Estimating near-verbatim extraction risk in language models with decoding-constrained beam search

A. Feder Cooper* Stanford & Yale
Mark A. Lemley Stanford
Christopher De Sa Cornell
Lea Duesterwald Cornell
Allison Casasola Stanford
Jamie Hayes Google DeepMind
Katherine Lee
Daniel E. Ho Stanford
Percy Liang Stanford

arXiv:2603.24917v3 [cs.CL] 31 Jul 2026

Abstract

Recent work shows that standard greedy-decoding extraction methods for quantifying memorization in LLMs miss how extraction risk varies across sequences. Probabilistic extraction—computing the probability of generating a target suffix given a prefix under a decoding scheme—addresses this, but is tractable only for verbatim memorization, missing near-verbatim instances that pose similar privacy and copyright risks. Quantifying near-verbatim extraction risk is expensive: the set of near-verbatim suffixes is combinatorially large, and reliable Monte Carlo (MC) estimation can require $\approx$100,000 samples per sequence. To mitigate this cost, we introduce decoding-constrained beam search, which yields deterministic lower bounds on near-verbatim extraction risk at a cost comparable to $\approx$20 MC samples per sequence. Across experiments, our approach surfaces information invisible to verbatim methods: many more extractable sequences, substantially larger per-sequence extraction mass, and patterns in how near-verbatim extraction risk manifests across model sizes and types of text.

1 Introduction

LLMs memorize portions of their training data, and it is sometimes possible to extract those memorized sequences in outputs (Carlini et al., 2021; 2023). Understanding memorization is important for assessing model quality, as it may indicate overfitting rather than generalization. Extraction of memorized training data also raises other concerns: it can expose private information (Brown et al., 2022; Nasr et al., 2023; Nolte et al., 2025) or reproduce copyrighted material present in the training data (Lee et al., 2023b; Cooper & Grimmelmann, 2024; Ahmed et al., 2026).

Most prior work measures extraction through *verbatim* comparisons: prompt an LLM with a training-data prefix, use greedy decoding to produce an output, and check whether that output *exactly* matches the corresponding training-data suffix (Lee et al., 2022). This approach undercounts memorization in two important ways. First, requiring verbatim matches misses *near-verbatim* memorization (Ippolito et al., 2023). Second, greedy decoding produces a single deterministic output, so it cannot capture how extraction risk varies across sequences. Recent work on *probabilistic* extraction—computing the probability of generating the target suffix under a decoding scheme—addresses the latter, but remains tractable only for verbatim memorization (Hayes et al. (2025b), Section 2). Computing *near-verbatim probabilistic* extraction is far more expensive, as it requires estimating probability mass over a combinatorially large set of near-verbatim suffixes.

To address this, we introduce a tractable method for estimating near-verbatim extraction risk: **decoding-constrained beam search**, a family of algorithms that produce a provably

*Corresponding author: a.feder.cooper@yale.edu

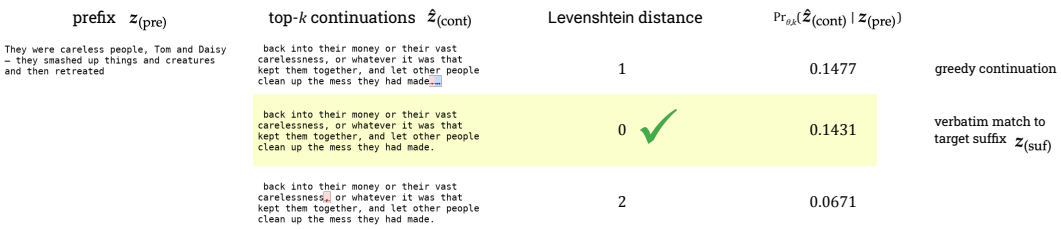


Figure 1: **Probabilistic extraction.** For θ = LLAMA 1 13B and a training sequence z from *The Great Gatsby*, we show prefix $z_{(\text{pre})} \coloneqq z_{1:a}$ and 3 continuations $\hat{z}_{(\text{cont})} \coloneqq \hat{z}_{a+1:a+T}$ under ϕ = top-k = 40 (Equation 1) with conditional probabilities $\Pr_{\theta,k}(\hat{z}_{(\text{cont})} \mid z_{(\text{pre})})$ (Equation 2). We diff each $\hat{z}_{(\text{cont})}$ with the target suffix $z_{(\text{suf})} \coloneqq z_{a+1:a+T}$ (character space: blue additions, red deletions) and compute the Levenshtein distance (token space, Equation 5). We highlight verbatim extraction (i.e., $\hat{z}_{(\text{cont})} = z_{(\text{suf})}$, $0.1431 \geq \tau_{\min} = 0.001$), which is *not* the greedy continuation (top row). All three $\hat{z}_{(\text{cont})}$ are *near-verbatim* matches to $z_{(\text{suf})}$ (Section 3).

correct deterministic lower bound on a sequence z's near-verbatim extraction probability $p^{\text{dist}}_{z,\varepsilon}$ for a given distance metric dist and tolerance ε (Section 4). While reliable Monte Carlo (MC) estimation can require $\approx$100,000 samples per sequence (Section 3), our experiments with top-k-constrained beam search (k-CBS) yield practically useful lower bounds at a cost comparable to $\approx$20 MC samples per sequence. We provide variants that integrate distance-based ε-viability pruning into the search, often yielding tighter lower bounds on $p^{\text{dist}}_{z,\varepsilon}$ at reduced runtime cost. We evaluate k-CBS across multiple model families, model sizes, and datasets (Section 5 & Appendix F). Our experiments show that near-verbatim probabilistic extraction reveals information invisible to verbatim methods: far more extractable sequences (e.g., 2.57% of sequences for OLMO 2 32B on Wikipedia, compared to 1.42% for verbatim probabilistic extraction); substantially larger per-sequence extraction risk (e.g., in some cases, from 0 verbatim risk to over 0.85 near-verbatim risk); and patterns in how near-verbatim extraction risk manifests across model sizes and types of text.

2 Background and related work

Let $\mathbb{V}$ denote the token **vocabulary** for a **large language model (LLM)** (e.g., $|\mathbb{V}| = 32{,}000$ for LLAMA 1). An LLM with **weights** θ maps a sequence of tokens $\boldsymbol{b} = (b_1, \ldots, b_n) \in \mathbb{V}^n$ to a logit vector $\boldsymbol{y} \in \mathbb{R}^{|\mathbb{V}|}$: $\theta : \mathbb{V}^n \to \mathbb{R}^{|\mathbb{V}|}$. A **decoding scheme** ϕ defines how logits are mapped to a next-token sampling distribution, $\text{softmax}_\phi : \mathbb{R}^{|\mathbb{V}|} \to \mathcal{P}(\mathbb{V})$, where $\mathcal{P}(\mathbb{V})$ is the set of probability distributions over $\mathbb{V}$. Together, (θ, ϕ) define an **autoregressive generation process**: given a **prompt** $\boldsymbol{b}_{1:i}$, at each step $t > i$ we compute $\boldsymbol{y}_t = \theta(\boldsymbol{b}_{1:t-1})$ and sample $b_t \sim \text{softmax}_\phi(\boldsymbol{y}_t)$; for j steps, this produces a **continuation** $\hat{\boldsymbol{b}}_{i+1:i+j}$.

We focus on **top-k decoding** (Fan et al., 2018). At step t, let $\mathbb{V}_k \subseteq \mathbb{V}$ be the set of the $k > 0$ tokens with the largest logits in $\boldsymbol{y}_t$. Top-k zeros out all but the k highest logits and renormalizes:

$$\Pr_{\theta,k}(b_t = v \mid \boldsymbol{b}_{1:t-1}) = \frac{\exp(\boldsymbol{y}_t[v])}{\sum_{u \in \mathbb{V}_k} \exp(\boldsymbol{y}_t[u])} \mathbf{1}\{v \in \mathbb{V}_k\}. \tag{1}$$

Verbatim memorization and extraction. Memorization refers to the encoding of specific training sequences in a model's weights (Feldman, 2020; Yeom et al., 2018; Shokri et al., 2017), such that the learned distribution assigns very high probability to them. For LLMs, this can make **extraction** possible: memorized sequences can sometimes be reproduced in outputs (Carlini et al., 2021). The most common way to assess memorization in LLMs is **discoverable extraction** (**greedy extraction**): split a training sequence z into an a-length **prefix** $z_{1:a}$ and a T-length **target suffix** $z_{a+1:a+T}$ (i.e., $z = z_{1:a} \parallel z_{a+1:a+T}$), prompt the LLM with $z_{1:a}$, use **greedy decoding** as a cheap proxy for deterministically generating a high-probability T-length continuation $\hat{z}_{a+1:a+T}$, and deem extraction successful if $\hat{z}_{a+1:a+T}$ *exactly* matches $z_{a+1:a+T}$ (i.e., $\mathbf{1}[\hat{z}_{a+1:a+T} = z_{a+1:a+T}]$) (Lee et al., 2022; Carlini et al., 2023).

Hayes et al. (2025b) show that greedy extraction greatly underestimates memorization, as memorized continuations that have very high generation probability may not be reachable

through greedy (top-1) decoding. They suggest a definition for **probabilistic extraction**, which accounts for the non-determinism of more typical decoding schemes ϕ (e.g., top-k with $k>1$) by computing the probability of generating the exact target $\boldsymbol{z}_{a+1:a+T}$ given $\boldsymbol{z}_{1:a}$:

$$p_{\boldsymbol{z}} \triangleq \Pr_{\theta,\phi}\left(\boldsymbol{z}_{a+1:a+T} \mid \boldsymbol{z}_{1:a}\right) = \prod_{t=a+1}^{a+T} \Pr_{\theta,\phi}\left(z_t \mid \boldsymbol{z}_{1:t-1}\right) = \exp\left(\sum_{t=a+1}^{a+T} \log \Pr_{\theta,\phi}\left(z_t \mid \boldsymbol{z}_{1:t-1}\right)\right). \quad (2)$$

For a threshold $\tau_{\text{min}} \in (0,1]$, verbatim probabilistic extraction is successful when the generation probability of $\boldsymbol{z}_{a+1:a+T}$ given $\boldsymbol{z}_{1:a}$ is at least τ_{min}, i.e., $\mathbf{1}[\, p_{\boldsymbol{z}} \geq \tau_{\text{min}} \,]$. Hayes et al. (2025b) justify studying probabilistic extraction using top-k decoding with $k=40$, for which Cooper et al. (2025) validate a conservative $\tau_{\text{min}}=0.001$. We examine the same settings here.

Figure 1 provides a concrete example. For LLAMA 1 13B and a training sequence from *The Great Gatsby*, we illustrate the generation of three continuations under top-k decoding (Equation 1) for the same prefix. Verbatim probabilistic extraction succeeds (middle row): $p_{\boldsymbol{z}}=0.1431>\tau_{\text{min}}$. Verbatim greedy extraction fails, as the greedy continuation (top row) is not a verbatim match to the target suffix. Probabilistic extraction also surfaces nuanced information about **extraction risk**. While greedy decoding flattens extraction success to a binary, yes-or-no outcome, probabilistic extraction under ϕ returns an extraction probability $p_{\boldsymbol{z}} \in [0,1]$. For training sequences A and B, if $p_{\boldsymbol{z}^{(A)}} > p_{\boldsymbol{z}^{(B)}}$, then in expectation LLM θ will leak A more often than B. In Figure 1, $p_{\boldsymbol{z}}=0.1431$ means that LLAMA 1 13B leaks the exact target suffix about 1 out of every 7 times it is prompted with the prefix. Finally, probabilistic extraction offers these benefits at comparable cost to greedy decoding. Verbatim extraction probability $p_{\boldsymbol{z}}$ (Equation 2) can be computed exactly via **teacher-forced inference**—a single forward pass over $\boldsymbol{z}$, with no sampling required (Cooper et al. (2025), Appendix C.3).

3 The challenge of quantifying near-verbatim extraction risk

For efficiency reasons, much prior work studies *verbatim* extraction (Lee et al., 2022; Nasr et al., 2023) (Section 2). However, as intuited in Figure 1, verbatim extraction underestimates memorization. For greedy extraction, accounting for *near-verbatim* continuations of a prefix reveals substantially more memorized sequences (Ippolito et al., 2023). (In Figure 1, near-verbatim greedy extraction succeeds.) For probabilistic extraction, near-verbatim continuations would similarly provide richer information about extraction risk. (In Figure 1, the probabilities of all three continuations contribute to near-verbatim extraction risk: verbatim risk is 0.1431, but near-verbatim risk computed over all near-verbatim suffixes is at least $0.1477+0.1431+0.0671=0.3579$.) Yet, as Hayes et al. (2025b) note, straightforward approaches for computing near-verbatim probabilistic extraction—e.g., Monte Carlo (MC) sampling—are computationally expensive. In this section, we explain this cost, motivating our algorithm for efficiently estimating near-verbatim probabilistic extraction (Section 4).

Near-verbatim extraction. To simplify indexing, denote a-length prefix $\boldsymbol{z}_{(\text{pre})} \coloneqq \boldsymbol{z}_{1:a}$, T-length target suffix $\boldsymbol{z}_{(\text{suf})} \coloneqq \boldsymbol{z}_{a+1:a+T}$, and T-length generated continuation $\hat{\boldsymbol{z}}_{(\text{cont})} \coloneqq \hat{\boldsymbol{z}}_{a+1:a+T}$. For both greedy and probabilistic extraction, we accommodate near-verbatim cases by introducing a distance metric dist and tolerance ε. For greedy extraction, success counts $\mathbf{1}[\, \mathsf{dist}(\hat{\boldsymbol{z}}_{(\text{cont})}, \boldsymbol{z}_{(\text{suf})}) \leq \varepsilon \,]$. (When $\varepsilon=0$, this reduces to verbatim extraction.)

For verbatim probabilistic extraction, success depends on the probability of the exact target, $p_{\boldsymbol{z}}$ (Equation 2). For the near-verbatim case, many continuations—e.g., all three in Figure 1—may qualify as extraction success, any of which may reasonably be sampled with a non-deterministic decoding scheme ϕ. We therefore define the near-verbatim extraction probability as the aggregate mass of the set of ε-viable continuations. For target suffix $\boldsymbol{z}_{(\text{suf})}$, denote the **$\varepsilon$-ball** of near-verbatim matches for distance dist as

$$\mathbb{B}_{\varepsilon}^{\mathsf{dist}}(\boldsymbol{z}_{(\text{suf})}) \triangleq \{\, \boldsymbol{v} \in \mathbb{V}^T : \mathsf{dist}(\boldsymbol{v}, \boldsymbol{z}_{(\text{suf})}) \leq \varepsilon \,\}. \quad (3)$$

The **near-verbatim extraction risk** $p_{\boldsymbol{z},\varepsilon}^{\mathsf{dist}}$ is the total mass on $\mathbb{B}_{\varepsilon}^{\mathsf{dist}}(\boldsymbol{z}_{(\text{suf})})$,

$$p_{\boldsymbol{z},\varepsilon}^{\mathsf{dist}} \triangleq \sum_{\boldsymbol{v} \in \mathbb{B}_{\varepsilon}^{\mathsf{dist}}(\boldsymbol{z}_{(\text{suf})})} \Pr_{\theta,\phi}\left(\boldsymbol{v} \mid \boldsymbol{z}_{(\text{pre})}\right), \quad (4)$$

and we count success with $\mathbf{1}\left[\, p^{\mathsf{dist}}_{z,\varepsilon} \geq \tau_{\min} \,\right]$. $\mathbb{B}^{\mathsf{dist}}_0(z_{(\text{suf})})$ contains only $z_{(\text{suf})}$, so $p^{\mathsf{dist}}_{z,0} = p_z$, i.e., reduces to the verbatim case.)

We consider two standard token-level distances $\text{dist} : \mathbb{V}^T \times \mathbb{V}^T \to \{0, 1, \ldots, T\}$—the **Hamming** and **Levenshtein** distances. For $\boldsymbol{b}, \boldsymbol{c} \in \mathbb{V}^T$,

$$\mathsf{Ham}(\boldsymbol{b}, \boldsymbol{c}) \triangleq \textstyle\sum_t \mathbf{1}[b_t \neq c_t]; \qquad \mathsf{Lev}(\boldsymbol{b}, \boldsymbol{c}) \triangleq \min\{d : \boldsymbol{b} \xrightarrow{d} \boldsymbol{c}\}. \tag{5}$$

Hamming (Ham) counts positional mismatches (unit-cost token substitutions). Levenshtein (Lev) is more general: it is the minimum number of (unit cost) substitution, insertion, or deletion edits d required to transform one sequence into the other. For near-verbatim greedy extraction, these metrics are cheap to compute: just as with verbatim extraction, there is only one suffix to evaluate—the greedy continuation. But to produce more nuanced information about extraction risk, the set of near-verbatim suffixes in $\mathbb{B}^{\mathsf{dist}}_\varepsilon(z_{(\text{suf})})$ can be enormous, which can make computing $p^{\mathsf{dist}}_{z,\varepsilon}$ enormously expensive.

Exactly computing $p^{\mathsf{dist}}_{z,\varepsilon}$ is intractable. For Hamming, the size of the ε-ball is $|\mathbb{B}^{\mathsf{Ham}}_\varepsilon(z_{(\text{suf})})| = \sum_{r=0}^{\varepsilon} \binom{T}{r} (|\mathbb{V}| - 1)^r$, where $\binom{T}{r}$ chooses the r substitution positions and $(|\mathbb{V}| - 1)^r$ counts possible substitutions at those positions. For $|\mathbb{V}| = 32{,}000$, $T = 50$, and just $\varepsilon \leq 2$, $|\mathbb{B}^{\mathsf{Ham}}_2(z_{(\text{suf})})| > 10^{12}$. For Levenshtein, the ε-ball is even larger (Appendix B.3.2). It is intractable to teacher force that many suffixes, and it should also be unnecessary. Many suffixes in the ε-ball will have 0 or extremely low probability (e.g., consider a continuation $\hat{z}_{(\text{cont})}$ that substitutes the token ␣jazz for ␣the in the target suffix $z_{(\text{suf})}$).

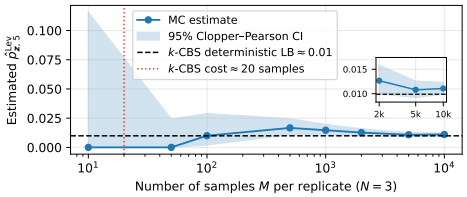


Figure 2: **Monte Carlo (MC) estimation.** For Levenshtein distance ≤ 5 ($\hat{p}^{\mathsf{Lev}}_{z,5}$), we plot convergence for a single sequence z from *The Great Gatsby* for LLAMA 2 7B, showing the pooled MC estimate with a 95% confidence interval over 3 replicates. Our algorithm (k-CBS, Section 4) produces a deterministic, provably correct lower bound (LB) of ≈ 0.01. It captures 89.4% of the mean MC estimate at $M = 10^4$ samples, at a cost of ≈ 20 MC samples (Appendix C.3)—a budget at which MC produces no hits.

Estimating $p^{\mathsf{dist}}_{z,\varepsilon}$ with MC is expensive. Alternatively, one could prompt M times with the prefix and estimate $p^{\mathsf{dist}}_{z,\varepsilon}$ as the proportion of sampled continuations (with decoding scheme ϕ) lying within the ε-ball of the target suffix. This is statistically unbiased, but also infeasible at scale. The probability of never hitting the ε-ball in M i.i.d. samples is $\Pr\left(\text{miss } \mathbb{B}^{\mathsf{dist}}_\varepsilon(z_{(\text{suf})})\right) = (1 - p^{\mathsf{dist}}_{z,\varepsilon})^M$. To guarantee a miss probability of at most δ requires $(1 - p^{\mathsf{dist}}_{z,\varepsilon})^M \leq \delta \iff M \geq \ln(1/\delta)/(-\ln(1 - p^{\mathsf{dist}}_{z,\varepsilon}))$. Even for a modest miss tolerance of $\delta = 0.05$, merely *detecting* an ε-ball with a relatively high mass of $p^{\mathsf{dist}}_{z,\varepsilon} = 0.01$ requires $M \approx 300$ samples. Further, detection only guarantees one hit; it does not guarantee that the MC estimate of $p^{\mathsf{dist}}_{z,\varepsilon}$ is *reliable*. Figure 2 shows thousands of samples are necessary for an accurate estimate for $p^{\mathsf{Lev}}_{z,5} \approx 0.01$; $M \approx 100{,}000$ samples would be needed for reliable estimation of $p^{\mathsf{dist}}_{z,\varepsilon} = \tau_{\min} = 0.001$ (Appendix C.1). Compared to teacher-forced inference for verbatim p_z, MC for $p^{\mathsf{dist}}_{z,\varepsilon}$ is roughly $M\times$ more expensive (Appendix C.3).

4 Decoding-constrained beam search

We show that near-verbatim extraction risk can be estimated at much lower cost than MC sampling, making it feasible to evaluate at scale. The overarching intuition is simple: for a given prefix, beam search is a decoding algorithm that tends to find a set of continuations that are high probability under the model θ (Section 4.1); memorized suffixes are especially high probability under the model (Section 2); and so, when a sequence is memorized, beam search should return a set of continuations that are near-verbatim matches to the memorized suffix—i.e., continuations in the ε-ball $\mathbb{B}^{\mathsf{dist}}_\varepsilon(z_{(\text{suf})})$. Following this intuition, we propose algorithms (Sections 4.2 & 4.3) that modify beam search to incorporate decoding scheme

Algorithm 1: Top-k Constrained Beam Search (k-CBS)

Input: LLM θ; prefix $\boldsymbol{z}_{(\text{pre})}$; suffix length T; beam width B; top-k parameter k; EOS token id; optional $\tau_{\min} > 0$

Output: Set of up to $B \cdot k$ pairs $(\hat{\boldsymbol{z}},\ \log p)$, where $\hat{\boldsymbol{z}} = \boldsymbol{z}_{(\text{pre})} \,\|\, \hat{\boldsymbol{z}}_{(\text{cont})}$ and $\log p = \log \Pr_{\theta,\phi}(\hat{\boldsymbol{z}}_{(\text{cont})} \mid \boldsymbol{z}_{(\text{pre})})$; or $\emptyset$ if early termination

Compute $\boldsymbol{y}_1(\boldsymbol{z}_{(\text{pre})})$ via forward pass on $\boldsymbol{z}_{(\text{pre})}$; `// Prefill`

$\mathbb{L}_0 \leftarrow \{(\boldsymbol{z}_{(\text{pre})},\ 0)\}$; `// Beam (max. capacity` $|\mathbb{L}_t| = B$`): pairs` $(\hat{\boldsymbol{z}},\ \log p)$

for $t = 1, \ldots, T$ **do**

  $\mathbb{C}_t \leftarrow \emptyset$; `// Candidate set for step` t

  foreach $(\hat{\boldsymbol{z}},\ \log p) \in \mathbb{L}_{t-1}$ **do**

    $\mathbb{V}_{t,k} \leftarrow \mathsf{TopK}_k(\boldsymbol{y}_t(\hat{\boldsymbol{z}}))$; $\boldsymbol{r}_t \leftarrow \mathsf{LogSoftmax}(\boldsymbol{y}_t(\hat{\boldsymbol{z}}))$; $Z_t \leftarrow \mathsf{LogSumExp}(\boldsymbol{r}_t[\mathbb{V}_{t,k}])$;

    foreach $\hat{z} \in \mathbb{V}_{t,k}$ **do**

      `// append token to partial history, update continuation log prob`

      $\hat{\boldsymbol{z}}' \leftarrow \hat{\boldsymbol{z}} \,\|\, \hat{z}$; $\log p' \leftarrow \log p + \boldsymbol{r}_t[\hat{z}] - Z_t$; $\mathbb{C}_t \leftarrow \mathbb{C}_t \cup \{(\hat{\boldsymbol{z}}',\ \log p')\}$

  if $t = T$ **then return** $\mathbb{C}_T$; `// Return up to` $B \cdot k$ `candidates`

  Remove from $\mathbb{C}_t$ any $(\hat{\boldsymbol{z}}', \log p')$ whose last token is EOS;

  $\mathbb{L}_t \leftarrow$ top-B elements of $\mathbb{C}_t$ by $\log p'$; `// Prune to beam width`

  if $\tau_{\min}$ *and* $\max_{(_,\log p)\in\mathbb{L}_t} \exp(\log p) < \tau_{\min}/(B \cdot k)$ **then return** $\emptyset$; `// LB cannot reach` $\tau_{\min}$

  Compute $\boldsymbol{y}_{t+1}(\hat{\boldsymbol{z}})$ for each $(\hat{\boldsymbol{z}}, \cdot) \in \mathbb{L}_t$;

ϕ, which lets us compute an inexpensive, deterministic lower bound on the near-verbatim extraction probability $p^{\mathsf{dist}}_{\boldsymbol{z},\varepsilon}$ (Equation 4). These algorithms have cost comparable to $B \ll M$ MC samples (Appendix C.3), where $B = 20$ works well in practice (Appendix F.7).

4.1 Beam search

Beam search (Lowerre, 1976) is a deterministic decoding algorithm for autoregressive language models. For LLM θ and a prompt $\boldsymbol{z}_{(\text{pre})}$, at each generation step $t \in 1, \ldots, T$ we maintain a **beam** of at most B $(a+t-1)$-length partial **histories** $\hat{\boldsymbol{z}} := \boldsymbol{z}_{(\text{pre})} \,\|\, \hat{\boldsymbol{z}}^{(\text{cont})}_{<t}$, each with its accumulated **score**, $\log p(\hat{\boldsymbol{z}}) := \log \Pr_\theta(\hat{\boldsymbol{z}}^{(\text{cont})}_{<t} \mid \boldsymbol{z}_{(\text{pre})})$. Then,

(A) For each $\hat{\boldsymbol{z}}$ in the beam, use θ to compute the next-token probability $\Pr_\theta(\hat{z}|\hat{\boldsymbol{z}})$ for every vocabulary token $\hat{z} \in \mathbb{V}$.

(B) Expand each history $\hat{\boldsymbol{z}}$ by every token $\hat{z} \in \mathbb{V}$, yielding a candidate set $\mathbb{C}_t$ of at most $B \cdot |\mathbb{V}|$ children $\hat{\boldsymbol{z}}' = \hat{\boldsymbol{z}} \| \hat{z}$ with updated scores $\log p(\hat{\boldsymbol{z}}') = \log p(\hat{\boldsymbol{z}}) + \log \Pr_\theta(\hat{z}|\hat{\boldsymbol{z}})$.

(C) Perform an across-beam prune, retaining the B highest-scoring $(a+t)$-length histories $\hat{\boldsymbol{z}}'$ in candidate set $\mathbb{C}_t$. These histories form the B-sized beam $\mathbb{L}_t$ for step $t+1$.

