* of tokenizers from most to least expensive is itself stable. Figure 7 answers this second question directly, by computing each tokenizer's mean tax on odd-indexed items and on even-indexed items

separately, then correlating the two rankings with Spearman's ρ. Every category exceeds $\rho = 0.87$, and 16 of 19 exceed $\rho = 0.99$, meaning a claim like "tokenizer X is unusually expensive for category Y" would hold regardless of which half of the item set happened to be sampled.

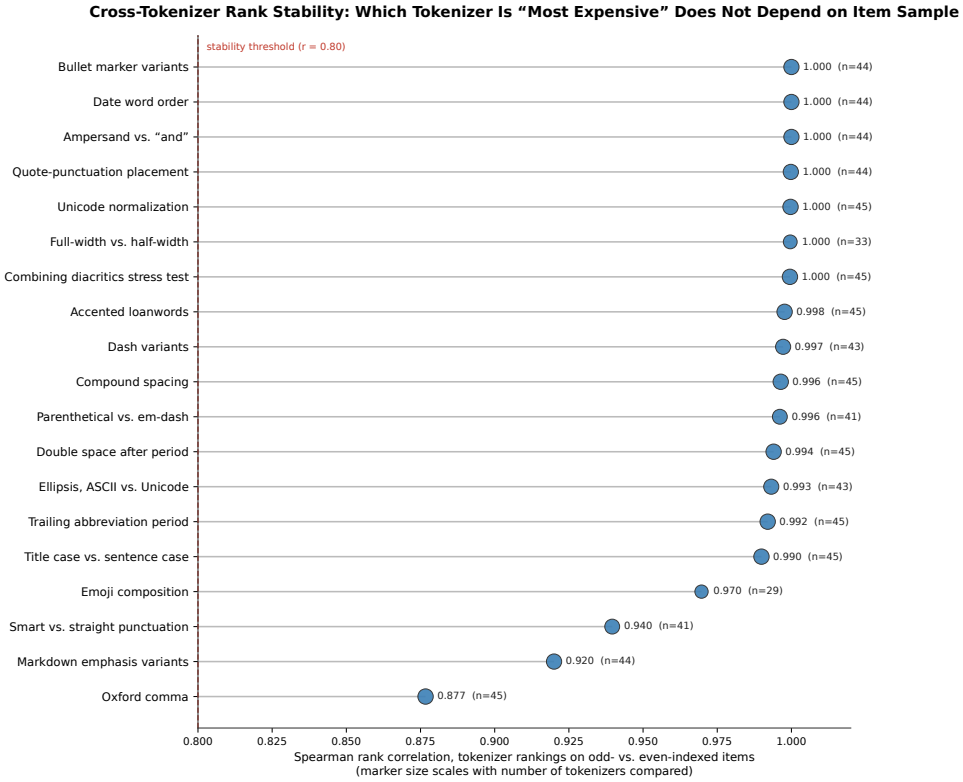

Figure 7: Cross-tokenizer rank stability for all 19 categories. Marker size scales with the number of tokenizers compared in that category, reduced for categories with high unknown-token exclusion, such as emoji composition and full-width vs. half-width.

B.3 Effect-Size Distribution

Figure 8 reports the median absolute Cohen's d for every category, together with the percentage of tokenizer-level tests within that category reaching a conventionally "large" effect size ($|d| \geq 0.8$). This complements the raw percentage-tax numbers in Figure 2: a category can have a modest raw tax but a large, consistent standardized effect if the within-category variance is small, and Figure 8 shows this is exactly what happens for several Tier B and C categories, such as dash variants, whose raw tax is under one percent but whose effect size is still non-trivial for a meaningful fraction of tokenizers.

