

Do Association-Reweighted BPE Merge Rules Help Data-Efficient Language Modeling?

Zy Li Phu Han Trinh Tran Lam Duong Andrew Fang

Department of Computer Science, Seattle University

{zli10, phan, ktran24, lduong, afang}@seattleu.edu

Abstract

Subword tokenization determines a language model’s symbol inventory and sequence length, yet standard BPE selects merges by frequency alone. We test whether association-reweighted merges help on BabyLM Strict-Small. Holding the merge budget and architecture fixed and approximately matching word exposure, we train 97M-parameter causal decoders with three seeds using standard BPE, the S-BPE merge score with fixed-budget stopping, and exponential positive-PMI reweighting (PMI-BPE). Neither alternative reliably improves held-out bits-per-byte (BPB) or BLiMP. PMI-BPE’s mean BPB is slightly higher than standard BPE’s, but no pairwise difference is statistically reliable at three seeds. Both alternatives alter the merge tables and measurably change compression, ruling out a trivial no-intervention explanation. Yet intrinsic association rankings depend on scoring measure and aggregation. Type-uniform rankings reverse between post-merge PMI and thresholded held-out PPMI, while application-weighted PPMI is nearly identical across tokenizers. These results do not rule out association-aware tokenization; they cover one merge budget, one operating point per score, and two evaluation axes. Within that scope, the tested scores do not reliably help, and intrinsic merge association does not consistently align with the BPB and BLiMP point-estimate orderings.

1 Introduction

Subword tokenization is the first learned component of a language model and one of its few discrete design choices. Byte-Pair Encoding (BPE) (Gage, 1994; Sennrich et al., 2016) and its relatives fix the input and output vocabulary. This shapes sequence lengths, the granularity of the units a model must predict, and how the finite training budget is spent. This choice is especially consequential in the data-efficient setting, where the corpus is small and fixed. The question is then how much can be learned per

word rather than per FLOP, and no additional data can compensate for a poor segmentation.

Standard BPE is greedy and purely frequency-driven: at each step it merges the adjacent pair with the highest raw bigram count, $\arg \max_{(a,b)} N(a,b)$. This objective is agnostic to whether a pair coheres as a linguistic unit or merely co-occurs often. A recent line of methods revisits exactly this merge rule, reweighting it by a statistical association score so that merges reflect *cohesion* rather than raw frequency. We use *cohesion* as shorthand for statistical association under the stated metric, not as a direct measure of morphological or semantic well-formedness. Most build on pointwise mutual information (Church and Hanks, 1990). S-BPE weights the merge count by a pointwise mutual information term (Vilar and Federico, 2021), and related proposals reweight or gate merges by significance or attribution signals (Nouri, 2026; Foroutan et al., 2026; Chizhov et al., 2026). The shared intuition is to prefer merges whose parts predict each other rather than merely co-occur. A broader literature likewise argues that tokenization quality is not reducible to compression (Zouhar et al., 2023; Goldman et al., 2024; Schmidt et al., 2024; Bostrom and Durrett, 2020), motivating tokenizers optimized for something other than sequence length.

The underlying question has a narrow, testable form: *does association-aware tokenization actually improve data-efficient language modeling?* We answer it in the stringent low-resource BabyLM Strict-Small setting (Warstadt et al., 2023; Choshen et al., 2026), a fixed 10M-word corpus of developmentally plausible text. We train small language models from scratch under three tokenizers held to a common merge budget and a common *word* budget. The three are standard greedy BPE and two association-based merge criteria. The first is the S-BPE merge score with fixed-budget stopping (Vilar and Federico, 2021), which weights the merge

count by a pointwise mutual information term. The second is an exponential positive-PMI reweighting $N(a, b) \exp(\beta \text{PPMI}(a, b))$, which we call PMI-BPE and adopt as a representative instantiation of the PMI-reweighting family (published members use related but distinct functionals (Nouri, 2026), and $\beta=0$ recovers greedy BPE). All three use a common implementation so that only the merge criterion differs. We evaluate on bits-per-byte (BPB) over a held-out slice and on BLiMP zero-shot grammatical judgments (Warstadt et al., 2020), with three seeds per tokenizer.

Our answer is negative. Both alternatives substantially alter the merge table, ruling out a trivial no-intervention explanation. Intrinsic *association* does not provide a stable tokenizer ranking in this experiment, because whether a reweighted tokenizer scores higher than standard BPE depends on the association measure. Under the type-uniform post-merge-PMI summary both reweighted tokenizers exceed standard BPE. Under the type-uniform thresholded-PPMI summary neither does, and PMI-BPE, which uses positive PMI in its merge score, ranks last. Moreover, weighting each merge by how often its rule is applied narrows the positive-PMI spread to a fraction of a percent, and weighting by final-token frequency leaves standard BPE, not the reweighted tokenizers, slightly ahead. These diagnostics provide little reason to expect substantial separation on either evaluation axis, and we detect none reliably at three seeds (Table 3). The larger of the two differences from standard BPE is PMI-BPE’s higher mean BPB, reflecting a compression–prediction tradeoff: the easier per-token predictions its finer segmentation affords do not offset the additional predictions per byte it requires. Our conclusion rests on the two evaluation axes, not on the intrinsic association scores. Those scores rank the tokenizers differently under different measures, so we treat them as descriptive and do not offer them as an explanation of the negative result.

We make the following contributions:

- **A controlled data-efficient study.** We compare standard BPE, PMI-BPE, and S-BPE (Vilar and Federico, 2021) at a fixed merge budget and approximately matched word exposure, training a 97M-parameter Llama-style decoder from scratch with three seeds per tokenizer (nine runs), and report BPB and BLiMP in the BabyLM Strict-Small setting (Choshen et al., 2026).

- **An aggregation analysis of merge association.** We compare type-uniform, merge-application-weighted, and final-token-weighted association summaries. On this corpus, their rankings depend on the association measure. Under thresholded held-out PPMI, much of the difference is associated with rarely applied or unseen merges.
- **A negative result.** Neither association-reweighted merge rule reliably improves BPB or BLiMP. PMI-BPE has the higher mean BPB of the two alternatives, a compression–prediction tradeoff (Section 5).