After T steps, the search returns the B highest-scoring T-length continuations $\hat{\boldsymbol{z}}_{(\text{cont})}$ that remain in the beam, which approximately maximize the conditional probability $\Pr_\theta(\hat{\boldsymbol{z}}_{(\text{cont})} \mid \boldsymbol{z}_{(\text{pre})})$. Because the joint log-probability of a continuation decomposes across time steps, a continuation must maintain a competitive cumulative score at *every* depth to survive pruning. Conversely, if at most depths a given continuation's score is near the top of the model's distribution, it is very likely to survive all across-beam prunes and appear in the final beam. Beam search therefore concentrates computational effort on a thin, high-mass region of candidate continuations rather than exploring the full, exponentially large space (Appendix C.2).

4.2 Computing a deterministic lower bound on extraction risk

To quantify near-verbatim probabilistic extraction, we introduce **decoding-constrained beam search**, which modifies beam search to operate under the distribution induced by decoding scheme ϕ, and therefore returns continuations with their exact probabilities under ϕ. Building on prior work on probabilistic extraction (Hayes et al., 2025b; Cooper et al., 2025), we focus on top-k decoding (Section 2) to develop **top-k-constrained beam search (k-CBS)** (Algorithm 1), but our approach can be applied with other ϕ (Appendix D.3).

We highlight key changes to beam search in red. For LLM θ, decoding scheme ϕ =top-k, and a prompt $z_{(\text{pre})}$, at each generation step $t \in 1, \ldots, T$ we maintain a beam of at most B $(a+t-1)$-length partial histories $\hat{z} := z_{(\text{pre})} \parallel \hat{z}_{<t}^{(\text{cont})}$, each with accumulated score $\log p(\hat{z}) := \log \Pr_{\theta,\phi}(\hat{z}_{<t}^{(\text{cont})} \mid z_{(\text{pre})})$. At each step t, $\mathbb{V}_{t,k}(\hat{z})$ is the set of top-k next tokens with the highest probabilities, given history $\hat{z}$.

(A) For each $\hat{z}$ in the beam, use θ with ϕ to get next-token probabilities $\Pr_{\theta,\phi}(\hat{z} \mid \hat{z})$; these probabilities are computed only over the top-k set $\mathbb{V}_{t,k}(\hat{z})$ (Equation 1).

(B) Expand each history $\hat{z}$ by the tokens in $\mathbb{V}_{t,k}(\hat{z})$, yielding at most $B \cdot k$ candidates. We update these candidates' scores according to the top-k distribution probabilities—i.e., $\log p(\hat{z}') = \log p(\hat{z}) + \log \Pr_{\theta,\phi}(\hat{z} \mid \hat{z})$.

(C) Perform an across-beam prune to B, except for at the final step T. At step T, return up to $B \cdot k$ T-length continuations $\hat{z}_{(\text{cont})}$ and their scores $\log \Pr_{\theta,\phi}(\hat{z}_{(\text{cont})} \mid z_{(\text{pre})})$—the exact probabilities under the top-k-renormalized decoding distribution.

Notably, beam search's well-known lack of output diversity (Li et al., 2016; Vijayakumar et al., 2018) is a *feature* here: for memorized sequences, it is useful for the beam to concentrate on the high-mass region near $z_{(\text{suf})}$, not to explore diverse alternatives.

From k-CBS outputs to a lower bound. For each returned $\hat{z}_{(\text{cont})}$, its score is exactly the conditional probability given the prefix under top-k decoding. For the sequence from *The Great Gatsby* in Figure 1, $B = 20$, and $k = 40$, so k-CBS (Algorithm 1) returns $B \cdot k = 800$ T-length continuations $\hat{z}_{(\text{cont})}$ and their probabilities—including the three near-verbatim $\hat{z}_{(\text{cont})}$ in the figure. We identify the $\hat{z}_{(\text{cont})}$ that are near-verbatim matches to the target $z_{(\text{suf})}$ (i.e., are in ε-ball $\mathbb{B}_{\varepsilon}^{\mathsf{dist}}(z_{(\text{suf})})$), and we sum their probabilities to get a lower bound on the near-verbatim extraction probability $p_{z,\varepsilon}^{\mathsf{dist}}$ with respect to top-k decoding (Equation 4).

More formally, let $\mathbb{F}$ denote the set of $(\hat{z}, \log p)$ pairs returned by k-CBS after T steps. We filter $\mathbb{F}$ to retain continuations $\hat{z}_{(\text{cont})}$ that satisfy $\mathsf{dist}(\hat{z}_{(\text{cont})}, z_{(\text{suf})}) \leq \varepsilon$, yielding $\mathbb{F}^{(\leq \varepsilon)} \subseteq \mathbb{F}$. Then,

$$\text{LB}_{\varepsilon,\mathsf{dist}} \triangleq \sum_{(_,\log p) \in \mathbb{F}^{(\leq \varepsilon)}} \exp(\log p) \;\leq\; p_{z,\varepsilon}^{\mathsf{dist}}. \tag{6}$$

This is a valid lower bound on $p_{z,\varepsilon}^{\mathsf{dist}}$: the candidates in $\mathbb{F}^{(\leq \varepsilon)}$ are a subset of all continuations within ε of $z_{(\text{suf})}$, and their probabilities under top-k are computed exactly (Appendix D). Unlike MC sampling (Section 3), because beam search is deterministic, this bound is deterministic; it requires no repeated trials. As shown in Figure 2, for beam width $B = 20$, k-CBS produces this bound at a cost comparable to just ≈ 20 MC samples—a budget at which MC produces no hits. For the sequence in Figure 1, $\text{LB}_{5,\mathsf{Lev}} = 0.716$—nearly $5\times$ the mass of the verbatim target suffix (i.e., the 0.1431 mass that verbatim probabilistic extraction captures).

We use $\hat{p}_{z,\varepsilon}^{\mathsf{dist}} = \text{LB}_{\varepsilon,\mathsf{dist}}$ to estimate $p_{z,\varepsilon}^{\mathsf{dist}}$, keeping in mind that it is an underestimate. k-CBS may miss near-verbatim continuations that were pruned from the beam, or miss extractable sequences entirely if no near-verbatim continuation survives. Nevertheless, our experiments (Section 5) show that k-CBS captures rich information about extraction risk in practice.

Why k-CBS produces useful lower bounds. The lower bound in Equation 6 is valid but could be vacuous (e.g., $\text{LB}_{\varepsilon,\mathsf{dist}} = 0$ if no near-verbatim path survives). However, for memorized sequences, it captures substantial mass because high-probability continuations are constrained to pick high-ranked tokens at almost every step—precisely the region that k-CBS explores. A simple counting argument illustrates why. For continuation $\hat{z}_{(\text{cont})}$, the conditional probability given prefix $z_{(\text{pre})}$ is the product of per-token conditional probabilities across T steps (Equation 2). For this product to exceed the extraction minimum $\tau_{\text{min}} = 0.001$, per token, the continuation must sustain a geometric-mean probability of at least $\tau_{\text{min}}^{1/T} = (0.001)^{1/50} \approx 0.87$—on average, enormous probability over $T = 50$ steps.

For instance, at most 3 steps can have per-token probability as low as 0.1, since $(0.1)^3 \cdot 1^{47} = 0.001 = \tau_{\text{min}}$. More generally, a token with probability p must be ranked at

least $\lceil 1/p \rceil$, since the tokens ranked above it each have probability $\geq p$ and probabilities sum to 1. Following similar reasoning, which we prove formally in the appendix,[1] a continuation can pick a token that falls below the top-10 on at most 2 of 50 steps; it can deviate from the distribution's top-1 token on at most 9 steps. This is exactly the region k-CBS explores—continuations that "hug" the top of the distribution's ranking at nearly every step. For memorized sequences, the near-verbatim $\hat{\boldsymbol{z}}_{(\text{cont})} \in \mathbb{B}_{\varepsilon}^{\mathsf{dist}}(\boldsymbol{z}_{(\text{suf})})$ that dominate $p_{\boldsymbol{z},\varepsilon}^{\mathsf{dist}}$ are therefore very likely to survive across-beam pruning and be captured by k-CBS. (See Appendix C.2.3 for further discussion, including a survival guarantee for heavy-mass paths.)

4.3 Constrained beam search with an ε-viable pruning rule

Baseline k-CBS (Algorithm 1) is agnostic to dist: it returns $B \cdot k$ candidates, which are post-processed to identify those within ε of the target. This generality comes at a cost. Throughout the search, beam capacity may be spent on continuations that can never satisfy reasonable $\mathsf{dist}(\hat{\boldsymbol{z}}_{(\text{cont})}, \boldsymbol{z}_{(\text{suf})}) \leq \varepsilon$, and the search runs for all T steps even when no viable paths remain. We can further modify k-CBS to remedy both of these issues with ε**-viability pruning**, which integrates a chosen distance metric dist and tolerance ε directly into the search. At each step t, for (B) we check whether each token-expanded partial history $\hat{\boldsymbol{z}}'$ can still produce a final continuation $\hat{\boldsymbol{z}}_{(\text{cont})}$ within distance ε of the target—whether there exists *any* possible completion of the continuation that satisfies $\mathsf{dist}(\hat{\boldsymbol{z}}_{(\text{cont})}, \boldsymbol{z}_{(\text{suf})}) \leq \varepsilon$. If not, the candidate is provably non-ε-viable; we do not include it in $\mathbb{C}_t$, freeing beam capacity for ε-viable candidates.

The incorporation of this ε-viability pruning rule requires only bookkeeping for each item in the beam—no additional model forward passes. It also enables early termination when no ε-viable candidates remain, often making it cheaper to run than baseline k-CBS for non-extractable sequences (Appendix C.3). The trade-off is that, unlike for baseline k-CBS (Appendix F.9), dist must be a token-level metric for which a lower bound on the final distance can be computed incrementally from streamable per-path state (Appendix E.3.1), and dist and ε must be specified upfront.

Hamming distance. For Ham (Equation 5), ε-viability is straightforward: we maintain a running mismatch count for each candidate in the beam. At step t, if the generated token $\hat{z}$ differs from the target token $z_t^{(\text{suf})}$, the count increments. Since positional mismatches cannot be undone, this count is monotonically non-decreasing; once it exceeds ε, the candidate is permanently non-ε-viable and is removed; it is not expanded further (Appendix E.1).

Levenshtein distance. For Lev (Equation 5), ε-viability is more subtle. Later insertions and deletions can repair earlier misalignments, so the minimum achievable distance can *decrease* between steps. We therefore track richer per-continuation state in the beam: the range of Lev distances between the partial t-length continuation and each j-length prefix $\boldsymbol{z}_{1:j}^{(\text{suf})}$ of the target suffix $\boldsymbol{z}_{(\text{suf})}$.[2] The continuation remains viable so long as the minimum of this range is at most ε; once the minimum exceeds ε, no possible completion can bring the distance back within the ε budget and the continuation is pruned (Appendix E.2).

All returned continuations are ε-viable by construction—no post-processing is needed. The algorithm also tracks an upper bound $\mathrm{UB} = \mathrm{LB} + \mathit{bank}$, where *bank* accumulates the mass of ε-viable continuations removed by the across-beam prune; in practice this bound is often loose, so we use $\hat{p}_{\boldsymbol{z},\varepsilon}^{\mathsf{dist}} = \mathrm{LB}_{\varepsilon,\mathsf{dist}}$. Full details and invariant proofs are in Appendix E.

[1] A token ranked R or worse contributes probability at most $1/R$. At decoding step t, let r_t be the rank of the realized token $\hat{z}_t^{(\text{cont})}$ (1 = highest probability). We can show that, for any integer $R \geq 2$, the number of steps $t \leq T$ for which $r_t \geq R$ is at most $\lfloor \ln \tau_{\min} / \ln(1/R) \rfloor$ (Appendix C.2.3).

[2] We propose a streamed version of the Ukkonen-banded Wagner & Fischer dynamic program. Define the dynamic programming table $D[t,j] \triangleq \mathsf{Lev}(\hat{\boldsymbol{z}}_{1:t}^{(\text{cont})}, \boldsymbol{z}_{1:j}^{(\text{suf})})$, with $D[0,j] = j$ and $D[t,0] = t$. At each step t, consider generated token $\hat{z}_t^{(\text{cont})}$ and each suffix token $z_j^{(\text{suf})}$ ($j \in [\max\{0, t-\varepsilon\},\ \min\{T, t+\varepsilon\}]$). Update the table to edit cost $D[t,j] = \min\{ \overbrace{D[t-1,j]+1}^{\text{delete } \hat{z}_t^{(\text{cont})}}, \overbrace{D[t,j-1]+1}^{\text{insert } z_j^{(\text{suf})}}, \overbrace{D[t-1,j-1]+\mathbf{1}[\hat{z}_t^{(\text{cont})} \neq z_j^{(\text{suf})}]}^{\text{match/substitute}} \}$.

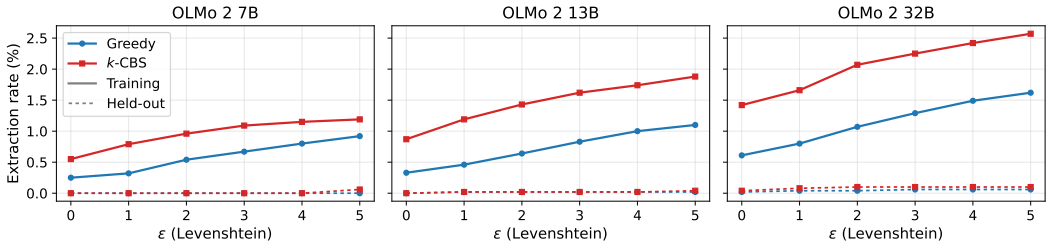


Figure 3: **Comparing extraction rates.** For OLMO 2 7B, 13B, and 32B, we show rates for verbatim ($\varepsilon = 0$) and near-verbatim extraction for $\text{Lev}\,\varepsilon \in \{1, \ldots, 5\}$. For greedy near-verbatim, one generates the single greedy $\hat{z}_{(\text{cont})}$ and checks if $\text{Lev}(\hat{z}_{(\text{cont})}, z_{(\text{suf})}) \leq \varepsilon$. We use a sample of 10,000 sequences from Wikipedia from OLMO 2's training data; to assess validity, we also run negative controls on 5,000 held-out sequences scraped from Wikipedia that post-date OLMO 2's training cutoff. Greedy rates are computed exactly. Probabilistic rates are computed with k-CBS (Section 4.2); they may miss some valid instances of extraction, and thus should be interpreted as lower bounds on extraction rates. We set $B = 20$ for k-CBS, so those results cost $\sim 20\times$ more than greedy extraction (Section 3).

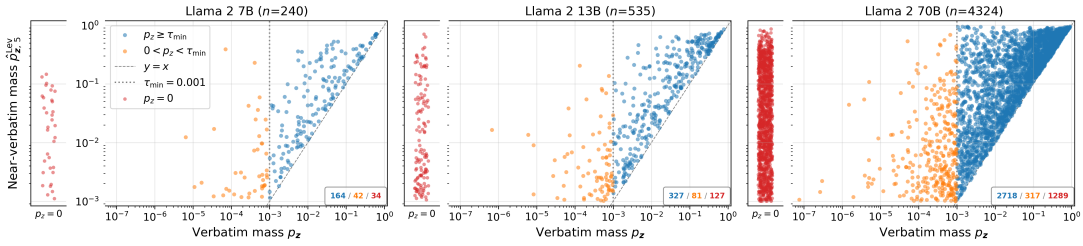


Figure 4: **Near-verbatim mass vs. verbatim mass.** LLAMA 2 on *The Great Gatsby*; each point is one sequence. Axes show near-verbatim ($p^{\text{Lev}}_{z,5}$, $\text{Lev}\,\varepsilon = 5$) vs. verbatim ($p_z$) extraction mass on a log–log scale. **Red**/**orange** points are "unlocked" by near-verbatim extraction (to the left of the τ_{min} dotted reference line, $p_z < \tau_{\text{min}}$, but $p^{\text{Lev}}_{z,5} \geq \tau_{\text{min}}$); **blue** points are verbatim-extractable ($p_z \geq \tau_{\text{min}}$). Points above the dashed $y = x$ line show increased extraction risk when near-verbatim mass is accounted for.

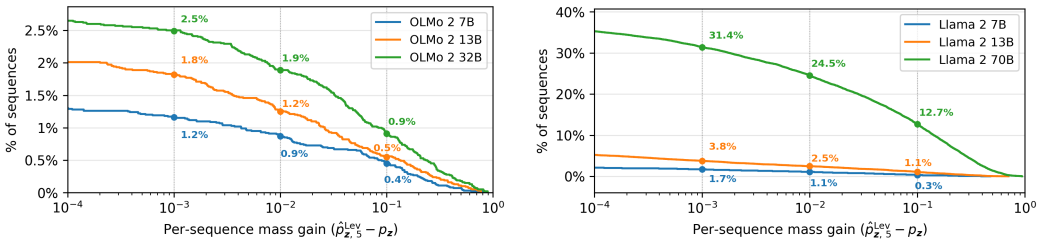
5 Experiments

In the main paper, we show results for k-CBS on two model families and datasets, which illustrate different patterns in near-verbatim extraction: (i) OLMO 2 (7B, 13B, 32B) (OLMo et al., 2025) on sequences from 10,000 Wikipedia articles from its training data (Dolma 1.7 (Soldaini et al., 2024)); and (ii) LLAMA 2 (7B, 13B, 70B) on *The Great Gatsby* from the Books3 corpus, which is known to be in LLAMA's training data (Touvron et al., 2023a; Kadrey v. Meta). In these experiments, we use baseline and Lev-pruned k-CBS with $k = 40$, beam width $B = 20$, $\varepsilon = 5$, prefix length $a = 50$, suffix length $T = 50$, and minimum extraction probability $\tau_{\text{min}} = 0.001$. Additional details (e.g., runtime, code, visualizations), extraction experiments, negative controls, and investigations of k-CBS algorithm parameters are in Appendix F.

5.1 Revealing additional extracted sequences and increased risk

Increased extraction rates. Figure 3 compares greedy and baseline k-CBS extraction rates (Appendix F.1) for OLMO 2 across model sizes and $\text{Lev}\,\varepsilon \in \{0, \ldots, 5\}$. k-CBS dominates greedy at every ε and model size, with both rates and gaps growing with model size. That is, greedy extraction under-counts extraction more for larger models. For OLMO 2 32B, the rate at $\varepsilon = 5$ reaches 2.57%, compared to 1.62% for greedy near-verbatim, 1.42% for baseline k-CBS at $\varepsilon = 0$ (verbatim), and 0.61% for greedy verbatim. Lines for held-out data (negative controls) stay flat near zero, supporting the validity of our extraction procedure (Appendix F.3.1).

Analyzing "unlocked" instances of extraction. Figure 4 plots each sequence's near-verbatim extraction mass ($\hat{p}^{\text{Lev}}_{z,5}$) against its verbatim mass (p_z) for LLAMA 2 on *The Great*



(a) OLMo 2 on Wikipedia (b) LLAMA 2 on *The Great Gatsby*

Figure 5: **CCDF of per-sequence near-verbatim mass gain.** For Lev $\varepsilon = 5$ mass minus verbatim mass ($\hat{p}^{\text{Lev}}_{z,5} - p_z$), a point (x, y) means y% of sequences have extraction-mass gain $\geq x$.

Gatsby. The **red** and **orange** points are sequences "unlocked" by near-verbatim extraction: they fall below the τ_{min} threshold for verbatim extraction, but are extractable when we account for near-verbatim mass. **Red** points have *zero* verbatim mass, while **orange** points have nonzero but sub-threshold (below $\tau_{\text{min}} = 0.001$) verbatim mass.

At 70B, 1,606 sequences are unlocked, of which 1,289 (80.3%) have zero verbatim mass. These unlocked sequences are not marginal: their mean near-verbatim mass is 0.086 (86× the extraction threshold τ_{min}), with a maximum of 0.863. Typical edits include spacing around punctuation (e.g., "money— that" vs. "money—that", Lev = 1, mass 0.777). **Blue** points are verbatim-extractable; those above the $y = x$ line show increased extraction risk when near-verbatim mass is included. The number of unlocked sequences grows from 76 (7B) to 1,606 (70B)—a 21× increase for a 10× increase in parameters. Interestingly, many unlocked sequences at smaller model sizes are verbatim extractable at larger ones (Appendix F.2.3).

5.2 Patterns in near-verbatim extraction risk

Risk increases scale with model size. Figure 4 suggests that larger models have more unlocked sequences with larger near-verbatim mass. Figure 5 quantifies this directly, showing the complementary CDF (CCDF) of absolute per-sequence mass gain ($\hat{p}^{\text{Lev}}_{z,5} - p_z$)—i.e., the additional extraction mass from near-verbatim continuations. Figure 5b shows that, for LLAMA 2 70B and *The Great Gatsby*, 12.7% of all sequences in the book (32.9% of extractable sequences) exhibit a ≥ 0.1 mass gain. For both CCDFs, the curves shift upward and rightward with model size: larger models have both *more* sequences with positive extraction risk gain and *larger* absolute per-sequence gains. Verbatim probabilistic extraction therefore produces increasingly large undercounts of near-verbatim extraction risk at larger model scales.

Per-sequence verbatim mass share varies by model size and text. The absolute mass gains described above raise a natural question: for a given extractable sequence, what relative portion of its total extraction risk comes from the verbatim continuation? For each extracted sequence z, we compute this as the **verbatim share**:

$$\text{verbatim_share}(z) \triangleq \frac{p_z}{\hat{p}^{\text{Lev}}_{z,5}} \times 100\%. \tag{7}$$

This percentage reflects how much the extraction mass concentrates on the verbatim sequence versus on near-verbatim variants (for z, the near-verbatim only share is $100\% - \text{verbatim_share}(z)$). Sequences with 0% verbatim share are those for which the extraction mass comes entirely from near-verbatim continuations ($p_z = 0$, e.g., the **red** "unlocked" points in Figure 4). Those with 100% verbatim share exhibit no increase in extraction risk when we account for near-verbatim mass ($p_z = \hat{p}^{\text{Lev}}_{z,5}$, e.g., the **blue** points on $y = x$ in Figure 4). Most sequences fall somewhere in between these extremes, like the one in Figure 6, for which verbatim share is 27.56%.

We can compute verbatim share for every extracted sequence and plot the distribution, as in Figure 7. We indicate the median to summarize the verbatim share of the typical extracted sequence. For instance, for LLAMA 2 7B on *The Great Gatsby* (Figure 7b), the median verbatim share is 27.53%: the median sequence resembles the example in Figure 6, with respect to the decomposition of verbatim and near-verbatim mass.

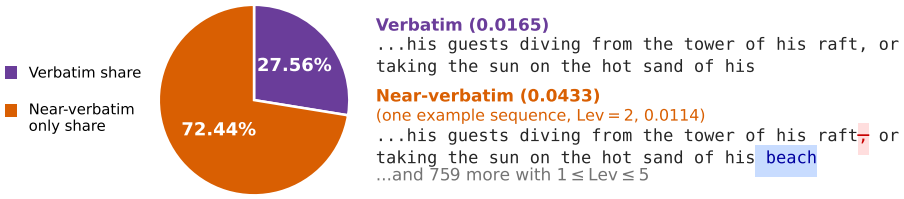
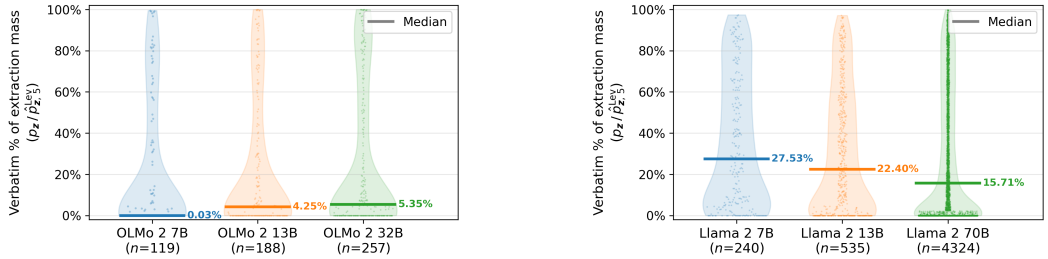


Figure 6: **Verbatim share.** For LLAMA 2 7B and one sequence from *The Great Gatsby*, we show the verbatim share (Equation 7): 27.56% of the total near-verbatim mass $p^{\text{Lev}}_{z,5} \approx 0.06$.



(a) OLMO 2 on Wikipedia

(b) LLAMA 2 on *The Great Gatsby*

Figure 7: **Distribution of verbatim share.** We compute verbatim share (Equation 7) for each extracted sequence and plot the distribution. Higher values indicate extraction mass is dominated by verbatim memorization; lower values indicate mass spread across near-verbatim variants. We annotate the median verbatim-mass shares.

In this setting, the verbatim share decreases substantially with model size. Larger LLAMA 2 models spread more of their extraction mass across near-verbatim variants of memorized *Gatsby* text. This appears to be driven by changes in the composition of the extractable set (e.g., the huge increase in **red** points in Figure 4). The set grows dramatically with model size (for *Gatsby*, 240 → 535 → 4,324), and newly extractable sequences at larger model sizes tend to have low verbatim share (median verbatim share of 12.9% for sequences first extractable at 70B). Through qualitative analysis, we observe near-verbatim edits include spelling, punctuation, and minor syntactic variation—possibly reflecting training on multiple editions or learned variation in natural language syntax (Appendix F.5.1).

The results for OLMO 2 on Wikipedia exhibit a different pattern. In contrast to LLAMA 2 on *Gatsby*, the median verbatim share *increases* slightly with model size, consistent with larger models concentrating more mass on the exact Wikipedia-article training text. However, verbatim share remains very low overall (Appendix F.3). For instance, half of extracted sequences for OLMO 2 7B have verbatim share below the 0.03% median—i.e., nearly half of all extractable sequences have zero verbatim mass ($p_z = 0$).

These patterns illustrate that near-verbatim extraction risk manifests differently depending on the type of text, model, and model size—underscoring the value of analysis beyond verbatim extraction. We defer investigation of the underlying causes to future work. Additional results on other datasets and models appear in Appendix F.[3]

6 Conclusion

We introduce decoding-constrained beam search (Section 4): a family of algorithms that produces deterministic lower bounds on near-verbatim extraction probability at a fraction of the cost of Monte Carlo estimation (Section 3). Our experiments show that accounting for near-verbatim probabilistic extraction (Section 2) reveals substantially more memorization and extraction risk, with rich variation across model sizes and types of text (Section 5). In future work, we will further investigate patterns in near-verbatim extraction risk, and explore decoding-constrained beam search as a memorization diagnostic tool during model training.

[3] PYTHIA models on Enron show a stark increase in verbatim share with model size (Appendix F.4).

Acknowledgments

We thank Ahmed Ahmed and Abe Hou for feedback on earlier versions of this work. Katherine Lee joined OpenAI as a full-time employee after contributing to this work.

References

Ahmed Ahmed, A. Feder Cooper, Sanmi Koyejo, and Percy Liang. Extracting books from production language models. *arXiv preprint arXiv:2601.02671*, 2026.

Jane Austen. *Pride and Prejudice*. Penguin Books Ltd, 1813.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate, 2016. URL https://arxiv.org/abs/1409.0473.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pp. 2397–2430. PMLR, 2023.

Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Audio Chord Recognition with Recurrent Neural Networks. In *International Society for Music Information Retrieval Conference*, 2013.

