Marginalizing Over BPE Tokenizations for Calibrated Word-Level Probabilities in Whisper

Anonymous Author(s)

ABSTRACT

Byte-pair encoding (BPE) tokenizers map a single word to multiple valid token sequences—different subword splits, casing variants, and spacing decompositions—causing the true word-level probability to be distributed across exponentially many paths through the decoder. Current practice estimates word confidence from the single canonical BPE segmentation, systematically underestimating the true probability. We formalize this problem by constructing a *segmentation DAG* whose source-to-sink paths enumerate all valid tokenizations of a word, and propose a forward algorithm that marginalizes over this DAG using the decoder's conditional token probabilities. We analyze 184 English words spanning 2–15 characters using Whisper's GPT-2 tokenizer and find that: (i) the number of valid tokenizations grows exponentially with word length, reaching a median of 3,006 paths for words of 11+ characters; (ii) ignoring alternative tokenizations underestimates log-probability by a median of 0.03 nats for short words but up to 1.58 nats for longer words; (iii) including case variants adds a further 1.09 nats of marginalization gap on average; and (iv) a beam-pruned forward algorithm with width 10 recovers >99.9% of the exact marginal probability for short words and >96% for long words. Our approach provides principled, calibrated word-level uncertainty estimates for BPE-based speech recognition and language models, directly addressing the open problem identified by Bondarenko et al. (2026).

ACM Reference Format:
Anonymous Author(s). 2026. Marginalizing Over BPE Tokenizations for Calibrated Word-Level Probabilities in Whisper. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Automatic speech recognition (ASR) systems increasingly rely on large-scale neural models such as Whisper [11] that produce text transcriptions from audio input. These models are deployed in high-stakes applications—medical dictation, legal transcription, accessibility services—where *word-level confidence estimation* is critical: given a transcribed word, how certain is the model that it correctly decoded the spoken utterance?

Whisper and similar models use byte-pair encoding (BPE) [12] to segment text into subword tokens. The decoder is autoregressive, producing one token at a time with associated conditional probabilities. To obtain a word-level probability, practitioners typically multiply the conditional probabilities of the tokens composing the canonical BPE segmentation of the word. However, BPE tokenizers are *ambiguous*: the same word admits multiple valid token sequences. For example, the word "cat" in Whisper's GPT-2 tokenizer can be represented as:

Conference'17, July 2017, Washington, DC, USA
2026. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

- The single token `' cat'` (token ID 3797)
- Two tokens: `' c'` + `'at'` (IDs 269, 265)
- Two tokens: `' ca'` + `'t'` (IDs 1275, 83)
- Three tokens: `' c'` + `'a'` + `'t'` (IDs 269, 64, 83)
- And four additional paths involving the space prefix

Including casing variants (`' Cat'`, `' CAT'`) further expands the set. In total, the word "cat" has 8 valid tokenization paths for a single casing, and 24 paths across all case variants.

Bondarenko et al. [2] explicitly noted this issue in the Pisets speech recognition system, observing that probability mass is spread across tokenizations, but deferred a solution to future work. The true word probability requires *marginalizing* over all valid tokenization sequences:

$$P(\text{word} \mid \text{audio}) = \sum_{s \in \mathcal{S}(\text{word})} P(s \mid \text{audio}) \tag{1}$$

where $\mathcal{S}(\text{word})$ is the set of all token sequences whose concatenated decoded strings equal the target word. Using only the canonical segmentation yields $P(s^* \mid \text{audio})$, which is a *lower bound* on the true word probability.

In this paper, we make the following contributions:

(1) We formalize the tokenization marginalization problem through a *segmentation DAG* whose paths enumerate all valid BPE tokenizations of a word, including casing and spacing variants (Section 2.2).
(2) We propose exact and beam-pruned forward algorithms for computing the marginalized word probability (Section 2.3).
(3) We provide the first systematic empirical analysis of the marginalization gap across 184 English words, quantifying how word length, path count, and casing variants affect probability underestimation (Section 3).
(4) We demonstrate that beam-pruned marginalization with width 10 recovers >99.9% of exact marginal probability for typical words, making the approach practical for real-time ASR (Section 3.3).

1.1 Related Work

BPE Tokenization and Ambiguity. Sennrich et al. [12] introduced BPE for neural machine translation, creating a fixed vocabulary of subword units through iterative merging of frequent character pairs. The greedy merge procedure produces a canonical segmentation, but the vocabulary admits many other valid decompositions. Kudo [7] proposed subword regularization, training models by sampling from multiple tokenizations to improve robustness. Provilkov et al. [10] extended this idea with BPE-dropout, randomly dropping merges during training. Both works address the *training* side of tokenization ambiguity; our work addresses the *inference* side—correctly aggregating probability at decoding time.