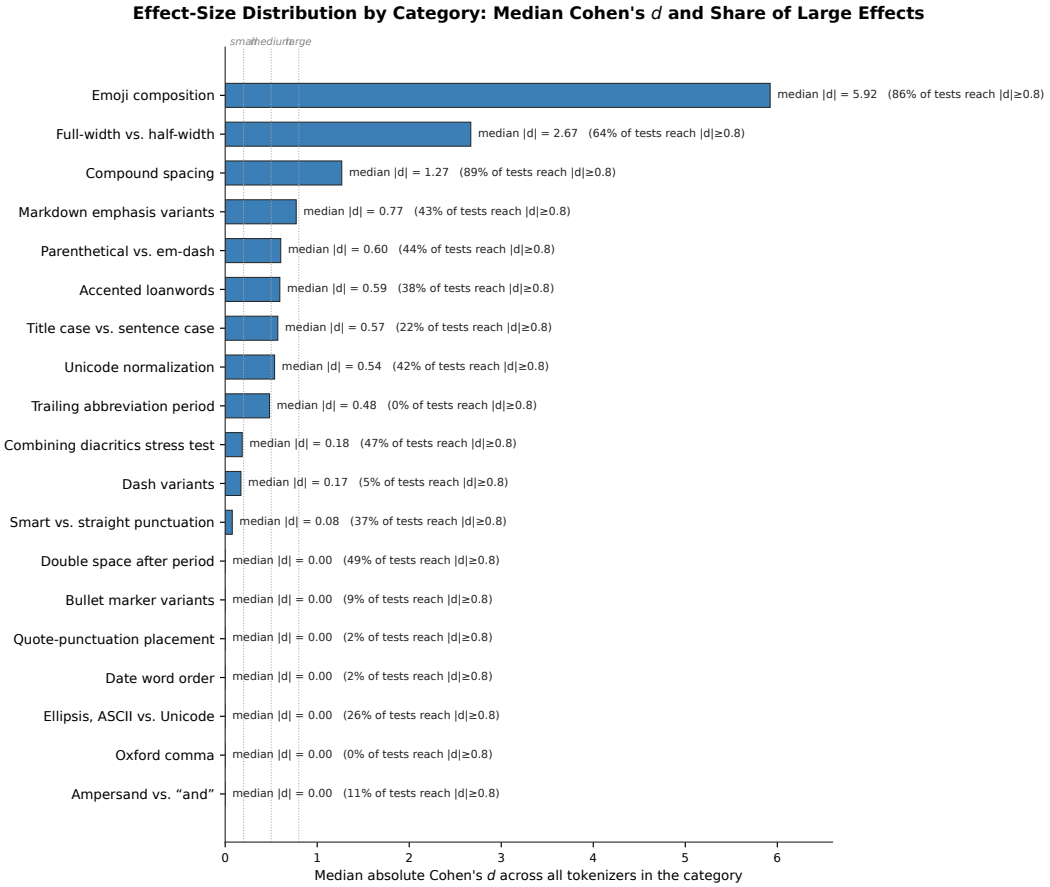

Figure 8: Median absolute Cohen's d per category, with the percentage of tokenizer-level tests reaching a large effect size ($|d| \geq 0.8$) annotated on each bar. Dotted reference lines mark Cohen's conventional small (0.2), medium (0.5), and large (0.8) thresholds.

Across all 810 tokenizer-category tests pooled together, 51.4% show a negligible effect size ($|d| < 0.2$), 8.0% small, 9.6% medium, and 31.0% large, confirming that this paper's significant results are not simply an artifact of very large sample sizes manufacturing statistical significance out of trivial effects; a substantial share of the significant results also carry a conventionally large standardized effect.

B.4 Tokenizer Style-Robustness Leaderboard

Averaging each tokenizer's absolute tax across all 19 categories produces a single, global measure of how sensitive that tokenizer is to formatting choice overall, independent of any single category. Figure 9 shows the ten most and ten least style-robust tokenizers of the 45 in our registry. The pattern by algorithm family is notable: the most style-robust tokenizers are dominated by SentencePiece unigram models and SentencePiece byte pair encoding with byte fallback, together with the one raw byte-level tokenizer in the registry, while the least style-robust tokenizers are, without exception, older byte-level byte pair encoding checkpoints in the GPT-2 lineage. This is consistent with this paper's broader finding that algorithm family matters, even though it explains less variance than formatting category itself (Figure 4).