Hannah Brown, Katherine Lee, Fatemehsadat Mireshghallah, Reza Shokri, and Florian Tramèr. What Does it Mean for a Language Model to Preserve Privacy? In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '22, pp. 2280–2292, New York, NY, USA, 2022. Association for Computing Machinery. URL https://doi.org/10.1145/3531146.3534642.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pp. 2633–2650, 2021.

Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. Membership inference attacks from first principles. In *2022 IEEE Symposium on Security and Privacy (SP)*, pp. 1897–1914. IEEE, 2022.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramèr, and Chiyuan Zhang. Quantifying Memorization Across Neural Language Models. In *International Conference on Learning Representations*, 2023.

A. Feder Cooper and James Grimmelmann. The Files are in the Computer: Copyright, Memorization, and Generative AI. *arXiv preprint arXiv:2404.12590*, 2024.

A. Feder Cooper, Katherine Lee, James Grimmelmann, Daphne Ippolito, Christopher Callison-Burch, Christopher A. Choquette-Choo, Niloofar Mireshghallah, Miles Brundage, David Mimno, Madiha Zahrah Choksi, Jack M. Balkin, Nicholas Carlini, Christopher De Sa, Jonathan Frankle, Deep Ganguli, Bryant Gipson, Andres Guadamuz, Swee Leng Harris, Abigail Z. Jacobs, Elizabeth Joh, Gautam Kamath, Mark Lemley, Cass Matthews, Christine McLeavey, Corynne McSherry, Milad Nasr, Paul Ohm, Adam Roberts, Tom Rubin, Pamela Samuelson, Ludwig Schubert, Kristen Vaccaro, Luis Villa, Felix Wu, and Elana Zeide. Report of the 1st Workshop on Generative AI and Law. *arXiv preprint arXiv:2311.06477*, 2023.

A. Feder Cooper, Christopher A Choquette-Choo, Miranda Bogen, Matthew Jagielski, Katja Filippova, Ken Ziyu Liu, Alexandra Chouldechova, Jamie Hayes, Yangsibo Huang, et al. Machine unlearning doesn't do what you think: Lessons for generative ai policy, research, and practice. *arXiv preprint arXiv:2412.06966*, 2024.

A. Feder Cooper, Aaron Gokaslan, Amy B. Cyphert, Christopher De Sa, Mark A. Lemley, Daniel E. Ho, and Percy Liang. Extracting memorized pieces of (copyrighted) books from open-weight language models. *arXiv preprint arXiv:2505.12546*, 2025.

A. Feder Cooper, Marika Swanberg, Jamie Hayes, Lea Duesterwald, Christopher De Sa, Daniel E. Ho, Mark A. Lemley, and Percy Liang. Extractable Memorization From First Principles. *arXiv preprint arXiv:2607.12649*, 2026.

Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical Neural Story Generation. In Iryna Gurevych and Yusuke Miyao (eds.), *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 889–898, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1082. URL https://aclanthology.org/P18-1082/.

Vitaly Feldman. Does learning require memorization? a short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, pp. 954–959, New York, NY, USA, 2020. Association for Computing Machinery.

F. Scott Fitzgerald. *The Great Gatsby*. Harper Perennial Classics; HarperCollins Publishers (Canada) Ltd., 1925.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The Pile: An 800GB Dataset of Diverse Text for Language Modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Gemini Team et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

Aaron Grattafiori et al. The Llama 3 Herd of Models, 2024. URL https://arxiv.org/abs/2407.21783.

Alex Graves. Sequence Transduction with Recurrent Neural Networks, 2012. URL https://arxiv.org/abs/1211.3711.

Jamie Hayes, Ilia Shumailov, Christopher A. Choquette-Choo, Matthew Jagielski, George Kaissis, Katherine Lee, Milad Nasr, Sahra Ghalebikesabi, Niloofar Mireshghallah, Meenatchi Sundaram Mutu Selva Annamalai, Igor Shilov, Matthieu Meeus, Yves-Alexandre de Montjoye, Franziska Boenisch, Adam Dziedzic, and A. Feder Cooper. Strong Membership Inference Attacks on Massive Datasets and (Moderately) Large Language Models. *arXiv preprint arXiv:2505.18773*, 2025a.

Jamie Hayes, Marika Swanberg, Harsh Chaudhari, Itay Yona, Ilia Shumailov, Milad Nasr, Christopher A. Choquette-Choo, Katherine Lee, and A. Feder Cooper. Measuring memorization in language models via probabilistic extraction. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 9266–9291, Albuquerque, New Mexico, April 2025b. Association for Computational Linguistics. URL https://aclanthology.org/2025.naacl-long.469/.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The Curious Case of Neural Text Degeneration, 2020. URL https://arxiv.org/abs/1904.09751.

Daphne Ippolito, Florian Tramer, Milad Nasr, Chiyuan Zhang, Matthew Jagielski, Katherine Lee, Christopher Choquette Choo, and Nicholas Carlini. Preventing Generation of Verbatim Memorization in Language Models Gives a False Sense of Privacy. In C. Maria Keet, Hung-Yi Lee, and Sina Zarrieß (eds.), *Proceedings of the 16th International Natural Language Generation Conference*, pp. 28–53, Prague, Czechia, September 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.inlg-main.3. URL https://aclanthology.org/2023.inlg-main.3/.

Kadrey v. Meta. URL https://www.courthousenews.com/wp-content/uploads/2025/02/kadrey-vs-meta-third-amended-complaint.pdf. No. 3:23-cv-03417-VC (Third Amended Consolidated Complaint).

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. Deduplicating Training Data Makes Language Models Better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, volume 1, pp. 8424–8445, 2022.

Katherine Lee, A. Feder Cooper, James Grimmelmann, and Daphne Ippolito. AI and Law: The Next Generation. *SSRN*, 2023a. http://dx.doi.org/10.2139/ssrn.4580739.

Katherine Lee, A. Feder Cooper, and James Grimmelmann. Talkin' 'Bout AI Generation: Copyright and the Generative-AI Supply Chain. *arXiv preprint arXiv:2309.08133*, 2023b.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. A Diversity-Promoting Objective Function for Neural Conversation Models. In Kevin Knight, Ani Nenkova, and Owen Rambow (eds.), *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 110–119, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1014. URL https://aclanthology.org/N16-1014/.

Bruce T. Lowerre. *The HARPY Speech Recognition System*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 1976. PhD Dissertation.

A. A. Milne. *Winnie the Pooh*. Dutton Children's Books; Penguin Group (USA) Inc., 1926.

Gene Myers. A fast bit-vector algorithm for approximate string matching based on dynamic programming. *J. ACM*, 46(3):395–415, May 1999. ISSN 0004-5411. doi: 10.1145/316542.316550. URL https://doi.org/10.1145/316542.316550.

Milad Nasr, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A. Feder Cooper, Daphne Ippolito, Christopher A. Choquette-Choo, Eric Wallace, Florian Tramèr, and Katherine Lee. Scalable Extraction of Training Data from (Production) Language Models. *arXiv preprint arXiv:2311.17035*, 2023.

Milad Nasr, Javier Rando, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A. Feder Cooper, Daphne Ippolito, Christopher A. Choquette-Choo, Florian Tramèr, and Katherine Lee. Scalable Extraction of Training Data from Aligned, Production Language Models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=vjel3nWP2a.

Gonzalo Navarro. A guided tour to approximate string matching. *ACM Comput. Surv.*, 33(1):31–88, March 2001. ISSN 0360-0300. doi: 10.1145/375360.375365. URL https://doi.org/10.1145/375360.375365.

Henrik Nolte, Michèle Finck, and Kristof Meding. Machine Learners Should Acknowledge the Legal Implications of Large Language Models as Personal Data, 2025. URL https://arxiv.org/abs/2503.01630.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Allyson Ettinger, Michal Guerquin, David Heineman, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James V. Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, Jake Poznanski, Valentina Pyatkin, Aman Rangapur, Michael Schmitz, Sam Skjonsberg, David Wadden, Christopher Wilhelm, Michael Wilson, Luke Zettlemoyer, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. 2 OLMo 2 Furious, 2025. URL https://arxiv.org/abs/2501.00656.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pp. 311–318, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://doi.org/10.3115/1073083.1073135.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pp. 3–18. IEEE, 2017.

Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, Valentin Hofmann, Ananya Harsh Jha, Sachin Kumar, Li Lucy, Xinxi Lyu, Nathan Lambert, Ian Magnusson, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Abhilasha Ravichander, Kyle Richardson, Zejiang Shen, Emma Strubell, Nishant Subramani, Oyvind Tafjord, Pete Walsh, Luke Zettlemoyer, Noah A. Smith, Hannaneh Hajishirzi, Iz Beltagy, Dirk Groeneveld, Jesse Dodge, and Kyle Lo. Dolma: an Open Corpus of Three Trillion Tokens for Language Model Pretraining Research, 2024. URL https://arxiv.org/abs/2402.00159.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to Sequence Learning with Neural Networks, 2014. URL https://arxiv.org/abs/1409.3215.

Gemma Team et al. Gemma 2: Improving Open Language Models at a Practical Size, 2024. URL https://arxiv.org/abs/2408.00118.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and Efficient Foundation Language Models, 2023a. URL https://arxiv.org/abs/2302.13971.

Hugo Touvron et al. Llama 2: Open Foundation and Fine-Tuned Chat Models, 2023b. URL https://arxiv.org/abs/2307.09288.

Esko Ukkonen. Algorithms for approximate string matching. *Inf. Control*, 64(1–3): 100–118, March 1985. ISSN 0019-9958. doi: 10.1016/S0019-9958(85)80046-2. URL https://doi.org/10.1016/S0019-9958(85)80046-2.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R. Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. Diverse Beam Search: Decoding Diverse Solutions from Neural Sequence Models, 2018. URL https://arxiv.org/abs/1610.02424.

Robert A. Wagner and Michael J. Fischer. The String-to-String Correction Problem. *J. ACM*, 21(1):168–173, January 1974. ISSN 0004-5411. doi: 10.1145/321796.321811. URL https://doi.org/10.1145/321796.321811.

Virginia Woolf. *Orlando*. Hogarth Press, 1928.

Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st computer security foundations symposium (CSF)*, pp. 268–282. IEEE, 2018.

Appendix Table of Contents

A Language models and verbatim training-data extraction **17**
A.1 Language models and decoding schemes 17
A.2 Verbatim extraction of training data 18
A.2.1 Defining verbatim extraction success 18
A.2.2 Observations about probabilistic extraction 19
A.2.3 Computing an extraction rate 20

B Problem setup for near-verbatim extraction **20**
B.1 Distance metrics 20
B.2 Success criteria for near-verbatim extraction 21
B.3 Quantifying near-verbatim extraction success in practice 21
B.3.1 Challenges for computing near-verbatim probabilistic extraction . . 22
B.3.2 Levenshtein never exceeds Hamming for T-length sequences 23

C An intuition for more efficient near-verbatim probabilistic extraction **24**
C.1 Monte Carlo sampling 24
C.2 An intuition for using a decoding-constrained beam search approach 26
C.2.1 Beam search 26
C.2.2 Modifying beam search with our approach 27
C.2.3 Mathematical intuition for good performance in practice 28
C.3 Comparing method costs with token evaluations 32

D Decoding-constrained beam search **34**
D.1 Notation 35
D.2 Top-k constrained beam search 37
D.2.1 Detailed description of baseline k-CBS 37
D.2.2 Invariants for baseline k-CBS 42
D.3 Nucleus sampling constrained beam search (p-CBS) 48

E ε-pruned k-CBS **50**
E.1 Hamming-ε-pruned k-CBS 50
E.2 Levenshtein-ε-pruned k-CBS 55
E.2.1 Wagner-Fischer dynamic program 55
E.2.2 Ukkonen's band and a viable pruning rule 57
E.2.3 Integrating the Levenshtein-pruning dynamic program into k-CBS . 58
E.3 Invariants for ε-pruned k-CBS 61
E.3.1 Distance-viability-pruned k-CBS: Notation and background 61
E.3.2 Distance-specific results for streaming generation ε-viability 63

E.3.3 General invariants that follow 64

F Details on experiments **67**

F.1 Metrics and visualizations for assessing extraction 67

F.1.1 Extraction rates 67

F.1.2 Visualizing "unlocked" sequences and per-sequence risk increase . . 68

F.1.3 Analyzing per-sequence risk composition 69

F.1.4 Heatmaps to visualize book extraction coverage 69

F.2 Experimental settings 70

F.2.1 OLMo 2 on Wikipedia 70

F.2.2 Pythia on emails 71

F.2.3 Llama on public domain books 71

F.2.4 Summary of reported experiments 73

F.3 Extended results for OLMo 2 on Wikipedia 74

F.3.1 Extracted sequences and rates 74

F.3.2 Extraction risk 77

F.4 Extended results for Pythia on emails 79

F.4.1 Extracted sequences and rates 79

F.4.2 Extraction risk 80

F.5 Extended results for Llama 2 on public domain books 82

F.5.1 Extracted sequences and rates 82

F.5.2 Extraction risk 87

F.5.3 Heatmaps and cross-model sequence analysis 88

F.6 Additional experiment with Llama 3.1 8B 92

F.7 Varying beam width 93

F.8 Comparing baseline and ε-viability pruned k-CBS 95

F.9 Examining baseline k-CBS outputs with other near-verbatim metrics 97

A Language models and verbatim training-data extraction

We give basic background and our notation for language models and decoding schemes (Appendix A.1). We then discuss background and related work concerning the state-of-the-art approach for measuring verbatim extraction (Appendix A.2).

A.1 Language models and decoding schemes

Let $\mathbb{V}$ denote the token **vocabulary** for a **large language model (LLM)**. For example, $|\mathbb{V}| = 32{,}000$ for LLAMA 1 models. An LLM with **weights** θ maps a sequence of tokens $\boldsymbol{b} = (b_1, \ldots, b_n) \in \mathbb{V}^n$ to a logit vector $\boldsymbol{y} \in \mathbb{R}^{|\mathbb{V}|}$: $\theta : \mathbb{V}^n \to \mathbb{R}^{|\mathbb{V}|}$. A **decoding scheme** ϕ defines how logits are mapped to a next-token sampling distribution, $\mathsf{softmax}_\phi : \mathbb{R}^{|\mathbb{V}|} \to \mathcal{P}(\mathbb{V})$, where $\mathcal{P}(\mathbb{V})$ is the set of probability distributions over $\mathbb{V}$. For example, temperature scaling or top-k filtering each define such transformations (discussed below). Together, (θ, ϕ) define an **autoregressive generation process**: given a **prompt** $\boldsymbol{b}_{1:i}$, at each step $t > i$ we (i) compute $\boldsymbol{y}_t = \theta(\boldsymbol{b}_{1:t-1})$ and (ii) sample $b_t \sim \mathsf{softmax}_\phi(\boldsymbol{y}_t)$, (iii) producing a **continuation**. After T iterations, we obtain a generated **continuation** $\hat{\boldsymbol{b}}_{a+1:a+T}$.

Logits and probabilities. Given a sequence $\boldsymbol{b}_{1:t-1}$, the model θ outputs for every token $v \in \mathbb{V}$ a real value $\boldsymbol{y}_t[v]$ called a **logit** associated with the next token b_t. ($\boldsymbol{y}_t \in \mathbb{R}^{|\mathbb{V}|}$ is the entire logit vector.) Logits are unnormalized scores, which are mapped to a probability distribution over $\mathbb{V}$ via the **softmax function**:

$$\mathsf{softmax}(\boldsymbol{y}_t)[v] \;=\; \frac{\exp(\boldsymbol{y}_t[v])}{\sum_{u \in \mathbb{V}} \exp(\boldsymbol{y}_t[u])}, \qquad v \in \mathbb{V}, \tag{8}$$

where the model's next-token conditional probability distribution is

$$\Pr(\cdot \mid \boldsymbol{b}_{1:t-1}; \theta) \;=\; \mathsf{softmax}(\boldsymbol{y}_t),$$

which is a function $\mathbb{R}^{|\mathbb{V}|} \to [0,1]^{|\mathbb{V}|}$. In particular, the probability of a given token v is

$$\Pr(b_t = v \mid \boldsymbol{b}_{1:t-1}; \theta) \;=\; \mathsf{softmax}(\boldsymbol{y}_t)[v].$$

Temperature scaling. A **temperature** parameter $\beta > 0$ can rescale the logits before applying softmax:

$$\Pr_\beta(\cdot \mid \boldsymbol{b}_{1:t-1}; \theta) \;=\; \mathsf{softmax}\Big(\tfrac{1}{\beta}\boldsymbol{y}_t\Big). \tag{9}$$

For a specific token v,

$$\Pr_\beta(b_t = v \mid \boldsymbol{b}_{1:t-1}; \theta) = \frac{\exp(\frac{\boldsymbol{y}_t[v]}{\beta})}{\sum_{u \in \mathbb{V}} \exp(\frac{\boldsymbol{y}_t[u]}{\beta})}. \tag{10}$$

Here, β is a hyperparameter of the decoding scheme that modifies the distribution used to sample the next token. When $\beta = 1$, we recover the base distribution of θ. As $\beta \to 0$, the distribution sharpens toward a point mass on the maximum-logit token, corresponding to deterministic greedy decoding (discussed below). As $\beta \to \infty$, the distribution approaches uniform, corresponding to random sampling where each token is equally likely.

Top-k filtering. Let $\mathbb{V}_k \subseteq \mathbb{V}$ be the set of the $k > 0$ tokens with largest logits at step t. **Top-k** filtering (Fan et al., 2018) masks out all other tokens by setting their logits to $-\infty$, which is equivalent to assigning them probability 0 after softmax (Equation 8). The remaining k logits are normalized to form a distribution:

$$\Pr_k(b_t = v \mid \boldsymbol{b}_{1:t-1}; \theta) \;=\; \begin{cases} \dfrac{\exp(\boldsymbol{y}_t[v])}{\sum_{u \in \mathbb{V}_k} \exp(\boldsymbol{y}_t[u])}, & v \in \mathbb{V}_k, \\ 0, & v \notin \mathbb{V}_k. \end{cases} \tag{11}$$

Temperature scaling and top-k filtering can be combined: first scale the logits by $\frac{1}{\beta}$, then mask all but the top-k tokens by setting their logits to $-\infty$, and finally apply softmax to normalize the remaining logits into a probability distribution. **Greedy decoding** is the special case $k = 1$ (temperature is immaterial): at each step t, the highest-probability (top-1) token is chosen deterministically.

A.2 Verbatim extraction of training data

Models are known to memorize portions, but not all, of their training data. **Memorization** refers to the encoding of specific training sequences in a model's weights (Feldman, 2020; Yeom et al., 2018; Hayes et al., 2025a; Cooper et al., 2023), such that the learned distribution assigns very high probability to them. For generative models like LLMs, this can make **extraction** possible: memorized sequences can sometimes be reproduced in outputs (Carlini et al., 2021; Lee et al., 2022).

We denote an LLM θ's **training dataset** by $\mathbb{D}$, which consists of training sequences of tokens. A given training sequence $\boldsymbol{z}$ can be split into a **prefix** $\boldsymbol{z}_{1:a}$ and a **target suffix** $\boldsymbol{z}_{a+1:a+T}$, i.e., $\boldsymbol{z} = \boldsymbol{z}_{1:a} \parallel \boldsymbol{z}_{a+1:a+T}$. We quantify **extraction** of such a training sequence using model θ and decoding scheme ϕ as follows. Given the prefix $\boldsymbol{z}_{1:a}$, we run T autoregressive steps where, at each step, θ produces a distribution over the vocabulary $\mathbb{V}$, ϕ transforms that distribution, and one token is sampled and appended. We denote the generated continuation by

$$\hat{\boldsymbol{z}}_{a+1:a+T} \;=\; \mathsf{generate}_{\theta,\phi}(\boldsymbol{z}_{1:a}, T).$$

In the extraction literature, it is common to set $a = T = 50$, i.e., use 100-token sequences (Carlini et al., 2023; Hayes et al., 2025b; Cooper et al., 2025). Most prior work studies extraction success in terms of an exact match between the generated continuation $\hat{\boldsymbol{z}}_{a+1:a+T}$ and the target suffix $\boldsymbol{z}_{a+1:a+T}$.

A.2.1 Defining verbatim extraction success

We use a predicate $s_{\theta,\phi}(\boldsymbol{z}) \in \{0, 1\}$ to indicate whether verbatim extraction of the target suffix succeeds under the chosen criterion (specified below).

Success with respect to greedy decoding. Greedy decoding, which we denote $\phi = \mathsf{greedy}$, is deterministic; for a given prompt, model, and output length, it always results in the same output. As a result, in the greedy case, verbatim extraction success reduces to exact string equality between the generated continuation and the ground-truth target suffix from the training data, i.e.,

$$s_{\theta,\mathsf{greedy}}(\boldsymbol{z}) \;\triangleq\; \mathbf{1}[\,\hat{\boldsymbol{z}}_{a+1:a+T} \;=\; \boldsymbol{z}_{a+1:a+T}\,]. \tag{12}$$

This is the most common approach for quantifying memorization in both research and technical reports for new model releases (Carlini et al., 2023; Lee et al., 2022; Gemini Team et al., 2024; Team et al., 2024; Grattafiori et al., 2024; Biderman et al., 2023). The associated metric is called **discoverable extraction** (Carlini et al., 2023).

Success with respect to non-deterministic sampling. Hayes et al. (2025b) provide an alternative way of measuring extraction, which accounts for the non-determinism of sampling from distributions produced by most decoding schemes. For their **probabilistic discoverable extraction** metric, one can pick a non-deterministic decoding scheme ϕ (e.g., base distribution $\beta = 1$, or top-k with $k = 40$), where the conditional next-token distribution can be used to compute suffix probabilities $p_{\boldsymbol{z}} \in [0, 1]$.

In this setting, Cooper et al. (2025) define the success predicate as follows. For a non-deterministic decoding scheme ϕ, the probability $p_{\boldsymbol{z}}$ of generating the exact T-token target suffix given an a-token prefix, under (θ, ϕ), is

$$p_{\boldsymbol{z}} \;\triangleq\; \Pr_{\theta,\phi}(\boldsymbol{z}_{a+1:a+T} \mid \boldsymbol{z}_{1:a}) \;=\; \prod_{t=a+1}^{a+T} \Pr_{\theta,\phi}(z_t \mid \boldsymbol{z}_{1:t-1}) \;=\; \exp\left(\sum_{t=a+1}^{a+T} \log \Pr_{\theta,\phi}(z_t \mid \boldsymbol{z}_{1:t-1}) \right). \tag{13}$$

This is the probability that the model θ generates the exact target suffix $z_{a+1:a+T}$, token by token under decoding scheme ϕ—i.e., the probability that $\hat{z}_{a+1:a+T}$ is equal to $z_{a+1:a+T}$. For a threshold $\tau_{\min} \in (0, 1]$, verbatim probabilistic extraction success occurs when this probability is at least $\tau_{\min}$:

$$s_{\theta,\phi}(\boldsymbol{z}; \tau_{\min}) \triangleq \mathbf{1}[\, p_z \geq \tau_{\min} \,]. \tag{14}$$

Cooper et al. (2025) set and validate a very conservative $\tau_{\min} = 0.001$ for top-k decoding with $\beta = 1$ and $k = 40$. For convenience, we do the same here.

A.2.2 Observations about probabilistic extraction

Three important observations follow from the above definitions for probabilistic discoverable extraction (which we refer to as **probabilistic extraction**, for short):

1. Greedy-decoded extraction is a special case of probabilistic extraction: under $\phi = \text{greedy}$, the per-step distribution is a point mass on the arg max token, so $p_z = 1$ if and only if the generated continuation equals the target suffix (otherwise $p_z = 0$), recovering Equation 12.
2. For stochastic ϕ, a single-run, one-shot equality indicator $\mathbf{1}[\mathsf{generate}_{\theta,\phi}(z_{1:a}, T) = z_{a+1:a+T}]$ is a Bernoulli random variable with mean p_z.
3. In practice, it is unnecessary to actually generate any continuations to estimate p_z; teacher-forced scoring (described below) computes p_z directly from the logits $\boldsymbol{y}$ in a single forward pass (Cooper et al., 2025).

Computing p_z without sampling (teacher-forced scoring). To evaluate p_z for verbatim extraction, we do not need to generate any tokens. Instead, we perform a single forward pass on the *entire* sequence from the training data $\boldsymbol{z} = z_{1:a} \,\|\, z_{a+1:a+T}$ to obtain next-token logit vectors $\boldsymbol{y}_t$ for $t = a+1, \ldots, a+T$ (each conditioned on $z_{1:t-1}$, the prefix and any earlier tokens in the suffix). We then apply the decoding scheme ϕ in *logit space* (e.g., temperature rescaling and/or top-k filtering), and obtain the probability of the ground-truth suffix token z_t by applying the softmax:

$$\Pr_{\theta,\phi}(z_t \mid z_{1:t-1}) \;=\; \mathsf{softmax}(\phi(\boldsymbol{y}_t))[z_t], \qquad t = a+1, \ldots, a+T. \tag{15}$$

Finally, $p_z = \prod_{t=a+1}^{a+T} \Pr_{\theta,\phi}(z_t \mid z_{1:t-1})$, or equivalently the sum of their log-probabilities (as in Equation 13). This procedure, called **teacher-forced scoring**, computes all T probabilities in one batched pass that reuses the model's internal parallelism. It avoids T sequential sampling steps from autoregressive generation and the associated overhead. Even with KV caching, in practice, autoregressive generation is more expensive than teacher forcing (Appendix C.3).

Benefits of probabilistic extraction over discoverable extraction. Prior work has shown that probabilistic extraction provides a more nuanced measure of memorization than greedy-decoded discoverable extraction (Hayes et al., 2025b; Cooper et al., 2025). In summary, probabilistic extraction:

- **Is more realistic.** Non-deterministic sampling schemes are the norm in practice, so probabilistic extraction more accurately reflects how LLMs are actually used.
- **Identifies more extractable sequences.** Greedy decoding systematically under-counts valid extraction. High-probability training sequences can be missed simply because greedy decoding always picks the locally highest-probability token at each step. By exploring more of the model's distribution, non-deterministic decoding schemes can surface valid extraction that greedy decoding fails to detect. This gap grows with model size (Cooper et al., 2025; Hayes et al., 2025b).
- **Quantifies extraction risk.** The probability p_z is the expected frequency with which the model outputs the suffix when prompted with the prefix. For example, $p_z = 0.5$ means

the model will reproduce the suffix about once every two prompts, while $p_z = 0.001$—our chosen τ_{min}—means about once in a thousand prompts. This scalar measure provides a direct notion of **extraction risk** that can be compared across sequences. Some sequences are more extractable (higher p_z) and therefore more vulnerable. In contrast, since greedy-decoded extraction is deterministic, it collapses this information into a single bit—was the suffix extractable or not? With probabilistic extraction, it is possible to analyze how extraction risk varies across different sequences (Cooper et al., 2025). This makes it especially useful for analyzing how different types of data are at risk of leakage at generation time.