Consistent with the analysis-oriented aims of the BabyLM effort (Warstadt et al., 2023), we report this negative result. We include the intrinsic measurements that document the tokenizer-level changes and how the intrinsic ranking depends on the association measure. Future work can then test whether association concentrated in frequent final tokens, or another model-relevant property, affects data-efficient learning.

2 Related Work

Association-based merge criteria. Standard BPE (Gage, 1994; Sennrich et al., 2016) grows a vocabulary by repeatedly merging the most frequent adjacent pair. This purely frequency-driven rule draws no distinction between statistically bound units (e.g. *morphemes*) and accidental within-word collocations, and a long line of work has replaced or augmented it with an *association* measure. WordPiece (Schuster and Nakajima, 2012; Wu et al., 2016) selects the merge that most increases training-corpus likelihood, a likelihood-ratio rule closely related to pointwise mutual information (PMI) (Church and Hanks, 1990). Vilar and Federico (2021) propose a statistical extension of BPE (S-BPE) that scores a pair using its count and a smoothed lower-bound term on the likelihood increase from merging it. Nouri (2026) combines a z -statistic under an independence null with an explicit compression-aware gain. The PMI-BPE tokenizer we study reweights each candidate’s frequency by an exponential of its *positive* PMI, with a strength β that recovers greedy BPE at $\beta=0$. S-BPE is a prior method. PMI-BPE is our controlled exponential instantiation of the broader association-reweighting idea, introduced as an experimental condition rather than a tokenizer contribution of

this paper. A parallel strand modifies the merge objective for goals orthogonal to cohesion: Parity-Aware BPE (Foroutan et al., 2026) applies a fair-max rule to reduce cross-lingual tokenization inequality, and SA-BPE (Chizhov et al., 2026) regularizes code-tokenizer training via source attribution and merge skipping. These works share our premise, that the greedy frequency rule is a design choice rather than a fixed default, but target fairness and code rather than data-efficient language modeling of English.

Data-efficient pretraining and BabyLM. The BabyLM challenge (Warstadt et al., 2023; Choshen et al., 2026) asks whether pretraining can be made sample-efficient on a fixed, developmentally plausible corpus of roughly 10M or 100M words, and it explicitly welcomes analysis and negative results. We use the Strict-Small (10M-word) corpus and training setting, and measure grammatical competence with BLiMP (Warstadt et al., 2020), a suite of minimal-pair paradigms scored zero-shot through the language-model evaluation harness (Biderman et al., 2024). Most BabyLM entries hold the tokenizer fixed and vary architecture, data curation, or objective. We instead make the tokenizer the independent variable and test whether association-based merging helps in this low-resource setting.

Tokenization and language-model quality. Whether tokenizer choice measurably affects downstream quality remains contested. Gallé (2019) attributes BPE’s effectiveness to the shorter sequences it produces, Rust et al. (2021) show that tokenizer fit strongly affects multilingual models, and Bostrom and Durrett (2020) find BPE suboptimal relative to a unigram tokenizer for pretraining. On intrinsic evaluation, Zouhar et al. (2023) cast tokenization as a noiseless channel, while Goldman et al. (2024) and Schmidt et al. (2024) caution that compression correlates only weakly with downstream accuracy. Mielke et al. (2021) survey the broader design space. Our aggregation analysis compares type-uniform, merge-application-weighted, and final-token-weighted summaries. It contributes an intrinsic measure tied to how often an association-reweighted merge rule is applied during encoding, and to how sensitive that diagnostic is to the association measure used to score it.

3 Method

We compare three tokenizers that differ only in the criterion used to score candidate merges during BPE training. All three share the same greedy agglomerative procedure of Gage (1994) and Senrich et al. (2016). Starting from a Unicode-character vocabulary, we repeatedly select the highest-scoring adjacent pair, add it as a new symbol, and re-count. We run 15,598 merge operations, corresponding to a target vocabulary size of $K = 16k$. The merge score is the only deliberate intervention. Any downstream difference follows from which merges are formed and in what order (and their consequences for vocabulary size, sequence length, and optimizer-step count), not from the encoding mechanism: each produces a standard merge table applied greedily at encode time, identically across all three.

3.1 Three merge criteria

Let $N(a, b)$ be the count of the adjacent pair (a, b) at the current stage of training, tallied within words only and weighted by word frequency, so no pair spans a word boundary. We define the positive pointwise mutual information (PPMI) of a pair as

$$\text{PPMI}(a, b) = \max\left(0, \log \frac{P(a, b)}{P_L(a) P_R(b)}\right), \quad (1)$$

the standard clipping of pointwise mutual information (Church and Hanks, 1990), where P_L and P_R are the left and right positional pair-marginals. Every observed pair remains eligible for merging. Let $\mathcal{R} = \{(x, y) : N(x, y) \geq 5\}$ be the retained pairs and $T = \sum_{(x, y) \in \mathcal{R}} N(x, y)$. For $(a, b) \in \mathcal{R}$, the three probabilities are estimated only over $\mathcal{R}$: $P(a, b) = N(a, b)/T$, $P_L(a) = \sum_{y: (a, y) \in \mathcal{R}} N(a, y)/T$, and $P_R(b) = \sum_{x: (x, b) \in \mathcal{R}} N(x, b)/T$. A pair outside $\mathcal{R}$ (that is, with $N(a, b) < 5$) receives $\text{PPMI} = 0$, so its score reduces to the frequency term $N(a, b)$. The held-out merge-association diagnostic below applies the same count-five threshold independently to development counts. All three criteria pick the next merge as $\arg \max_{(a, b)} s(a, b)$, differing only

in the score s :

$$\text{Standard BPE: } s(a, b) = N(a, b), \quad (2)$$

$$\text{PMI-BPE: } s(a, b) = N(a, b) e^{\beta \text{PPMI}(a, b)}, \quad (3)$$

$$\text{S-BPE: } s(a, b) = N(a, b) [\log p_n(ab) - \log p_n(a) - \log p_n(b)]. \quad (4)$$

PPMI clips PMI at zero. In Eq. 4, $N(a, b)$ is the number of adjacent (a, b) occurrences the merge consumes, which equals the count of the newly formed unit ab (for overlapping same-symbol pairs we use the non-overlapping application count). The $p_n(\cdot)$ are unigram probabilities computed from the *post-merge* counts, with add-one (Laplace) smoothing, following Vilar and Federico (2021).