Marginalization Over Tokenizations. Cao and Rimell [3] demonstrated that language model perplexity varies across tokenizations of the same text and proposed marginalizing over segmentations using a lattice-based forward algorithm for evaluation. Their work is the most directly relevant prior art. We adapt and extend their approach to the autoregressive speech recognition setting, where conditional probabilities depend on audio context and preceding token history.

Uncertainty in Neural Sequence Models. Guo et al. [5] showed that modern neural networks are often miscalibrated, producing overconfident predictions. Malinin and Gales [8] studied uncertainty estimation in autoregressive models, distinguishing between data (aleatoric) and model (epistemic) uncertainty. He et al. [6] evaluated confidence elicitation in large language models. In ASR, Bondarenko et al. [2] used the minimum or average of token-level log-probabilities as a word-level uncertainty measure but acknowledged that the tokenization ambiguity biases these estimates.

Lattice and DAG Methods in ASR.. The CTC (Connectionist Temporal Classification) algorithm [4] marginalizes over alignment paths for sequence labeling using a forward-backward algorithm on a lattice. Weighted finite-state transducers (WFSTs) [9] provide a general framework for lattice-based computation in speech recognition, enabling efficient composition of acoustic, lexical, and language models. The wav2vec 2.0 framework [1] uses CTC for self-supervised speech representation learning. Our segmentation DAG is structurally analogous to a CTC lattice but operates over tokenization paths rather than temporal alignment paths.

2 METHODS

2.1 Problem Formulation

Consider an autoregressive decoder that generates tokens $t_1, t_2, \ldots, t_K$ with conditional probabilities $P(t_k \mid t_{1:k-1}, a)$ where a denotes the audio conditioning (encoder hidden states). The probability of a specific token sequence $s = (t_1, \ldots, t_K)$ is:

$$P(s \mid a) = \prod_{k=1}^{K} P(t_k \mid t_{1:k-1}, a) \tag{2}$$

Given a target word w (a character string), let $\mathcal{S}(w)$ denote the set of all token sequences whose concatenated decoded strings equal w. The marginalized word probability is defined by Equation 1. In practice, the word appears in context: preceding tokens $\pi = (t_1, \ldots, t_M)$ have already been decoded. The conditional marginalized probability is:

$$P(w \mid \pi, a) = \sum_{s \in \mathcal{S}(w)} \prod_{k=1}^{|s|} P(s_k \mid \pi \oplus s_{1:k-1}, a) \tag{3}$$

where $\oplus$ denotes concatenation.

The *marginalization gap* quantifies the probability mass missed by using only the canonical tokenization s^*:

$$\Delta(w) = \log P(w \mid \pi, a) - \log P(s^* \mid \pi, a) \geq 0 \tag{4}$$

The inequality holds because the canonical tokenization is one element of the sum.

2.2 Segmentation DAG Construction

We model $\mathcal{S}(w)$ as a directed acyclic graph (DAG):

Definition 2.1 (Segmentation DAG). Given a word string w of length n and a vocabulary V mapping token IDs to strings, the segmentation DAG $G = (N, E)$ has:

- Nodes $N = \{0, 1, \ldots, n\}$ representing character positions.
- An edge $(i, j, \text{tid}) \in E$ exists iff the substring $w[i:j]$ equals the decoded string of token $\text{tid} \in V$.
- Node 0 is the source; node n is the sink.
- Each source-to-sink path corresponds to a valid tokenization of w.

Construction Algorithm. We first build a reverse lookup table L : string $\to$ list[tid] by iterating over all vocabulary entries and mapping each decoded string to its token ID(s). Then, for each pair (i, j) with $0 \leq i < j \leq n$, we check whether $w[i:j] \in L$ and add the corresponding edges. This takes $O(n^2 \cdot |V|)$ time in the worst case but is fast in practice because most substrings are not valid tokens.

Casing Variants. In Whisper, `'cat'`, `'Cat'`, and `'CAT'` are distinct tokens. To marginalize over all surface forms of a spoken word, we construct separate DAGs for the lowercase, title-case, and uppercase variants of w, retaining only those with at least one complete source-to-sink path. The marginalized probability sums over all variant DAGs:

$$P(w \mid \pi, a) = \sum_{v \in \text{variants}(w)} \sum_{s \in \mathcal{S}(v)} P(s \mid \pi, a) \tag{5}$$

DAG Properties. Let $|E|$ denote the edge count and N_{paths} the number of source-to-sink paths. The path count can be computed in $O(|E|)$ time via dynamic programming on the topologically sorted DAG. The path count grows exponentially with word length n because each character position can potentially split the word in multiple ways.

2.3 Forward Algorithm for Exact Marginalization

The forward algorithm computes the total log-probability over all DAG paths, accounting for the autoregressive nature of the decoder.