A.2.3 Computing an extraction rate

Now, let $\mathbb{Z}$ be a set of sequences $\boldsymbol{z}$ drawn from some training dataset $\mathbb{D}$ (i.e., $\mathbb{Z} \subseteq \mathbb{D}$). We can compute an **extraction rate** over $\mathbb{Z}$ with

$$\text{extraction_rate}(\mathbb{Z}; s) \triangleq \frac{1}{|\mathbb{Z}|} \sum_{\boldsymbol{z} \in \mathbb{Z}} \mathbf{1}[\, s(\boldsymbol{z}) \,], \tag{16}$$

where $s(\boldsymbol{z})$ is a success predicate defined according to the chosen criterion. In this work, for verbatim extraction, this predicate can either be the greedy success condition in Equation 12 or the more general probabilistic success condition in Equation 14. We will use extraction rates as one of the main metrics in this work. (We will also visualize distributions over extraction probabilities for a set $\mathbb{Z}$, in order to convey how risk varies across sequences.) Equation 16 is general enough to also apply to near-verbatim extraction metrics.

B Problem setup for near-verbatim extraction

The extraction metrics discussed in Appendix A all register success only when the generated continuation is an *exact* match to the target suffix. Even if one token is off—an additional space is inserted, a deleted comma—both success criteria would return 0. Because of this dependence on strict equality with the target suffix, both metrics underestimate memorization (Lee et al., 2022; Ippolito et al., 2023).

To address this, it is possible to extend both greedy-decoded discoverable extraction and probabilistic extraction (Appendix A.2) to capture instances of near-exact (**near-verbatim**) memorization. In our context of comparing generated suffixes to target suffixes of the same length T, we choose a distance metric $\text{dist} : \mathbb{V}^T \times \mathbb{V}^T \to \mathbb{R}_{\geq 0}$ and a tolerance $\varepsilon \in \mathbb{R}_{\geq 0}$. Using this metric, if the distance between the generated continuation and the target suffix is at most ε, we count the sequence as extracted.

In this appendix, we describe the two distance metrics that we consider in this paper (Appendix B.1), the success criteria for near-verbatim extraction (Appendix B.2), and considerations for computing near-verbatim extraction in practice (Appendix B.3).

B.1 Distance metrics

We consider two standard token-sequence distance measures. For $\boldsymbol{b}, \boldsymbol{c} \in \mathbb{V}^T$,

$$\text{Hamming}(\boldsymbol{b}, \boldsymbol{c}) \triangleq \sum_{t=1}^{T} \mathbf{1}[\, b_t \neq c_t \,], \tag{17}$$

$$\text{Levenshtein}(\boldsymbol{b}, \boldsymbol{c}) \triangleq \min\{\, d : \boldsymbol{b} \xrightarrow{d \text{ edits}} \boldsymbol{c} \,\}. \tag{18}$$

The Hamming distance counts positional mismatches: it computes the distance with respect to token substitutions, where each substitution has unit cost. The Levenshtein distance is more general: it allows substitutions, insertions, and deletions, each with unit cost, and is the minimum number of such edits required to transform one sequence into the other. Normalized versions of these metrics are also often useful, e.g.,

$$\text{Norm_Levenshtein}(\boldsymbol{b}, \boldsymbol{c}) \triangleq \frac{\text{Levenshtein}(\boldsymbol{b}, \boldsymbol{c})}{T} \in [0, 1].$$

Since we always consider comparisons for T-length sequences, we use unnormalized distance metrics. We will also often abbreviate Hamming as Ham and Levenshtein as Lev.

Comparative remark. For equal-length token sequences, the Hamming and Levenshtein distances need not coincide. The Hamming distance counts the number of mismatched positions, while the Levenshtein may use insertions and deletions to find a shorter edit path. For example, suppose $\boldsymbol{b}$ is obtained from $\boldsymbol{c}$ by inserting a token at the beginning and deleting one at the end. The Levenshtein assigns distance 2, while the Hamming registers T mismatches, making the two sequences appear more dissimilar than they actually are. We will revisit this below in Appendix B.3.1.

Character-based variants. Although we compute distances over tokens, the same definitions apply to characters. In that case, decoded token sequences may yield different character lengths, but the Levenshtein distance remains well-defined. (Hamming requires the two strings being compared to have equal length, so may not apply to character-based distances.) Since our analysis focuses on tokens, we omit explicit character-based definitions.

B.2 Success criteria for near-verbatim extraction

For both discoverable extraction and probabilistic extraction, we can generalize the success criteria to accommodate near-verbatim memorization.

Discoverable extraction. For the greedy-decoded continuation $\hat{\boldsymbol{z}}_{a+1:a+T}$ and target suffix $\boldsymbol{z}_{a+1:a+T}$, we define success under a chosen distance metric dist and tolerance $\varepsilon \in \mathbb{R}_{\geq 0}$ as

$$s^{\mathsf{dist}}_{\theta,\mathsf{greedy},\varepsilon}(\boldsymbol{z}) \triangleq \mathbf{1}\Big[\,\mathsf{dist}\big(\hat{\boldsymbol{z}}_{a+1:a+T}, \boldsymbol{z}_{a+1:a+T}\big) \leq \varepsilon\,\Big]. \tag{19}$$

When $\varepsilon = 0$, this reduces to the verbatim extraction success criterion in Equation 12.

Probabilistic extraction. For verbatim extraction, the success criterion depends on the probability of the exact target $p_{\boldsymbol{z}}$. Now, with tolerance ε, multiple suffixes may satisfy the success criterion—any of which may reasonably be sampled with a non-deterministic decoding scheme ϕ. We therefore need to define extraction probability as the aggregate mass over the set of all such ε-viable suffixes.

To do so, for target suffix $\boldsymbol{z}_{a+1:a+T}$, we define the **ε-ball of near-verbatim suffixes** for distance metric dist as

$$\mathbb{B}^{\mathsf{dist}}_{\varepsilon}\big(\boldsymbol{z}_{a+1:a+T}\big) \triangleq \big\{\, \boldsymbol{v} \in \mathbb{V}^T : \mathsf{dist}\big(\boldsymbol{v}, \boldsymbol{z}_{a+1:a+T}\big) \leq \varepsilon \,\big\}. \tag{20}$$

For an LLM θ and decoding scheme ϕ, the total probability mass on this set is

$$\begin{aligned} p^{\mathsf{dist}}_{\boldsymbol{z},\varepsilon} &\triangleq \Pr_{\theta,\phi}\big(\boldsymbol{v} \in \mathbb{B}^{\mathsf{dist}}_{\varepsilon}(\boldsymbol{z}_{a+1:a+T}) \,\big|\, \boldsymbol{z}_{1:a}\big) \\ &= \mathbb{E}_{\boldsymbol{v}\sim \Pr_{\theta,\phi}(\cdot|\boldsymbol{z}_{1:a})}\left[\mathbf{1}\left\{\mathsf{dist}(\boldsymbol{v}, \boldsymbol{z}_{a+1:a+T}) \leq \varepsilon\right\}\right] \\ &= \sum_{\boldsymbol{v}\in\mathbb{B}^{\mathsf{dist}}_{\varepsilon}(\boldsymbol{z}_{a+1:a+T})} \Pr_{\theta,\phi}\left(\boldsymbol{v} \mid \boldsymbol{z}_{1:a}\right). \end{aligned} \tag{21}$$

The near-verbatim success criterion is then

$$s^{\mathsf{dist}}_{\theta,\phi,\varepsilon}(\boldsymbol{z}; \tau_{\min}) \triangleq \mathbf{1}\big[\, p^{\mathsf{dist}}_{\boldsymbol{z},\varepsilon} \;\geq\; \tau_{\min}\,\big]. \tag{22}$$

When $\varepsilon = 0$, $\mathbb{B}^{\mathsf{dist}}_{0}(\boldsymbol{z}_{a+1:a+T})$ contains only the verbatim suffix, so $p^{\mathsf{dist}}_{\boldsymbol{z},0} = p_{\boldsymbol{z}}$, and Equation 22 reduces to Equation 14.

B.3 Quantifying near-verbatim extraction success in practice

Having modified the extraction success criteria to allow for near-verbatim matches to the target suffix, we next begin to discuss how one might go about instantiating these criteria in practice.

Computing greedy-decoded, near-verbatim discoverable extraction is cheap. For greedy decoding, near-verbatim success is straightforward to compute in practice: prompt with the prefix $z_{1:a}$, generate the deterministic greedy continuation $\hat{z}_{a+1:a+T}$, compute the distance to the target suffix $\mathsf{dist}(\hat{z}_{a+1:a+T}, z_{a+1:a+T})$, and declare extraction success if that distance does not exceed ε. This requires no additional forward passes through the model beyond those needed to generate the greedy continuation.[4] Because greedy decoding is deterministic, there is only one suffix to evaluate.

In contrast, computing near-verbatim probabilistic extraction exhibits other challenges.

B.3.1 Challenges for computing near-verbatim probabilistic extraction

The set of near-verbatim suffixes can be enormous. While, in principle, Equation 22 only involves a slight modification of the verbatim probabilistic extraction success criterion, in practice, it is significantly more expensive to compute.

For the Hamming distance (Equation 17), the number of T-length suffixes within radius ε around a T-length target suffix over vocabulary $\mathbb{V}$ is

$$\left|\mathbb{B}_{\varepsilon}^{\mathsf{Hamming}}(z_{a+1:a+T})\right| \;=\; \sum_{r=0}^{\varepsilon} \binom{T}{r} (|\mathbb{V}|-1)^r. \tag{23}$$

Here $\binom{T}{r}$ chooses the r substitution positions, and $(|\mathbb{V}|-1)^r$ counts all possible substitutions at those positions. For instance, consider $|\mathbb{V}| = 32{,}000$ (as is the case for LLAMA 1 models) and $T = 50$:

$$\begin{aligned}
\varepsilon = 0 \quad &: 1 && \text{(single verbatim match)},\\
\varepsilon \le 1 \quad &: 1 + \binom{50}{1}(31{,}999) && = 1{,}599{,}951,\\
\varepsilon \le 2 \quad &: 1 + \binom{50}{1}(31{,}999) + \binom{50}{2}(31{,}999)^2 && \approx 1.254 \times 10^{12}.
\end{aligned}$$

So, even when just setting $\varepsilon = 2$, the number of near-verbatim suffixes is already in the trillions.

For the Levenshtein distance (Equation 18), as noted in Appendix B.1, the computed distance can be smaller than the Hamming distance for the same pair of T-length sequences, since an insertion-deletion pair may substitute for multiple mismatches. Put differently, the Levenshtein distance can effectively "shift" a sequence by pairing a deletion with an insertion to realign tokens, whereas the Hamming distance can only compare fixed positions; as a result, the Levenshtein can need fewer edits to transform one sequence to another. Therefore, for the same ε, the Levenshtein ε-ball is always at least as large as the Hamming ε-ball, and can be strictly larger (Theorem 1).

No single-forward-pass, teacher-forced analogue. In the verbatim case, the probability p_z can be computed with a single teacher-forced forward pass through the model (Equation 15). By contrast, recall that computing $p_{z,\varepsilon}^{\mathsf{dist}}$ for near-verbatim extraction requires

$$p_{z,\varepsilon}^{\mathsf{dist}} \;\triangleq \sum_{v \in \mathbb{B}_{\varepsilon}^{\mathsf{dist}}(z_{a+1:a+T})} \Pr_{\theta,\phi}(v \mid z_{1:a}). \tag{21}$$

If we were to use the same measurement approach, this would involve one teacher-forced evaluation for each v in the ε-ball. Given the size of the Hamming ball (Equation 23)—and the potentially larger Levenshtein ball (Theorem 1)—this is prohibitively expensive even for very small ε. (We discuss cost in more detail in Appendix C.3, in the context of providing an intuition for other ways of computing near-verbatim probabilistic extraction.)

[4] This applies to the setup for computing discoverable extraction that takes a known sequence from the training data, splits it into a prefix and target suffix, and evaluates the generated continuation against the target suffix. Other approaches compare the generated continuation against an entire corpus of training data—not just the target suffix. For efficiency, such work often uses a suffix array to search for the verbatim generated continuation among the training data (Lee et al., 2022; Nasr et al., 2023; 2025). This data structure enables fast exact substring searches.

B.3.2 *Levenshtein never exceeds Hamming for T-length sequences*

We show that, for sequences of the same length, the Levenshtein distance (Equation 18) never exceeds the Hamming distance (Equation 17). As a result, for the same ε and T-length target suffix, the ε-ball of the Hamming distance is a subset of the ε-ball of the Levenshtein distance. This follows intuitively by definition: both metrics allow substitution edits, but the Levenshtein distance may find a cheaper edit path by additionally allowing for insertions and deletions.

Theorem 1. *Fix a token vocabulary $\mathbb{V}$ and $T \in \mathbb{N}$. For all $\boldsymbol{b}, \boldsymbol{c} \in \mathbb{V}^T$,*

$$\mathsf{Levenshtein}(\boldsymbol{b}, \boldsymbol{c}) \;\leq\; \mathsf{Hamming}(\boldsymbol{b}, \boldsymbol{c}). \tag{24}$$

Further:

1. ***(Inclusion)** For every $\boldsymbol{b} \in \mathbb{V}^T$ and $\varepsilon \geq 0$, $\mathbb{B}_\varepsilon^{\mathsf{Ham}}(\boldsymbol{b}) \subseteq \mathbb{B}_\varepsilon^{\mathsf{Lev}}(\boldsymbol{b})$.*
2. ***(Equality for $\varepsilon \in \{0, 1\}$)** If $\varepsilon \in \{0, 1\}$, then, for every $\boldsymbol{b} \in \mathbb{V}^T$, $\mathbb{B}_\varepsilon^{\mathsf{Ham}}(\boldsymbol{b}) = \mathbb{B}_\varepsilon^{\mathsf{Lev}}(\boldsymbol{b})$.*
3. ***(Strictness for $\varepsilon \in \{2, \ldots, T-1\}$; existence)** If $T \geq 3$ and $|\mathbb{V}| \geq T$, then there exists $\boldsymbol{b} \in \mathbb{V}^T$ such that for every ε with $2 \leq \varepsilon < T$,*

$$\mathbb{B}_\varepsilon^{\mathsf{Ham}}(\boldsymbol{b}) \;\subsetneq\; \mathbb{B}_\varepsilon^{\mathsf{Lev}}(\boldsymbol{b}).$$

4. ***(Saturation)** For every $\boldsymbol{b} \in \mathbb{V}^T$, every $\varepsilon \geq T$, and $\mathsf{dist} \in \{\mathsf{Hamming}, \mathsf{Levenshtein}\}$, $\mathbb{B}_\varepsilon^{\mathsf{dist}}(\boldsymbol{b}) = \mathbb{V}^T$.*

Proof. First, note that if $\boldsymbol{b}$ and $\boldsymbol{c}$ differ at $r = \mathsf{Hamming}(\boldsymbol{b}, \boldsymbol{c})$ positions, then performing those r position-wise substitutions is a valid edit script from $\boldsymbol{b}$ to $\boldsymbol{c}$ of cost r (Equation 17). By the definition of the Levenshtein distance (Equation 18), $\mathsf{Levenshtein}(\boldsymbol{b}, \boldsymbol{c}) \leq r$.We can now take each case in turn.

1. **(Inclusion)** By Equation 24, if $\boldsymbol{c} \in \mathbb{B}_\varepsilon^{\mathsf{Ham}}(\boldsymbol{b})$, then $\mathsf{Levenshtein}(\boldsymbol{b}, \boldsymbol{c}) \leq \mathsf{Hamming}(\boldsymbol{b}, \boldsymbol{c}) \leq \varepsilon$. And so $\boldsymbol{c} \in \mathbb{B}_\varepsilon^{\mathsf{Lev}}(\boldsymbol{b})$—i.e., every member of $\mathbb{B}_\varepsilon^{\mathsf{Ham}}(\boldsymbol{b})$ must also be a member of $\mathbb{B}_\varepsilon^{\mathsf{Lev}}(\boldsymbol{b})$.
2. **(Equality for $\varepsilon \in \{0, 1\}$)** For $\varepsilon = 0$ both ε-balls are $\{\boldsymbol{b}\}$.

   For $\varepsilon = 1$, this corresponds to $\mathsf{Levenshtein}(\boldsymbol{b}, \boldsymbol{c}) \leq 1$ with $\boldsymbol{b}, \boldsymbol{c} \in \mathbb{V}^T$. In this case, the single edit cannot be an insertion or deletion, as this would change the length to $T+1$ or $T-1$, respectively. So, the only possibilities are either (a) no edit (i.e., $\boldsymbol{b} = \boldsymbol{c}$) or (b) a single substitution. Therefore, it is also the case that $\mathsf{Hamming}(\boldsymbol{b}, \boldsymbol{c}) \leq 1$. Combining with (1), this means the two ε-balls are equal.
3. **(Strictness for $\varepsilon \in \{2, \ldots, T-1\}$; existence)** The reason $\mathbb{B}_\varepsilon^{\mathsf{Ham}}(\boldsymbol{b})$ can be strictly contained in $\mathbb{B}_\varepsilon^{\mathsf{Lev}}(\boldsymbol{b})$ is that Levenshtein distance allows an insertion-deletion pair to "shift" tokens and realign two sequences. For example, if $T \geq 3$ and the vocabulary $\mathbb{V}$ is large enough to choose pairwise distinct tokens $v_1, \ldots, v_T$, consider

   $$\boldsymbol{b} \coloneqq (v_1, v_2, \ldots, v_T), \qquad \boldsymbol{c} \coloneqq (v_2, v_3, \ldots, v_T, v_1).$$

   Then $\mathsf{Hamming}(\boldsymbol{b}, \boldsymbol{c}) = T$, because every position mismatches. But $\boldsymbol{c}$ can be obtained from $\boldsymbol{b}$ by deleting v_1 and then inserting it at the end, i.e., with two edits

   $$\boldsymbol{b} \xrightarrow{\text{delete } v_1} (v_2, \ldots, v_T) \xrightarrow{\text{insert } v_1 \text{ at end}} \boldsymbol{c},$$

   so $\mathsf{Levenshtein}(\boldsymbol{b}, \boldsymbol{c}) \leq 2$. Since $\boldsymbol{b} \neq \boldsymbol{c}$ (eliminating $\varepsilon = 0$) and a single edit cannot map one T-token sequence to another unless it is a substitution, which here can fix at most one of the $T \geq 3$ mismatched positions (eliminating $\varepsilon = 1$), we have $\mathsf{Levenshtein}(\boldsymbol{b}, \boldsymbol{c}) = 2$. Thus, for any $\varepsilon \in \{2, \ldots, T-1\}$, $\boldsymbol{c}$ lies in $\mathbb{B}_\varepsilon^{\mathsf{Lev}}(\boldsymbol{b})$ but not in $\mathbb{B}_\varepsilon^{\mathsf{Ham}}(\boldsymbol{b})$, proving strict inclusion for this $\boldsymbol{b}$.

4. **(Saturation)** For any $\boldsymbol{c} \in \mathbb{V}^T$, by definition, $\mathsf{Hamming}(\boldsymbol{b}, \boldsymbol{c}) \leq T$, since there are at most T mismatched positions. And so, by Equation 24, $\mathsf{Levenshtein}(\boldsymbol{b}, \boldsymbol{c}) \leq \mathsf{Hamming}(\boldsymbol{b}, \boldsymbol{c}) \leq T$. Therefore, once $\varepsilon \geq T$, every $\boldsymbol{c}$ is included; the ε-ball stabilizes to be every sequence in $\mathbb{V}^T$.

□

C An intuition for more efficient near-verbatim probabilistic extraction

Even though the set of near-verbatim suffixes can be enormous (Appendix B.3.1), many sequences in that set will be very unlikely (if not 0-probability) under the model θ and decoding scheme ϕ. Imagine taking a particular piece of a famous, memorized text and replacing the token for `_the` with the token for `_jazz`, `_eight`, or `_github`, using the LLAMA 1 tokenizer; those would all be within $\varepsilon \leq 1$, but are very unlikely to be sequences with any meaningful probability. The point is, we should be able to derive a significantly more efficient approach to estimating near-verbatim probabilistic extraction—one that does not enumerate the entire $\mathbb{B}^{\mathsf{dist}}_{\varepsilon}$, but only sequences with non-zero or non-trivial mass.

In this appendix, we first describe a simple Monte Carlo baseline that samples continuations from θ under ϕ (Appendix C.1). While this is an intuitive approach, it turns out to be prohibitively expensive for producing useful and reliable estimates of $p^{\mathsf{dist}}_{\boldsymbol{z},\varepsilon}$. We then describe the intuition behind the approach that we take in this paper—a modification of beam search that produces a deterministic lower bound on $p^{\mathsf{dist}}_{\boldsymbol{z},\varepsilon}$ at a fraction of the cost (Appendix C.2). We provide quantitative intuition for why this approach succeeds in practice, and compare it with MC sampling. Finally, we define a common unit of cost—token evaluations—that allows direct comparison across methods (Appendix C.3). This shows why our beam-based approach is significantly cheaper than MC in terms of token evaluations, and enables us to compare with greedy discoverable extraction. We do not go into formal details about our beam-search-based algorithm here, and instead defer this discussion to Appendices D and E.

C.1 Monte Carlo sampling

Instead of enumerating the entire set $\mathbb{B}^{\mathsf{dist}}_{\varepsilon}(\boldsymbol{z}_{a+1:a+T})$ and computing the conditional probability for each continuation given the prefix, a straightforward alternative is to estimate $p^{\mathsf{dist}}_{\boldsymbol{z},\varepsilon}$ via sampling. Denote a training-data sequence $\boldsymbol{z}$, composed of the a-length prefix $\boldsymbol{z}_{1:a} \equiv \boldsymbol{z}_{(\text{pre})}$ and the T-length target suffix $\boldsymbol{z}_{a+1:a+T} \equiv \boldsymbol{z}_{(\text{suf})}$—i.e., $\boldsymbol{z} \coloneqq \boldsymbol{z}_{(\text{pre})} \| \boldsymbol{z}_{(\text{suf})}$. Denote a generated T-length continuation $\hat{\boldsymbol{z}}_{a+1:a+T} \equiv \hat{\boldsymbol{z}}_{(\text{cont})}$. Given prefix $\boldsymbol{z}_{(\text{pre})}$, sample M i.i.d. T-length continuations $\{\hat{\boldsymbol{z}}^{(1)}_{(\text{cont})}, \ldots, \hat{\boldsymbol{z}}^{(M)}_{(\text{cont})}\} \sim \Pr_{\theta,\phi}(\cdot \mid \boldsymbol{z}_{(\text{pre})})$. Define the empirical, sampling-based estimate for $p^{\mathsf{dist}}_{\boldsymbol{z},\varepsilon}$ as

$$\hat{p}^{\mathsf{dist}}_{\boldsymbol{z},\varepsilon} \;=\; \frac{1}{M} \sum_{w=1}^{M} \mathbf{1}\big[\, \mathsf{dist}(\hat{\boldsymbol{z}}^{(w)}_{(\text{cont})}, \boldsymbol{z}_{(\text{suf})}) \leq \varepsilon \,\big], \quad \text{where } \hat{\boldsymbol{z}}^{(w)}_{(\text{cont})} \underset{\text{i.i.d. } \theta,\phi}{\sim} \Pr(\cdot \mid \boldsymbol{z}_{(\text{pre})}). \tag{25}$$

That is, we prompt the model M times with the prefix $\boldsymbol{z}_{(\text{pre})}$ and estimate $p^{\mathsf{dist}}_{\boldsymbol{z},\varepsilon}$ as the proportion of sampled continuations (according to decoding scheme ϕ) lying within the ε-ball of the target suffix. This estimator is unbiased—$\mathbb{E}[\hat{p}^{\mathsf{dist}}_{\boldsymbol{z},\varepsilon}] = p^{\mathsf{dist}}_{\boldsymbol{z},\varepsilon}$—and straightforward to compute. However, there are two important reasons why it is expensive for estimating near-verbatim extraction probabilities.

1) There is no guarantee MC will sample from a high-probability ε-ball. Each of the M i.i.d. samples independently either hits or misses the ε-ball. The probability of *never* hitting it in M samples is

$$\Pr\left(\text{miss } \mathbb{B}^{\mathsf{dist}}_{\varepsilon}(\boldsymbol{z}_{(\text{suf})})\right) = (1 - p^{\mathsf{dist}}_{\boldsymbol{z},\varepsilon})^M. \tag{26}$$

To guarantee a miss probability of at most δ, we require

$$(1 - p^{\mathsf{dist}}_{\boldsymbol{z},\varepsilon})^M \leq \delta \iff M \geq \frac{\ln(1/\delta)}{-\ln(1 - p^{\mathsf{dist}}_{\boldsymbol{z},\varepsilon})}.$$

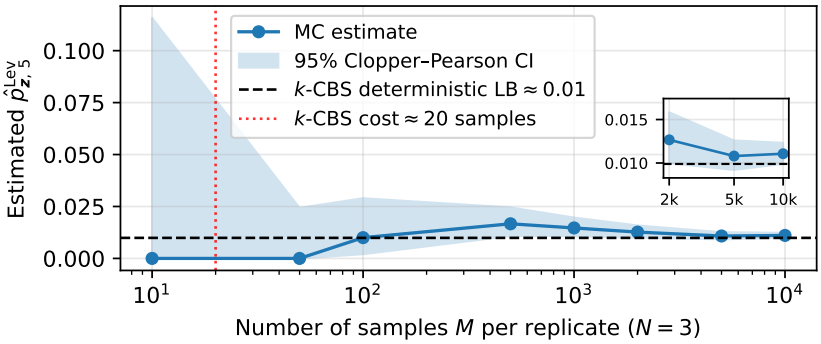


Figure 8: **Monte Carlo (MC) estimation of near-verbatim extraction probability.** For Levenshtein distance ≤ 5 ($p^{\text{Lev}}_{z,5}$), we plot convergence for a single sequence z from *The Great Gatsby* for LLAMA 2 7B, showing the pooled MC estimate with a 95% confidence interval over 3 replicates. Our algorithm (k-CBS) produces a deterministic, provably correct lower bound (LB) of $\approx$0.01, which captures 89.4% of the mean MC estimate at $M = 10^4$ samples, at a cost of $\approx$20 MC samples (see Appendix C.3 for cost accounting)—a budget at which MC produces no hits.

Table 1 shows the minimum number of samples required for several values of $p^{\text{dist}}_{z,\varepsilon}$ and δ, computed using this formula.

Table 1: **Samples required to hit at least once.** Minimum number of MC samples M to hit the ε-ball at least once with probability $\geq 1 - \delta$.

$p^{\text{dist}}_{z,\varepsilon}$	$\lceil M \rceil$		
	$\delta = 0.005$	$\delta = 0.05$	$\delta = 0.5$
10^{-1}	51	29	7
10^{-2}	528	299	69
10^{-3}	5,296	2,995	693

Even for modest miss tolerance (e.g., $\delta = 0.05$), merely *detecting* an ε-ball with mass $p^{\text{dist}}_{z,\varepsilon} = 10^{-3}$ requires on the order of 3,000 samples. Note, however, that detection only guarantees *one* hit; it does not guarantee that the MC estimate of $p^{\text{dist}}_{z,\varepsilon}$ is accurate.