Standard BPE (Eq. 2) is the greedy compression baseline: it merges the most frequent pair, a purely count-driven objective that ignores whether the two symbols are statistically associated beyond what their marginal frequencies predict.

PMI-BPE (Eq. 3) reweights the frequency score by a Boltzmann/exponential factor in positive PMI, so that among pairs of comparable frequency the more *cohesive* pair (higher PPMI) is preferred. A strength parameter β controls the cohesion prior: $\beta = 0$ reduces Eq. 3 exactly to standard BPE (Eq. 2), and larger β trades compression for association. We take this exponential form as a representative instantiation of the PMI-reweighting family. Published members reweight or gate merges by related association signals (Nouri, 2026) through different specific functionals. We fix $\beta = 0.5$ and do not sweep it on the BabyLM corpus (a limitation). This fixed, untuned operating point tests whether a cohesion prior provides any benefit on a 10M-word budget.

S-BPE (Eq. 4) is the statistical BPE extension of Vilar and Federico (2021). It scores the merge of (a, b) by a lower-bound term on the corpus log-likelihood increase from treating ab as a unit: the pair count times the change in unigram log-probability, computed from the post-merge unigram counts with add-one smoothing (their Eq. 7 and Appendix A). We faithfully implement the published S-BPE merge score. A common shortcut instead scores pairs by their pre-merge bigram PMI. That is a distinct approximation, and we found the two yield measurably different merge tables, so we

report the faithful criterion here. Under our fixed-budget stopping, all 15,598 selected merges have a positive S-BPE score, so fixed-budget stopping does not take the algorithm beyond the zero-score point.

3.2 Merge-association metric and its aggregations

To characterize what each merge table encodes about association, we score every merge by the PPMI of its pair *at merge time*, evaluated on held-out text. For the m -th merge, forming pair (a_m, b_m) , we compute $c_m = \text{PPMI}(a_m, b_m)$ using bigram and unigram statistics of (a_m, b_m) on the held-out dev slice under the partial merge table in effect just before merge m is applied. To suppress unstable estimates for rare held-out pairs, we score a pair whose held-out bigram count is below five as zero (Appendix E reports the ranking’s sensitivity to this choice). This measures, for each merge the tokenizer chose to create, how strongly its two constituents co-occur beyond chance on text the tokenizer did not train on.

We aggregate the per-merge scores $\{c_m\}$ three ways. The **type-uniform** aggregation is the plain mean over merges,

$$\bar{c}_{\text{type}} = \frac{1}{M} \sum_{m=1}^M c_m, \quad (5)$$

which counts every merge *type* once, regardless of how often it is used. The **merge-application-weighted** aggregation weights each merge by how often its rule is applied when the dev text is tokenized,

$$\bar{c}_{\text{app}} = \frac{\sum_m w_m c_m}{\sum_m w_m}, \quad (6)$$

where w_m is the number of times merge rule m is applied while encoding the dev slice. This is *not* the number of final token occurrences: a final token can absorb several merge applications, base symbols involve none, and the model consumes only final token IDs, never the merge rules. As an intrinsic proxy for the token level, we also report a **final-token-weighted** aggregation,

$$\bar{c}_{\text{token}} = \frac{\sum_t f_{\text{dev}}(t) g(t)}{\sum_t f_{\text{dev}}(t)}, \quad (7)$$

where $f_{\text{dev}}(t)$ is the dev frequency of final token type t and $g(t)$ is the association c_m of the merge that forms t , using the highest-priority creating

merge when a string is reachable by more than one merge order (base tokens contribute $g=0$). Even this treats t as an atomic ID and so remains an intrinsic proxy.

What the aggregations do and do not measure.

None of these statistics is a quantity the language model observes or optimizes: the model is trained on final token IDs, not on merge rules or their derivations. A merge rule applied often during encoding need not correspond to a frequent final token, so we report all three as intrinsic diagnostics, with the scoring measure stated, rather than as “what the model sees.” The gap between $\bar{c}_{\text{type}}$ and the usage-weighted forms is nonetheless diagnostic: if a tokenizer’s association advantage lives in rarely-applied merges, it contributes little to $\bar{c}_{\text{app}}$ or $\bar{c}_{\text{token}}$. All three aggregations, under both held-out scoring measures, are reported in Table 2; the type-uniform ordering reverses between them (Section 6). Our two evaluation axes, BPB and BLiMP, are reported in Table 3.

4 Experimental Setup

Data. We use the BabyLM-2026 Strict-Small corpus (Choshen et al., 2026), a fixed budget of 10M words drawn from six developmental and naturalistic English sources: CHILDES (2.84M words), Project Gutenberg (2.56M), OpenSubtitles (2.28M), Simple Wikipedia (1.53M), the spoken portion of the BNC (0.76M), and Switchboard (0.02M). This mixture mirrors the quantity and character of language a child is exposed to. The Strict-Small setting fixes the word budget, so all methods are compared under identical data. The release includes no official development split. We therefore hold out every 20th corpus line as a deterministic 5% development slice. These lines are short, about nine words on average, i.e. sentence- or utterance-sized rather than document-sized. This gives 9.50M training words (~ 1.05 M lines) and 0.50M held-out development words. Because we split by line rather than by source document, adjacent lines from the same source, for instance one CHILDES conversation, can fall on opposite sides of the split. The held-out slice may therefore share local context with training. We use it only for BPB, and treat that number as indicative rather than as an unbiased generalization estimate. BLiMP, our main external measure, is a minimal-pair benchmark and is unaffected. The dev slice is never seen during tokenizer training or language-model optimization.

