TOKENPROBE: WHAT TOKEN COUNTS REVEAL ABOUT BLACK-BOX TOKENIZERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Token-count interfaces reveal how many tokens a tokenizer emits, but not necessarily where the token boundaries lie. We construct two byte-level BPE tokenizers that share a reachable vocabulary and agree on the count of every input, yet segment some inputs differently; no number of adaptive count queries can distinguish them. Independent copies yield exponentially many count-equivalent segmentation functions, and a distributional lower bound quantifies the error of any point reconstruction. We then characterize what a transcript does identify: a task output is determined exactly when all consistent tokenizers agree on it. TOKENPROBE operationalizes this characterization through finite-class elimination, driven by standard active-query objectives, and task-specific certification. Exhaustive enumeration verifies the construction on all $88{,}572$ nonempty strings over $\{a, b, c\}$ of length at most ten. In a fully enumerated family of 24 BPE tokenizers with uniformly weighted targets, identifying the count-response class yields perfect held-out counts, yet a fixed representative correctly segments only 95.40% of target–input pairs. Segmentation unanimity certifies 90.64% of pairs without error, and boundary-level certificates still determine 69.22% of internal byte positions on the remaining pairs. These results establish a sharp separation between estimating token counts and recovering a tokenizer's segmentation behavior.

1 INTRODUCTION

A tokenizer determines the units in which a language-model application measures context length, meters usage, and processes text. Its design affects multilingual performance and the cost of processing the same content across languages (Rust et al., 2021; Ahia et al., 2023). When the implementation is hidden, an application may observe only token counts, decoded pieces, or generated text, and these interfaces reveal different information about the underlying computation.

Token counts already support useful forms of extraction. Debenedetti et al. (2024) recover vocabulary entries through a truncation side channel, and Gupta (2026) uses a count endpoint to build a vocabulary-backed local count estimator. The `ctok` project likewise targets count reconstruction and explicitly does not claim to recover the true segmentation (Land, 2026). These practical efforts raise a structural question: can counts determine token boundaries, even when the vocabulary is known?

In general, they cannot. A two-merge BPE construction makes this precise: with only the merges ab and bc, their relative priority determines whether abc is segmented as (ab, c) or (a, bc). The two resulting tokenizers share the same reachable vocabulary and produce identical counts on every byte string, so no number of adaptive count queries can determine their boundaries.

This limit calls for a task-specific notion of recovery. Within a declared hypothesis class, we retain every tokenizer consistent with the query transcript. Agreement among them certifies a downstream output; disagreement exposes the remaining alternatives. Tasks differ in what they require: a count upper bound may suffice for context budgeting, whereas span alignment requires particular token boundaries. The same transcript can therefore settle one task while leaving another unresolved.

We make three contributions. First, we establish a full-domain information limit within rank-based BPE, extend it to exponentially many count-equivalent segmentation functions, and derive a lower bound on the segmentation error of any count-only reconstruction (Section 4). Second, we characterize task identification through the outputs shared by all transcript-consistent tokenizers, which certifies

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

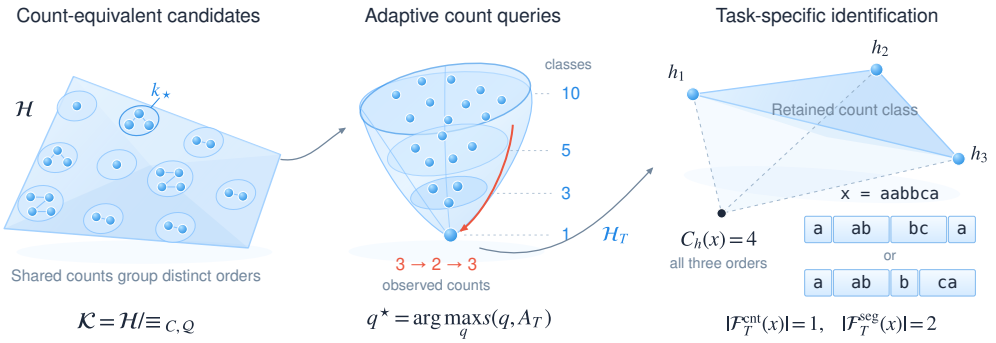


Figure 1: TOKENPROBE refines count-response classes while retaining within-class ambiguity. Left: the 24 priority orders of the merges $\{ab, bc, ca, aa\}$ form ten count-response classes on strings over $\{a, b, c\}$ of lengths one through five. Center: the responses `caaa` $\mapsto 3$, `abca` $\mapsto 2$, and `aabc` $\mapsto 3$ successively leave five, three, and one class. Right: the final class contains three orders, which all predict four tokens for $x =$ `aabbca` but admit two segmentations. Points denote individual orders (left and right) or response classes (center); shaded regions indicate schematic groupings.

counts or context-fit decisions even when boundaries remain unresolved (Section 5). Third, we implement this characterization in TOKENPROBE (Section 6 and Figure 1) and evaluate counts, complete segmentations, and boundary certificates on the same exhaustively enumerated population, measuring the queries needed for class identification and for task resolution (Sections 7 and 8). TOKENPROBE contributes the recovery framework; its query policies use established active-learning objectives, whose costs we compare under matched information.

2 RELATED WORK

Tokenization and extraction. BPE (Sennrich et al., 2016), WordPiece (Song et al., 2021), and unigram tokenization (Kudo, 2018) differ in how they select pieces from a vocabulary, and implementations add normalization and pretokenization (Kudo & Richardson, 2018). Equal vocabularies therefore need not imply equal behavior. Prior studies document the effects of tokenization across languages (Rust et al., 2021; Ahia et al., 2023). Hayase et al. (2024) infer training-data mixtures from an *observed* merge list; we ask what behavioral access identifies before such a list is available.

Debenedetti et al. (2024) use a fixed context window to test singleton membership and recursively compose candidate pieces; their GPT-2 experiment reports approximately 8.2×10^8 queries for complete vocabulary recovery. The practitioner tool `ctoc` combines paired count queries, cross-tokenizer candidates, and frequency-based ordering into a greedy estimator (Gupta, 2026). The `ctok` implementation returns a minimum-cost tiling and explicitly distinguishes it from the hidden tokenizer's actual boundaries (Land, 2026). Count emulation and the practical count–boundary distinction thus predate this work. Our result is a stronger information limit within ordinary rank-based BPE: the same reachable vocabulary and the complete count function are compatible with different segmentation functions.

Active learning and system interfaces. Version spaces and query selection are classical tools in exact learning (Angluin, 1987) and generalized binary search (Mussmann & Liang, 2018). We apply the standard elimination principle to *response classes* rather than to redundant tokenizer configurations, and retain the underlying configurations for a different purpose: certifying downstream outputs. Our query bound follows the classical analysis, whereas the BPE constructions determine which tokenizer behaviors the observations can separate. Model extraction seeks a surrogate with matching predictions (Tramèr et al., 2016; Jagielski et al., 2020); here, the prediction to match may be a count, a segmentation, or an identifier sequence. Side channels that reveal token lengths (Weiss et al., 2024) likewise expose a particular observation, not necessarily the entire tokenizer.