2) Even when MC hits the ε-ball, the estimate $\hat{p}^{\text{dist}}_{z,\varepsilon}$ can be unreliable. Each term in Equation 25 is a Bernoulli random variable with success probability $p^{\text{dist}}_{z,\varepsilon}$. Because $\hat{p}^{\text{dist}}_{z,\varepsilon}$ is the mean of M such i.i.d. variables, its variance and standard error are

$$\text{Var}[\hat{p}^{\text{dist}}_{z,\varepsilon}] = \frac{p^{\text{dist}}_{z,\varepsilon}(1 - p^{\text{dist}}_{z,\varepsilon})}{M}, \qquad \text{SE}[\hat{p}^{\text{dist}}_{z,\varepsilon}] = \sqrt{\frac{p^{\text{dist}}_{z,\varepsilon}(1 - p^{\text{dist}}_{z,\varepsilon})}{M}}.$$

To achieve a relative standard error of η (i.e., $\text{SE}[\hat{p}^{\text{dist}}_{z,\varepsilon}] \leq \eta \cdot p^{\text{dist}}_{z,\varepsilon}$), we need

$$\sqrt{\frac{p^{\text{dist}}_{z,\varepsilon}(1 - p^{\text{dist}}_{z,\varepsilon})}{M}} \leq \eta \cdot p^{\text{dist}}_{z,\varepsilon} \iff M \geq \frac{1 - p^{\text{dist}}_{z,\varepsilon}}{\eta^2\, p^{\text{dist}}_{z,\varepsilon}} \approx \frac{1}{\eta^2\, p^{\text{dist}}_{z,\varepsilon}} \quad \text{when } p^{\text{dist}}_{z,\varepsilon} \text{ is small.}$$

For the minimum extraction threshold $p^{\text{dist}}_{z,\varepsilon} \approx \tau_{\min} = 10^{-3}$ and $\eta = 10\%$, this gives $M \gtrsim 10^5$; for $\eta = 1\%$, $M \gtrsim 10^7$. In other words, *reliable* estimation of small $p^{\text{dist}}_{z,\varepsilon}$ is orders of magnitude more expensive than merely detecting the ε-ball.

Empirical illustration. We illustrate this cost empirically in Figure 8, which shows MC convergence for a single sequence from *The Great Gatsby* under LLAMA 2 7B with top-k ($k = 40$) decoding. At 30 pooled samples (across $N = 3$ replicates at 10^1), the 95%

confidence interval for $p^{\mathsf{dist}}_{z,\varepsilon}$ spans from 0 to over 0.10—the estimate is consistent with both "no extraction" and "10% extraction probability." The estimate only stabilizes near the true value after thousands of samples. By contrast, our approach produces a deterministic lower bound on $p^{\mathsf{dist}}_{z,\varepsilon}$ at a cost equivalent to roughly 20 MC samples.

While MC sampling is unbiased and simple, its cost scales as $1/p^{\mathsf{dist}}_{z,\varepsilon}$ for detection and roughly $1/(\eta^2\, p^{\mathsf{dist}}_{z,\varepsilon})$ for reliable estimation—prohibitive at scale even when $p^{\mathsf{dist}}_{z,\varepsilon}$ is moderately large (e.g., ≈ 0.01; see Figure 8). The fundamental limitation is that MC explores the sequence space blindly: most samples land far from the ε-ball and contribute no useful information. This motivates an alternative approach that concentrates computation on the high-mass region of the sequence space, yielding a deterministic lower bound on $p^{\mathsf{dist}}_{z,\varepsilon}$ at a fraction of the cost.

C.2 An intuition for using a decoding-constrained beam search approach

In this work, we propose a method for obtaining a useful, efficient lower bound on near-verbatim extraction probability. With far fewer evaluations than MC sampling would require, we use a variant of **beam search** to produce a downward-biased (but still correct) lower bound on $p^{\mathsf{dist}}_{z,\varepsilon}$. We provide brief background on beam search (Appendix C.2.1), describe the modifications that turn it into an efficient procedure for lower-bounding $p^{\mathsf{dist}}_{z,\varepsilon}$ (Appendix C.2.2), and then provide mathematical intuition for why this approach succeeds in practice (Appendix C.2.3).

C.2.1 Beam search

Beam search (Lowerre, 1976) is a standard decoding algorithm for autoregressive language models that approximately maximizes the conditional probability of a continuation given a prefix (Graves, 2012; Boulanger-Lewandowski et al., 2013; Sutskever et al., 2014; Bahdanau et al., 2016). Denote the length-$(t-1)$ prefix of $\hat{z}_{(\text{cont})}$ by $\hat{z}^{(\text{cont})}_{<t} := (\hat{z}^{(\text{cont})}_1, \ldots, \hat{z}^{(\text{cont})}_{t-1})$. Then, for $z_{(\text{pre})}$ and $\hat{z}_{(\text{cont})}$, the model θ defines

$$\Pr_\theta(\hat{z}_{(\text{cont})} \mid z_{(\text{pre})}) \;=\; \prod_{t=1}^{T} \Pr_\theta\big(\hat{z}^{(\text{cont})}_t \,\big|\, z_{(\text{pre})} \| \hat{z}^{(\text{cont})}_{<t}\big),$$

or, in log space,

$$\log \Pr_\theta(\hat{z}_{(\text{cont})} \mid z_{(\text{pre})}) \;=\; \sum_{t=1}^{T} \log \Pr_\theta\big(\hat{z}^{(\text{cont})}_t \,\big|\, z_{(\text{pre})} \| \hat{z}^{(\text{cont})}_{<t}\big).$$

Beam search maintains, at each depth t, a **beam** of at most B partial continuations, each with its accumulated log-probability. At $t = 0$ the beam contains only the prefix $z_{(\text{pre})}$ with score 0. At step t:

1. For each partial history $\hat{z}$ in the beam ($z_{(\text{pre})} \| \hat{z}^{(\text{cont})}_{<t}$, the prefix concatenated with the generated continuation so far), the model produces next-token probabilities $\Pr_\theta(\hat{z} \mid \hat{z})$ over $\hat{z} \in \mathbb{V}$.
2. Each beam element is expanded by each of these tokens, yielding candidate children $\hat{z}' = \hat{z} \| \hat{z}$ with updated scores (the conditional log probability of the generated continuation so far, given the prefix):
$$\log p(\hat{z}') \;=\; \log p(\hat{z}) + \log \Pr_\theta(\hat{z} \mid \hat{z}).$$
3. For the next iteration $(t+1)$, beam search performs an across-beam prune to keep only (at most) the B unique highest-scoring partial sequences and discards the rest. This is done for efficiency, i.e., to prevent explosive blow-up of the number of sequences under consideration.

After T steps, the algorithm returns the B highest-scoring complete continuations in the beam as approximate maximum-probability sequences under the model.

Why beam search favors high-probability sequences. The joint log-probability of a continuation decomposes additively across time steps. Therefore, a high-probability continuation must maintain a high cumulative log-probability at *every* depth in order to remain competitive and stay in the beam. If at some step t a partial continuation falls far behind many competing sequences, any extension of it will remain disadvantaged when ranked.

Beam search exploits this structure by pruning low-scoring partial continuations as soon as they fall outside the current top-B. Partial continuations that consistently receive high next-token probabilities remain in the top-B and are expanded further; those that accumulate several low-probability token choices are quickly outrun by alternatives and permanently removed via the across-beam prune.

In this sense, beam search concentrates computational effort on a thin, high-mass region of candidate continuations rather than exploring the full, exponentially large space of possible sequences. Our approach leverages this property: we impose additional constraints to turn beam search into an efficient procedure for lower-bounding the near-verbatim extraction probability $p^{\mathsf{dist}}_{z,\varepsilon}$.

C.2.2 Modifying beam search with our approach

The general idea behind our modification is simple. Rather than scoring candidate sequences under the full model distribution, we implement expansion and scoring to respect the decoding policy ϕ that we choose for computing $p^{\mathsf{dist}}_{z,\varepsilon}$. In doing so, beam search (with width B) will return B candidate sequences that are viable under ϕ, with their associated probabilities under the ϕ-constrained and renormalized probability distribution. (In fact, we return $B \cdot k$ such candidates by not performing the last across-beam prune at step T.)

In particular, for top-k decoding, we keep a beam of width B. For a partial history $\hat{z}$ in the beam at step t, we denote

$$\mathrm{S}_t(\hat{z}) := \mathsf{TopK}_k\big(\boldsymbol{y}_t(\hat{z})\big)$$

to be the set of k tokens in $\mathbb{V}$ with the largest logits in $\boldsymbol{y}_t(\hat{z})$ (renaming $\mathbb{V}_{t,k}$ from Equation 11 and Algorithm 1). At each step t, maintain a beam of at most B partial continuations, each with its accumulated log-probability. At $t = 0$ the beam contains only the prefix $z_{(\mathrm{pre})}$ with score 0. At step t:

1. For each partial history $\hat{z}$ in the beam, the model θ and top-k decoding ϕ produce next-token probabilities $\Pr_{\theta,\phi}(\hat{z} \mid \hat{z})$ over only the top-k-token set, $\hat{z} \in \mathrm{S}_t(\hat{z})$.
2. Each beam element is expanded by each of these k tokens, yielding $B \cdot k$ candidate children $\hat{z}' = \hat{z} \| \hat{z}$ with updated scores

$$\log p(\hat{z}') \;=\; \log p(\hat{z}) + \log \Pr_{\theta,\phi}(\hat{z} \mid \hat{z}).$$

3. For the next iteration $(t+1)$, beam search performs an across-beam prune to keep only the B unique highest-scoring partial sequences and discards the rest.

After T steps, the algorithm returns all $B \cdot k$ complete continuations (i.e., without performing a final across-beam prune). Similar to traditional beam search, these continuations approximate the maximum-probability sequences under the truncated distribution induced by the model and the top-k decoding policy. We refer to this procedure as **top-k constrained beam search**, or k-CBS for short, and we refine it to attempt to get tighter bounds on $p^{\mathsf{dist}}_{z,\varepsilon}$ throughout this paper. Note, also, that we could use other decoding policies, though we focus on top-k in this work (Appendix D.3).

Connection to memorization. Intuitively, k-CBS should recover near-verbatim (and possibly the exact verbatim) suffixes whenever the sequence is memorized. When memorized, the training-data suffix $z_{(\mathrm{suf})}$ and small perturbations of it in $\mathbb{B}^{\mathsf{dist}}_{\varepsilon}(z_{(\mathrm{suf})})$ typically carry unusually high conditional probability under $\Pr_{\theta,\phi}(\cdot \mid z_{(\mathrm{pre})})$, so the partial continuations corresponding to those suffixes remain among the highest-scoring across depths. As a result,

for a comfortably sized beam width B, these near-verbatim paths are unlikely to be pruned; they are likely to survive to the final beam as B high-probability candidates. We can then post-process the returned continuations by applying a distance filter $\mathsf{dist}(\hat{\boldsymbol{z}}_{(\text{cont})}, \boldsymbol{z}_{(\text{suf})}) \leq \varepsilon$. That is, let $\mathbb{F}^{(\leq\varepsilon)}$ denote the set of filtered continuations; we can sum their masses,

$$\sum_{\hat{\boldsymbol{z}}_{(\text{cont})} \in \mathbb{F}^{(\leq\varepsilon)}} \Pr_{\theta,\phi}(\hat{\boldsymbol{z}}_{(\text{cont})} \mid \boldsymbol{z}_{(\text{pre})}),$$

yielding a deterministic, rigorous (but biased-downward) lower bound on $p^{\mathsf{dist}}_{\boldsymbol{z},\varepsilon}$ (with respect to θ and top-k for ϕ). We will miss (potentially many) near-verbatim suffixes with this approach, but if the sequence is memorized and therefore high probability, its (near-)verbatim variants should appear in the final beam. (Again, we actually return all $B \cdot k$ candidates from the last iteration, rather than performing the final across-beam prune to width B, as these are all T-length continuations under the decoding policy.)

Moreover, rather than applying the distance filter only at the end, we can incorporate ε-viability filtering *during* the search: at each depth, prune any partial continuation that can no longer yield a final suffix within distance ε of $\boldsymbol{z}_{(\text{suf})}$. This further focuses the beam on near-verbatim paths, freeing beam capacity and potentially yielding tighter lower bounds (we develop these pruned variants in Appendix E).

While it is typically a drawback that beam search does not always produce diverse outputs (Li et al., 2016; Vijayakumar et al., 2018), in our setting this is a *feature*: we explicitly focus on a small set of high-probability continuations that lie close to the training suffix, rather than exploring a wide variety of semantically different but low-probability alternatives.

Working assumptions (informal). Informally, k-CBS should produce a useful lower bound on $p^{\mathsf{dist}}_{\boldsymbol{z},\varepsilon}$, given that two conditions hold (which are both realistic for a memorized sequence):

1. **Token rank.** At most steps t, the true suffix token z_t lies in the model's top-k set, and deviations are rare enough that the full continuation remains within $\mathbb{B}^{\mathsf{dist}}_{\varepsilon}(\boldsymbol{z}_{(\text{suf})})$.
2. **Beam dominance.** The verbatim path and/or its near-verbatim variants maintain cumulative log-probabilities that stay within the beam's top-B at each depth, so they survive all across-beam prunes.

We formalize these conditions quantitatively in Appendix C.2.3 below, and show that they follow naturally from the structure of high-mass continuations without distributional assumptions.

Cost comparison to MC. Even though k-CBS provides a downward-biased lower bound on $p^{\mathsf{dist}}_{\boldsymbol{z},\varepsilon}$, it tends to recover more probability mass than MC sampling at smaller compute budgets because it concentrates computation on the (greedily) highest-mass continuations. We defer a detailed cost comparison to Appendix C.3.

C.2.3 Mathematical intuition for good performance in practice

The algorithm (Appendices D & E) guarantees that the downward-biased lower bound is *valid*—it never exceeds the true $p^{\mathsf{dist}}_{\boldsymbol{z},\varepsilon}$. But a lower bound of 0 is also valid. So, why should we expect the lower bound to be *useful*—to capture a substantial fraction of $p^{\mathsf{dist}}_{\boldsymbol{z},\varepsilon}$? In this appendix, we provide quantitative intuition. The key observation is that any continuation with probability $\geq \tau_{\min}$ (our extraction threshold)—or even slightly smaller—must pick high-rank tokens at almost every step. As a result, such continuations take up spots in the beam; it is likely that our algorithm retains them.

Setup. Consider a complete history $\hat{\boldsymbol{z}} := (\hat{z}_1, \ldots, \hat{z}_T)$. We write $\hat{\boldsymbol{z}} := \boldsymbol{z}_{(\text{pre})} \| \hat{\boldsymbol{z}}_{(\text{cont})}$, where $\boldsymbol{z}_{(\text{pre})}$ is the training-data prefix and $\hat{\boldsymbol{z}}_{(\text{cont})}$ is a generated continuation given $\boldsymbol{z}_{(\text{pre})}$. The

probability of such a continuation decomposes as

$$\Pr_\theta(\hat{z}_{(\text{cont})} \mid z_{(\text{pre})}) = \prod_{t=1}^{T} \Pr_\theta(\hat{z}_t^{(\text{cont})} \mid z_{(\text{pre})} \| \hat{z}_{<t}^{(\text{cont})}),$$

where $\hat{z}_{<t}^{(\text{cont})}$ denotes the length-$(t-1)$ prefix of $\hat{z}_{(\text{cont})}$. Abbreviate

$$p_t \coloneqq \Pr_\theta(\hat{z}_t^{(\text{cont})} \mid z_{(\text{pre})} \| \hat{z}_{<t}^{(\text{cont})})$$

for the conditional probability of the realized token at step t.

Lemma 2 (Low-probability budget from a sequence-mass floor τ)**.** *Fix a minimum mass threshold $\tau \in (0, 1]$. Let $\hat{z}_{(\text{cont})} \in \mathbb{V}^T$ be a continuation with $\Pr_\theta(\hat{z}_{(\text{cont})} \mid z_{(\text{pre})}) \geq \tau$. Then for any $\alpha \in (0, 1)$,*

$$\#\{\, t \leq T : \; p_t \leq \alpha \,\} \;\leq\; \left\lfloor \frac{\ln \tau}{\ln \alpha} \right\rfloor.$$

Proof. If exactly c of the p_t are $\leq \alpha$ and on all other steps we use the bound $p_t \leq 1$, then

$$\Pr_\theta(\hat{z}_{(\text{cont})} \mid z_{(\text{pre})}) = \prod_{t=1}^{T} p_t \leq \alpha^c.$$

The assumption that $\Pr_\theta(\hat{z}_{(\text{cont})} \mid z_{(\text{pre})}) \geq \tau$ forces $\alpha^c \geq \tau$. Taking the ln (noting that $\ln \alpha < 0$ for $\alpha \in (0, 1)$),

$$\ln \tau \;\leq\; c \ln \alpha \quad\Longrightarrow\quad c \;\leq\; \frac{\ln \tau}{\ln \alpha},$$

and the integer bound follows by applying $\lfloor \cdot \rfloor$. □

We can translate this point to the rank-ordering of tokens at every step.

Corollary 3 (Rank-bucket consequences)**.** *At step t, let $r_t \in \{1, 2, \ldots, |\mathbb{V}|\}$ be the rank of the realized token (1 = highest), and let p_t be its conditional probability. Then for any integer $R \geq 2$ and any $\alpha \in (0, 1)$:*

(a) ***Outside-head budget.*** *Since $r_t \geq R \Rightarrow p_t \leq 1/R$ (by the pigeonhole principle on the probability simplex),*

$$\#\{\, t \leq T : \; r_t \geq R \,\} \;\leq\; \left\lfloor \frac{\ln \tau}{\ln(1/R)} \right\rfloor.$$

(b) ***Head coverage via a probability floor.***

$$\#\{\, t \leq T : \; p_t > \alpha \,\} \;\geq\; T \;-\; \left\lfloor \frac{\ln \tau}{\ln \alpha} \right\rfloor, \quad \textit{and on those steps} \quad r_t \leq \left\lfloor \tfrac{1}{\alpha} \right\rfloor.$$

Proof. **(a) Outside-head budget.** Fix a step t and sort the step-t probabilities in nonincreasing order as $p_{(1)}^{(t)} \geq p_{(2)}^{(t)} \geq \ldots \geq p_{(|\mathbb{V}|)}^{(t)}$. Because the probabilities at step t sum to 1,

$$\sum_{i=1}^{R} p_{(i)}^{(t)} \;\leq\; 1 \quad\Longrightarrow\quad R \cdot p_{(R)}^{(t)} \;\leq\; 1 \quad\Longrightarrow\quad p_{(R)}^{(t)} \;\leq\; \frac{1}{R}.$$

If the realized token has rank $r_t \geq R$, then by definition $p_t \leq p_{(R)}^{(t)} \leq 1/R$. Let $c \coloneqq \#\{t \leq T : r_t \geq R\}$, i.e., c is the number of steps where the rank of the realized token meets or exceeds R. On those c steps, $p_t \leq 1/R$. (Apply Lemma 2 with $\alpha = 1/R$.)

(b) Head coverage. This is the complement of Lemma 2. That is, set $c \;=\; \#\{\, t \leq T : \; p_t \leq \alpha \,\} \;\leq\; \left\lfloor \frac{\ln \tau}{\ln \alpha} \right\rfloor$. Since there are exactly T steps,

$$\#\{t \leq T : p_t > \alpha\} \;=\; T - c \;\geq\; T - \left\lfloor \tfrac{\ln \tau}{\ln \alpha} \right\rfloor.$$

On steps with $p_t > \alpha$, at most $\lfloor 1/\alpha \rfloor$ tokens can have probability $\geq \alpha$ (otherwise the total mass at that step would exceed 1), and so $r_t \leq \lfloor 1/\alpha \rfloor$. □

Note that we write the above results in terms of the base model distribution θ. Nevertheless, these results hold for *any* model θ and scoring rule ϕ that induces a probability distribution—we deliberately avoid distributional assumptions so that the intuition applies broadly. To translate these for a decoding rule ϕ, the ranks of realized tokens are limited to a subset of $\{1, 2, \ldots, |\mathbb{V}|\}$—i.e., a subset of the size of the vocabulary (e.g, k for top-k).

Instantiations for $\tau = \tau_{\min} = 10^{-3}$ and $T = 50$.

- **Greedy steps** ($\alpha = 0.5$). At most $\lfloor \ln 10^{-3} / \ln 0.5 \rfloor = 9$ steps can have $p_t \leq 0.5$. On any step, at most 1 token can have probability > 0.5; this is the case for the remaining ≥ 41 steps—i.e., at each of these steps, it is necessarily the model's top-1 (greedy) token.
- **Top-10 steps** ($\alpha = 0.1$). At most 3 steps can have $p_t \leq 0.1$; on the remaining ≥ 47 steps, the realized token has rank $r_t \leq 10$.
- **Top-k steps** ($R = 40$). By Corollary 3(a), at most $\lfloor \ln 10^{-3} / \ln(1/40) \rfloor = 1$ step can have the realized token at rank ≥ 40. Under top-k decoding with $k = 40$, this means at most one step where the realized token is at the very bottom of the top-k set; on all other steps it lies strictly above rank 40.

These bounds are deliberately worst-case. For instance, the $\alpha = 0.1$ case allows a highly atypical situation in which ten different tokens all have probability exactly 0.1 at many steps. In realistic, extracted model outputs, the distribution is much peakier. Therefore, Lemma 2 and Corollary 3 are conservative: they only require that high-mass continuations use high-ranked tokens on almost all steps.

Why high-mass paths survive the beam. The instantiations above show that a continuation with probability $\geq \tau_{\min}$ is constrained to pick high-rank tokens on almost every step. Concretely, the geometric mean of the per-token conditional probabilities for such a continuation is

$$\left(\tau_{\min}\right)^{1/T} = (10^{-3})^{1/50} \approx 0.87,$$

meaning the typical (geometric-mean) per-step conditional probability is roughly 87%. (For $\tau = 10^{-4}$, the geometric mean is still $\approx 83\%$.)

For a language model, sustaining $\approx 87\%$ conditional probability over 50 consecutive tokens is very high—matching the case of memorization. We observe empirically that continuations at this mass level are overwhelmingly (near-)verbatim matches to the training data. (We confirm this with negative-control experiments on non-training data, where k-CBS does not detect near-verbatim matches above $\tau_{\min}$.) Such continuations maintain competitive cumulative log-probabilities at every depth, because they consistently pick high-rank tokens. With beam width $B = 20$ (what we choose in practice, see Appendix F.7), the beam has ample capacity to retain these paths.

A path is pruned from the beam only if B other paths outrank it at some depth—a point we make precise below. Because the rank-budget constraints force high-mass paths to stay near the top of the cumulative-probability ranking at each step, these paths naturally maintain their competitive position across depths. This is the core reason our approach of decoding constrained beam search produces useful (not just valid) lower bounds for memorized sequences, and it is fundamentally different from MC sampling, which has no mechanism to concentrate on these paths.

A rigorous floor: heavy-mass path survival. We can prove a rigorous sufficient condition for very high-mass sequences, requiring no assumptions beyond the mass conservation property of the top-k tree (proven in Appendix D).

Lemma 4 (Heavy-mass path survival under across-beam top-B)**.** *Fix a depth $t \in \{1, \ldots, T\}$. Let $\mathbb{C}_t$ be the candidate set at depth t (the union of all children formed from the beam at depth $t-1$, possibly after any optional ε-viability filtering). Let $p'(\cdot) > 0$ denote the cumulative path probability used for ranking.*

If a length-t partial continuation $\hat{\boldsymbol{z}}_{1:t} \in \mathbb{C}_t$ satisfies

$$p'(\hat{\boldsymbol{z}}_{1:t}) \;>\; \frac{1}{B+1},$$

then $\hat{\boldsymbol{z}}_{1:t}$ is kept by the across-beam top-B selection at depth t (with deterministic tie-breaking). Therefore, if a full path $\hat{\boldsymbol{z}}_{1:T}$ has $p'(\hat{\boldsymbol{z}}_{1:t}) > \frac{1}{B+1}$ for every $t = 1, \ldots, T$, then it survives all across-beam prunes and is returned by k-CBS (or a pruned variant of k-CBS).

Proof. Write $\boldsymbol{x} := \hat{\boldsymbol{z}}_{1:t}$ for the partial continuation in question, and suppose for contradiction that $p'(\boldsymbol{x}) > \frac{1}{B+1}$ but $\boldsymbol{x}$ is not kept by the top-B selection. Then there exist B distinct candidates $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_B \in \mathbb{C}_t$ that each outrank $\boldsymbol{x}$, meaning

$$p'(\boldsymbol{w}_i) \;\geq\; p'(\boldsymbol{x}) \quad \text{for each } i = 1, \ldots, B.$$

Summing over these B candidates and adding $p'(\boldsymbol{x})$:

$$\sum_{i=1}^{B} p'(\boldsymbol{w}_i) \;+\; p'(\boldsymbol{x}) \;\geq\; B \cdot p'(\boldsymbol{x}) \;+\; p'(\boldsymbol{x}) \;=\; (B+1) \cdot p'(\boldsymbol{x}) \;>\; (B+1) \cdot \frac{1}{B+1} \;=\; 1.$$

But $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_B$ and $\boldsymbol{x}$ are $B+1$ distinct elements of $\mathbb{C}_t$, so their probabilities are a subset of the total mass at depth t. Under top-k renormalization, all depth-t candidates sum to exactly 1 (at most 1 with viability filtering; see Appendix D), so their total probability cannot exceed 1—a contradiction.

Applying this argument at each depth $t = 1, \ldots, T$ establishes the second claim by induction. □

This lemma applies to both the baseline and ε-pruned k-CBS variants, provided the path passes the viability filter at every depth. For our setting ($B = 20$), the threshold is $1/(B+1) \approx 0.048$—any continuation maintaining cumulative probability above $\approx 5\%$ at every depth is guaranteed to appear in the final beam. There can be at most B such paths (since $B+1$ of them would violate mass conservation), so the beam has capacity for all of them.

Comparison with MC at the heavy-mass floor. Even this high-probability floor dominates MC sampling on both detection and reliability. Consider a continuation with probability $p > 1/(B+1)$. From Table 1, MC requires $m \geq \ln(1/\delta)/(-\ln(1-p))$ samples merely to detect this continuation with confidence $1-\delta$. Using $-\ln(1-p) \approx p$ for moderate p, this is approximately $m \geq \ln(1/\delta)/p$. For $B = 20$, $p = 1/21$, and $\delta = 0.05$, this gives $m \gtrsim 21 \cdot \ln(20) \approx 63$ samples just for detection. And reliable estimation of its mass is even more expensive (Appendix C.1). By contrast, k-CBS recovers this continuation *deterministically*—with certainty and with its exact probability under the scoring distribution.

Beyond the floor. The $1/(B+1)$ threshold is a sufficient condition for survival of the search, not a necessary one. In practice, k-CBS (and the pruned variants) captures continuations with probability far below this floor. For most extractable sequences ($p^{\mathsf{dist}}_{z,\varepsilon} \geq \tau_{\min}$), no individual continuation exceeds $1/(B+1)$; those that survive the beam have individual probability closer to $\tau_{\min}$. We cannot provide a comparable guarantee for these lower-probability continuations without distributional assumptions.