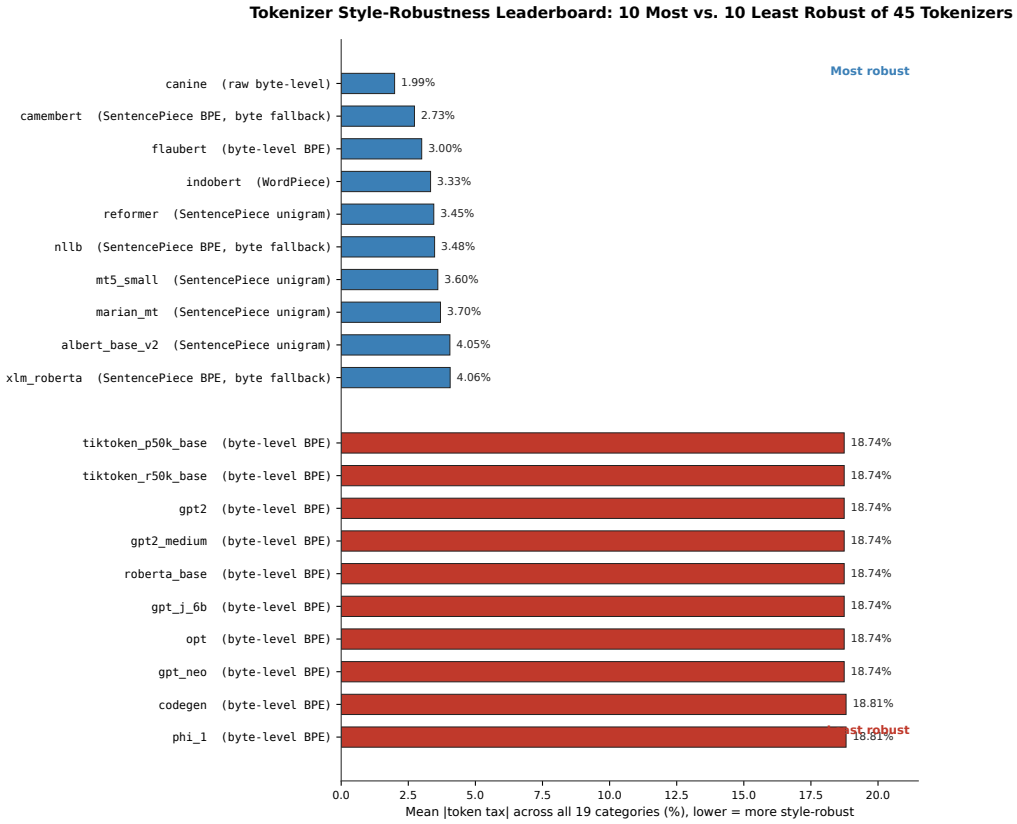

Figure 9: Tokenizer style-robustness leaderboard: the 10 most and 10 least style-robust tokenizers, measured as mean absolute token tax across all 19 categories, out of 45 tokenizers total. Every one of the 10 least robust tokenizers uses byte-level byte pair encoding.

B.5 Robustness Diagnostics Summary

Table 4 summarizes the robustness checks reported across this appendix and the main text in one place.

Table 4: Robustness diagnostics across all 19 categories.

Diagnostic	**Result**
Categories with split-half reliability > 0.8	19 of 19
Lowest split-half reliability observed	0.939 (quote-punctuation placement)
Categories with cross-tokenizer rank stability > 0.8	19 of 19
Lowest rank stability observed	0.877 (Oxford comma)
Categories with algorithm families differing significantly (Kruskal–Wallis)	19 of 19
Tokenizer-category results significant but practically trivial ($\lvert\text{tax}\rvert < 1\%$ or $\lvert d\rvert < 0.2$)	127 of 810
Outlier tokenizer flags ($\lvert z\rvert > 2$ within category)	24
Tests with a negligible effect size ($\lvert d\rvert < 0.2$)	51.4% of 810
Tests with a large effect size ($\lvert d\rvert \geq 0.8$)	31.0% of 810

C Additional Diagnostic Tables

C.1 Subtype-Level Breakdown for the Three Conflicted Categories

Table 5 reports the subtype-level mean tax for the three categories flagged for within-category direction conflicts in Section 5.4 and visualized in Figure 5a.