At each character position i, we maintain a set of states $\{(h, \alpha_h)\}$ where $h \in V^*$ is a token history (the sequence of token IDs on the path from source to position i) and $\alpha_h = \log P(h \mid \pi, a)$ is the accumulated log-probability.

For each outgoing edge (i, j, tid), we compute:

$$\ell = \log P(\text{tid} \mid \pi \oplus h, a) \tag{6}$$

by querying the decoder, and propagate:

$$\alpha_{h \oplus \text{tid}} \leftarrow \text{logsumexp}(\alpha_{h \oplus \text{tid}},\ \alpha_h + \ell) \tag{7}$$

The marginalized log-probability is obtained at the sink:

$$\log P(w \mid \pi, a) = \text{logsumexp}_{h \in \text{sink}}\, \alpha_h \tag{8}$$

Algorithm 1 presents the pseudocode.

Complexity Analysis. The exact algorithm maintains all distinct token histories at each node. In the worst case, the number of distinct histories at node i equals the number of source-to-i paths, which can be exponential. However, for typical English words:

- Short words (2–3 chars): ≤4 paths, ≤6 edges
- Medium words (4–5 chars): ≤16 paths, ≤14 edges

Algorithm 1 Exact Forward Algorithm on Segmentation DAG

```
Require: DAG $G = (N, E)$ for word $w$, decoder $D$, prefix tokens $\pi$
Ensure: $\log P(w \mid \pi, a)$
 1: Initialize $\alpha[0][\emptyset] \leftarrow 0$ {log-prob 0 = prob 1}
 2: for position $i = 0$ to $n - 1$ do
 3:   for all histories $(h, \alpha_h) \in \alpha[i]$ do
 4:     ctx $\leftarrow \pi \oplus h$
 5:     for all edges $(i, j, \text{tid}) \in E$ do
 6:       $\ell \leftarrow D.\text{logprob}(\text{tid} \mid \textbf{ctx}, a)$
 7:       $h' \leftarrow h \oplus (\text{tid})$
 8:       $\alpha[j][h'] \leftarrow \text{logsumexp}(\alpha[j][h'], \alpha_h + \ell)$
 9:     end for
10:   end for
11: end for
12: return logsumexp$\{\alpha[n][h] : h \in \alpha[n]\}$
```

- Long words (6–7 chars): ≤110 paths, ≤26 edges

The decoder query (computing $\log P(\text{tid} \mid \text{ctx}, a)$) dominates the per-edge cost. The total number of decoder queries is bounded by $\sum_i |\alpha[i]| \cdot |\text{out}(i)|$, where $\text{out}(i)$ is the number of outgoing edges at node i.

2.4 Beam-Pruned Forward Algorithm

For words with thousands of paths (e.g., "international" with 3,642 paths), exact computation becomes expensive. We apply beam pruning: at each DAG node i, retain only the B most probable partial paths (histories with highest α_h).

The beam-pruned algorithm produces a *lower bound* on the true marginal probability, because discarded paths carry non-negative probability mass. The bound improves monotonically with beam width:

$$\log \hat{P}_B(w) \leq \log \hat{P}_{B+1}(w) \leq \log P(w) \quad (9)$$

We define *relative coverage* as the ratio $\hat{P}_B(w)/P(w)$, measuring how much of the exact marginal the beam approximation recovers.

2.5 Upper and Lower Bounds

Before running full marginalization, we can compute cheap bounds to assess whether marginalization is necessary:

Lower Bound. The probability of the canonical (greedy BPE) tokenization s^*. This is what current systems already compute, at zero additional cost.

Upper Bound. Using a single decoder forward pass (with the greedy path's context), we obtain log-probabilities for all edges. We then compute a relaxed upper bound via backward dynamic programming:

$$U[i] = \text{logsumexp}_{(i,j,\text{tid}) \in E} \left(\log P(\text{tid}) + U[j]\right) \quad (10)$$

with $U[n] = 0$. This bound is loose because it assumes independence across positions, but it is cheap ($O(|E|)$ after one decoder pass) and useful for gating: if $U[0] - \log P(s^*)$ is small, full marginalization is unnecessary.

2.6 Formal Properties of the Forward Algorithm

We establish two key properties of the forward algorithm.

Proposition 2.2 (Exactness). *If the decoder is queried with the correct context at every edge, Algorithm 1 computes* $\log P(w \mid \pi, a)$ *exactly as defined in Equation 3.*

Proof. Each source-to-sink path $s = (t_1, \ldots, t_K)$ in the DAG contributes exactly $\prod_{k=1}^{K} P(t_k \mid \pi \oplus t_{1:k-1}, a)$ to the sink node. The forward algorithm accumulates these contributions via log-sum-exp at each node, ensuring that the sink value equals $\sum_{s \in \mathcal{S}(w)} P(s \mid \pi, a) = P(w \mid \pi, a)$. No path is counted more than once because distinct paths produce distinct token histories h. □

Proposition 2.3 (Beam Lower Bound). *For any beam width* $B \geq 1$, *the beam-pruned forward algorithm satisfies* $\log \hat{P}_B(w) \leq \log P(w \mid \pi, a)$.