Nevertheless, the results in this appendix show why it is likely that such paths survive the search. As the geometric-mean argument above shows, even at $\tau = \tau_{\min} = 10^{-3}$, it is intuitively the case that high-mass paths maintain very high per-token probabilities and stay competitive in the beam.

We deliberately avoid imposing distributional assumptions that would let us formalize this observation further. The rank-budget and heavy-survival results hold for any model and any scoring rule that induces a valid probability distribution, and the practical tightness of the lower bound is an empirical finding that we document in our experiments (Section 5 and Appendix F).

C.3 Comparing method costs with token evaluations

We compare the cost of different extraction methods in terms of **token evaluations**—the number of tokens processed by the model in forward passes. These costs show why our beam-search-based approach is far more efficient than Monte Carlo sampling for estimating $p^{\mathsf{dist}}_{z,\varepsilon}$. We also include greedy decoding and teacher-forced likelihood as reference points, since we report these metrics in our experiments. Throughout, we consider a single prefix–suffix pair with prefix length a and target suffix length T.

Greedy decoding (verbatim and near-verbatim discoverable extraction). Given a prefix of length a, greedy decoding produces exactly one (deterministic) continuation of length T. The prefill processes the a prefix tokens, producing logits that predict the first suffix token. Then $T-1$ autoregressive decode steps each process one token (the last decode step predicts the Tth suffix token), for a total of

$$\underbrace{a}_{\text{prefill}} + \underbrace{(T-1)}_{\text{decode steps}} \quad \text{token evaluations.}$$

The downstream check—whether the generated continuation matches the target suffix verbatim or near-verbatim (i.e., within distance ε for a chosen metric dist)—is post-processing that does not involve the model, so the cost is the same for both verbatim and near-verbatim discoverable extraction. For $a = 50$ and $T = 50$, this is 99 token evaluations. (With EOS tokens allowed, decoding may terminate early, producing fewer than T tokens.) This is the standard way to evaluate extraction, originating from Lee et al. (2022) and Carlini et al. (2023).

Teacher-forced likelihood (verbatim probabilistic extraction). To compute the verbatim probability of a fixed suffix of length T under a given prefix, we evaluate the next-token distributions along the full prefix + suffix via teacher forcing—a single forward pass over $a + T$ positions:

$$\underbrace{a+T}_{\text{one pass over prefix + suffix}} \quad \text{token evaluations.}$$

From this single pass, we obtain logits at every position, which can then be post-processed with respect to any decoding scheme ϕ to get the suffix probability under that scheme, at no additional token-evaluation cost (Appendix A.2.2). Verbatim probabilistic (discoverable) extraction originates from Hayes et al. (2025b), and was made more efficient in practice in Cooper et al. (2025). Greedy decoding and teacher forcing thus require essentially the same number of token evaluations ($a + T - 1$ vs. $a + T$).

Monte Carlo sampling (near-verbatim probabilistic extraction). Under the MC baseline (Appendix C.1), we draw M independent T-length continuations from the prefix and check each for near-verbatim similarity to the target. With KV-cache reuse of the shared prefix across all M samples,[5] the cost is

$$\underbrace{a}_{\text{one prefill}} + \underbrace{(T-1)\,M}_{M \text{ samples} \times (T-1) \text{ decode steps each}} \quad \text{token evaluations.} \tag{27}$$

For $a = 50$, $T = 50$, and $M = 3{,}000$ (the order of magnitude required merely to detect an ε-ball with mass $p^{\mathsf{dist}}_{z,\varepsilon} = 10^{-3}$ at 95% confidence; Table 1), this gives $50 + 49 \cdot 3{,}000 = 147{,}050$ token evaluations—roughly $1{,}500\times$ the cost of teacher-forced verbatim extraction for the same pair. For reliable estimation with 10% relative standard error, the required sample size grows to $M \gtrsim 10^5$ (Appendix C.1), and the cost scales proportionally. (This is the worst case among extractable sequences: M scales as $1/p^{\mathsf{dist}}_{z,\varepsilon}$, so sequences with larger $p^{\mathsf{dist}}_{z,\varepsilon}$ require fewer samples.)

[5] This is a lower bound on cost: in principle, the prefix KV cache can be computed once and cloned for each sample. In practice, standard generation APIs (e.g., HuggingFace `model.generate()`) re-process the prefix for each batch chunk, yielding a per-sample cost closer to $a + T - 1$ and a total closer to $M(a + T - 1)$. We present the shared-prefill cost as the theoretical minimum; either way, the dominant cost is the $(T-1) \cdot M$ decode term.

k-CBS (near-verbatim probabilistic extraction). Our beam-search-based approach maintains a beam of width B under top-k decoding. Write $\mathbb{L}_t$ for the beam at depth t (partial continuations of length t). The cost breaks down as follows:

1. **Prefill.** Process the a prefix tokens ($\mathbb{L}_0$, a single element) in one forward pass, producing logits for the first suffix position. Select the top-B candidates from the top-k tokens $\rightarrow \mathbb{L}_1$ (no additional forward pass needed). Cost: a token evaluations.
2. **Decode steps $t = 1, \ldots, T-1$.** Compute logits for each element of $\mathbb{L}_t$ ($|\mathbb{L}_t|$ token evaluations), expand each by the top-k tokens ($B \cdot k$ candidates), and prune to the top-$B \rightarrow \mathbb{L}_{t+1}$. At the final step ($t = T-1$), return all $B \cdot k$ candidates without pruning (Appendix D). Cost: $\sum_{t=1}^{T-1} |\mathbb{L}_t|$ token evaluations; without viability pruning, $|\mathbb{L}_t| = B$ at every depth, giving $(T-1) \cdot B$.

The total is therefore

$$\underbrace{a}_{\text{prefill}} + \underbrace{(T-1)\,B}_{(T-1)\text{ steps} \times B\text{ beams}} \quad \text{token evaluations,} \tag{28}$$

returning up to $B \cdot k$ unique continuations.

For $a = 50$, $T = 50$, $B = 20$, and $k = 40$, this is $50 + 49 \cdot 20 = 1{,}030$ token evaluations, returning up to $B \cdot k = 800$ candidates. This is roughly $10\times$ the cost of greedy extraction. By comparison, MC requires $\approx$147,000 token evaluations merely for detection ($m = 3{,}000$; Table 1) and $\approx$4,900,000 for reliable estimation ($m = 10^5$)—roughly $140\times$ and $4{,}800\times$ more than k-CBS, respectively.

Equal-budget comparison. Setting $M = B$ in Equation 27 gives the same token-evaluation cost as Equation 28: both equal $a + (T-1)\,B$. For $B = 20$: MC returns 20 random samples, while k-CBS returns up to 800 deterministic candidates to evaluate for near-verbatim extraction. At this budget ($M = B = 20$), MC is overwhelmingly unlikely to produce even a single hit for an ε-ball with mass $p^{\text{dist}}_{z,\varepsilon} = 10^{-3}$ (Table 1; see also Figure 8). Even for $p^{\text{dist}}_{z,\varepsilon} = 10^{-2}$, MC at $M = 20$ has only an $\approx$18% chance of producing a single hit ($0.99^{20} \approx 0.82$ miss probability). Moreover, k-CBS concentrates its budget on high-mass continuations by greedily retaining the highest-scoring partial sequences at each depth (Appendix C.2.3), rather than sampling uniformly from the full distribution as MC does.

ε-pruned variants and early stopping. The ε-viability-pruned variants of k-CBS (Appendix E) can be cheaper still. At each step, any partial continuation that can no longer produce a final suffix within distance ε of the target is pruned, potentially reducing the beam size below B. Writing $|\mathbb{L}_t|$ for the number of surviving beam elements at depth t, the cost becomes

$$\underbrace{a}_{\text{one prefill}} + \underbrace{\sum_{t=1}^{T-1} |\mathbb{L}_t|}_{\text{one eval per beam element per depth}} \leq a + (T-1)\,B \quad \text{token evaluations.}$$

If the viable beam empties entirely at some step $t^\star < T$, decoding halts after

$$a + \sum_{t=1}^{t^\star} |\mathbb{L}_t| \leq a + t^\star \cdot B \quad \text{token evaluations.}$$

For example, with $a = 50$, $B = 20$, and $t^\star = 3$, a rough upper bound is $50 + 20 \cdot 3 = 110$ token evaluations—comparable to the cost of greedy extraction. By contrast, MC has no analogous early stopping and costs $a + (T-1) \cdot m$ regardless of whether the sequence is extractable. The same cost reduction applies when using the minimum-extraction-probability early termination criterion (Appendix E), which halts a sequence once the best beam's cumulative probability falls below $\tau_{\min}/(B \cdot k)$. In practice, with batched processing, the wall-clock time for a batch is determined by the slowest-to-terminate sequence; early-terminated sequences are padded until the batch completes, so the wall-clock savings depend on the fraction of sequences that terminate early.

Token evaluations as a cost metric. Each token evaluation involves passing a token through the model's feed-forward and projection layers (whose per-token cost is fixed, since these layers apply the same weight matrices to each token's representation without attending to other positions). In contrast, the attention component costs scale with the KV-cache length (i.e., the number of preceding tokens that each new token must attend to). At the sequence lengths in our setting ($a + T = 100$), the per-token feed-forward and projection cost overwhelmingly dominates: the crossover where per-step attention FLOPs match per-token feed-forward FLOPs occurs at KV-cache lengths that are orders of magnitude beyond our setting.[6] Token-evaluation ratios are therefore a reasonable way to approximate FLOP ratios for our experimental conditions.

However, token evaluations do not capture all factors that affect wall-clock runtime. Teacher forcing processes all $a + T$ tokens in a single parallel forward pass—effectively one large matrix multiplication with one kernel launch and excellent GPU utilization (using causal masking rather than a KV cache, which also frees memory for larger batch sizes). Autoregressive methods (greedy, MC, and k-CBS) instead require $T - 1$ sequential decode steps, each incurring a separate kernel launch plus per-step KV-cache read/write overhead. As a result, teacher forcing is substantially faster in wall-clock time than greedy decoding despite requiring essentially the same number of token evaluations. For one prefix–suffix pair, greedy processes 1 token per decode step, MC processes M, and k-CBS processes B. Batching across multiple sequences improves GPU utilization for all autoregressive methods, though k-CBS requires $\approx B\times$ more KV-cache memory per sequence than greedy, limiting batch sizes accordingly. k-CBS also incurs per-step KV-cache gather operations (reordering cache rows to match the pruned beam) that are not reflected in the token-evaluation count.

Overall, token evaluations provide a fair common unit for comparing the computational work performed by each method, but the wall-clock advantage of teacher forcing over autoregressive methods is larger than the token-evaluation ratio alone would suggest. In practice, for larger models (e.g., 70B), the per-token compute cost dominates fixed per-step overhead, and observed wall-clock ratios approach the token-evaluation ratios. For smaller models (e.g., 7B), kernel-launch and cache-management overhead represent a larger fraction of total time, widening the gap. Viability pruning and early termination help offset this overhead by reducing the number of decode steps.

Summary for our experimental conditions. Table 2 summarizes the token-evaluation costs at our experimental settings.

Table 2: **Token-evaluation cost comparison** for a single prefix–suffix pair ($a = 50$, $T = 50$, $B = 20$, $k = 40$).

Method	Token evals	vs. greedy	Candidates returned
Greedy / Teacher forcing	99 / 100	$1\times$	1
k-CBS ($B = 20$)	1,030	$\approx 10\times$	up to 800 (deterministic)
MC ($M = 20$, same budget)	1,030	$\approx 10\times$	20 (random)
MC ($M = 3{,}000$, detection)	147,050	$\approx 1{,}500\times$	3,000 (random)
MC ($M = 10^5$, estimation)	4,900,050	$\approx 49{,}500\times$	10^5 (random)

D Decoding-constrained beam search

The prior appendix gives a mathematical intuition and cost analysis for our approach: **decoding-constrained beam search**. Here, we provide more details on our baseline algorithmic approach and invariants concerning the lower bound (and trivial upper bound) on

[6] For LLAMA 2 7B ($d = 4096$, $d_{\text{ff}} = 11008$), one token evaluation costs $\approx 25{,}000\times$ more FLOPs than one attention position lookup; for LLAMA 2 70B ($d = 8192$, $d_{\text{ff}} = 28672$), the ratio is $\approx 59{,}000\times$. At a KV-cache length of 100, the total attention cost per token is therefore $100/25{,}000 \approx 0.4\%$ of the feed-forward cost at 7B scale, and even less at 70B.

$p^{\mathsf{dist}}_{z,\varepsilon}$ that this approach returns. Our main focus in this paper is to implement decoding-constrained beam search for top-k, but this is not a strict requirement; any decoding policy that produces a valid probability distribution could be adapted accordingly. Therefore, we focus here on presenting details on the top-k version of our algorithm, starting with notation (Appendix D.1) before describing the concrete algorithm (**baseline top-k constrained beam search (CBS)**) in detail (Appendix D.2.1) and its invariants (Appendix D.2.2).

In brief, this algorithm is beam search, replacing the full softmax distribution with the renormalized top-k distribution and disabling the last across-beam prune. For beam width B and T iterations, this algorithm returns up to $B \cdot k$ T-length continuations of the input prefix, each with the associated conditional probability computed with respect to top-k decoding. These continuations can then be post-processed to check for set membership in the near-verbatim ε-ball $\mathbb{B}^{\mathsf{dist}}_{\varepsilon}$, for a chosen distance dist and tolerance ε; we sum over the probabilities of those continuations to produce a deterministic lower bound on $p^{\mathsf{dist}}_{z,\varepsilon}$ and (often very loose) upper bound on $p^{\mathsf{dist}}_{z,\varepsilon}$.

We close this appendix with a discussion of how one might also apply decoding-constrained beam search for nucleus sampling as the decoding policy (Appendix D.3).

As we show in Appendix E, we can generally improve on this approach with the addition of ε-viability pruning rule ($\mathsf{Hamming}$ or $\mathsf{Levenshtein}$) to the search procedure, which prunes non-ε-viable continuations from the beam rather than retaining and returning them at step T. This approach, of course, trades off the generality that comes from post-processing in the baseline approach we discuss in this appendix, as it bakes in a distance metric to the search process. However, this comes with benefits: without any additional token evaluations—just some additional bookkeeping—the pruned variants often return tighter lower and upper bounds for near-verbatim extractable sequences, also often at lower cost. The cost of all of these methods is discussed in Appendix C.3. Decoding-constrained beam search is significantly cheaper than MC sampling, and (under a matched compute budget) returns a significantly more useful (and deterministic) estimate of $p^{\mathsf{dist}}_{z,\varepsilon}$ (Appendix C.1).

D.1 Notation

We describe some common notation for our algorithms and proofs.

Token sequences.

- $z_{(\text{pre})}$: the a-token prefix from the training data. $z_{(\text{suf})}$: the corresponding T-token ground-truth target suffix. The training-data sequence is $z = z_{(\text{pre})} \| z_{(\text{suf})}$.
- $z_{(\text{cont})}$: an arbitrary T-token continuation of $z_{(\text{pre})}$.
- $\hat{z}_{(\text{cont})}$: an actual, generated T-token continuation of $z_{(\text{pre})}$. A history is the concatenation of a prefix and a generated continuation: $\hat{z} = z_{(\text{pre})} \| \hat{z}_{(\text{cont})}$.
- $\hat{z}^{(\text{cont})}_{1:t}$: the first t tokens of a generated continuation ($\hat{z}^{(\text{cont})}_{1:0} = \emptyset$, the empty token sequence). At step t of Algorithm 2, each input beam element holds a history $\hat{z} = z_{(\text{pre})} \| \hat{z}^{(\text{cont})}_{1:t-1}$; after expansion, the updated history is $\hat{z}' = z_{(\text{pre})} \| \hat{z}^{(\text{cont})}_{1:t}$.

This notation lets us refer to prefixes, suffixes, and partial continuations directly, avoiding index arithmetic on the full history $\hat{z}$.

Candidate sets and beam operations. At depth $t \in \{0, \ldots, T\}$, denote

- $\mathbb{L}_t$: the beam at depth t (at most B elements), obtained by expanding $\mathbb{L}_{t-1}$, applying across-beam pruning.
- $\mathbb{C}_t$: all children formed from $\mathbb{L}_{t-1}$ by one-step top-k expansion (exactly $|\mathbb{L}_{t-1}| \cdot k$ before EOS removal; EOS candidates are removed and recorded but do not enter the beam).
- $\mathbb{U}_t$: the (at most) top-B of $\mathbb{C}_t$ by cumulative probability (across-beam prune); $\mathbb{L}_t \leftarrow \mathbb{U}_t$.
- $\mathbb{F}$, the set of returned finals.

Per-step t notation. Given a beam element with current history $\hat{z}$ (the prefix $z_{(\text{pre})}$ concatenated with a continuation of length $t-1$),

$$\begin{aligned}
\boldsymbol{y}_t(\hat{\boldsymbol{z}}) \in \mathbb{R}^{|\mathbb{V}|} &\triangleq \text{logit vector from } \theta \text{ for the next token given the history } \hat{\boldsymbol{z}};\\
\mathbb{S}_t(\hat{\boldsymbol{z}}) \leftarrow \mathtt{TopK}_k\big(\boldsymbol{y}_t(\hat{\boldsymbol{z}})\big) &\triangleq \text{the set of } k \text{ tokens in } \mathbb{V} \text{ with the largest logits in } \boldsymbol{y}_t(\hat{\boldsymbol{z}});\\
\boldsymbol{r}_t(\hat{\boldsymbol{z}}) \in \mathbb{R}^{|\mathbb{V}|} \leftarrow \mathtt{LogSoftmax}\big(\boldsymbol{y}_t(\hat{\boldsymbol{z}})\big) &\triangleq \text{log probs vector over } \mathbb{V}, \text{ obtained from log softmax on } \boldsymbol{y}_t(\hat{\boldsymbol{z}}),\\
&\quad \boldsymbol{r}_t(\hat{\boldsymbol{z}})[v] = \log \Pr_{\theta}(z = v \mid \hat{\boldsymbol{z}}) \quad \forall v \in \mathbb{V} \text{ (full } \theta \text{ distribution, before top-}k);\\
Z_t(\hat{\boldsymbol{z}}) \leftarrow \mathtt{LogSumExp}\big(\boldsymbol{r}_t(\hat{\boldsymbol{z}})[\mathbb{S}_t(\hat{\boldsymbol{z}})]\big) &\triangleq \text{log of the total probability mass on } \mathbb{S}_t(\hat{\boldsymbol{z}}) \text{ (normalizing constant for top-}k),\\
&\quad Z_t(\hat{\boldsymbol{z}}) \;=\; \log \sum_{u \in \mathbb{S}_t(\hat{\boldsymbol{z}})} \exp\big(\boldsymbol{r}_t(\hat{\boldsymbol{z}})[u]\big) \;=\; \log \sum_{u \in \mathbb{S}_t(\hat{\boldsymbol{z}})} \Pr_{\theta}(u \mid \hat{\boldsymbol{z}}).
\end{aligned}$$

Then, for the update rule, selecting any $\hat{z} \in \mathbb{S}_t(\hat{\boldsymbol{z}})$ under ϕ (top-k decoding) has probability

$$\Pr_{\theta,\phi}(\hat{z} \mid \hat{\boldsymbol{z}}) = \exp(\boldsymbol{r}_t(\hat{\boldsymbol{z}})[\hat{z}] - Z_t(\hat{\boldsymbol{z}})),$$

and the log-prob update is

$$\boldsymbol{r}_t(\hat{\boldsymbol{z}})[\hat{z}] - Z_t(\hat{\boldsymbol{z}}) = \log \Pr_{\theta,\phi}(\hat{z} \mid \hat{\boldsymbol{z}}).$$

In all of our algorithms, we focus on top-k decoding and set temperature $\beta = 1$. However, we could also use temperature scaling with different settings in the decoding policy ϕ. If temperature $\beta > 0$ is applied, logits are rescaled by $1/\beta$ before the softmax:

$$\boldsymbol{y}_t^{(\beta)}(\hat{\boldsymbol{z}}) \;=\; \tfrac{1}{\beta}\,\boldsymbol{y}_t(\hat{\boldsymbol{z}}), \qquad \boldsymbol{r}_t^{(\beta)}(\hat{\boldsymbol{z}}) \;=\; \log \mathsf{softmax}\big(\boldsymbol{y}_t^{(\beta)}(\hat{\boldsymbol{z}})\big) \;=\; \log \mathsf{softmax}\big(\tfrac{1}{\beta}\,\boldsymbol{y}_t(\hat{\boldsymbol{z}})\big). \tag{29}$$

Because scaling by a positive β preserves order, the top-k set is unchanged:

$$\mathbb{S}_t(\hat{\boldsymbol{z}}) \;=\; \mathtt{TopK}_k\big(\boldsymbol{y}_t^{(\beta)}(\hat{\boldsymbol{z}})\big) \;=\; \mathtt{TopK}_k\big(\boldsymbol{y}_t(\hat{\boldsymbol{z}})\big).$$

For $\phi = \beta$, we define the temperature-aware log-normalizer over this constrained set from the log-probs:

$$Z_t^{(\beta)}(\hat{\boldsymbol{z}}) \;\triangleq\; \mathtt{LogSumExp}\big(\boldsymbol{r}_t^{(\beta)}(\hat{\boldsymbol{z}})[\mathbb{S}_t(\hat{\boldsymbol{z}})]\big) \;=\; \log \sum_{u \in \mathbb{S}_t(\hat{\boldsymbol{z}})} \Pr_{\theta,\beta}(u \mid \hat{\boldsymbol{z}}).$$

The per-step t, per-next-token $\hat{z}$ top-k log-probability update is then

$$\boldsymbol{r}_t^{(\beta)}(\hat{\boldsymbol{z}})[\hat{z}] - Z_t^{(\beta)}(\hat{\boldsymbol{z}}) \;=\; \log \Pr_{\theta,(\beta,k)}(\hat{z} \mid \hat{\boldsymbol{z}}), \qquad \hat{z} \in \mathbb{S}_t(\hat{\boldsymbol{z}}). \tag{30}$$

To see that this simplifies to a log softmax (Equation 8) over the top-k scaled logits, let

$$C_t \triangleq \log \sum_{w \in \mathbb{V}} \exp(\tfrac{1}{\beta}\,\boldsymbol{y}_t(\hat{\boldsymbol{z}})[w])$$

denote the full-vocabulary log-partition function. For token $\hat{z}$, we can expand $\boldsymbol{r}_t^{(\beta)}$ from its definition (Equation 29):

$$\begin{aligned}
\boldsymbol{r}_t^{(\beta)}(\hat{\boldsymbol{z}})[\hat{z}] \;&=\; \log \mathsf{softmax}\big(\tfrac{1}{\beta}\,\boldsymbol{y}_t(\hat{\boldsymbol{z}})\big)[\hat{z}]\\
&=\; \log \frac{\exp\big(\tfrac{1}{\beta}\,\boldsymbol{y}_t(\hat{\boldsymbol{z}})[\hat{z}]\big)}{\sum_{w \in \mathbb{V}} \exp\big(\tfrac{1}{\beta}\,\boldsymbol{y}_t(\hat{\boldsymbol{z}})[w]\big)}\\
&=\; \log \exp\big(\tfrac{1}{\beta}\,\boldsymbol{y}_t(\hat{\boldsymbol{z}})[\hat{z}]\big) \;-\; \log \sum_{w \in \mathbb{V}} \exp\big(\tfrac{1}{\beta}\,\boldsymbol{y}_t(\hat{\boldsymbol{z}})[w]\big)\\
&=\; \frac{1}{\beta}\,\boldsymbol{y}_t(\hat{\boldsymbol{z}})[\hat{z}] \;-\; C_t.
\end{aligned}$$

Now, expanding $Z_t^{(\beta)}$:

$$\begin{aligned} Z_t^{(\beta)}(\hat{z}) \;&=\; \log \sum_{u \in \mathsf{S}_t(\hat{z})} \exp\left(\boldsymbol{r}_t^{(\beta)}(\hat{z})[u]\right) \;=\; \log \sum_{u \in \mathsf{S}_t(\hat{z})} \exp\left(\frac{1}{\beta}\,\boldsymbol{y}_t(\hat{z})[u] - C_t\right) \\ &=\; \log \sum_{u \in \mathsf{S}_t(\hat{z})} \left[\exp\left(\frac{1}{\beta}\,\boldsymbol{y}_t(\hat{z})[u]\right) \cdot \exp(-C_t)\right] \\ &=\; \log \sum_{u \in \mathsf{S}_t(\hat{z})} \exp\left(\frac{1}{\beta}\,\boldsymbol{y}_t(\hat{z})[u]\right) \;-\; C_t. \end{aligned}$$

Subtracting:

$$\begin{aligned} \boldsymbol{r}_t^{(\beta)}(\hat{z})[\hat{z}] - Z_t^{(\beta)}(\hat{z}) \;&=\; \left(\frac{1}{\beta}\,\boldsymbol{y}_t(\hat{z})[\hat{z}] - C_t\right) \;-\; \left(\log \sum_{u \in \mathsf{S}_t(\hat{z})} \exp\left(\frac{1}{\beta}\,\boldsymbol{y}_t(\hat{z})[u]\right) - C_t\right) \\ &=\; \frac{1}{\beta}\,\boldsymbol{y}_t(\hat{z})[\hat{z}] \;-\; \log \sum_{u \in \mathsf{S}_t(\hat{z})} \exp\left(\tfrac{1}{\beta}\,\boldsymbol{y}_t(\hat{z})[u]\right), \end{aligned}$$

That is, the update reduces to a log softmax over the top-k scaled logits. Setting $\beta = 1$ recovers the update that we use throughout: $\boldsymbol{r}_t(\hat{z})[\hat{z}] - Z_t(\hat{z}) = \log \Pr_{\theta,\phi}(\hat{z} \mid \hat{z})$ (and for $\beta = 1$, ϕ is just top-k without temperature). (We will generally refer to the decoding strategy as ϕ, but this means in practice, with respect to the notation introduced here, we set $\beta = 1$ with top-k, i.e., $(\beta, k) = (1, k)$.)

D.2 Top-k constrained beam search

In Algorithm 2, we describe the baseline approach for the top-k constrained beam search algorithm: a slight variation on beam search that replaces the full softmax distribution with the renormalized top-k distribution and omits the last step's across-beam prune to width B. This approach returns a deterministic lower bound on $p^{\mathsf{dist}}_{z,\varepsilon}$ and (almost always very loose) upper bound on $p^{\mathsf{dist}}_{z,\varepsilon}$, computed over an ε-viable filtered subset of the returned sequences (for a chosen distance metric and ε).

We describe the algorithm in detail (Appendix D.2.1) and then prove several invariants (Appendix D.2.2). These invariants are straightforward, but are nevertheless important because they justify the soundness of our approach more formally than the intuition provided in Section 4 and Appendix C. The novelty of our work comes from connecting these previously disconnected observations in the service of making near-verbatim probabilistic extraction computationally feasible.