Table 1: Structural information revealed by each interface about complete behavior on the declared domain (not finite-query guarantees). "Pieces + IDs" includes the decoded value of each emitted ID; "raw history" is the merge-priority order.

Interface	Counts	Boundaries	Emitted IDs	Raw history
Decoded model text	not in general	not in general	no	no
Exact input counts	yes	not in general	no	no
Decoded pieces	yes	yes	no	no
Pieces + IDs	yes	yes	yes	no

3 Observation Model and Recovery Targets

Let Σ be the byte alphabet. A deterministic tokenizer h maps an input $x \in \mathcal{X} \subseteq \Sigma^*$ to a finite sequence of token identifiers $h(x) = (t_1, \ldots, t_k)$. Let d_h decode each identifier into bytes, and write

$$S_h(x) = (d_h(t_1), \ldots, d_h(t_k)), \qquad C_h(x) = k. \tag{1}$$

We assume a byte-faithful domain: the pieces of $S_h(x)$ are nonempty and concatenate to x, and no special tokens are inserted, so byte boundaries and decoded-piece sequences represent a segmentation equivalently. Section B treats normalization and template effects, which can violate this assumption.

The principal oracles are the count oracle $\mathcal{O}^h_{\text{cnt}}(x) = C_h(x)$ and the segment oracle $\mathcal{O}^h_{\text{seg}}(x) = S_h(x)$. An identifier oracle returns both pieces and IDs; IDs alone need not reveal the pieces when the decoder is unknown. A decoded-text interface, by contrast, is a conditional distribution $P_{M,h}(y \mid x)$ of a tokenizer and a language model M, not a deterministic function of tokenizer behavior alone. Table 1 summarizes what each interface reveals.

Unlike the raw vocabulary V_h, which may contain never-emitted entries, the reachable vocabulary

$$V_{\text{eff}}(h; \mathcal{X}) = \{s : s \text{ occurs in } S_h(x) \text{ for some } x \in \mathcal{X}\} \tag{2}$$

is determined by behavior, as is the singleton set $V_1(h; \mathcal{X}) = \{s \in \mathcal{X} \setminus \{\epsilon\} : C_h(s) = 1\}$. One count query decides membership in V_1 but neither enumerates an unbounded candidate universe nor recovers unreachable entries. Context-dependent rules can also make V_1 differ from V_{eff}.

Definition 1 (Oracle equivalence and transcript consistency)**.** *For an oracle $\mathcal{O}$ and a query set $\mathcal{Q}$, write $h \equiv_{\mathcal{O},\mathcal{Q}} h'$ if $\mathcal{O}^h(q) = \mathcal{O}^{h'}(q)$ for every $q \in \mathcal{Q}$. Given a declared class $\mathcal{H}$ and a transcript $T = ((q_i, y_i))_{i=1}^t$, the version space is*

$$\mathcal{H}_T = \{h \in \mathcal{H} : \mathcal{O}^h(q_i) = y_i \text{ for every } i\}. \tag{3}$$

We call the equivalence classes of $\equiv_{\mathcal{O},\mathcal{Q}}$ *response classes*. A finite transcript may leave several of them consistent, and the members of a single remaining class may still disagree on a task that $\mathcal{O}$ does not identify; identification statements are therefore relative to the query set.

4 Identifiability: Equal Counts, Different BPE Boundaries

The construction below operates within ordinary rank-based BPE. Encoding starts from bytes, repeatedly applies the highest-priority merge available among adjacent pairs, and stops when no listed merge applies. Occurrences of the same pair are merged from left to right.

Theorem 2 (Full-domain BPE separation)**.** *Let a, b, c be distinct bytes, and let h_L and h_R each have a vocabulary consisting of all single bytes and exactly two additional tokens, ab and bc. Their only merges are ordered $ab \prec bc$ in h_L and $bc \prec ab$ in h_R ($\prec$ denotes higher priority). Both have identical reachable and singleton vocabularies, and for all $x \in \Sigma^*$,*

$$C_{h_L}(x) = C_{h_R}(x) = |x| - N_{ab}(x) - N_{bc}(x) + N_{abc}(x), \tag{4}$$

where $N_u(x)$ counts occurrences of u in x. Nevertheless, $S_{h_L}(abc) = (ab, c)$ and $S_{h_R}(abc) = (a, bc)$.

Proof. Call an occurrence of ab or bc in the input byte sequence a merge opportunity. Because both rules have single-byte operands, no merge creates a new opportunity. Occurrences of the same rule cannot overlap, and the only overlap between different rules is the pair of edges in an abc triple. Such triples do not overlap one another, and neither of their edges overlaps any other opportunity. Hence, under either order, every isolated opportunity contributes one merge and every abc triple contributes exactly one. With $N_{ab} + N_{bc}$ opportunities and N_{abc} overlapping pairs, this gives Equation (4). Both length-two tokens are emitted on their own strings and every byte is reachable, so the vocabularies coincide. Finally, the two priority orders select different edges of abc. □

The two orders thus remain indistinguishable under repeated, arbitrarily long, and adaptive count queries. The construction also admits independent copies, so the number of count-equivalent segmentation functions can grow exponentially.

Corollary 3 (Independent unresolved merge orders; proof in Section A.1)**.** *For any m with $3m \le |\Sigma|$, there exist 2^m distinct BPE segmentation functions with the same reachable vocabulary and the same count function on Σ^*. A single segment query distinguishes all members of this family.*

Each ambiguous decision is local and has a short witness under segment access; count access discards precisely the information needed to resolve it. The obstacle is therefore an uninformative interface rather than a hard search problem: no count-query budget can recover the missing priorities.

Corollary 4 (Segmentation error from indistinguishable counts)**.** *Let $\mathcal{D}$ be a distribution over inputs and let $p = \Pr_{x\sim\mathcal{D}}[S_{h_L}(x) \neq S_{h_R}(x)]$. Any, possibly adaptive and randomized, procedure that issues only count queries and outputs a point segmentation predictor $\widehat{S}$ has expected error $\Pr_{x\sim\mathcal{D}}[\widehat{S}(x) \neq S_h(x)] \ge p/2$ for some target $h \in \{h_L, h_R\}$.*

Proof. Because $C_{h_L} = C_{h_R}$, the count transcript, and hence $\widehat{S}$, has the same distribution under both targets, even for adaptive randomized queries. On each disagreement input, a point prediction agrees with at most one target, so the two expected errors sum to at least p and their maximum is at least $p/2$. □

For the two-tokenizer class, choosing either tokenizer with probability $1/2$ attains the bound. The disagreement event is exactly the occurrence of abc (Section A.7), so its probability governs both the cost of a forced point answer and the fraction of inputs whose segmentation remains unresolved. The construction thus rules out universal segmentation recovery from counts within rank-based BPE, even with a known reachable vocabulary. Recovering a particular tokenizer depends on the structural restrictions placed on the candidate class: more queries can separate distinct count-response classes, but distinguishing the two orders here requires a richer observation.