All three tokenizers and all language models share this exact split.

Tokenizers. We compare the three merge criteria of Section 3: standard BPE, PMI-BPE ($\beta=0.5$), and S-BPE (Vilar and Federico, 2021). All three are trained with a common budget of 15,598 merge operations (target $K = 16$ k) on the 9.50M-word training split, from an identical 402-character base. This yields realized vocabularies of 15,982 / 16,000 / 15,986 token IDs (Appendix C). Realized vocabulary sizes differ by under 0.2%, and optimizer steps vary as a consequence of sequence length. Table 1 reports intrinsic properties of the three tokenizers on the dev slice; Section 5 reads them off.

Language model. For each tokenizer we train a 97M-parameter Llama-style decoder from scratch: $d_{\text{model}} = 768$, 12 layers, 12 attention heads, context length 1024, a SwiGLU feed-forward network with hidden width 2048, rotary position embeddings, RMSNorm, and tied input/output embeddings. The tied embedding is sized to each tokenizer’s realized vocabulary, so totals range 97.23–97.24M parameters, differing by under 0.02% across tokenizers. Training uses AdamW (peak learning rate 3×10^{-4} , cosine decay to $0.1 \times$ peak, warmup set to 10% of each run’s optimizer steps (~ 185), weight decay 0.1, gradient clipping 1.0) in bfloat16 on a single NVIDIA RTX 5090, with an effective batch of 65,536 tokens per step (micro-batch $16 \times$ context $1024 \times$ gradient accumulation 4). Each model performs ~ 10 passes over the fixed training text, roughly 1,850 steps. We set `max_steps per tokenizer` so that every tokenizer is trained on the fixed 9.50M-word text the same number of times ($\sim 10\times$), matched up to a dropped final partial block (Appendix A). Because PMI-BPE encodes the corpus into more tokens, holding the word budget fixed means its longer sequences cost proportionally more optimizer steps. This is a compute cost of the longer sequences. We do not correct for it, because the quantity we hold constant is the linguistic data. Appendix A reports the exact per-tokenizer step counts (PMI-BPE takes about 3.5% more steps than standard BPE). We train 3 random seeds per tokenizer, for 9 language-model runs in total, and report mean and sample standard deviation across seeds.

Evaluation. We evaluate on two axes. First, *BPB* on the held-out 0.50M-word dev slice, computed as the summed per-token negative log-likelihood

Tokenizer	B/tok $\uparrow$	Fert.	%split
Standard BPE	4.24	1.255	16.3
PMI-BPE ($\beta=0.5$)	4.09	1.300	18.0
S-BPE	4.20	1.266	16.4

Table 1: Compression properties on the held-out dev slice (target $K=16k$). Fert. is tokens per whitespace word and %split the fraction of words split into more than one token. Higher bytes/token is better compression. Merge-association summaries are in Table 2.

divided by (bytes $\times \ln 2$). BPB normalizes by bytes rather than tokens, so it compares models on a common scale despite their different sequence lengths and vocabulary segmentations. Second, *BLiMP* zero-shot accuracy (Warstadt et al., 2020): we score the stock blimp task (67 minimal-pair paradigms) in the lm-evaluation-harness (Biderman et al., 2024) via a model adapter for our from-scratch checkpoints. This is not the official blimp_filtered variant used for the BabyLM leaderboard, so our accuracies are not directly comparable to the official baseline. BLiMP is part of the BabyLM evaluation suite (Warstadt et al., 2023; Choshen et al., 2026). BPB is an additional tokenizer-comparable held-out language-modeling metric. BPB and BLiMP results (mean $\pm$ sample SD over 3 seeds per tokenizer) are reported in Table 3.

5 Results

Intrinsic metrics. Table 1 reports the three tokenizers on the held-out dev slice. The reweighting changes which merges are formed: the PMI-BPE and S-BPE merge tables share only 38% and 43% of their merges with standard BPE (Jaccard over the set of ordered merge rules). Standard BPE compresses best and splits the fewest words; PMI-BPE compresses worst, with the highest fertility, and S-BPE sits close to standard.

Table 2 reports mean merge association under both held-out scoring measures and all three aggregations. No aggregation makes the reweighted tokenizers uniformly more associated. Under thresholded held-out PPMI, standard BPE ranks highest type-uniformly and PMI-BPE last, despite PMI-BPE using PPMI in its own training-time merge score, and the final-token weighting ranks standard BPE highest as well. Under merge-application weighting the spread across all three collapses to below 0.3%. Scored instead by post-merge PMI, the type-uniform ordering reverses and both

Measure	Standard	PMI-BPE	S-BPE
PPMI, type-unif.	4.08	3.82	3.97
PPMI, app-wtd	2.131	2.133	2.130
PPMI, token-wtd	3.40	3.22	3.16
Post-merge PMI, type-unif.	5.66	6.24	6.00
Post-merge PMI, app-wtd	11.34	11.42	11.51
Post-merge PMI, token-wtd	4.98	4.91	4.75

Table 2: Mean merge association under two held-out scoring measures (thresholded PPMI and post-merge PMI), aggregated type-uniformly, by merge-application count, and by final-token frequency. Only the type-uniform ordering fully reverses between the two measures (see text).

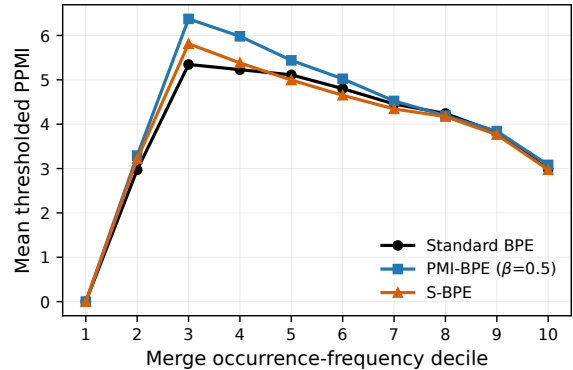


Figure 1: Mean held-out thresholded PPMI (minimum count five) of merges that fire on the dev slice, binned into deciles by application count (1 = rarest applied, 10 = most). Bins pool merges across tokenizers. PMI-BPE leads across most deciles, with standard BPE slightly ahead around the eighth. S-BPE is lowest in the more frequent deciles. See text for how this relates to the type-uniform means of Table 2.

reweighted tokenizers exceed standard BPE (Section 6). Figure 1 bins the merges that fire on the dev slice into deciles by application count and plots each decile’s mean PPMI. It conditions on fired merges, so it excludes merges that never fire on dev, which the type-uniform mean of Table 2 includes as zeros. The reweighted tokenizers place more merges in that zero-scored tail, which is why PMI-BPE ranks above standard BPE across most deciles here yet below it in the type-uniform mean.