D.2.1 Detailed description of baseline k-CBS

Algorithm 2 implements our **baseline top-k constrained beam search (k-CBS)** algorithm for identifying candidates for high-probability continuations of the prefix under top-k decoding. Because memorized sequences from the training data are by definition very high probability under θ and ϕ, if a sequence $z_{(\text{suf})}$ is memorized, we expect this procedure to return a set of candidates $\hat{z}_{(\text{cont})}$ that contains near-verbatim matches (and possibly the verbatim match) to the target suffix $z_{(\text{suf})}$ (Appendix C.2). Per sequence tested, this type of search procedure provides a deterministic, correct lower bound on $p^{\mathsf{dist}}_{z,\varepsilon}$; for beam width B, it is approximately $\frac{B}{2}\times$ more expensive than greedy-decoded discoverable extraction, but orders of magnitude cheaper than an unbiased Monte Carlo estimate of $p^{\mathsf{dist}}_{z,\varepsilon}$ (Appendix C.3). A more condensed version of Algorithm 2 is in the main paper in Algorithm 1.

To summarize, the algorithm takes a prefix $z_{(\text{pre})}$ and greedily (i.e., not optimally) searches for the highest-probability length-T continuations under top-k constrained decoding. It begins with a single prefill forward pass that processes $z_{(\text{pre})}$ and produces logits $\boldsymbol{y}_1(z_{(\text{pre})})$ for the first suffix position. The beam $\mathbb{L}_0$ is initialized with the prefix and a cumulative

Algorithm 2: Top-k Constrained Beam Search (k-CBS)

Input: LLM θ; prefix $\boldsymbol{z}_{(\text{pre})}$ of length a; suffix length T; beam width B; top-k parameter k for decoding policy ϕ ($B \leq k^2$, since each of at most k beams expands to k candidates; $k \ll |\mathbb{V}|$); EOS token id; optional extraction threshold $\tau_{\min} > 0$

Output: Set $\mathbb{F}$ of up to $B \cdot k$ pairs $(\hat{\boldsymbol{z}},\ \log p)$, where $\hat{\boldsymbol{z}} = \boldsymbol{z}_{(\text{pre})} \,\|\, \hat{\boldsymbol{z}}_{(\text{cont})}$ is the full history and $\log p = \log \Pr_{\theta,\phi}(\hat{\boldsymbol{z}}_{(\text{cont})} \mid \boldsymbol{z}_{(\text{pre})})$; or $\mathbb{F} = \emptyset$ if early termination is triggered

Notation (per step t for partial sequence $\hat{\boldsymbol{z}}$): Given a beam element with current history $\hat{\boldsymbol{z}}$ (the prefix $\boldsymbol{z}_{(\text{pre})}$ concatenated with a continuation of length $t-1$), let:

- $\boldsymbol{y}_t(\hat{\boldsymbol{z}}) \in \mathbb{R}^{|\mathbb{V}|}$: logits from θ for the next token given $\hat{\boldsymbol{z}}$;
- $\mathrm{S}_t(\hat{\boldsymbol{z}}) \subset \mathbb{V}$, $|\mathrm{S}_t(\hat{\boldsymbol{z}})| = k$: k tokens in $\mathbb{V}$ with the largest logits in $\boldsymbol{y}_t(\hat{\boldsymbol{z}})$;
- $\boldsymbol{r}_t(\hat{\boldsymbol{z}}) \in \mathbb{R}^{|\mathbb{V}|}$: log probs over $\mathbb{V}$, $\boldsymbol{r}_t(\hat{\boldsymbol{z}})[v] = \log \Pr_\theta(v \mid \hat{\boldsymbol{z}})\ \forall v \in \mathbb{V}$;
- $Z_t(\hat{\boldsymbol{z}}) \in \mathbb{R}$: normalizing constant for top-k, $Z_t(\hat{\boldsymbol{z}}) = \log \sum_{u \in \mathrm{S}_t(\hat{\boldsymbol{z}})} \exp\big(\boldsymbol{r}_t(\hat{\boldsymbol{z}})[u]\big)$.

Update rule (top-k decoding): Selecting any $\hat{z} \in \mathrm{S}_t(\hat{\boldsymbol{z}})$ under top-k has probability $\Pr_{\theta,\phi}(\hat{z} \mid \hat{\boldsymbol{z}}) = \exp(\boldsymbol{r}_t(\hat{\boldsymbol{z}})[\hat{z}] - Z_t(\hat{\boldsymbol{z}}))$, i.e., $\log \Pr_{\theta,\phi}(\hat{z} \mid \hat{\boldsymbol{z}}) = \boldsymbol{r}_t(\hat{\boldsymbol{z}})[\hat{z}] - Z_t(\hat{\boldsymbol{z}})$.

Beam state. Maintain $\mathbb{L}_t$ as pairs $(\hat{\boldsymbol{z}}, \log p)$, where $\log p$ is the accumulated top-k decoding log-probability of the partial sequence $\hat{\boldsymbol{z}}$ so far. Max. beam capacity $|\mathbb{L}_t| = B$.

Compute $\boldsymbol{y}_1(\boldsymbol{z}_{(\text{pre})})$ via forward pass on $\boldsymbol{z}_{(\text{pre})}$; `// Prefill: a token evals`

$\mathbb{L}_0 \leftarrow \{(\boldsymbol{z}_{(\text{pre})},\ 0)\}$; `// Beam: pairs (ẑ, log p)`

for $t = 1, \ldots, T$ **do**

  $\mathbb{C}_t \leftarrow \emptyset$; `// Candidate set for step t`

  foreach $(\hat{\boldsymbol{z}},\ \log p) \in \mathbb{L}_{t-1}$ **do**

    $\mathrm{S}_t(\hat{\boldsymbol{z}}) \leftarrow \mathtt{TopK}_k\big(\boldsymbol{y}_t(\hat{\boldsymbol{z}})\big)$;

    $\boldsymbol{r}_t(\hat{\boldsymbol{z}}) \leftarrow \mathtt{LogSoftmax}\big(\boldsymbol{y}_t(\hat{\boldsymbol{z}})\big)$;

    $Z_t(\hat{\boldsymbol{z}}) \leftarrow \mathtt{LogSumExp}\big(\boldsymbol{r}_t(\hat{\boldsymbol{z}})[\mathrm{S}_t(\hat{\boldsymbol{z}})]\big)$;

    foreach $\hat{z} \in \mathrm{S}_t(\hat{\boldsymbol{z}})$ **do**

      $\hat{\boldsymbol{z}}' \leftarrow \hat{\boldsymbol{z}} \,\|\, \hat{z}$; `// append token to partial history`

      $\log p' \leftarrow \log p + \boldsymbol{r}_t(\hat{\boldsymbol{z}})[\hat{z}] - Z_t(\hat{\boldsymbol{z}})$; `// update continuation log prob`

      $\mathbb{C}_t \leftarrow \mathbb{C}_t \cup \{(\hat{\boldsymbol{z}}',\ \log p')\}$;

  if $t = T$ **then** `// Final step: return all candidates (up to B · k)`

    return $\mathbb{F} \leftarrow \mathbb{C}_T$;

  `// Non-final step (t < T): EOS handling and beam pruning`

  foreach $(\hat{\boldsymbol{z}}',\ \log p') \in \mathbb{C}_t$ *with latest* $\hat{z} = \mathrm{EOS}$ **do**

    Record $(\hat{\boldsymbol{z}}',\ \log p',\ t)$ as early-termination path; remove from $\mathbb{C}_t$;

  if $\mathbb{C}_t = \emptyset$ **then return** $\emptyset$;

  $\mathbb{U}_t \leftarrow$ top-B elements of $\mathbb{C}_t$ ranked by $\log p'$;

  $\mathbb{L}_t \leftarrow \mathbb{U}_t$; `// Prune to beam width`

  if $\tau_{\min}$ *and* $\max_{(_,\log p) \in \mathbb{L}_t} \exp(\log p) < \tau_{\min}/(B \cdot k)$ **then**

    return $\mathbb{F} \leftarrow \emptyset$; `// Early termination`

  Compute $\boldsymbol{y}_{t+1}(\hat{\boldsymbol{z}})$ for each $(\hat{\boldsymbol{z}}, \cdot) \in \mathbb{L}_t$; `// B token evals`

log-probability of zero. The main loop then iterates over suffix positions $t = 1, \ldots, T$. At each step, every partial history $\hat{\boldsymbol{z}}$ in the beam is expanded: the top-k token set $\mathrm{S}_t(\hat{\boldsymbol{z}})$ is produced from the current logits, the log-probability vector $\boldsymbol{r}_t(\hat{\boldsymbol{z}})$ and normalizing constant $Z_t(\hat{\boldsymbol{z}})$ are computed, and each candidate token $\hat{z} \in \mathrm{S}_t(\hat{\boldsymbol{z}})$ is appended to form an extended history $\hat{\boldsymbol{z}}' = \hat{\boldsymbol{z}} \,\|\, \hat{z}$ with updated cumulative log-probability (with respect to top-k) $\log p' = \log p + \boldsymbol{r}_t(\hat{\boldsymbol{z}})[\hat{z}] - Z_t(\hat{\boldsymbol{z}})$. These candidates are collected into the set $\mathbb{C}_t$. (At $t = 1$, the beam $\mathbb{L}_0$ contains a single element (the prefix), so $|\mathbb{C}_1| = k$: the prefix paired with each of the k tokens in $\mathrm{S}_1(\boldsymbol{z}_{(\text{pre})})$. For all subsequent steps $t \geq 2$, $|\mathbb{L}_{t-1}| = B$, so $|\mathbb{C}_t| = B \cdot k$.) On the final

step ($t = T$), up to $B \cdot k$ candidate T-length histories are returned in the output set $\mathbb{F}$.[7] The remaining candidates are pruned to the top B by cumulative log-probability to form the new beam $\mathbb{L}_t$, and a forward pass on the B selected tokens produces logits for the next position.

Optional early termination. When the extraction threshold τ_{min} is provided, the algorithm can terminate before completing all T steps if it becomes impossible for the final output to accumulate at least τ_{min} total mass within $\mathbb{B}_{\varepsilon}^{\mathsf{dist}}(z_{(\text{suf})})$. The key observation is that cumulative log-probabilities are monotonically non-increasing along any path: at each step t, the update adds $\log \Pr_{\theta,\phi}(\hat{z} \mid \hat{\boldsymbol{z}}) \leq 0$, so no descendant of a beam element $(\hat{\boldsymbol{z}}, \log p) \in \mathbb{L}_t$ can have cumulative probability exceeding $\exp(\log p)$. Therefore, $\max_{(_, \log p) \in \mathbb{L}_t} \exp(\log p)$ upper-bounds the probability of every final candidate at depth T. Even if all $B \cdot k$ final candidates achieved this maximum and all fell within $\mathbb{B}_{\varepsilon}^{\mathsf{dist}}(z_{(\text{suf})})$, the total near-verbatim mass would be at most $B \cdot k \cdot \max_{(_, \log p) \in \mathbb{L}_t} \exp(\log p)$. When this quantity is strictly less than τ_{min}—equivalently, $\max_{(_, \log p) \in \mathbb{L}_t} \exp(\log p) < \tau_{\text{min}} / (B \cdot k)$—the lower bound from this prefix can never reach τ_{min}. If we provide a minimum (validated) extraction probability τ_{min} (Appendix F), we can then allow the algorithm to terminate early returning $\mathbb{F} = \emptyset$, as this sequence cannot be near-verbatim extractable with respect to τ_{min}. When early termination is not triggered, the algorithm behaves identically to the case without τ_{min}.

The algorithm performs 1 prefill forward pass (a token evaluations) followed by at most $T - 1$ decoding forward passes (B token evaluations each), for a total of at most $a + (T - 1) \cdot B$ token evaluations per sequence (Appendix C.3). We follow the notation in Appendix D.1.

Returned suffix candidates. There are no duplicate sequences in the output candidates. This follows directly from the semantics of Algorithm 2, but is worth making explicit for clarity: every output *token* sequence is guaranteed to be unique (Lemma 5). However, it is possible that in *character* space there are duplicates—that unique token sequences decode to the same string of characters. Because our probability mass computations are done in token space, we do not need to account for character-space duplicates in a special way. Every unique token sequence contributes to our extraction mass calculations. We return up to $B \cdot k$ such unique sequences. Unlike traditional beam search, which prunes to B at the final step, we omit this last across-beam prune: all T-length candidates already have valid probabilities with respect to top-k decoding, so retaining all of them in $\mathbb{F}$ is free and preserves mass that would otherwise be discarded (Lemma 10). Because these $B \cdot k$ sequences contain the top-B (i.e., those that would survive the last prune), the mass of these $B \cdot k$ sequences dominates (Corollary 11).

The HuggingFace API makes it very simple to implement changes to beam search to respect our top-k scoring rule using the `LogitProcessor` abstraction. (Using beam search with top-k enabled does not do this by default; it is a stochastic variant of beam-search with top-k sampling.) However, this is not the case for skipping the last iteration across-beam prune, or how we handle EOS. We need to implement our own search, batching, and KV caching to achieve this.

Computing the near-verbatim extraction bounds from suffix candidates. Finally, we can apply our chosen distance metric—in this paper, either Hamming (Equation 17) or Levenshtein (Equation 18)—with tolerance ε to the returned suffix candidates, in order to test which are near-verbatim matches to the target suffix $z_{(\text{suf})}$. (One could in principle apply any edit-distance or semantic similarity metric.) This provides a rigorous lower bound on $p_{z,\varepsilon}^{\mathsf{dist}}$ (Equation 21): instead of estimating the near-verbatim suffix probability via Monte Carlo sampling (Appendix C.1), we sum the probabilities of the suffix candidates that are in the ε-ball $\mathbb{B}_{\varepsilon}^{\mathsf{dist}}(z_{(\text{suf})})$.

[7] The reason why this is "up to $B \cdot k$" and not exactly that number is that, on non-final steps, any candidate whose last token is EOS is recorded as an early-termination path and removed from $\mathbb{C}_t$; it is possible, in degenerate cases with many EOS sequences, that there are fewer than the maximum possible, though we do not observe this in practice.

As noted above, $\mathbb{F}$ is the set of final returned tuples (up to $B \cdot k$ of them) of T-length candidates and their respective log probabilities under top-k (Algorithm 2). Because the algorithm's semantics respect top-k decoding, the search explores a subset of the full probability distribution of top-k continuations (i.e., mass summing to 1; see Lemma 9). Therefore, the returned mass of the final continuations in $\mathbb{F}$ is ≤ 1 (in practice, almost certainly < 1). We therefore define **covered mass** as the total mass of all returned continuations:

$$\text{covered_mass}(\mathbb{F}) := \sum_{(_,\log p)\in\mathbb{F}} \exp(\log p). \tag{31}$$

Of course, since the total probability is 1, the *un*covered mass—i.e., mass that is not captured by our algorithm—is equivalent to $1 - \text{covered_mass}(\mathbb{F})$.

To estimate near-verbatim extraction, we then filter the returned suffixes according to the chosen distance metric dist and tolerance ε. That is, on the outputs of Algorithm 2, for each $\hat{z} = z_{(\text{pre})} \| \hat{z}_{(\text{cont})}$, we evaluate

$$\mathbb{F}^{(\leq\varepsilon)} := \{(\hat{z}, \log p) : (\hat{z}, \log p) \in \mathbb{F} \text{ and } \hat{z}_{(\text{cont})} \in \mathbb{B}_{\varepsilon}^{\mathsf{dist}}(z_{(\text{suf})})\}. \tag{32}$$

After this filtering operation, we can compute the (deterministic) lower bound of $p_{z,\varepsilon}^{\mathsf{dist}}$ as

$$p_{z,\varepsilon}^{\mathsf{dist}} \;\geq\; \text{LB}_{\varepsilon,\mathsf{dist}} \;\triangleq\; \sum_{(_,\log p)\in\mathbb{F}^{(\leq\varepsilon)}} \exp(\log p). \tag{33}$$

From the above covered mass (Equation 31) and lower bound (Equation 33), we can also compute a (typically very) loose upper bound on $p_{z,\varepsilon}^{\mathsf{dist}}$:

$$p_{z,\varepsilon}^{\mathsf{dist}} \;\leq\; \text{UB}_{\varepsilon,\mathsf{dist}} \;\triangleq\; \text{LB}_{\varepsilon,\mathsf{dist}} \;+\; \big(1 - \text{covered_mass}(\mathbb{F})\big), \tag{34}$$

which we elaborate on in Proposition 14. Note that $\text{UB}_{\varepsilon,\mathsf{dist}} \leq 1$ always holds, just as $\text{LB}_{\varepsilon,\mathsf{dist}} \geq 0$ trivially holds; the formula captures the fact that any uncovered mass could in principle all be ε-viable. For our baseline algorithm for k-CBS, we use a subset of the covered mass to produce an ε-viable lower bound; a simple upper bound, then, is any potentially viable mass that is *not* covered by the algorithm's output continuations. Of course, this can be very loose, if this uncovered mass is quite large.

A simple illustrative example. To give a sense of what the algorithm returns, we use the same simple illustrative example from Cooper et al. (2025)— a famous quote from *The Great Gatsby*:

$z_{(\text{pre})}$: `They were careless people, Tom and Daisy – they smashed up things and creatures and then retreated`

$z_{(\text{suf})}$: `back into their money or their vast carelessness, or whatever it was that kept them together, and let other people clean up the mess they had made.`

We use $z_{(\text{pre})}$ as the prompt and will compare generated continuations $\hat{z}_{(\text{cont})}$ against the target suffix $z_{(\text{suf})}$ to determine near-verbatim extraction success. We use LLAMA 1 13B, which was trained on the `Books3` corpus (Touvron et al., 2023a; Lee et al., 2023b). *The Great Gatsby* is included in `Books3`. With the LLAMA 1 tokenizer, the prefix is 25 tokens and the suffix is 32 tokens. We run baseline k-CBS (Algorithm 2) to get a lower bound on the near-verbatim extraction probability, $p_{z,\varepsilon}^{\mathsf{dist}}$, with $\beta = 1$, $B = 20, k = 40$. Table 3 shows results for the 10 highest-probability continuations that the algorithm returns.

There are several interesting observations about these results:

- **Verbatim discoverable extraction would fail, returning** 0**.** The first row corresponds to the greedy-decoded continuation, which is *not* a verbatim match to the target suffix. If we were to perform traditional discoverable extraction with greedy decoding, we would output the first row's $\hat{z}_{(\text{cont})}$, and we would not identify this output as successful extraction, since it fails strict equality with $z_{(\text{suf})}$.

- **Near-verbatim discoverable extraction succeeds, returning 1.** Using $\varepsilon = 1$ near-verbatim discoverable extraction (either Hamming or Levenshtein), we would identify the generation in row 1 as successful near-verbatim extraction.
- **Verbatim probabilistic extraction succeeds, returning $p_z = 0.1431$.** Probabilistic extraction shows that the verbatim suffix has $p_z = 0.1431$—a very high probability that we would count as valid extraction (e.g., exceeding a reasonable τ_{min}). Note that the top-ranked continuation (row 1) is also the greedy-decoded continuation. This provides a notion of extraction risk, absent from greedy discoverable extraction.
- **Near-verbatim probabilistic extraction reveals higher extraction risk.** In this paper, we consider maximum distances of $\varepsilon = 5$ for both Levenshtein and Hamming. For this sequence, $p^{\text{Lev}}_{z,5} \geq 0.7155$ (over all $B \cdot k = 20 \cdot 40 = 800$ candidates). With even just $\varepsilon = 1$ for the Levenshtein distance, $p^{\text{Lev}}_{z,1} \geq 0.4681$, over $3\times$ the extraction risk compared to verbatim p_z. This shows that verbatim probabilistic extraction can greatly underestimate extraction risk. All of the continuations above are effectively the same text, with only slight variations in punctuation.

Table 3: **Example k-CBS output.** Top-10 generations ($|\hat{z}_{(\text{cont})}| = 32$ tokens) for LLAMA 1 13B baseline k-CBS ($B = 20$, $k = 40$) for the 25-token prefix $z_{(\text{pre})}$: `They were careless people, Tom and Daisy - they smashed up things and creatures and then retreated`. This is a quote from *The Great Gatsby* (Fitzgerald, 1925), a book contained in LLAMA 1's training data. We diff each $\hat{z}_{(\text{cont})}$ with $z_{(\text{suf})}$, showing deletions in red and crossed out, and additions in blue. All exact-matched text is in black. Row 2 (highlighted in yellow) shows the verbatim generation of the target suffix, i.e., $\hat{z}_{(\text{cont})} = z_{(\text{suf})}$; there is no diff highlighting.

	$\hat{z}_{(\text{cont})}$	$\Pr_{\theta,\phi}(\hat{z}_{(\text{cont})} \mid z_{(\text{pre})})$	Lev	Ham
1	back into their money or their vast carelessness, or whatever it was that kept them together, and let other people clean up the mess they had made.…	0.1477	1	1
2	back into their money or their vast carelessness, or whatever it was that kept them together, and let other people clean up the mess they had made.	0.1431	0	0
3	back into their money or their vast carelessness, or whatever it was that kept them together, and let other people clean up the mess they had made.	0.0671	2	22
4	back into their money or their vast carelessness, or whatever it was that kept them together, and let other people clean up the mess they had made..	0.0535	1	1
5	back into their money or their vast carelessness, or whatever it was that kept them together, and let other people clean up the mess they had made.…	0.0409	3	22
6	back into their money or their vast carelessness, or whatever it was that kept them together, and let other people clean up the mess they had made .	0.0385	1	1
7	back into their money or their vast carelessness, or whatever it was that kept them together, and let other people clean up the mess they had made. …	0.0286	1	1
8	back into their money or their vast carelessness, or whatever it was that kept them together, and let other people clean up the mess they had made..	0.0214	3	22
9	back into their money or their vast carelessness, or whatever it was that kept them together, and let other people clean up the mess they had made . .	0.0186	3	22
10	back into their money or their vast carelessness, or whatever it was that kept them together, and let other people clean up the mess they had made.	0.0119	1	1

D.2.2 *Invariants for baseline k-CBS*

Using the notation in Appendix D.1, we first show that the semantics of Algorithm 2 ensure that every sequence $\hat{\boldsymbol{z}}$ returned in $\mathbb{F}$—the collection of pairs $(\hat{\boldsymbol{z}}, \log p)$—is unique.

Lemma 5 (No token-level duplicates)**.** *For a set of sequence-*log *prob pairs $\mathbb{A} \subseteq \mathbb{V}^i \times \mathbb{R}$ (for a fixed length i), define the projection onto sequences by*

$$\pi_1(\mathbb{A}) \;:=\; \{\, \hat{\boldsymbol{z}} \in \mathbb{V}^i : \exists\, \ell \in \mathbb{R} \text{ with } (\hat{\boldsymbol{z}}, \ell) \in \mathbb{A} \,\}.$$

*As in Algorithm 2, assume that we do not maintain partial histories with an EOS until possibly at the final step T, so all child sequences at depth t have the same length $a+t$ (the a-length prefix and t generated tokens so far). Let $\mathbb{L}_{t-1} \subseteq \mathbb{V}^{a+t-1} \times \mathbb{R}$ be the beam at depth $t-1$, and $\mathbb{F} \subseteq \mathbb{V}^{a+T} \times \mathbb{R}$ the set of returned final sequence-*log *prob pairs. Denote $\mathbb{L}_{t-1}^{\text{seq}} := \pi_1(\mathbb{L}_{t-1})$ and $\mathbb{F}^{\text{seq}} := \pi_1(\mathbb{F})$—i.e., the set of partial histories in the beam at $t-1$ and the set of full sequences at T, respectively. Define the set of unpruned child sequences at depth t as*

$$\mathbb{C}_t^{\text{seq}} \;=\; \{\, \hat{\boldsymbol{z}} \| \hat{z} \;:\; \hat{\boldsymbol{z}} \in \mathbb{L}_{t-1}^{\text{seq}},\, \hat{z} \in \mathbb{S}_t(\hat{\boldsymbol{z}}) \,\}.$$

Then $\mathbb{C}_t^{\text{seq}}$ contains no duplicates for every t, and the returned final sequences satisfy $\mathbb{F}^{\text{seq}} = \mathbb{C}_T^{\text{seq}}$ and contain no duplicates.

Proof. Consider the map $f : (\hat{\boldsymbol{z}}, \hat{z}) \mapsto \hat{\boldsymbol{z}} \| \hat{z}$, from $\bigcup_{\hat{\boldsymbol{z}} \in \mathbb{L}_{t-1}^{\text{seq}}} \big(\{\hat{\boldsymbol{z}}\} \times \mathbb{S}_t(\hat{\boldsymbol{z}})\big)$ into $\mathbb{V}^{a+t}$. Given that we do not maintain histories with an EOS until possibly at T, all $\hat{\boldsymbol{z}} \in \mathbb{L}_{t-1}^{\text{seq}}$ have the same length $a+t-1$. If $\hat{\boldsymbol{z}}_1 \neq \hat{\boldsymbol{z}}_2$, let i be the first index where they differ; then $(\hat{\boldsymbol{z}}_1 \| \hat{z}_1)_i \neq (\hat{\boldsymbol{z}}_2 \| \hat{z}_2)_i$, so $\hat{\boldsymbol{z}}_1 \| \hat{z}_1 \neq \hat{\boldsymbol{z}}_2 \| \hat{z}_2$. If $\hat{\boldsymbol{z}}_1 = \hat{\boldsymbol{z}}_2$ but $\hat{z}_1 \neq \hat{z}_2$ (which is guaranteed since $\mathbb{S}_t(\hat{\boldsymbol{z}})$ is a set), the last token differs, so $\hat{\boldsymbol{z}}_1 \| \hat{z}_1 \neq \hat{\boldsymbol{z}}_2 \| \hat{z}_2$. Therefore, f is injective on its domain, and $\mathbb{C}_t^{\text{seq}}$ has no duplicates at all steps t. Since the final sequences at depth T are the unpruned children at T, $\mathbb{F}^{\text{seq}} = \mathbb{C}_T^{\text{seq}}$ and is also duplicate-free. □

Next, we show that Algorithm 2 has a local optimality guarantee among the unique child histories (Lemma 5) enumerated at each depth. (Finite-width beam search need not recover the globally highest-probability sequences among all possible k^T paths. Importantly, our lower bound result does not rely on global optimality.)