Other interfaces leave different ambiguities: unrestricted decoded-text generation need not identify the tokenizer (Section A.4), finitely many count queries cannot recover the count function of an unbounded dictionary class (Section A.5), and even complete segmentation behavior need not identify commuting merge priorities (Section A.6).

5 Task-Specific Recovery Certificates

Assume that the target belongs to $\mathcal{H}$ and observations are exact. A task assigns each tokenizer h an output $f_x(h)$ on input x, such as the count $C_h(x)$, the segmentation $S_h(x)$, or the context-fit decision $\mathbf{1}\{C_h(x) \le L\}$ for a token budget L. The identified output set is

$$\mathcal{F}_T(x) = \{f_x(h) : h \in \mathcal{H}_T\}. \tag{5}$$

Proposition 5 (Task identification from a transcript)**.** *For a realizable transcript, so that $\mathcal{H}_T \neq \emptyset$, some output is correct for every consistent target if and only if $|\mathcal{F}_T(x)| = 1$.*

Unanimity therefore licenses a task answer; otherwise, any point answer is wrong for at least one consistent target. With finitely many distinct outputs, $m = |\mathcal{F}_T(x)|$, the minimax error of a randomized answer on a fixed input is $1 - 1/m$ (Section A.2). A selective predictor returns the unique output when $m = 1$ and abstains otherwise; on an evaluation set E, its certified coverage is $|E|^{-1} \sum_{x\in E} \mathbf{1}\{|\mathcal{F}_T(x)| = 1\}$. These guarantees quantify over all consistent hypotheses; unanimity within a truncated candidate pool has only an empirical interpretation (Section A.9).

Context and boundary tasks. The count envelope

$$[\underline{C}_T(x), \overline{C}_T(x)] = \left[\min_{h \in \mathcal{H}_T} C_h(x),\ \max_{h \in \mathcal{H}_T} C_h(x)\right] \tag{6}$$

is the smallest interval containing all consistent counts. An input certainly fits budget L if $\overline{C}_T(x) \le L$, certainly exceeds it if $\underline{C}_T(x) > L$, and is unresolved otherwise. Individual byte boundaries can likewise be settled without identifying the whole segmentation. Because refinement only shrinks identified sets, a task may be answered before the response class is identified, whereas disagreement within the target's class cannot be removed by further queries from $\mathcal{Q}$ (Proposition 9).

Partial boundary information. A unique segmentation is a demanding output: disagreement at a single position forces abstention on the entire string. Applications that need only particular token boundaries can use finer tasks. Let $B_h(x) \subseteq \{1, \dots, |x| - 1\}$ be the set of internal byte offsets at which $S_h(x)$ places a boundary, and define

$$B_T^-(x) = \bigcap_{h \in \mathcal{H}_T} B_h(x), \qquad B_T^+(x) = \bigcup_{h \in \mathcal{H}_T} B_h(x). \tag{7}$$

An offset in $B_T^-(x)$ is a boundary under every consistent tokenizer, an offset outside $B_T^+(x)$ is a boundary under none, and only offsets in $B_T^+(x) \setminus B_T^-(x)$ remain unresolved. These coordinate-wise conclusions are exact. The full set of consistent segmentations carries additional joint constraints: an arbitrary labeling of the unresolved offsets need not be realized by any consistent tokenizer (Section 8).

6 TokenProbe: Candidate Refinement and Task Prediction

TokenProbe maintains a set of executable candidate tokenizers to determine which task outputs remain compatible with the observations. Given a declared finite class, it separates two operations: selecting probes among distinct response signatures, and evaluating tasks over all remaining *raw hypotheses*, i.e., the individual members of $\mathcal{H}$.

Finite-class formulation. Fix a finite set $\mathcal{Q}$ of allowable queries. Let $\mathcal{K} = \mathcal{H}/\equiv_{\mathcal{O},\mathcal{Q}}$ be the set of K response classes, each identified by its signature $(\mathcal{O}^h(q))_{q \in \mathcal{Q}}$, and let $A_T \subseteq \mathcal{K}$ contain the classes that survive transcript T. Collapsing hypotheses into classes serves only query selection; the full version space $\mathcal{H}_T$ remains available for Equation (5). For $A \subseteq \mathcal{K}$, writing $\mathcal{O}^k(q)$ for the common response of class k, define the split score

$$s(q, A) = 1 - \max_y \frac{|\{k \in A : \mathcal{O}^k(q) = y\}|}{|A|}. \tag{8}$$

Each class receives equal weight, so duplicating a configuration within a class does not change the query policy. Maximizing s minimizes the size of the largest cell that can survive the query.

This worst-cell criterion is the standard generalized-binary-search objective (Mussmann & Liang, 2018), applied to distinct response signatures; Section A.3 states its contraction bound. It identifies what the oracle can distinguish while preserving within-class ambiguity for task prediction. With cached responses, scoring all queries costs $O(|\mathcal{Q}||A_T|)$ operations per step; constructing and simulating candidates incurs a separate, local cost (Section C).

Executable candidates and prediction. Each candidate specifies its preprocessing, token bytes, merge operands, and a total merge-priority order. The implementation builds a response matrix $R_{h,q} = \mathcal{O}^h(q)$, groups identical rows, and retains the constituent hypotheses of every group. In the controlled evaluation, the builder enumerates all $4!$ orders of four declared merges, and the query generator enumerates all strings of lengths one through five; no completion, sampling, or pruning is used. Candidate simulation is local computation, and each requested target response costs one oracle call.

At prediction time, TokenProbe evaluates every retained hypothesis on the new input and returns either a singleton task output or the alternative outputs together with a disagreeing witness pair. When

TOKENPROBE **with task certificates**

1. **Input:** exact oracle $\mathcal{O}$; finite class $\mathcal{H}$ and query set $\mathcal{Q}$; query budget B; downstream tasks f_x.
2. Under the declared observation model, compute each hypothesis's response signature on $\mathcal{Q}$ and group equal signatures into $\mathcal{K}$. Set $T = \emptyset$, $\mathcal{H}_T = \mathcal{H}$, and $A_T = \mathcal{K}$.
3. While $|A_T| > 1$ and $|T| < B$: select $q \in \arg\max_{q' \in \mathcal{Q}} s(q', A_T)$, breaking ties by length and then byte order; observe $y = \mathcal{O}(q)$ and append (q, y) to T; restrict $\mathcal{H}_T$ and A_T to consistent raw hypotheses and classes. If none remain, report a *class/observation mismatch*.
4. **Return:** $\mathcal{H}_T$, the surviving classes, and $\mathcal{F}_T(x)$ for each requested task. Certify singleton output sets; otherwise, return the set or abstain.

Algorithm 1: The implemented finite-class procedure. Query selection operates on distinct response signatures; certification uses all consistent raw hypotheses.

a point prediction is required, it is taken from the lowest-index retained candidate, independently of held-out labels. Query selection and point selection serve different purposes: a good probe shrinks the set of response classes, whereas an arbitrary representative need not resolve segmentation within the final class.