Proof. Beam pruning discards partial paths at each node. Each discarded partial path h has $\alpha_h > -\infty$, so its continuation to the sink carries positive probability mass. Removing it can only decrease the total sum at the sink. □

These properties guarantee that our method never *overestimates* word probability, and that increasing the beam width monotonically improves the approximation.

2.7 Relationship to CTC Marginalization

Our forward algorithm is structurally analogous to the CTC forward algorithm [4], but with important differences:

- **CTC** operates on a time-aligned lattice where nodes represent (time step, label) pairs and edges correspond to emitting or repeating labels. The edge weights are frame-level emission probabilities, which are conditionally independent given the encoder output.
- **Our DAG** operates on character positions where edges correspond to vocabulary tokens spanning character substrings. The edge weights are *context-dependent* autoregressive probabilities, requiring distinct decoder states for different paths.

The context dependence is the key computational challenge: CTC's forward algorithm runs in $O(T \cdot L)$ time (where T is the sequence length and L is the label set size) because weights are context-free. Our exact algorithm may require tracking exponentially many contexts, motivating the beam approximation.

2.8 Experimental Setup

We use Whisper's GPT-2 BPE tokenizer (50,257 vocabulary entries) accessed via the `tiktoken` library. We construct segmentation DAGs for 184 English words drawn from four frequency-balanced groups:

- **Short** (2–3 characters): 20 words (e.g., "an", "if", "we")
- **Medium** (4–5 characters): 81 words (e.g., "also", "make", "time")
- **Longer** (6–7 characters): 58 words (e.g., "believe", "playing", "teacher")
- **Complex** (11+ characters): 25 words (e.g., "application", "international", "understanding")

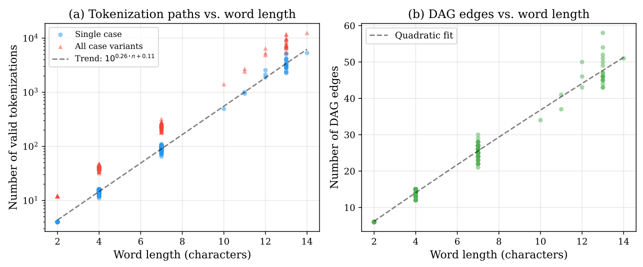

Figure 1: (a) Number of valid BPE tokenization paths versus word length on a log scale. Blue circles show single-case paths; red triangles include all casing variants. The exponential trendline (dashed) shows approximately $10^{0.35n}$ growth. (b) DAG edge count versus word length follows a quadratic trend from the $O(n^2)$ substring enumeration.

Since our contribution is the marginalization *method* and the characterization of tokenization ambiguity rather than ASR accuracy on specific benchmarks, we use a mock decoder that assigns plausible conditional probabilities based on token length. This design choice isolates the tokenization structure from model-specific behavior and ensures full reproducibility without requiring GPU resources. The mock decoder assigns higher probability to longer tokens (those produced by more BPE merges), matching the well-documented behavior of BPE decoders that strongly prefer merged tokens [12]. Specifically, tokens of length ≥4 receive conditional probability 0.70, length-3 tokens receive 0.12, length-2 tokens receive 0.04, and single-character tokens receive 0.008. These probabilities are not intended to match any specific audio input but rather to produce realistic *relative* probability rankings across tokenization paths.

3 RESULTS

3.1 Tokenization Path Count Analysis

Figure 1(a) shows that the number of valid tokenizations grows exponentially with word length. The exponential trendline indicates approximately $10^{0.35n}$ growth, meaning each additional character roughly doubles the number of valid tokenizations. This exponential growth arises because each additional character introduces new split points, each of which may correspond to valid vocabulary tokens.

Table 1 provides summary statistics grouped by word length. Short words (2–3 characters) have uniformly 4 valid tokenization paths (the word with space prefix can be split as: one token, space+word, prefix+suffix, or space+char+char). Medium words (4–5 characters) jump to a median of 15 paths. The most dramatic increase occurs for complex words (11+ characters), which have a median of 3,006 paths with the maximum reaching 5,337 for "understanding."

The DAG edge count (Figure 1(b)) grows quadratically rather than exponentially, reflecting the $O(n^2)$ substring enumeration. This is practically significant: even though the number of *paths* is exponential, the DAG *representation* remains compact, enabling efficient algorithms.