Language modeling. Table 3 reports BPB on the held-out slice and BLiMP zero-shot accuracy, each averaged over three seeds.

First, *no association-based tokenizer reliably changes BPB relative to standard BPE*. The three means are close: standard 1.640 ± 0.003 , S-BPE 1.636 ± 0.007 , PMI-BPE 1.647 ± 0.003 . With three seeds, all primary 95% confidence intervals for the

tokenizer differences include zero (Welch intervals over the three independent runs per tokenizer; Appendix B). S-BPE’s mean is fractionally lower than standard’s, but the difference (-0.004) is mixed across seeds and small relative to across-seed variation. The larger of the two differences from standard BPE is PMI-BPE’s higher BPB, worse than standard by 0.007 , but its 95% confidence interval still includes zero. PMI-BPE fragments most (lowest bytes/token) and also has the highest mean BPB. S-BPE fragments slightly more than standard BPE yet has a slightly lower mean BPB, so the compression-to-BPB relationship is not monotone across the three.

Per-token loss is *not* comparable across tokenizers: each scores a different number of targets over the same text, so a finer segmentation mechanically lowers the loss per token. We list NLL/token in Table 3 only as a factor of the identity $\text{BPB} = (\text{NLL}/\text{token}) / (\text{bytes}/\text{scored token} \cdot \ln 2)$, and draw no tokenizer-quality conclusion from it. BPB normalizes by *bytes* and is the tokenizer-comparable quantity. In that identity, PMI-BPE spends more token predictions per byte, and the mechanically easier per-token predictions do not offset them, so its mean BPB sits above standard BPE’s; whether they offset is an empirical balance, not a consequence of segmentation alone, as S-BPE shows.

Second, *BLiMP differences are inconclusive*. All three tokenizers score within about one point of one another (57.3 – 57.8%), with overlapping across-seed variation. On the point estimate PMI-BPE is highest and S-BPE lowest, the reverse of the BPB ordering. But with three seeds we detect no reliable difference (a paradigm-level paired bootstrap gives 95% intervals that include zero; Appendix B). BLiMP is a minimal-pair benchmark, not normalized by bytes and less directly coupled to compression. It too provides no reliable evidence of an association benefit.

6 Discussion

Our result is negative, within the scope we test: on the 10M-word Strict-Small corpus (9.5M words for training), at one merge budget, one operating point per association score, and two evaluation axes, association-reweighted tokenization does not reliably improve data-efficient language modeling. With three seeds neither alternative differs reliably from standard BPE on either axis (Appendix B). On BPB, S-BPE’s slightly lower mean falls within

Tokenizer	NLL/tok [†]	BPB ↓	BLiMP ↑
Standard BPE	4.82	1.640 ± 0.003	57.5 ± 0.5
PMI-BPE ($\beta=0.5$)	4.67	1.647 ± 0.003	57.8 ± 0.9
S-BPE	4.77	1.636 ± 0.007	57.3 ± 0.2

Table 3: Language-model results: BPB and BLiMP zero-shot accuracy, mean $\pm$ sample standard deviation over 3 seeds. [†]NLL/token (nats; across-seed mean) is *not* comparable across tokenizers and is listed only as a factor of the identity $\text{BPB} = (\text{NLL}/\text{token}) / (\text{bytes}/\text{scored token} \cdot \ln 2)$ (Appendix C). No value is boldfaced: no pairwise difference is significant at three seeds (Section 5).

across-seed variation, and PMI-BPE’s higher mean is the larger of the two differences from standard BPE, but its 95% interval still includes zero. On BLiMP the differences are inconclusive.

A single intrinsic diagnostic cannot explain the negative result, because the intrinsic evidence is too measure-dependent. The reweighting does reshape the merge table, but whether that counts as higher association depends on the measure. Scored by the post-merge PMI component of S-BPE’s score (with no count threshold) on held-out text, both reweighted tokenizers exceed standard BPE type-uniformly, and both remain above standard under application weighting, though PMI-BPE and S-BPE exchange order (Table 2). Under our thresholded held-out PPMI diagnostic (count threshold five), the type-uniform ordering reverses: standard BPE scores highest, and PMI-BPE ranks last despite incorporating PPMI in its training-time merge score. The application-weighted spread across all three tokenizers then falls to a fraction of a percent (Eq. 6). The two measures disagree partly because the reweighted tokenizers concentrate merges on rare pairs assigned high scores by the smoothed post-merge measure: those pairs are assigned zero by our minimum-count threshold under thresholded positive PMI, and either way their rules are applied seldom. Intrinsic association is therefore too sensitive to the scoring measure to explain the absence of a reliably detected difference on either axis, and type-uniform association in particular is an unreliable summary.

The observed PMI-BPE point-estimate difference reflects a compression–prediction tradeoff. BPB is the tokenizer-comparable held-out metric used here: unlike per-token loss, which is not comparable across segmentations because each tokenizer scores a different number of targets, BPB is

expressed per byte. PMI-BPE’s finer segmentation spends more token predictions per byte, the easier per-token predictions it affords do not offset them, and by BPB it has the worse mean (Table 3).