7 CONTROLLED COMPUTATIONAL EVALUATION

We ask how the observation interface affects identification, how efficiently a response class can be found, and which downstream outputs that class determines. The controlled BPE family permits exhaustive enumeration of hypotheses, query responses, and held-out certificates; its vocabulary and merge candidates are fixed throughout (Table 2). The supplement includes the encoder, query policies, decision trees, and raw traces.

Table 2: Complete specification of the known-family experiment. All policies receive the same prior information, and target responses are the only oracle observations.

Component	Specification
Candidate family	All $4! = 24$ priority orders of the known merges ab, bc, ca, aa; all candidates share byte operands and vocabulary.
Encoding	Identity preprocessing; no special tokens; the highest-priority available pair is merged, with occurrences merged from left to right.
Queries / held-out inputs	All strings over $\{a, b, c\}$ of lengths 1–5 / 6–8 (363 and 9,477 strings).
Filtering / prediction	Retain every transcript-consistent order, without pruning or completion; point outputs use the lowest candidate ID.
Evaluation / cost	Uniform over the same 24 targets for every oracle and policy; one call per requested response; candidate simulation is local.

Full-domain counterexample: finite verification. We enumerate all $88{,}572$ nonempty strings over $\{a, b, c\}$ of length at most ten and compare a rank-priority BPE encoder with an independent implementation of the closed-form count in Equation (4). The all-input conclusion follows from Theorem 2; the enumeration checks the implementation and quantifies boundary disagreement on an explicit finite population.

The two encoders never disagree on a count, and the closed form holds on every string. Their segmentations nevertheless differ on $23{,}068$ strings (26.04%), and on 27.59% of the strings of length ten (Figure 2, left). Every count transcript is thus identical under both encoders, while a substantial fraction of segmentations differs.

Known-class identification. We enumerate all 24 priority orders of $\{ab, bc, ca, aa\}$ and all 363 queries over $\{a, b, c\}$ of lengths one through five. In addition to the count and segment oracles, a singleton oracle returns $\mathbf{1}\{C_h(q) = 1\}$. With classes defined by complete response vectors on these queries, the singleton, count, and segment oracles induce one, ten, and eighteen classes.

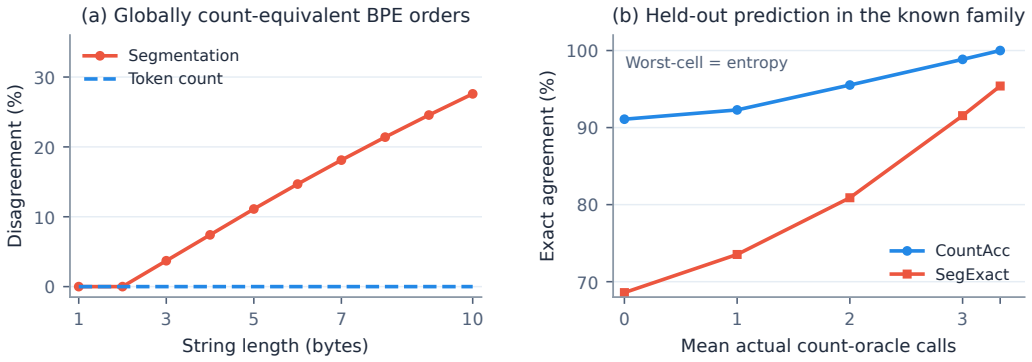


Figure 2: Controlled count–boundary separation. Left: segmentation disagreement between the rank-swapped tokenizers of Theorem 2, by input length. Right: held-out prediction accuracy versus mean actual count calls at call caps $0, 1, 2, 3, 4$; the last cap uses $10/3$ calls on average. All 24 raw targets receive equal weight, and worst-cell and entropy selection give identical curves.

Joint count and boundary evaluation. After class identification, we retain all raw hypotheses in the identified class and evaluate every string of lengths six through eight (9,477 inputs). Point predictions use the lowest-index retained hypothesis, fixed independently of test outputs. Every oracle is evaluated under the same uniform distribution over the 24 raw targets, and each target is evaluated on all held-out strings. We report count accuracy (CountAcc), exact segmentation accuracy (SegExact), micro-averaged F1 over internal byte boundaries, and the joint fraction of pairs with a correct count but an incorrect segmentation (C✓/S×).

Table 3: Joint held-out evaluation (%). Coverage is the fraction of target–input pairs with task-specific unanimity; accepted predictions have zero error on their certified task. All rows use the same 24-target distribution, and F1 pools boundary counts before taking the ratio.

					Coverage	
Observation	CountAcc	SegExact	Bound. F1	C✓/S×	Count	Seg.
Singleton	91.09	68.58	91.10	22.50	80.23	41.25
Count	100.00	95.40	98.98	4.60	100.00	90.64
Segments	100.00	100.00	100.00	0.00	100.00	100.00

Identifying the count class determines every held-out count, yet the point tokenizer missegments 4.60% of target–input pairs (Table 3). Count unanimity accepts these errors; segmentation unanimity excludes them, accepting 90.64% of pairs with no segmentation error. The least-resolved count class certifies only 80.51% of segmentations. CountAcc exceeds SegExact at every call cap (Figure 2, right). The gap stems from which output is identified: it persists after count-class identification and is unaffected by the choice between the two query policies.

Dependence on input length. Table 4 fixes the target distribution and stratifies the held-out inputs by byte length. The count-correct segmentation error rises from 3.53% at length six to 4.85% at length eight, while count errors remain zero. Boundary F1 stays near 99%: a small fraction of misplaced offsets can coexist with a larger whole-string error. Over the full test domain, the byte-length median, P90, and maximum are all eight, and the corresponding target-token lengths are six, seven, and eight. These are explicit finite domains, not estimates of accuracy on natural-language documents.

Query efficiency under matched information. All policies receive the same candidate predictions and query set, and query only strings whose responses differ among the surviving classes. We compare worst-cell splitting, entropy maximization, shortest informative queries, and uniformly sampled informative queries. Random selection uses twenty seeds; deterministic ties are broken by

Table 4: Length-resolved count-oracle evaluation after class identification (%). All count predictions are exact, so MAE and under- and overestimation rates are zero at every length. Targets receive equal weight; token-length quantiles are computed over all target–input pairs at each byte length.

Bytes	Inputs	Token median/P90/max	SegExact	Bound. F1	C✓/S×	Seg. cov.
6	729	4 / 5 / 6	96.47	98.97	3.53	92.79
7	2,187	5 / 6 / 7	95.78	98.98	4.22	91.42
8	6,561	6 / 7 / 8	95.15	98.98	4.85	90.14

Table 5: Queries needed to identify the target's response class. Means weight the 24 raw targets equally; worst cases range over all targets and, for random selection, all twenty seeds. The mean-optimal and depth-optimal tree searches attain the same displayed cost pair. All policies use the same candidate predictions and informative queries.