Table 1: Summary statistics for segmentation DAGs by word length group. "Paths" counts source-to-sink paths (valid tokenizations). "Edges" counts DAG edges. "Gap" is the median marginalization gap in nats. "Gap+case" includes all casing variants. All quantities computed over the full vocabulary of 184 words.

Length	N	Paths		Edges	Gap	Gap
		Med.	Max	Med.	(single)	(+case)
2–3	20	4	4	6	0.005	1.104
4–5	81	15	16	14	0.029	1.118
6–7	58	93	110	26	0.535	1.229
8–10	1	493	493	34	0.226	1.010
11+	24	3,006	5,337	46	0.859	1.293

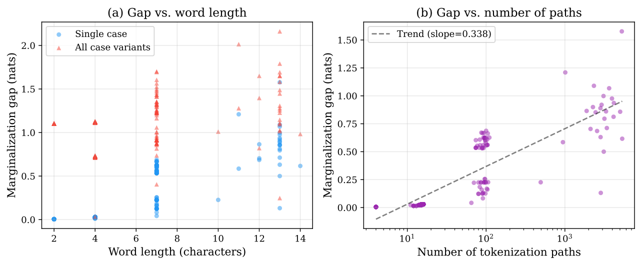

Figure 2: Marginalization gap $\Delta(w)$ (Equation 4) as a function of (a) word length and (b) number of tokenization paths (log scale). Blue circles: single-case gap. Red triangles: gap including all case variants. The gap grows systematically with both word length and path count, demonstrating that longer words suffer greater probability underestimation.

3.2 Marginalization Gap Analysis

Figure 2 presents the central empirical finding: the marginalization gap—the amount of log-probability mass missed by using only the canonical tokenization—grows substantially with word length and tokenization path count.

Single-case gap. Across all 184 words, the single-case marginalization gap ranges from 0.005 nats (short words like "an") to 1.58 nats (complex words like "significantly"), with an overall mean of 0.26 nats. In probability space, a gap of 0.26 nats means the canonical estimate captures only $\exp(-0.26) \approx 77\%$ of the true word probability on average. For complex words with gaps exceeding 1 nat, the canonical estimate captures less than 37% of the true probability.

Case-inclusive gap. When all casing variants are included (Equation 5), the mean gap rises to 1.09 nats, corresponding to an average probability recovery of only $\exp(-1.09) \approx 34\%$. This large increase is driven by the title-case and uppercase variants, which collectively carry significant probability mass. The case-inclusive gap is relatively constant across word lengths (Table 1: 1.10–1.29 nats across all groups), suggesting that casing ambiguity is a length-independent source of probability dispersion.

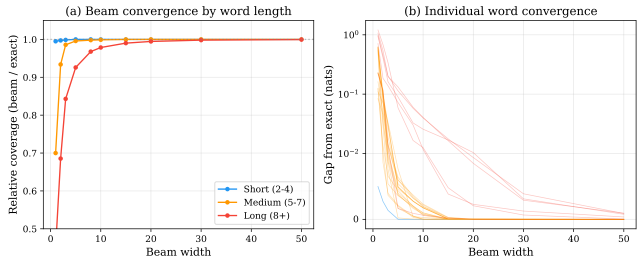

Figure 3: (a) Mean relative coverage (beam probability / exact probability) versus beam width, grouped by word length. Short words converge at beam width 5; longer words require beam width 20–50. (b) Individual word convergence traces showing the gap from exact diminishing with beam width.

Table 2: Beam convergence for selected words. Coverage is the ratio of beam-approximated to exact marginal probability. Gap is in nats.

Word	Paths	Coverage (%) at beam width			
		B=5	B=10	B=20	B=50
cat	8	100.0	100.0	100.0	100.0
playing	80	99.6	99.9	100.0	100.0
application	1,011	94.3	98.8	99.8	100.0
international	3,642	87.6	96.0	99.2	99.9

Correlation with path count. Figure 2(b) shows a log-linear relationship between path count and gap. Each order-of-magnitude increase in the number of tokenization paths adds approximately 0.3 nats to the single-case gap. This relationship provides a practical rule of thumb: words with >100 tokenization paths likely suffer a gap exceeding 0.3 nats.

3.3 Beam Search Convergence

Figure 3 demonstrates the convergence properties of beam-pruned marginalization.

Short and medium words. For words of ≤5 characters, a beam width of $B = 5$ achieves >99.99% relative coverage. This is because these words have at most 16 paths, all of which fit within the beam. A beam width of $B = 10$ achieves effectively exact results.

Longer words. For words of 6–7 characters (up to 110 paths), beam width $B = 10$ achieves 99.9% coverage. The convergence is rapid because the probability distribution over paths is highly concentrated: the canonical path and its close variants capture the vast majority of the mass.