Two implications follow for small-data tokenizer design. First, intrinsic association is an unreliable predictor of measured model quality. Reported type-uniformly, it can point in either direction depending on the association measure. Its application-weighted form differs by about 0.14% across tokenizers (PMI-BPE marginally highest), and its final-token-weighted form ranks standard BPE highest. No intrinsic summary matches both point-estimate orderings: application-weighted PPMI happens to match the BLiMP point-estimate ordering, but its spread is below 0.3% and the BLiMP differences are inconclusive. Intrinsic association should be reported and read with the scoring measure stated. Second, we test PMI-BPE at a single fixed strength ($\beta=0.5$) and S-BPE at the published merge score with fixed-budget stopping. A strength or stopping rule calibrated on larger corpora should not be assumed to carry over, and the useful setting, if any, would need to be re-tuned for each corpus size. Association-reweighted tokenization is not ruled out in principle. One possibility is to concentrate association in the merges that become frequent final tokens, rather than in the rare pairs a PMI prior most rewards.

7 Conclusion

At the tested operating points (a shared 15,598-merge budget targeting $K=16k$, PMI-BPE at $\beta=0.5$, and S-BPE with fixed-budget stopping), neither association-reweighted merge rule yields a reliably detected improvement over standard BPE for our 97M-parameter models on the BabyLM Strict-Small corpus, on either held-out BPB or BLiMP. With three seeds we detect no significant difference among the three tokenizers on BPB or BLiMP. PMI-BPE’s higher mean BPB is the larger of the two differences from standard BPE, but its 95% interval includes zero. The easier per-token predictions its finer segmentation affords do not offset the additional token predictions per byte it requires. The reweighting does change the tokenizer, so the absence of a reliably detected difference on either axis is not explained by identical tokenizations. We were unable to trace it to a specific intrinsic cause, because the tokenizers’ ranking by association varies with both the measure and the

aggregation: the type-uniform ordering reverses between post-merge PMI and thresholded PPMI. For small-data tokenizer design, this cautions against treating intrinsic association, and type-uniform association in particular, as a predictor of language-model quality. One possibility is to concentrate association in the merges that become frequent final tokens.

Limitations

Our study is narrow in scope. It covers a single language (English) and a single model scale (97M parameters). Larger models, or morphologically richer languages where association may align better with meaningful units, could behave differently. We compare three merge criteria at one merge budget (target $K=16k$). Vocabulary size is an influential hyperparameter that we neither swept nor tuned per merge criterion, so we cannot exclude that the three rank differently at another budget. We performed no hyperparameter tuning of any kind, on the tokenizers or the language models. PMI-BPE is tested at a single strength ($\beta=0.5$) and S-BPE at a fixed-budget stopping rule rather than its published likelihood-gain stopping criterion, so we cannot exclude that another operating point behaves differently on this corpus. With only three runs per tokenizer, the confidence intervals are wide. The zero-spanning intervals do not establish equivalence and remain compatible with small effects in either direction. Our intrinsic association findings are measure-relative: the ordering of tokenizers by merge association reverses between a thresholded positive-PMI measure and post-merge PMI, so we report the BPB and BLiMP results as primary and treat the intrinsic diagnostics as descriptive. The training-time PMI bonus uses a fixed count-five cutoff that we do not sweep. Appendix E varies only the held-out diagnostic cutoff and shows the intrinsic ranking is sensitive to that evaluation choice. The held-out development slice is split by line rather than by source, so its BPB may share local context with training (Section 4). BLiMP, our main external measure, is unaffected by this train/development split. Our two evaluation axes are held-out language modeling (BPB) and grammatical minimal-pair accuracy (BLiMP); neither is a broad downstream-task evaluation. Other BabyLM evaluations, including the fine-tuning suite, are left to future work. Finally, WordPiece and other likelihood-based association

criteria are discussed but not trained end-to-end in the language-modeling comparison.

References

- Stella Biderman, Hailey Schoelkopf, Lintang Sutawika, Leo Gao, Jonathan Tow, Baber Abbasi, Alham Fikri Aji, Pawan Sasanka Ammanamanchi, Sidney Black, Jordan Clive, Anthony DiPofi, Julien Etzaniz, Benjamin Fattori, Jessica Zosa Forde, Charles Foster, Jeffrey Hsu, Mimansa Jaiswal, Wilson Y. Lee, Haonan Li, and 11 others. 2024. Lessons from the trenches on reproducible evaluation of language models. *arXiv preprint arXiv:2405.14782*.
- Kaj Bostrom and Greg Durrett. 2020. Byte pair encoding is suboptimal for language model pretraining. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4617–4624.
- Pavel Chizhov, Egor Bogomolov, and Ivan P. Yamshchikov. 2026. From where words come: Efficient regularization of code tokenizers through source attribution. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 39053–39073.
- Leshem Choshen, Ryan Cotterell, Mustafa Omer Gul, Jaap Jumelet, Tal Linzen, Aaron Mueller, Suchir Salhan, Raj Sanjay Shah, Alex Warstadt, and Ethan Gotlieb Wilcox. 2026. BabyLM turns 4 and goes multilingual: Call for papers for the 2026 BabyLM workshop. *arXiv preprint arXiv:2602.20092*.
- Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29.
- Negar Foroutan, Clara Meister, Debjit Paul, Joel Niklaus, Sina Ahmadi, Antoine Bosselut, and Rico Sennrich. 2026. Parity-aware byte-pair encoding: Improving cross-lingual fairness in tokenization. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7514–7538.
- Philip Gage. 1994. A new algorithm for data compression. *The C Users Journal*, 12(2):23–38.
- Matthias Gallé. 2019. Investigating the effectiveness of BPE: The power of shorter sequences. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1375–1381.
- Omer Goldman, Avi Caciularu, Matan Eyal, Kris Cao, Idan Szpektor, and Reut Tsarfaty. 2024. Unpacking tokenization: Evaluating text compression and its correlation with model performance. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2274–2286.
- Sabrina J. Mielke, Zaid Alyafeai, Elizabeth Salesky, Colin Raffel, Manan Dey, Matthias Gallé, Arun Raja, Chenglei Si, Wilson Y. Lee, Benoît Sagot, and Samson Tan. 2021. Between words and characters: A brief history of open-vocabulary modeling and tokenization in NLP. *arXiv preprint arXiv:2112.10508*.
- Azam Nouri. 2026. Significance-gain pair encoding for LLMs: A statistical alternative to frequency-based subword merging. *arXiv preprint arXiv:2603.19261*.
- 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.
- Craig W. Schmidt, Varshini Reddy, Haoran Zhang, Alec Alameddine, Omri Uzan, Yuval Pinter, and Chris Tanner. 2024. Tokenization is more than compression. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 678–702.
- Mike Schuster and Kaisuke Nakajima. 2012. Japanese and Korean voice search. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152.
- Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725.
- David Vilar and Marcello Federico. 2021. A statistical extension of byte-pair encoding. In *Proceedings of the 18th International Conference on Spoken Language Translation (IWSLT 2021)*, pages 263–275.
- Alex Warstadt, Aaron Mueller, Leshem Choshen, Ethan Wilcox, Chengxu Zhuang, Juan Ciro, Rafael Mosquera, Bhargavi Paranjabe, Adina Williams, Tal Linzen, and Ryan Cotterell. 2023. Findings of the BabyLM challenge: Sample-efficient pretraining on developmentally plausible corpora. In *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning (CoNLL)*, pages 1–34.
- Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. 2020. BLiMP: The benchmark of linguistic minimal pairs for English. *Transactions of the Association for Computational Linguistics*, 8:377–392.
- Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, and 12 others. 2016.