	Count (10 classes)		Segments (18 classes)	
Policy	Mean	Worst	Mean	Worst
Exact tree (reference)	3.29	4	3.08	4
Shortest informative	3.38	4	4.42	5
Random informative	3.52	6	4.36	7
Entropy (Mussmann & Liang, 2018)	3.33	4	3.13	4
TOKENPROBE (worst-cell)	3.33	4	3.25	4

increasing length and then byte order. Every policy stops when one response class remains, and all runs identify the correct class.

As references, dynamic programming computes exact optimal decision trees for the mean query cost under the same uniform raw-target distribution and, separately, for the maximum path length (Section D). These offline references use the same information and allowable queries; constructing candidate classes is excluded from the oracle-cost metric.

The exact optimal mean costs are $79/24 \approx 3.29$ queries for counts and $37/12 \approx 3.08$ for segments. Entropy and worst-cell selection both use $10/3 \approx 3.33$ count queries on average. With segment access, entropy uses $25/8 = 3.125$ calls and worst-cell uses $13/4 = 3.25$. Both policies attain the optimal maximum depth of four, but neither attains the optimal mean: their scores weight distinct response classes equally, whereas the evaluation and the optimal reference weight raw targets equally. The count–boundary separation therefore does not rely on any advantage of worst-cell selection. Class identification here is relative to the declared 363-query set; exact held-out counts are an exhaustive observation on lengths six through eight, not a proof of full-domain equivalence among the 24 orders.

8 FROM CLASS IDENTIFICATION TO TASK RESOLUTION

Class identification is a natural stopping criterion for learning observable behavior, but applications request particular outputs. Using the same held-out population, we examine two consequences of this distinction: a task may be resolved before the class is identified, and an identified class may still leave part of a segmentation unresolved.

One count, two boundary configurations. The run in Figure 1 queries `caaa`, `abca`, and `aabc` and receives counts 3, 2, and 3, shrinking the ten count classes to five, three, and finally one. The surviving class contains three orders:

$$ab \prec bc \prec aa \prec ca, \qquad ab \prec aa \prec bc \prec ca, \qquad ab \prec aa \prec ca \prec bc.$$

On $x = \texttt{aabbca}$, they produce the segmentations (a, ab, bc, a) and (a, ab, b, ca). Both contain four tokens, so the context-fit decision for x is settled for every budget. The internal boundary set, however, is either $\{1, 3, 5\}$ or $\{1, 3, 4\}$: offsets 1 and 3 are certain boundaries, offset 2 is certainly not a boundary, and offsets 4 and 5 remain unresolved. Exactly one of the latter two is a boundary; marking both, or neither, contradicts every consistent candidate. The output set preserves this dependence, which separate per-offset labels would lose.

This class certifies $8{,}744$ of the $9{,}477$ held-out segmentations (92.27%) and leaves 733 unresolved. Six of the ten count classes leave some held-out segmentations unresolved. Class size alone does not predict this uncertainty: two classes with two orders each settle every held-out segmentation, whereas one three-order class settles only 80.51%. What matters is disagreement on the application inputs, not the number of retained parameterizations.

Stopping for a requested output. Fix an input x and a task f_x. Along a count-query trajectory, the task can stop at the first transcript T with $|\mathcal{F}_T(x)| = 1$, which may occur after zero queries if the supplied candidates already agree. If the final response class disagrees on f_x, no further query from $\mathcal{Q}$ can settle it. More generally, a task is resolvable for every consistent target with queries from $\mathcal{Q}$ exactly when it is constant within each surviving $\mathcal{Q}$-equivalence class (Proposition 9). This criterion separates uncertainty that more allowable queries can remove from uncertainty that requires queries outside $\mathcal{Q}$, a stronger interface, or further assumptions.

We replay the worst-cell count trajectory for every target–input pair, stopping at the first task agreement or at class identification. The probes have lengths one through five and the requested inputs lengths six through eight; restricting probes to $\mathcal{Q}$ tests transfer from a fixed probe domain and does not model a limitation of real count APIs. Before any query, the candidates agree on 80.23% of counts and 41.25% of segmentations. Table 6 reports a fresh run for each requested input, not amortized recovery across inputs.

Table 6: Task calls under two access settings, averaged over the 24 targets and 9,477 inputs. Upper block: the fixed count-query trajectory in $\mathcal{Q}$, with unresolved tasks run until class identification. Lower block: querying the requested input x is also permitted. All resolved outputs are exact.

Requested output	Resolved (%)	Calls, resolved only	Calls, all pairs
Probes restricted to $\mathcal{Q}$			
Exact count	100.00	0.453	0.453
Complete segmentation	90.64	1.366	1.531
Count-response class	100.00	3.333	3.333
Direct access to the requested input			
Count: unanimity or query x	100.00	0.198	0.198

Direct-query reference. If x itself can be queried, we return the unanimous candidate count when one exists and query x otherwise. Only $1{,}874$ of the $9{,}477$ inputs require a call, for a mean cost of $1{,}874/9{,}477 \approx 0.198$, below the 0.453 of the restricted trajectory. For exact single-input count prediction with unit call costs, this cost is optimal, since an initially ambiguous count requires at least one observation. The restricted-probe result thus quantifies early stopping along a given policy, not a saving over direct count access. Querying x reveals its count, but not necessarily its boundaries.

Information within unresolved segmentations. Among the $21{,}284$ target–input pairs whose final class does not determine the complete segmentation, $98{,}270$ of the $141{,}962$ internal byte positions (69.22%, pooled over positions) still have a determined boundary label: $75{,}459$ are certain boundaries and $22{,}811$ are certain non-boundaries. Whole-string abstention therefore discards information that a boundary-level task can use, whereas the remaining offsets require an additional observation or assumption.

9 Discussion and Conclusion

Token counts identify a coarser object than token boundaries. Our two-rule BPE construction establishes this separation with a known reachable vocabulary and unrestricted count queries. In the controlled family, count-class identification yields perfect held-out counts, yet a fixed representative missegments 4.60% of target–input pairs. Task-specific agreement certifies determined outputs, including partial boundary information. The information limit is independent of query policy. The controlled results concern the declared finite family; broader recovery requires explicit candidate and interface models (Sections B and C).

AI USE STATEMENT

In this work, we used generative AI tools (large language models) solely for language polishing, including improving grammar, clarity, and readability. These tools were not used to generate substantive content or to perform any research tasks, including developing methods or theoretical frameworks, formulating claims or hypotheses, constructing proofs, designing experiments, generating or processing data, analyzing or interpreting results, writing code, searching the literature, translating text, or preparing figures. All AI-assisted revisions were reviewed by the authors to ensure that they preserved the intended meaning and technical accuracy. We take full responsibility for the final content of this work, including all claims, results, and references.

REFERENCES

Orevaoghene Ahia, Sachin Kumar, Hila Gonen, Jungo Kasai, David Mortensen, Noah 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*, pp. 9904–9923. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.614. URL https://aclanthology.org/2023.emnlp-main.614/.

Dana Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75(2):87–106, 1987.