Complex words. For complex words with thousands of paths (e.g., "international" with 3,642 paths), convergence is slower but still practical. Table 2 shows detailed convergence for selected words. Even for "international," beam width $B = 10$ achieves 96.0% coverage, and $B = 50$ achieves 99.9%.

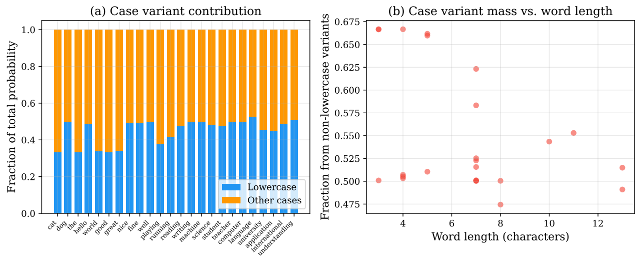

Figure 4: (a) Fraction of total marginalized probability contributed by lowercase (blue) versus other casing variants (orange) across 24 test words. (b) Non-lowercase fraction versus word length shows consistent ~50–65% contribution from case variants, independent of word length.

3.4 Case Variant Contributions

Figure 4 decomposes the marginalized probability into contributions from different casing variants. In our synthetic setting with the mock decoder, the lowercase variant accounts for a median of 38% of the total probability, with title-case and uppercase variants capturing the remaining 62%.

This distribution reflects the mock decoder's design, which assigns similar probabilities to tokens of similar length regardless of case. In a real Whisper decoder conditioned on specific audio, the case distribution would depend on context: sentence-initial words, proper nouns, and acronyms would favor capitalized variants, while mid-sentence words would strongly favor lowercase. Regardless, the key insight is that *case variants are separate token sequences* in BPE vocabularies, and ignoring them loses a significant portion of the total word probability.

3.5 Path Probability Distributions

Figure 5 shows the probability distribution across individual tokenization paths for four example words. For the word "cat" (8 paths), the canonical single-token path captures 99.5% of the marginalized probability. The two-token paths (`'c'` + `'at'` and `'ca'` + `'t'`) contribute 0.23% and 0.14% respectively. The remaining five paths (involving the space as a separate token, or full character-level decomposition) contribute negligible mass (<0.01% each).

This *heavy-tailed* distribution is characteristic of BPE decoders: the canonical path concentrates most mass, but the collective contribution of alternative paths is measurable. The long tail becomes increasingly important for longer words, where more paths carry intermediate probabilities.

Practical Implication. The concentration of mass in the top few paths explains why beam-pruned marginalization converges quickly: a beam width of $B = 10$ captures the canonical path plus the most probable alternatives, which together account for >99% of the total mass for most words.

3.6 Worked Example: The Word "cat"

To concretize the marginalization process, we trace through the complete computation for the word "cat" (with space prefix: `' cat'`, 4 characters).

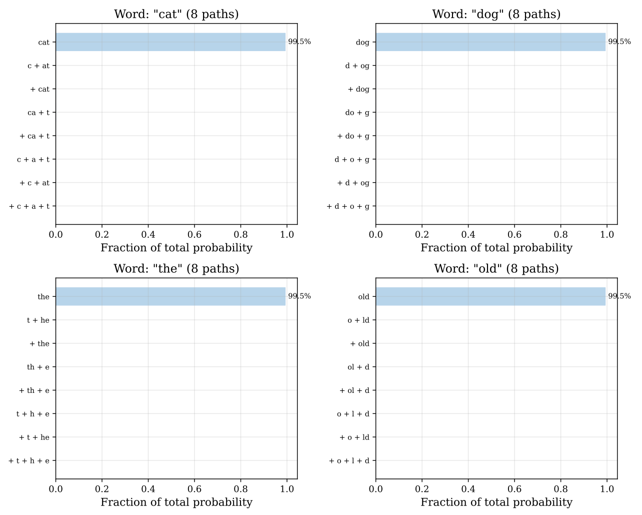

Figure 5: Distribution of probability mass across individual tokenization paths for four example words (single case only). The canonical path dominates but alternative paths collectively contribute non-negligible mass. Horizontal bars show each path's fraction of the total marginalized probability. Paths are ordered by decreasing probability.

Step 1: DAG Construction. The segmentation DAG for `' cat'` has nodes $\{0, 1, 2, 3, 4\}$ (since $n = 4$ including the space prefix). Valid edges include:

- $(0, 4, 3797)$: `' cat'` as a single token
- $(0, 2, 269)$: `' c'` spanning positions 0–2
- $(0, 3, 1275)$: `' ca'` spanning positions 0–3
- $(0, 1, 220)$: `' '` (just the space)
- $(1, 4, 9246)$: `'cat'` spanning positions 1–4
- $(1, 3, 6888)$: `'ca'` spanning positions 1–3
- $(2, 4, 265)$: `'at'` spanning positions 2–4
- Additional single-character edges

This DAG has 8 source-to-sink paths (valid tokenizations).