Google’s neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Vilém Zouhar, Clara Meister, Juan Gastaldi, Li Du, Mrinmaya Sachan, and Ryan Cotterell. 2023. Tokenization and the noiseless channel. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5184–5207.

A Per-tokenizer exposures

Table 4 gives the exact training exposure of each tokenizer. All three are trained for ~ 10 passes over the same 9.50M-word training text, so *word* exposure is matched up to a dropped final partial block (87, 371, and 700 tokens for standard BPE, PMI-BPE, and S-BPE, each $< 0.006\%$ of tokens, with the dropped suffix differing by segmentation). They are not matched on compute. A tokenizer that fragments more encodes the fixed word budget into more tokens, and therefore performs more optimizer steps at a fixed effective batch of 65,536 tokens. PMI-BPE takes 1,884 steps to standard BPE’s 1,820 (about 3.5% more), and S-BPE 1,834 (about 0.8% more). We hold the linguistic data constant and do not correct for this. The extra steps are a cost of longer sequences. BPB does not normalize for this additional training compute. The comparison is word-exposure-matched rather than compute-matched.

Tokenizer	Train tok.	Vocab	Steps	Pos.	Ep.
Standard BPE	11.93M	15,982	1,820	119.3M	10.0
PMI-BPE ($\beta=0.5$)	12.35M	16,000	1,884	123.5M	10.0
S-BPE	12.02M	15,986	1,834	120.2M	10.0

Table 4: Per-tokenizer training exposure. “Train tok.” is the encoded training token count. “Steps” is the optimizer steps for 10 epochs at an effective batch of 65,536 tokens. “Pos.” is the total token positions processed (steps $\times$ batch). “Ep.” is the epochs over the fixed 9.50M-word text. Word exposure is matched up to a sub-block tail. Optimizer steps are not.

B Per-seed results and significance

Table 5 lists all nine runs. Because the three tokenizers have different vocabularies, matched seed indices do not guarantee matched initialization or data order. Our primary analysis therefore uses unpaired Welch intervals over the three independent runs per tokenizer, and reports the matched-seed (paired) differences only as exploratory. All primary 95% confidence intervals for the two differ-

ences from standard BPE include zero (Welch t intervals with Welch–Satterthwaite degrees of freedom). For BPB, the Welch mean difference from standard BPE is +0.007 for PMI-BPE (95% CI $[-0.001, +0.015]$) and -0.004 for S-BPE (95% CI $[-0.019, +0.011]$). For BLiMP, the Welch differences are +0.4 points for PMI-BPE (95% CI $[-1.5, +2.3]$) and -0.2 for S-BPE (95% CI $[-1.5, +1.1]$). A paradigm-level paired bootstrap (10,000 resamples over the 67 paradigms, averaged across seeds, which does not propagate seed uncertainty) gives tighter but still zero-spanning intervals, $[-0.9, +1.7]$ and $[-1.1, +0.6]$ respectively. The remaining pair, PMI-BPE versus S-BPE, also spans zero (BPB +0.011, 95% CI $[-0.004, +0.026]$; BLiMP +0.6 points, 95% CI $[-1.7, +2.8]$), so no pairwise difference is reliable. All Welch intervals are computed from unrounded per-seed scores. The exploratory matched-seed analysis agrees: the only directionally consistent effect is PMI-BPE’s higher BPB, higher at each of the three matched seed indices, whose interval nonetheless includes zero.

C Tokenizer implementation

All three tokenizers share one implementation, differing only in the merge score (Section 3). **Initialization:** the base vocabulary is the set of Unicode characters observed in the training corpus. Each whitespace-delimited word is suffixed with a four-character end-of-word marker $</w>$ and split into characters, so the marker is itself built from the characters $<, /, w, >$. The base is identical across all three tokenizers (402 characters). **Normalization:** runs of whitespace are collapsed to single spaces and leading/trailing whitespace is stripped. There is no lowercasing or Unicode normalization, so case is preserved. **Word boundaries:** BPE is applied within words only and merges never cross whitespace. Decoding concatenates symbols and replaces $</w>$ with a space. **Special tokens:** the tokenizer adds none, and at encoding time an unseen symbol maps to id 0. Seven dev word-tokens (of $\sim 0.50\text{M}$) contain a character unseen in training and are mapped this way. Excluding those lines changes BPB by under 0.0001 (they are 0.004% of dev bytes), below the precision we report. BLiMP contains no character outside the training base, so no BLiMP token is out-of-vocabulary. The corpus contains no literal occurrence of the marker string $</w>$. **Merge selection:** at each step the highest-