Edoardo Debenedetti, Giorgio Severi, Nicholas Carlini, Christopher A. Choquette-Choo, Matthew Jagielski, Milad Nasr, Eric Wallace, and Florian Tramèr. Privacy side channels in machine learning systems. In *33rd USENIX Security Symposium*, 2024.

Rohan Gupta. Reverse engineering Claude's token counter. Author technical report, February 2026. URL https://grohan.co/2026/02/10/ctoc/. Published February 10, 2026.

Jonathan Hayase, Alisa Liu, Yejin Choi, Sewoong Oh, and Noah A. Smith. Data mixture inference attack: BPE tokenizers reveal training data compositions. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-0285.

Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. High accuracy and high fidelity extraction of neural networks. In *29th USENIX Security Symposium*, 2020.

Taku Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 2018.

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*, 2018.

Sander Land. ctok: Claude's tokenizer, offline, kinda. Software repository, 2026. URL https://github.com/sanderland/ctok. Accessed September 26, 2026.

Stephen Mussmann and Percy Liang. Generalized binary search for split-neighborly problems. In *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics*, 2018.

Phillip Rust, Jonas Pfeiffer, Ivan Vulić, 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 and the 11th International Joint Conference on Natural Language Processing*, 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*, 2016.

Xinying Song, Alex Salcianu, Yang Song, Dave Dopson, and Denny Zhou. Fast WordPiece tokenization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 2089–2103, 2021. doi: 10.18653/v1/2021.emnlp-main.160. URL https://aclanthology.org/2021.emnlp-main.160/.

Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction APIs. In *25th USENIX Security Symposium*, 2016.

Roy Weiss, Daniel Ayzenshteyn, Guy Amit, and Yisroel Mirsky. What was your prompt? a remote keylogging attack on AI assistants. *arXiv preprint arXiv:2403.09751*, 2024.

A Additional Proofs and Assumptions

A.1 Proof of Corollary 3

Proof. Choose m disjoint byte triples (a_i, b_i, c_i), include the merges $a_i b_i$ and $b_i c_i$ for each triple, and choose their relative priority independently across triples. Rules from different triples share no bytes, so they can neither overlap nor create opportunities for one another. Applying Theorem 2 to each triple, every choice has the count function

$$C(x) = |x| - \sum_{i=1}^{m} \left[N_{a_i b_i}(x) + N_{b_i c_i}(x) - N_{a_i b_i c_i}(x) \right]. \tag{9}$$

Every added token is reachable on its own string, so all 2^m tokenizers share the same reachable vocabulary. On $x = a_1 b_1 c_1 \cdots a_m b_m c_m$, every member emits $2m$ tokens, and its segmentation reveals the relative priority within each triple, hence all m choices. In particular, the 2^m segmentation functions are distinct. □

A.2 Proof of Proposition 5 and the Minimax Error

For $m = |\mathcal{F}_T(x)| < \infty$, the fixed-input statement extends to randomized answer rules A:

$$\inf_{A} \sup_{h \in \mathcal{H}_T} \Pr[A(T, x) \neq f_x(h)] = 1 - \frac{1}{m}. \tag{10}$$

Proof. If $|\mathcal{F}_T(x)| = 1$, the unique output is correct for every consistent target. If at least two outputs remain, each is contradicted by some consistent target that produces the same transcript. For randomized answers, some of the m distinct values receives probability at most $1/m$, and a consistent target attains that value, so the worst-case error is at least $1 - 1/m$. Conversely, sampling uniformly from the m values errs with probability exactly $1 - 1/m$ against every target. □

A.3 Finite-Class Query Bound

For the response classes and split score in Section 6, define

$$\beta = \min_{\substack{A \subseteq \mathcal{K} \\ |A| > 1}} \max_{q \in \mathcal{Q}} s(q, A). \tag{11}$$

Proposition 6 (Finite-class query cost)**.** *Suppose that $h^\star \in \mathcal{H}$, observations are exact, and $K \geq 2$. Greedy maximization of $s(q, A_T)$ identifies the target's $\mathcal{Q}$-equivalence class within*

$$\min \left\{ K - 1, \left\lceil \frac{\log K}{-\log(1 - \beta)} \right\rceil \right\} \tag{12}$$

queries. If each query has at most $R \geq 2$ outcomes on $\mathcal{H}$, every exact policy needs at least $\lceil \log_R K \rceil$ queries in the worst case. For $K = 1$, class identification requires no queries.

Proof. Distinct signatures differ on some query, so every A with $|A| > 1$ admits a query with $s(q, A) > 0$, and each greedy step eliminates at least one class. Moreover, the surviving cell has size at most $(1 - \beta)|A|$ at every step, so $|A_t| \leq (1 - \beta)^t K$, which yields the logarithmic bound. For the lower bound, an R-ary decision tree of depth t has at most R^t leaves, whereas exact identification requires at least K. □

A.4 Decoded-Text Access and the Composite Model

Proposition 7 (Unrestricted-model equivalence)**.** *Let h and h' be lossless tokenizers on a common domain of finite input and output strings. For any conditional distribution $P(y \mid x)$ induced by a model paired with h, there exists an unrestricted token-level autoregressive model paired with h' that induces the same distribution over decoded outputs.*

Proof. Losslessness makes $y \mapsto h'(y)$ injective. Push $P(\cdot \mid x)$ forward to a distribution over the finite token sequences $h'(y)$, appending an end-of-sequence event. The chain rule factors this distribution into autoregressive conditionals. The replacement model uses these conditionals given the losslessly encoded input $h'(x)$, from which x is recoverable, and behaves arbitrarily on unreachable prefixes. Decoding its outputs recovers P. □

The replacement model need not belong to a fixed architecture or parameter budget, so the proposition does not rule out tokenizer inference under additional architectural assumptions. Timing, context truncation, and token accounting also provide observations beyond the decoded-output distribution.

A.5 No Finite-Query Universal Dictionary Recovery

Proposition 8. *When dictionary entries may be arbitrarily long, no procedure that terminates after finitely many exact count queries on every lossless dictionary tokenizer can always recover the tokenizer's total count function on Σ^*.*

Proof. Run the procedure against the tokenizer that emits individual bytes, and let ℓ exceed the lengths of the finitely many queries in the resulting transcript. Choose z with $|z| > \max\{\ell, 1\}$, and consider the dictionary tokenizer with all bytes plus the token z, applied greedily to occurrences of z. It agrees with the byte tokenizer on every queried string, since none contains z; by induction over the queries, adaptivity yields the same transcript. Yet the count of z is one under the modified tokenizer and $|z|$ under the byte tokenizer, so no single returned function is correct for both. □

This rules out deterministic exact guarantees. Randomization does not help if the answer must always be correct: against the byte tokenizer, almost-sure termination implies that, for some finite length bound, a positive-probability set of runs makes only shorter queries, and adding a longer token leaves those runs indistinguishable. The statement is not a lower bound for distributional approximation over a fixed test distribution.