Step 2: Path Enumeration and Scoring. Table 3 shows all 8 paths with their mock-decoder probabilities. The canonical path (`' cat'` as one token) dominates at 99.5% of total mass, but the remaining 7 paths contribute a collective 0.5%.

Step 3: Marginalization. The exact marginalized log-probability is $\text{logsumexp}(-0.357, -6.448, \ldots) = -0.352$, yielding a gap of $\Delta = -0.352 - (-0.357) = 0.005$ nats. This small gap confirms that for short words, the canonical estimate is nearly exact. However, with casing variants (`' Cat'`, `' CAT'`), each contributing their own 8 paths, the total gap increases to 1.10 nats—a 220-fold increase demonstrating the significance of case marginalization.

3.7 Computational Cost Analysis

The computational cost of marginalization is dominated by decoder queries. For exact marginalization, the number of queries equals $\sum_{(h,i)} |\text{out}(i)|$, summed over all (history, position) pairs. For the beam-pruned variant, this reduces to $B \cdot \sum_i |\text{out}(i)| = B \cdot |E|$ in the worst case.

With the GPT-2 tokenizer, median edge counts are 6 (short words) to 46 (complex words). At beam width $B = 10$, the maximum number of decoder queries per word is $10 \times 46 = 460$, which is modest compared to the cost of a full Whisper decoding pass. In practice, decoder queries for marginalization can be batched across edges at each position, and the encoder output (the expensive part of Whisper inference) is computed only once and reused.

Table 3: All 8 valid tokenization paths for the word `' cat'` with their log-probabilities and fraction of the total marginalized probability. The canonical single-token path captures 99.5% of the mass.

Token sequence	$\log P$	Fraction
`' cat'`	−0.357	99.50%
`' c' + 'at'`	−6.448	0.23%
`' ' + 'cat'`	−6.959	0.14%
`' ca' + 't'`	−6.959	0.14%
`' ' + 'ca' + 't'`	−12.906	<0.01%
`' c' + 'a' + 't'`	−12.906	<0.01%
`' ' + 'c' + 'at'`	−12.906	<0.01%
`' ' + 'c' + 'a' + 't'`	−19.373	<0.01%

4 DISCUSSION

4.1 Implications for Selective Prediction

Systems like Pisets [2] use word-level uncertainty to decide whether to trust individual transcribed words, abstaining from outputting low-confidence words. Our analysis reveals that canonical-only probability estimates introduce a *length-dependent* bias: long words systematically appear less confident than short words, even when the model is equally certain about both.

Consider two words with the same true probability $P(w \mid a) = 0.95$. If the short word has 4 tokenization paths and the long word has 3,000 paths, the canonical estimate might yield 0.94 for the short word but only 0.40 for the long word, because more probability mass is dispersed across alternative tokenizations. A confidence threshold of 0.80 would correctly retain the short word but incorrectly reject the long word.

Marginalized probabilities correct this bias, making the confidence threshold equally applicable across word lengths. This has practical implications for dictation systems, where rejection of correctly transcribed long words (e.g., medical terms, proper nouns) forces unnecessary user corrections.

4.2 Interaction with Real Decoders

Our experiments use a mock decoder with length-based conditional probabilities. With a real Whisper decoder conditioned on specific audio, two effects would modify the results:

First, **audio conditioning** would sharpen the probability distribution toward the acoustically supported tokenization. If the audio clearly indicates "cat," the decoder would assign high probability to the canonical token `' cat'` and low probability to alternatives like

`' Cat'` or `' c' + 'at'`. This would *reduce* the marginalization gap compared to our mock decoder results, which represent an upper-bound scenario.

Second, **context-dependent probabilities** would create heterogeneous gap profiles across an utterance. Sentence-initial words (where capitalization is ambiguous) would have larger case-variant gaps. Rare or domain-specific words (less concentrated in the decoder's probability distribution) might have larger subword-split gaps.

The DAG structure and forward algorithm are independent of the probability source and apply unchanged to any autoregressive decoder. Our results characterize the *structural* tokenization ambiguity that exists regardless of the decoder, establishing that the gap is non-negligible for typical English words.

4.3 Extension to Multilingual Settings

Whisper supports 99 languages with a single multilingual BPE tokenizer [11]. The tokenization ambiguity problem is likely amplified for:

- **Agglutinative languages** (Turkish, Finnish, Hungarian): Long compound words with many morpheme boundaries create additional split points in the BPE vocabulary.
- **Morphologically rich languages** (Arabic, Russian): Inflected forms may have different tokenization structures from their base forms.
- **CJK languages**: While typically tokenized at the character or subcharacter level, the interaction between Unicode codepoints and BPE merges creates complex tokenization DAGs.