scoring adjacent pair is merged, with ties broken deterministically by symbol-id order (a min-heap keyed by $(-s(a, b), a, b)$). There is no candidate-elimination floor, so every observed pair stays eligible. In PMI-BPE, a pair with $N(a, b) < 5$ receives $\text{PPMI} = 0$, reducing its score to the ordinary frequency term, and the held-out diagnostic independently applies the same count-five cutoff. **Logarithms** are natural throughout (the merge scores and the association metric). The PPMI uses a count threshold of five and clips at zero on the held-out dev slice. **Realized vocabulary:** all three target $K = 16,000$ from the identical 402-character base, but the realized number of distinct token IDs differs slightly (15,982 / 16,000 / 15,986). A few merges produce a symbol string reachable by more than one merge order and so add no new ID. We report the realized sizes. The $<0.2\%$ difference is small relative to the merge budget rather than provably without effect. **BLiMP scoring:** we use `lm-evaluation-harness v0.4.11` over the standard `blimp` group (67 paradigms, 1,000 minimal pairs each). We sum each sentence’s conditional token log-likelihoods from the second token onward. Because the model has no BOS, EOS, or special token, the first token is unscored, and the delimiter space is dropped by whitespace pre-tokenization. A pair counts as correct when the grammatical sentence has the higher summed log-likelihood, and accuracy is the equally-weighted mean over paradigms. In all 67,000 pairs the two sentences share the same first token under every tokenizer, so the unscored first token cancels exactly and cannot bias the comparison.

Language-model inputs. The encoded training corpus is a single concatenated token stream, chunked into non-overlapping 1024-token blocks. Attention is causal within a block and does not cross block boundaries, so a block may span multiple corpus lines and sources. Training blocks are shuffled and the final incomplete block is dropped, with no padding. For BPB we process the concatenated dev stream in 1024-token contexts with stride 1024, each context predicting its following 1024 tokens (so the first prediction in each window uses short context, identically for all three tokenizers). The last target of a window is the first context token of the next, so no target is scored twice, and the final partial window is retained with its padded positions masked out of the loss. Every dev token except the first is therefore scored exactly once: 623,151, 645,468, and 628,317

scored targets for standard BPE, PMI-BPE, and S-BPE. NLL per token is the summed target NLL divided by these scored targets. BPB is the summed target NLL in bits divided by the UTF-8 byte count of the scored targets: the total dev byte count minus the first token’s five bytes, 2,640,322 for all three tokenizers (the dev text and its shared first token are identical across tokenizers). Only the first token is unscored, so the Table 3 decomposition $\text{BPB} = (\text{NLL}/\text{token}) / (\text{bytes}/\text{scored token} \cdot \ln 2)$ is an exact identity. The model uses no dropout. RMSNorm uses $\epsilon=10^{-5}$, and AdamW uses $\beta_1=0.9$, $\beta_2=0.95$, $\epsilon=10^{-8}$. The corpus is the Hugging Face dataset `BabyLM-community/BabyLM-2026-Strict-Small` (default configuration, revision `c92ab16`).

Tokenizer	Seed	BPB ↓	BLiMP (%) ↑
Standard BPE	0	1.6378	57.41
	1	1.6440	58.03
	2	1.6385	56.97
PMI-BPE	0	1.6489	56.78
	1	1.6488	58.38
	2	1.6428	58.38
S-BPE	0	1.6280	57.13
	1	1.6393	57.24
	2	1.6414	57.45

Table 5: Per-seed BPB and BLiMP accuracy for all nine runs.

D Merge association under multiple measures

Table 2 reports each tokenizer’s mean merge association under the two scoring measures we use: thresholded held-out PPMI (count threshold five, clipped at zero), and post-merge PMI (the association component of S-BPE’s score, $\log p_n(ab) - \log p_n(a) - \log p_n(b)$, with no threshold), each aggregated type-uniformly, by merge-application count ($\bar{c}_{\text{app}}$), and by final-token frequency ($\bar{c}_{\text{token}}$). The intrinsic ranking depends on both the measure and the aggregation. Under PPMI, standard BPE has the highest association type-uniformly and by final-token weight, and application weighting reduces the spread to near zero. Under post-merge PMI the type-uniform ordering fully reverses. Under application weighting, both reweighted tokenizers move above standard BPE, with S-BPE highest. The final-token aggregation again ranks standard BPE highest (4.98 vs. 4.91 and 4.75). The final-token aggregation is the closest of the

three proxies to the model’s final input units. It thus favors standard BPE under both measures, yet standard BPE does not reliably win downstream (Table 3). No intrinsic summary matches both downstream point-estimate orderings: application-weighted PPMI happens to match the BLiMP point-estimate ordering (PMI-BPE > standard > S-BPE), but its cross-tokenizer spread is below 0.3% and the BLiMP differences are inconclusive. Intrinsic association therefore describes the tokenizers but does not consistently align with downstream point estimates in this experiment. The PPMI/post-merge-PMI disagreement is partly driven by rare pairs that receive positive post-merge PMI but are assigned zero by our minimum-count rule under the thresholded held-out PPMI diagnostic. PMI-BPE places the most merges in the zero-scored tail: 1.5% never fire on dev and 19.5% fire below the count threshold, versus 0.2%/10.5% for standard BPE and 0.5%/12.9% for S-BPE. Figure 1 conditions on merges that fire ($w_m > 0$). Its within-bin means do not determine the type-uniform mean, because the tokenizers place different proportions of their merge inventories in each frequency range, including the never-applied group.

E Count-threshold sensitivity

The type-uniform held-out PPMI ranking is itself sensitive to the minimum-count threshold (Table 6). At thresholds 1 and 3 the reweighted tokenizers score highest, led by PMI-BPE. At 5 (our setting) and 10 standard BPE does. Because the reversal occurs within the range of thresholds examined here, this reinforces that the type-uniform ranking is parameter-dependent and should not be read as a predictor of downstream quality.

Min. count	Standard	PMI-BPE	S-BPE
1	4.23	4.81	4.44
3	4.32	4.44	4.37
5	4.08	3.82	3.97
10	2.75	2.60	2.68

Table 6: Type-uniform held-out PPMI by minimum-count threshold. The threshold-5 row matches Table 2.