A.6 Unreachable Entries and Commuting Merge Priorities

If a raw vocabulary entry is never emitted on the declared domain, any representation change that removes it while preserving the tokenizer's function is invisible to functional oracles. The qualifier matters: in a concrete implementation, removing a rule can change downstream segmentation.

For a constructive example, take four distinct bytes a, b, c, d and only the merges ab and cd. No two opportunities for these rules overlap, and neither rule creates an opportunity for the other, so swapping their ranks preserves the segmentation of every byte string. Even a pieces-and-identifiers oracle with identical ID assignments therefore cannot recover their rank order. This differs from Theorem 2, where swapping the ranks changes segmentation but preserves counts.

A.7 Where the Separated BPE Tokenizers Disagree

In Theorem 2, all merge opportunities outside abc triples are shared, whereas the two tokenizers select different edges in every such triple. Hence

$$S_{h_L}(x) \neq S_{h_R}(x) \quad \iff \quad N_{abc}(x) > 0. \tag{13}$$

This characterization provides an independent check on the disagreement labels in the exhaustive experiment. Occurrences are counted by scanning all byte offsets, including overlapping starts; since its bytes are distinct, the pattern abc cannot overlap itself.

A.8 Rank and Identifier Conventions

The construction in Corollary 3 concerns merge priorities. Assigning fixed identifiers to the shared decoded pieces makes those identifiers independent of the priority choices. Implementations that encode merge rank in the token identifier impose an additional representation convention; if that convention is known and identifiers are exposed, rank information may leak through the identifier channel even when counts conceal it. Our count and decoded-segment results require no such convention.

A.9 Certification under Incomplete Class Search

Suppose that an implementation retains only $\widetilde{\mathcal{H}}_T \subsetneq \mathcal{H}_T$. Its output sets may then be strict subsets of those in Equation (5), and a singleton under the approximation need not be a singleton under the full class. Replacing exhaustive enumeration by beam search is therefore an implementation change that weakens the guarantee, not merely an acceleration. Conservative outer approximations of task-output sets can be retained instead, but their validity must be established separately.

A.10 Partial Boundaries and Task-Specific Stopping

The certain and possible boundary sets in Equation (7) retain coordinate-wise information, whereas the full segmentation set also records which combinations of boundaries are jointly realizable.

Refinement and stopping. As exact observations accumulate, $\mathcal{H}_{T'} \subseteq \mathcal{H}_T$ for every extension T' of T. Consequently, $\mathcal{F}_{T'}(x) \subseteq \mathcal{F}_T(x)$, the count envelope contracts, and certified predictions remain certified. A task can therefore be completed before the target's response class has been identified. Conversely, some uncertainty may persist even after every allowable query has been answered.

Proposition 9 (When further queries can settle a task)**.** *Fix a finite query set $\mathcal{Q}$ and a realizable transcript. An adaptive procedure using queries from $\mathcal{Q}$ can determine $f_x(h)$ for every $h \in \mathcal{H}_T$ if and only if f_x is constant on each equivalence class of $\mathcal{H}_T$ under $\equiv_{\mathcal{O},\mathcal{Q}}$.*

Proof. If two consistent targets in the same class have different task outputs, every allowable query gives the same answer under both, so no subsequent transcript can resolve their disagreement. Conversely, querying all of the finite set $\mathcal{Q}$ determines the target's class, and constancy within the class then determines the task output. □

This yields two stopping rules with different roles. Unanimity of $\mathcal{F}_T(x)$ suffices to answer the task immediately, whereas disagreement within the identified response class means that further queries from $\mathcal{Q}$ have no value for that task. For multiple requested inputs, the same conditions apply to the joint output vector.

B Interfaces, Normalization, and Template Effects

Normalization. Under a lossy normalization N, a single-token response on s shows only that $N(s)$ is emitted as one token; it does not establish that the original bytes of s form a vocabulary item. Likewise, equal counts for two Unicode variants do not prove that the variants normalize identically. Claims about the decoder require decoded pieces, documentation, or an explicit family assumption. If byte-faithfulness cannot be established, one should evaluate normalized-domain behavior or treat template-conditioned counts as the target.

Template-conditioned counts. The observation model used to construct response signatures must match the service interface. For example, a fixed wrapper can expose

$$C_{\mathrm{api}}(x) = C_h(\mathsf{pre} \| x \| \mathsf{suf}) + \kappa, \tag{14}$$

where pre and suf are template strings and κ is a fixed special-token contribution. Subtracting $C_{\mathrm{api}}(\epsilon)$ removes an additive offset only when the wrapper and the input tokenize independently. Consider a tokenizer with the single merge $u + v \mapsto uv$, a hidden prefix $\mathsf{pre} = u$, an empty suffix, and $\kappa = 0$. Then $C_{\mathrm{api}}(v) = C_{\mathrm{api}}(\epsilon) = 1$, whereas $C_h(v) = 1$, so empty-input subtraction would assign zero tokens to a nonempty input.

For accounting and context planning, the wrapped count is often itself the relevant target, and the same version-space construction applies by simulating Equation (14) under each hypothesis. Recovering the unwrapped tokenizer instead requires known template boundaries or enough structure to infer their interactions. In particular, the singleton test must use an oracle for the intended domain: any offset correction is part of the observation model and must be justified before candidate labels are used to eliminate hypotheses.

Noisy probes. For at most B logical binary queries, suppose that repeated observations have conditionally independent errors, each with probability at most $\eta < 1/2$. Repeating each query an odd number

$$r \geq \frac{2}{(1-2\eta)^2} \log \frac{B}{\delta} \tag{15}$$

of times and taking the majority reproduces the exact transcript with probability at least $1 - \delta$. The result follows from a conditional Hoeffding bound and a union bound over queries, and it covers adaptive queries. It does not cover systematic rounding, unknown template interactions, or drifting timing measurements; these mechanisms require their own observation models and are not used in the reported experiments.

C Candidate Representation and Computation

Response matrix and certificates. The implementation caches the response matrix $R_{h,q} = \mathcal{O}^h(q)$, groups identical rows, and stores the raw hypotheses belonging to each group. This separates two operations that are easily conflated: a query partitions rows by their responses in one column, whereas a certificate evaluates all raw hypotheses in the retained rows on the requested input. Equal row weights prevent a class with many equivalent parameterizations from dominating probe selection; they are a design choice for query efficiency and do not change the logical content of the certificates.

For a new application input, a representative tokenizer can supply a tentative output, which is then compared with the outputs of the remaining class members. Agreement permits immediate certification; disagreement returns a witness pair and the corresponding alternative outputs. Repeated application inputs can be cached independently of the probe transcript, and certificates remain valid after refinement because consistency sets only shrink. These operations require no additional oracle calls; their only cost is local simulation of the retained hypotheses.

Candidate discovery and larger classes. For vocabulary discovery, a declared candidate set S can combine corpus substrings, public vocabulary candidates, and family-specific composition rules. An exact count query labels each $s \in S$ by whether $C_h(s) = 1$; coverage is measured within S, and the resulting membership labels then constrain the hypothesis class. For larger classes, overlap strings provide potential conflict probes, but a probe is useful only if the simulated oracle outputs actually differ. Failure of a bounded search to find a witness is reported as search-limited uncertainty; it establishes equivalence only when the finite allowable query set has been exhausted or a separate equivalence argument applies.