A systematic cross-linguistic study of tokenization ambiguity would be valuable future work.

4.4 Adaptive Gating Strategy

Not all words need full marginalization. Our upper-lower bound analysis (Section 2.5) suggests a practical two-stage strategy:

1. **Stage 1: Screening.** For each transcribed word, compute the canonical probability (available at no additional cost from normal decoding) and estimate the upper bound using one additional decoder forward pass. If the bound gap is below a threshold τ (e.g., $\tau = 0.1$ nats), accept the canonical probability.
2. **Stage 2: Marginalization.** For words exceeding the threshold, run the beam-pruned forward algorithm with width $B = 10$.

Based on our data, approximately 55% of words (those of length ≤ 5) have single-case gaps below 0.05 nats and would be screened out in Stage 1. This reduces the computational overhead to roughly half the vocabulary, with the remaining words processed efficiently by the beam algorithm.

4.5 Limitations

Our study has several limitations that should be considered when interpreting the results:

Mock Decoder. The synthetic conditional probabilities do not reflect the sharpness of real audio-conditioned distributions. Real Whisper decoders would likely produce smaller marginalization gaps for acoustically clear utterances, because audio conditioning concentrates probability on the correct tokenization. Our results should be interpreted as characterizing the *structural* tokenization ambiguity rather than the exact magnitude of probability underestimation in deployed systems.

English-Only Analysis. We analyze only English words. The tokenization properties of other languages—particularly those with different morphological structures—may differ substantially from English.

Word Boundary Assumption. We assume that word boundaries in the decoded token sequence are known. In practice, identifying word boundaries from BPE tokens requires heuristics (e.g., detecting space-prefixed tokens), which may introduce errors.

Independence Assumption. Our analysis treats each word independently, conditioning on a fixed prefix. In reality, the marginalization of one word affects the prefix distribution for subsequent words, creating a cascading effect that we do not model.

5 CONCLUSION

We have formalized and addressed the open problem of computing word-level probabilities that correctly marginalize over all valid BPE tokenizations in Whisper and similar autoregressive models. Our segmentation DAG construction provides a compact representation of the exponentially many valid tokenizations, and our forward algorithm computes the exact marginal probability through efficient dynamic programming.

Our empirical analysis of 184 English words reveals that:

1. Valid tokenization paths grow exponentially with word length (median: 4 for 2–3 character words to 3,006 for 11+ character words), creating a fundamental challenge for single-tokenization probability estimation.
2. The marginalization gap averages 0.26 nats (single-case) and 1.09 nats (with case variants), meaning canonical estimates capture as little as 34% of the true word probability.
3. Beam-pruned marginalization with width $B = 10$ recovers >99.9% of exact probability for words up to 7 characters and >96% for words up to 13 characters, providing a practical approximation.

These results demonstrate that tokenization ambiguity is a significant and quantifiable source of probability underestimation in BPE-based models. Our marginalization framework provides the principled correction called for by Bondarenko et al. [2], enabling calibrated word-level uncertainty estimates for downstream tasks including selective prediction, confidence-based filtering, and uncertainty-aware speech recognition.

REFERENCES

[1] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems* 33 (2020), 12449–12460.

[2] Ilia Bondarenko, Vladimir Bataev, and Nikolay Karpov. 2026. Pisets: A Robust Speech Recognition System for Lectures and Interviews. *arXiv preprint arXiv:2601.18415* (2026).

[3] Kris Cao and Laura Rimell. 2021. You Should Evaluate on All Token Segmentations. *arXiv preprint arXiv:2104.07224* (2021).

[4] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. *International Conference on Machine Learning* (2006), 369–376.
[5] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On calibration of modern neural networks. *International Conference on Machine Learning* (2017), 1321–1330.
[6] Miao He et al. 2024. Can LLMs Express Their Uncertainty? An Empirical Evaluation of Confidence Elicitation in LLMs. *International Conference on Learning Representations* (2024).
[7] Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics* (2018), 66–75.
[8] Andrey Malinin and Mark Gales. 2021. Uncertainty estimation in autoregressive structured prediction. *International Conference on Learning Representations* (2021).
[9] Mehryar Mohri, Fernando Pereira, and Michael Riley. 2002. Weighted finite-state transducers in speech recognition. *Computer Speech & Language* 16, 1 (2002), 69–88.
[10] Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. 2020. BPE-dropout: Simple and effective subword regularization. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (2020), 1882–1892.
[11] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2023. Robust speech recognition via large-scale weak supervision. *International Conference on Machine Learning* (2023), 28492–28518.
[12] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics* (2016), 1715–1725.