Table 5: Subtype-level mean tax for categories with internal direction conflicts. A positive value means the second-listed form of that subtype costs more; a negative value means the first-listed form costs more.

Category	**Subtype**	**Mean tax (%)**	n
Parenthetical vs. em-dash	Cause-and-effect clause	−3.36	1,025
	Appositive noun phrase	0.45	615
Dash variants	Double-hyphen vs. em-dash	−3.29	615
	Hyphen vs. en-dash, numeric score	1.50	1,290
	Hyphen vs. en-dash, date range	1.07	4,300
	Parenthetical vs. em-dash	0.39	615
Quote-punctuation placement	Exclamation mark	3.13	660
	Comma	−0.65	1,320
	Question mark	0.61	660
	Period	−0.24	880

C.2 Unknown-Token Exclusion by Category

Table 6 reports the unknown-token exclusion rate for every category in full precision, extending Figure 5b.

Table 6: Unknown-token exclusion rate by category, sorted descending.

Category	**Exclusion rate (%)**	**Severity**
Emoji composition	40.40	High
Full-width vs. half-width	26.67	High
Combining diacritics stress test	9.78	Moderate
Smart vs. straight punctuation	8.89	Moderate
Parenthetical vs. em-dash	8.89	Moderate
Unicode normalization form	6.26	Moderate
Title case vs. sentence case	5.70	Moderate
Dash variants	5.28	Moderate
Bullet marker variants	5.19	Moderate
Ellipsis, ASCII vs. Unicode	4.83	Low
Markdown emphasis variants	3.33	Low
Accented loanwords	2.81	Low
Ampersand vs. “and”	2.22	Low
Date word order	2.22	Low
Quote-punctuation placement	2.22	Low
Double space after period	0.04	Low
Oxford comma	0.00	Low
Compound spacing	0.00	Low
Trailing abbreviation period	0.00	Low

D Reproducibility Details

D.1 Environment

All tokenization was performed with Python 3, the HuggingFace `transformers` library, and OpenAI's `tiktoken` library, on standard CPU hardware. No GPU or model inference was required, since only tokenizer encoding was performed. Bootstrap resampling used a fixed random seed for full reproducibility of confidence intervals. All figures in this paper were generated directly from the raw statistical output with matplotlib, using data hardcoded from the corresponding result files at the time of writing; the figure-generation scripts are included with the released code.

D.2 Reproducibility Checklist

- All claims investigated in this work are clearly stated in Section 1.
- Source code for dataset construction, tokenization, statistical analysis, and figure generation will be made publicly available upon publication.
- Raw, unaggregated data, every tokenizer-pair token count, will be made available upon publication.
- Randomness in this work is limited to bootstrap resampling for confidence intervals and the random splits used in the split-half reliability and rank-stability diagnostics, all computed with a fixed seed.
- The number of comparisons used to compute each result is reported in every results table and figure.
- Every category in the dataset is reported in every table and figure; no category or tokenizer was excluded from the final analysis for any reason other than the unknown-token quality control described in Section 3.5.
- Analysis includes measures of variation and confidence beyond single-value summaries: bootstrap confidence intervals, effect sizes, split-half reliability, rank stability, and both parametric and non-parametric significance tests throughout.
- Multiple-comparison correction, Benjamini–Hochberg false discovery rate control, is applied across the full set of 810 tokenizer-category significance tests.

D.3 Dataset License and Availability

The full 2,828-pair dataset, all 45 tokenizer configurations, and the complete statistical analysis and figure-generation pipeline will be released under a permissive open license upon publication, allowing free use for research and reproduction.