D Controlled Evaluation Details

The BPE engine starts from the three ASCII bytes, scans the current adjacent pairs, selects the lowest-ranked (highest-priority) available pair, merges all nonoverlapping occurrences of that pair from left to right, and repeats. No normalization, unknown-token rule, prefix marker, or special token is present.

The identification class contains all $4! = 24$ priority orders of the four byte-pair rules ab, bc, ca, aa, and every target has the same vocabulary. The query set consists of all $3 + 9 + 27 + 81 + 243 = 363$ nonempty strings over $\{a, b, c\}$ of length at most five, ordered by length and lexicographically within each length. Each oracle induces its own partition of the same 24 configurations. Equivalence is determined by exact equality of the entire 363-response signature, not by hashes alone; hashes in the artifact serve only to verify trace integrity.

Greedy worst-cell selection follows Equation (8). The entropy policy maximizes Shannon entropy over the same equally weighted surviving signatures. Random informative selection samples uniformly from the currently informative unused queries with seeds $0, \ldots, 19$, and shortest informative selection takes the earliest such query in the fixed order. The supplement also records fixed shortest-first and random-order controls, both of which may query uninformative strings; any query a baseline issues, informative or not, costs one call. All policies stop as soon as one signature remains. Main-text means give each of the 24 raw targets equal weight, whereas distinct signatures are equally weighted in the greedy query objectives; the scoring weights and the evaluation distribution are separate choices.

The supplied traces record the query strings, answers, and number of surviving classes after each step.

Exact decision-tree references. For a surviving set S of response classes, let $\mathcal{P}_q(S)$ be the nonempty response cells induced by query q, and let $W(S)$ be the number of raw targets in those classes. Queries with only one cell are excluded. The minimum weighted total leaf depth Λ and the minimum maximum depth D satisfy

$$\begin{aligned} \Lambda(S) &= W(S) + \min_q \sum_{U \in \mathcal{P}_q(S)} \Lambda(U), \\ D(S) &= 1 + \min_q \max_{U \in \mathcal{P}_q(S)} D(U), \end{aligned} \tag{16}$$

with $\Lambda(S) = D(S) = 0$ when $|S| = 1$. Because raw targets are equally weighted, $\Lambda(S)/W(S)$ is the optimal mean. Each class contributes its number of raw members to $W(S)$, and these weights are inherited by every child. Both recurrences terminate because an informative query yields strict subsets. Since repeating a previous query is uninformative, no separate query-history state is needed. Queries that induce the same partition are interchangeable for these objectives; the implementation keeps the earliest such query in the declared order.

Mean optimization breaks ties by maximum depth and then by query order. To obtain the smallest mean among depth-optimal trees, a second solver minimizes total leaf depth subject to a depth budget $d = D(S)$, passing budget $d - 1$ to every child. This allows a child to use extra depth when doing so reduces the mean without increasing the root's worst case; minimizing each child's depth independently would not, in general, optimize this secondary objective. In this instance, both solvers return the same cost pair for each oracle: $(79/24, 4)$ for counts, $(37/12, 4)$ for segments, and $(0, 0)$ for the singleton predicate. The artifact includes both reconstructed weighted trees, a traversal for every raw target, and the dynamic-programming values; class-uniform references are provided in separate files. These are exact references for a finite family, not a scalable alternative for unknown tokenizer configurations.

Certification uses all raw hypotheses in the retained signature, including those collapsed for query selection. Test inputs have lengths six through eight and are therefore disjoint from $\mathcal{Q}$. Coverage is computed exactly over this finite population, so no confidence intervals are needed. Variation across random-policy seeds reflects query order, not sampling uncertainty about the test population. Source, query, and result metadata are included with the regeneration script.

D.1 Joint Count, Segmentation, and Length Evaluation

The joint analysis uses the same 24 candidates, 363 probes, and disjoint 9,477-string test domain. Hypothesis IDs follow the fixed permutation order of (ab, bc, ca, aa), and the point predictor is the retained candidate with the smallest ID; this choice uses no test labels. Every main-text oracle comparison, budget curve, and query-cost comparison uses equal weights over the same 24 raw targets. For completeness, the artifact also provides oracle-class-uniform and count-class-uniform summaries, labeled as alternative aggregations; all reported results use the common target distribution.

Let $\widehat{B}(x)$ and $B_h(x)$ be the predicted and true sets of internal byte offsets, excluding the mandatory start and end positions. Boundary F1 is $2\sum |\widehat{B} \cap B_h| / \sum (|\widehat{B}| + |B_h|)$, with sums over all target–input pairs, which receive equal weight. SegExact requires equality of the full decoded-piece sequence. The joint error rate is $\mathbb{E}[\mathbf{1}\{\widehat{C} = C_h, \widehat{S} \neq S_h\}]$, not an error rate conditioned on correct counts. Acceptance and its conditional error are computed separately for count and segmentation unanimity. Because the realized target is retained at every step, all accepted outputs are correct for their respective tasks.

Over the full test domain, the byte-length median, P90, and maximum are all eight, and the target-token counterparts are six, seven, and eight. For call caps $0, 1, 2, 3, 4$, the mean actual calls are $0, 1, 2, 3, 10/3$. The curve includes zero-call predictions from the supplied candidate family and charges each requested exact count once. Hypothesis construction is local computation, and this experiment has no vocabulary-discovery stage. The complete enumeration involves no input-sampling uncertainty; it does not, however, estimate accuracy on longer natural-language inputs or on a hidden production tokenizer.

D.2 Task Resolution and Boundary Accounting

For each raw target h and held-out input x, the task-resolution script replays the stored worst-cell count trajectory. It includes the initial version space at step zero and filters all 24 orders after each observed response. The stopping time is the first step with a singleton task-output set; if no such step exists, the run consumes the entire trajectory and is marked unresolved. Conditional mean calls average over resolved pairs only, whereas unconditional mean calls also charge unresolved pairs for their complete trajectories. All pairs receive equal weight. This procedure evaluates an early-stopping rule along a fixed policy, not a task-optimal query policy.

The separate direct-query reference computes initial count agreement over the same 24 candidates for every held-out input. If the candidates agree, it returns the shared value without a call; otherwise, it queries that input once. Its mean cost is therefore the fraction of inputs with initial disagreement, and all its counts are exact. Unlike the trajectory analysis, this reference permits queries outside $\mathcal{Q}$. Its agreement counts match the zero-call entries of the recorded task analysis.

At the final class, the script intersects and unions the internal boundary sets as in Equation (7). For unresolved whole-string segmentations, the denominator is the number of internal byte positions summed over target–input pairs. A position is resolved if every candidate assigns it the same binary boundary label, so the numerator includes both certain boundaries and certain non-boundaries. The per-target records also report these two components separately. The witness in Section 8 is recorded in `heldout_witnesses.json`.