Proposition 6 (Local optimality of the beam prunes)**.** *Let $\mathbb{L}_{t-1}$ be the beam at depth $t-1$ in Algorithm 2. Form the set $\mathbb{C}_t$ of all child sequences $\hat{\boldsymbol{z}}'$ and their respective* log *probabilities produced at depth t prior to pruning:*

$$\mathbb{C}_t := \{\, (\hat{\boldsymbol{z}}', \log p') : (\hat{\boldsymbol{z}}, \log p) \in \mathbb{L}_{t-1},\, \hat{z} \in \mathbb{S}_t(\hat{\boldsymbol{z}}),\, \hat{\boldsymbol{z}}' = \hat{\boldsymbol{z}} \parallel \hat{z},\, \log p' = \log p + \boldsymbol{r}_t(\hat{\boldsymbol{z}})[\hat{z}] - Z_t(\hat{\boldsymbol{z}}) \,\},$$

where log p *is the running* log*-probability for the partial history $\hat{\boldsymbol{z}}$ with a continuation of length $t-1$, and* log p' *is the running* log*-probability for the new partial history $\hat{\boldsymbol{z}}'$ with a continuation of length t. For $t < T$, any candidate whose last token is EOS is removed from $\mathbb{C}_t$ before pruning. There are at most $B \cdot k$ such pairs in $\mathbb{C}_t$ (exactly k at $t = 1$ since $|\mathbb{L}_0| = 1$, and at most $B \cdot k$ for $t \geq 2$). After the intermediate prune to width B, the new beam $\mathbb{L}_t$ consists of the B pairs $(\hat{\boldsymbol{z}}', \log p')$ from $\mathbb{C}_t$ with the largest values of* log p'*. (In practice, ties are broken by a fixed, deterministic rule; such ties are extremely rare.) At $t = T$, we do not perform the final across-beam prune; the returned set is exactly the unpruned set $\mathbb{F} = \mathbb{C}_T$, which contains the top-B pairs by* log p' *as a subset.*

Proof. By construction, at each depth $t < T$ the algorithm enumerates all children $\mathbb{C}_t$ of the current beam $\mathbb{L}_{t-1}$ (removing EOS candidates), sorts $\mathbb{C}_t$ by descending log p', and keeps the top B as the set $\mathbb{U}_t$. Therefore, $\mathbb{L}_t = \mathbb{U}_t$ is exactly the set of the B pairs in $\mathbb{C}_t$ with the largest log p'. At $t = T$, no prune is applied: $\mathbb{F} = \mathbb{C}_T$ contains up to $B \cdot k$ candidates, which trivially includes whatever the top-B subset would have been. □

Next, we show that the per-step log probability that Algorithm 2 accumulates for each sequence is equivalent to that sequence's top-k decoding log probability.

Lemma 7 (Per-step log probability equals top-k decoding log probability)**.** *For any history $\hat{z}$—consisting of the prefix $z_{(\text{pre})}$ and tokens generated so far—and any token $\hat{z} \in \mathbb{S}_t(\hat{z})$,*

$$r_t(\hat{z})[\hat{z}] - Z_t(\hat{z}) = \log \Pr_{\theta,\phi}(\hat{z} \mid \hat{z}),$$

where ϕ is top-k renormalization, i.e.,

$$\Pr_{\theta,\phi}(\hat{z} \mid \hat{z}) \;=\; \frac{\Pr_\theta(\hat{z} \mid \hat{z})}{\sum_{u \in \mathbb{S}_t(\hat{z})} \Pr_\theta(u \mid \hat{z})} \quad \textit{for } \hat{z} \in \mathbb{S}_t(\hat{z}), \qquad \Pr_{\theta,\phi}(v \mid \hat{z}) = 0 \textit{ for } v \notin \mathbb{S}_t(\hat{z}).$$

Proof. By definition,

$$r_t(\hat{z}) = \log \mathsf{softmax}(y_t(\hat{z})) = \log \Pr_\theta(\cdot|\hat{z}).$$

So,

$$\exp\big(r_t(\hat{z})[v]\big) = \Pr_\theta(v \mid \hat{z}) \quad \forall v \in \mathbb{V},$$

and

$$Z_t(\hat{z}) = \log \sum_{u \in \mathbb{S}_t(\hat{z})} \exp\big(r_t(\hat{z})[u]\big) = \log \sum_{u \in \mathbb{S}_t(\hat{z})} \Pr_\theta(u \mid \hat{z}).$$

Therefore, for any $\hat{z} \in \mathbb{S}_t(\hat{z})$,

$$\begin{aligned} r_t(\hat{z})[\hat{z}] - Z_t(\hat{z}) &= \log \Pr_\theta(\hat{z} \mid \hat{z}) - \log \sum_{u \in \mathbb{S}_t(\hat{z})} \Pr_\theta(u \mid \hat{z}) \\ &= \log \frac{\Pr_\theta(\hat{z} \mid \hat{z})}{\sum_{u \in \mathbb{S}_t(\hat{z})} \Pr_\theta(u \mid \hat{z})} \\ &= \log \Pr_{\theta,\phi}(\hat{z} \mid \hat{z}). \end{aligned}$$

□

Corollary 8 (Path log-probability equals sum of top-k increments)**.** *Let $z_{(\text{pre})} \in \mathbb{V}^a$ be a prefix of length a, and let $\hat{z}_{(\text{cont})} = (\hat{z}_1, \ldots, \hat{z}_T)$ be any length-T continuation produced by Algorithm 2. For $t = 1, \ldots, T$, define the step-t history $\hat{z} := z_{(\text{pre})} \,\|\, \hat{z}_{1:t-1}^{(\text{cont})} = \hat{z}_{1:a+t-1}$ Let $r_t(\hat{z}) \in \mathbb{R}^{|\mathbb{V}|}$ denote the* log*-probability vector with entries $r_t(\hat{z})[u] := \log \Pr_\theta(u \mid \hat{z})$. Let $\mathbb{S}_t(\hat{z})$ be the step-t top-k set and*

$$Z_t(\hat{z}) \;:=\; \texttt{LogSumExp}\big(\, r_t(\hat{z})[\mathbb{S}_t(\hat{z})] \,\big) \;=\; \log \sum_{u \in \mathbb{S}_t(\hat{z})} \exp\big(r_t(\hat{z})[u]\big).$$

Then the log*-probability of $\hat{z}_{(\text{cont})}$ under the top-k policy ϕ decomposes as*

$$\log \Pr_{\theta,\phi}\big(\hat{z}_{(\text{cont})} \mid z_{(\text{pre})}\big) \;=\; \sum_{t=1}^{T} \Big(r_t(\hat{z})[\hat{z}_t] \;-\; Z_t(\hat{z})\Big).$$

Proof. By Lemma 7, at step t the log-increment from choosing $\hat{z}_t \in \mathbb{S}_t(\hat{z})$ given history $\hat{z}$ is $r_t(\hat{z})[\hat{z}_t] - Z_t(\hat{z}) = \log \Pr_{\theta,\phi}(\hat{z}_t \mid \hat{z})$. Summing over $t = 1, \ldots, T$ yields $\log \Pr_{\theta,\phi}(\hat{z}_{(\text{cont})} \mid z_{(\text{pre})}) = \sum_{t=1}^{T} \log \Pr_{\theta,\phi}(\hat{z}_t \mid \hat{z})$, the standard autoregressive factorization. □

Following from above, we can show that the entire mass is conserved under top-k normalization. We use this below to show that final-step mass is preserved when the last pruning step is disabled.

Lemma 9 (Frontier mass identity under top-k renormalization)**.** *Fix a prefix $z_{(\text{pre})}$ and decoding policy ϕ (top-k). Let u be a partial continuation of length at most T, and for brevity let*

$$\Pr_{\theta,\phi}(u) \;:=\; \Pr_{\theta,\phi}(u \mid z_{(\text{pre})})$$

denote its ϕ-computed conditional probability. At any step $t < T$, the children of a length-t continuation $\boldsymbol{u}$ are the one-token extensions

$$\{\, \boldsymbol{u} \| v : \; v \in \mathrm{S}_{t+1}(\boldsymbol{z}_{(\mathrm{pre})} \| \boldsymbol{u}) \,\}$$

where $\mathrm{S}_{t+1}(\boldsymbol{z}_{(\mathrm{pre})} \| \boldsymbol{u})$ is the top-k set at step $t{+}1$ given history $\boldsymbol{z}_{(\mathrm{pre})} \| \boldsymbol{u}$.

A set $\mathbb{A}$ of pairs $(\boldsymbol{w}, \log p)$*—$\boldsymbol{w}$ is a partial continuation and $\log p = \log \Pr_{\theta,\phi}(\boldsymbol{w} \mid \boldsymbol{z}_{(\mathrm{pre})})$—is a* **frontier** *if the sequence components form a cut: (i) no $\boldsymbol{w}$ is a prefix of another (i.e., no ancestor-descendant pairs), and (ii) every length-T continuation has exactly one $\boldsymbol{w} \in \mathbb{A}$ as a prefix (i.e., initial segment of the continuation). Then the total mass of any frontier equals* 1*:*

$$\sum_{(_, \log p) \in \mathbb{A}} \exp(\log p) \;=\; 1.$$

Returned finals and pruned nodes. *Let $\mathbb{F}$ be the set of returned finals and their* log *probabilities, pairs $(\hat{\boldsymbol{z}}_{(\mathrm{cont})}, \log p)$ at depth T. Let $\mathbb{R}_{\mathrm{prune}}$ be the set of all pruned nodes and their* log *probabilities: these are pairs $(\boldsymbol{u}, \log p)$ for any depth at which the node $\boldsymbol{u}$ (and its* log *probability) was removed by pruning. (By construction, once a node is pruned, none of its descendants are ever constructed, so they cannot be pruned later. Siblings and nodes on different branches can both be in $\mathbb{R}_{\mathrm{prune}}$, but there are no ancestor-descendant pairs inside $\mathbb{R}_{\mathrm{prune}}$.)*

The returned set $\mathbb{F}$ contains all depth-T children that survived earlier pruning, so $\mathbb{R}_{\mathrm{prune}}$ contains prunes only from depths $< T$. Then $\mathbb{F} \cup \mathbb{R}_{\mathrm{prune}}$ is a frontier and

$$\underbrace{\sum_{(_, \log p) \in \mathbb{F}} \exp(\log p)}_{\text{computed from } \hat{\boldsymbol{z}}_{(\mathrm{cont})}} \;+\; \underbrace{\sum_{(\boldsymbol{u}, \log p) \in \mathbb{R}_{\mathrm{prune}}} \exp(\log p)}_{\text{leftover computed from } \boldsymbol{u}} \;=\; 1.$$

That is, $\mathbb{F} \cup \mathbb{R}_{\mathrm{prune}}$ is a frontier/cut because each root-to-T path intersects it exactly once—either it is kept as a final in $\mathbb{F}$ or at the first node on that path that was pruned in $\mathbb{R}_{\mathrm{prune}}$.

Proof. For any length-t continuation $\boldsymbol{u}$ with $t < T$, top-k renormalization ensures that the conditional probabilities of the next token sum to 1:

$$\sum_{v \in \mathrm{S}_{t+1}(\boldsymbol{z}_{(\mathrm{pre})} \| \boldsymbol{u})} \Pr_{\theta,\phi}(v \mid \boldsymbol{z}_{(\mathrm{pre})} \| \boldsymbol{u}) = 1.$$

Combining this with the autoregressive chain rule,

$$\Pr_{\theta,\phi}(\boldsymbol{u} \| v \mid \boldsymbol{z}_{(\mathrm{pre})}) = \Pr_{\theta,\phi}(\boldsymbol{u} \mid \boldsymbol{z}_{(\mathrm{pre})}) \cdot \Pr_{\theta,\phi}(v \mid \boldsymbol{z}_{(\mathrm{pre})} \| \boldsymbol{u}),$$

we get one-step conservation:

$$\begin{aligned} \sum_{v \in \mathrm{S}_{t+1}(\boldsymbol{z}_{(\mathrm{pre})} \| \boldsymbol{u})} \Pr_{\theta,\phi}(\boldsymbol{u} \| v \mid \boldsymbol{z}_{(\mathrm{pre})}) &= \sum_{v \in \mathrm{S}_{t+1}(\boldsymbol{z}_{(\mathrm{pre})} \| \boldsymbol{u})} \Pr_{\theta,\phi}(\boldsymbol{u} \mid \boldsymbol{z}_{(\mathrm{pre})}) \, \Pr_{\theta,\phi}(v \mid \boldsymbol{z}_{(\mathrm{pre})} \| \boldsymbol{u}) \\ &= \Pr_{\theta,\phi}(\boldsymbol{u} \mid \boldsymbol{z}_{(\mathrm{pre})}) \cdot \underbrace{\sum_{v \in \mathrm{S}_{t+1}(\boldsymbol{z}_{(\mathrm{pre})} \| \boldsymbol{u})} \Pr_{\theta,\phi}(v \mid \boldsymbol{z}_{(\mathrm{pre})} \| \boldsymbol{u})}_{=1} \;=\; \Pr_{\theta,\phi}(\boldsymbol{u} \mid \boldsymbol{z}_{(\mathrm{pre})}), \end{aligned} \tag{35}$$

i.e., the mass of a parent node $\boldsymbol{u}$ equals the total mass of its k children (before pruning).

Now let $\mathrm{Desc}_T(\boldsymbol{w}) = \{\boldsymbol{x} \in \mathbb{V}^T : \boldsymbol{w} \text{ is a prefix of } \boldsymbol{x}\}$, where here "prefix of" refers to an initial segment of the continuation, not the input prefix $\boldsymbol{z}_{(\mathrm{pre})}$). This is the set of depth-T descendants of $\boldsymbol{w}$ (the full-length-T continuations beginning with $\boldsymbol{w}$) under the top-k policy. By repeatedly applying the one-step conservation along the subtree rooted at $\boldsymbol{w}$ (i.e., push mass all the way down to depth T), we get the **descendant sum identity**:

$$\sum_{\boldsymbol{x} \in \mathrm{Desc}_T(\boldsymbol{w})} \Pr_{\theta,\phi}(\boldsymbol{x} \mid \boldsymbol{z}_{(\mathrm{pre})}) = \Pr_{\theta,\phi}(\boldsymbol{w} \mid \boldsymbol{z}_{(\mathrm{pre})}). \tag{36}$$

Simply put, this is per-node conservation: the probabilities of all depth-T descendants of node $\boldsymbol{w}$ (i.e., the subtree with $\boldsymbol{w}$ as the root) sum to the probability of $\boldsymbol{w}$ under top-k. By induction on depth: the base case is Equation 35 (one-step conservation); for the inductive step, apply one-step conservation to each leaf of the current frontier of the subtree, replacing each leaf's mass with the sum of its children's masses, until all leaves are at depth T.

If $\mathbb{A}$ is a frontier, the sets of descendants $\{\mathrm{Desc}_T(\boldsymbol{w}) : (\boldsymbol{w}, \log p) \in \mathbb{A}\}$ for different $\boldsymbol{w}$ are disjoint and their union is the full set of all depth-T continuations (every path hits the frontier). From per-node conservation above (descendant sum identity, Equation 36), summing over $(\boldsymbol{w}, \log p) \in \mathbb{A}$ gives

$$\begin{aligned}
\sum_{(_, \log p) \in \mathbb{A}} \exp(\log p) &= \sum_{(\boldsymbol{w}, \log p) \in \mathbb{A}} \Pr_{\theta,\phi}(\boldsymbol{w} \mid \boldsymbol{z}_{(\mathrm{pre})}) \\
&= \sum_{(\boldsymbol{w}, \log p) \in \mathbb{A}} \; \sum_{\boldsymbol{x} \in \mathrm{Desc}_T(\boldsymbol{w})} \Pr_{\theta,\phi}(\boldsymbol{x} \mid \boldsymbol{z}_{(\mathrm{pre})}) \qquad \text{(descendant sum, Eq. 36)} \\
&= \sum_{\text{all } \boldsymbol{x} \text{ of length } T} \Pr_{\theta,\phi}(\boldsymbol{x} \mid \boldsymbol{z}_{(\mathrm{pre})}) \qquad \text{(disjoint partition)} \\
&= 1. \qquad \text{(descendant sum at the root, which has mass 1)}
\end{aligned}$$

Finally, by construction, every root-to-length-T path ends either at a returned final in $\mathbb{F}$ or at its first pruned ancestor in $\mathbb{R}_{\mathrm{prune}}$ (the step where it left the beam), and the identity follows. □

Since Lemma 9 holds for any frontier—regardless of the depths at which its nodes appear—it holds in particular when the frontier includes all depth-T nodes that survived pruning. At T, we simply keep more in $\mathbb{F}$ rather than allocating mass to $\mathbb{R}_{\mathrm{prune}}$, since we do not perform the final across-beam prune. This is also why we can track the mass of partial paths before T when we hit an EOS; we can effectively treat those paths as parents that contain the mass of children that we do not explore further.

With the conservation of mass, per-step probabilities, and overall path probabilities shown above, we quantify how probability mass behaves at the final expansion when the last pruning operation is disabled.

Lemma 10 (Final-step mass preservation without pruning)**.** *Let $\mathbb{L}_{T-1} \subseteq \mathbb{V}^{a+T-1} \times \mathbb{R}$ be the beam at depth $T-1$ in Algorithm 2, and let*

$$\mathbb{C}_T \;=\; \left\{ (\hat{\boldsymbol{z}} \,\|\, \hat{z},\ \log p + \boldsymbol{r}_T(\hat{\boldsymbol{z}})[\hat{z}] - Z_T(\hat{\boldsymbol{z}})) \;:\; (\hat{\boldsymbol{z}}, \log p) \in \mathbb{L}_{T-1},\ \hat{z} \in \mathbb{S}_T(\hat{\boldsymbol{z}}) \right\}$$

be the set of up to $B \cdot k$ unpruned children at depth T paired with their associated log *probabilities. With the final prune disabled, the total probability associated with the returned final sequences equals the beam mass at depth $T-1$:*

$$\sum_{(_, \log p') \in \mathbb{C}_T} \exp(\log p') \;=\; \sum_{(_, \log p) \in \mathbb{L}_{T-1}} \exp(\log p).$$

Proof. By Lemma 7 (and Corollary 8), each child from parent $(\hat{\boldsymbol{z}}, \log p)$ has

$$\exp(\log p') = \exp(\log p) \cdot \exp(\boldsymbol{r}_T(\hat{\boldsymbol{z}})[\hat{z}] - Z_T(\hat{\boldsymbol{z}})) = \exp(\log p) \cdot \Pr_{\theta,\phi}(\hat{z} \mid \hat{\boldsymbol{z}}).$$

Therefore,

$$\sum_{(_, \log p') \in \mathbb{C}_T} \exp(\log p') = \sum_{(\hat{\boldsymbol{z}}, \log p) \in \mathbb{L}_{T-1}} \left[\exp(\log p) \cdot \sum_{\hat{z} \in \mathbb{S}_T(\hat{\boldsymbol{z}})} \Pr_{\theta,\phi}(\hat{z} \mid \hat{\boldsymbol{z}}) \right] = \sum_{(\hat{\boldsymbol{z}}, \log p) \in \mathbb{L}_{T-1}} \exp(\log p),$$

since $\sum_{\hat{z} \in \mathbb{S}_T(\hat{\boldsymbol{z}})} \Pr_{\theta,\phi}(\hat{z} \mid \hat{\boldsymbol{z}}) = 1$ by top-k renormalization. □

As an immediate consequence, we capture more mass by not pruning at step T.

Corollary 11 (Pruned vs. unpruned final sequences)**.** *Let $\mathbb{C}_T$ be the unpruned set of up to $B \cdot k$ child sequences and their* log *probabilities at depth T. Let $\mathbb{F}_{(B)} \subseteq \mathbb{C}_T$ be the B final sequences and their* log *probabilities that would be kept if we performed the last prune. Then*

$$\sum_{(_, \log p') \in \mathbb{F}_{(B)}} \exp(\log p') \;\le\; \sum_{(_, \log p') \in \mathbb{C}_T} \exp(\log p') \;=\; \sum_{(_, \log p) \in \mathbb{L}_{T-1}} \exp(\log p).$$

Proof. Since $\mathbb{F}_{(B)} \subseteq \mathbb{C}_T$ and all terms are non-negative, the first inequality is monotonicity of finite sums. The second equality is by Lemma 10. □

From the above, we now turn from probability mass accounting to the lower bound of the near-verbatim probability: summing probabilities of *any* subset of the ε-ball yields a certified lower bound by monotonicity.

Theorem 12 (Lower bound on near-verbatim mass)**.** *We denote the ε-ball around the T-length target suffix $z_{(\text{suf})}$ for distance metric* $\mathsf{dist} \in \{\mathsf{Hamming}, \mathsf{Levenshtein}\}$ *as $\mathbb{B}_\varepsilon^{\mathsf{dist}}(z_{(\text{suf})}) = \{v \in \mathbb{V}^T : \mathsf{dist}(v, z_{(\text{suf})}) \le \varepsilon\}$ (Equation 20). Let $\mathbb{F}_{(B)}$ be the set of up to B pairs of final T-length continuations and their* log *probabilities $(\hat{z}_{(\text{cont})}, \log p)$ that would be returned by Algorithm 2 with the final prune enabled. Define the beam-based estimated mass lower bound to be*

$$\mathrm{LB}^{(B)}_{\varepsilon,\mathsf{dist}} \;:=\; \sum_{(\hat{z}_{(\text{cont})}, \log p) \in \mathbb{F}_{(B)} \,:\, \hat{z}_{(\text{cont})} \in \mathbb{B}_\varepsilon^{\mathsf{dist}}(z_{(\text{suf})})} \exp(\log p).$$

Then

$$\mathrm{LB}^{(B)}_{\varepsilon,\mathsf{dist}} \;\le\; \sum_{v \in \mathbb{B}_\varepsilon^{\mathsf{dist}}(z_{(\text{suf})})} \Pr_{\theta,\phi}(v \mid z_{(\text{pre})}) \;=\; p^{\mathsf{dist}}_{z,\varepsilon} \quad \textit{(by Equation 21).}$$

Proof. By Lemma 7 and Corollary 8, for each $(\hat{z}_{(\text{cont})}, \log p) \in \mathbb{F}_{(B)}$ we have $\exp(\log p) = \Pr_{\theta,\phi}(\hat{z}_{(\text{cont})} \mid z_{(\text{pre})})$. So

$$\mathrm{LB}^{(B)}_{\varepsilon,\mathsf{dist}} = \sum_{(\hat{z}_{(\text{cont})}, \log p) \in \mathbb{F}_{(B)} \,:\, \hat{z}_{(\text{cont})} \in \mathbb{B}_\varepsilon^{\mathsf{dist}}(z_{(\text{suf})})} \Pr_{\theta,\phi}(\hat{z}_{(\text{cont})} \mid z_{(\text{pre})}).$$

By Lemma 5, each sequence appears in $\mathbb{F}_{(B)}$ exactly once (no duplicates), so $\{\, \hat{z}_{(\text{cont})} : (\hat{z}_{(\text{cont})}, \log p) \in \mathbb{F}_{(B)}$ and $\hat{z}_{(\text{cont})} \in \mathbb{B}_\varepsilon^{\mathsf{dist}}(z_{(\text{suf})}) \,\} \subseteq \mathbb{B}_\varepsilon^{\mathsf{dist}}(z_{(\text{suf})})$. Monotonicity of finite sums then yields

$$\mathrm{LB}^{(B)}_{\varepsilon,\mathsf{dist}} \le \sum_{v \in \mathbb{B}_\varepsilon^{\mathsf{dist}}(z_{(\text{suf})})} \Pr_{\theta,\phi}(v \mid z_{(\text{pre})}) = p^{\mathsf{dist}}_{z,\varepsilon} \quad \text{(Equation 21).}$$

□

The same monotonicity argument applies if we keep up to $B \cdot k$ final pairs of continuations and their log probs at depth T (i.e., no final across-beam prune to size B).

Corollary 13 (Lower bound for the no-final-prune)**.** *Let $\mathbb{C}_T$ be the set of up to $B \cdot k$ child pairs $(\hat{z}_{(\text{cont})}, \log p)$ at depth T before the final prune to B sequences (as in Proposition 6 with $t = T$). Define*

$$\mathrm{LB}^{(B\cdot k)}_{\varepsilon,\mathsf{dist}} \;:=\; \sum_{(\hat{z}_{(\text{cont})}, \log p) \in \mathbb{C}_T \,:\, \hat{z}_{(\text{cont})} \in \mathbb{B}_\varepsilon^{\mathsf{dist}}(z_{(\text{suf})})} \exp(\log p).$$

Then

$$\mathrm{LB}^{(B\cdot k)}_{\varepsilon,\mathsf{dist}} \;\le\; \sum_{v \in \mathbb{B}_\varepsilon^{\mathsf{dist}}(z_{(\text{suf})})} \Pr_{\theta,\phi}(v \mid z_{(\text{pre})}) \;=\; p^{\mathsf{dist}}_{z,\varepsilon} \;\textit{(Equation 21), and}\; \mathrm{LB}^{(B)}_{\varepsilon,\mathsf{dist}} \;\le\; \mathrm{LB}^{(B\cdot k)}_{\varepsilon,\mathsf{dist}}.$$

Proof. By Lemma 5, $\mathbb{C}_T$ has no token-level duplicates, so Theorem 12 applies exactly with $\mathbb{F}_{(B)}$ replaced by $\mathbb{C}_T$, yielding $\mathrm{LB}^{(B\cdot k)}_{\varepsilon,\mathsf{dist}} \le p^{\mathsf{dist}}_{z,\varepsilon}$. Since $\mathbb{F}_{(B)} \subseteq \mathbb{C}_T$ (final prune keeps the top B from the $B \cdot k$ candidates in $\mathbb{C}_T$), we have $\{\hat{z}_{(\mathrm{cont})} : (\hat{z}_{(\mathrm{cont})}, \log p) \in \mathbb{F}_{(B)},\ \hat{z}_{(\mathrm{cont})} \in \mathbb{B}^{\mathsf{dist}}_{\varepsilon}(z_{(\mathrm{suf})})\} \subseteq \{\hat{z}_{(\mathrm{cont})} : (\hat{z}_{(\mathrm{cont})}, \log p) \in \mathbb{C}_T,\ \hat{z}_{(\mathrm{cont})} \in \mathbb{B}^{\mathsf{dist}}_{\varepsilon}(z_{(\mathrm{suf})})\}$. So, by monotonicity of finite sums, $\mathrm{LB}^{(B)}_{\varepsilon,\mathsf{dist}} \le \mathrm{LB}^{(B\cdot k)}_{\varepsilon,\mathsf{dist}}$. Lemma 10 and Corollary 11 additionally show that total mass (not just near-verbatim mass) is preserved when the final prune is disabled. □

We can also show that Algorithm 2 produces a loose (almost always very loose) upper bound from the covered mass it returns.

Proposition 14 (Loose upper bound from covered mass)**.** *Let* $\mathrm{LB}_{\varepsilon,\mathsf{dist}}$ *be the beam-based lower bound from Theorem 12 or Corollary 13 (the superscript* (B) *or* $(B \cdot k)$ *is suppressed since the result holds for either), and (as in Equation 31) define the (unfiltered) covered mass*

$$\mathrm{covered_mass}(\mathbb{F}) \;:=\; \sum_{(_,\log p)\in\mathbb{F}} \exp(\log p),$$

and define the upper bound

$$\mathrm{UB}_{\varepsilon,\mathsf{dist}} \;:=\; \mathrm{LB}_{\varepsilon,\mathsf{dist}} \;+\; (1 - \mathrm{covered_mass}(\mathbb{F})).$$

Then for any distance $\mathsf{dist} \in \{\mathsf{Hamming}, \mathsf{Levenshtein}\}$*, the near-verbatim mass satisfies*

$$p^{\mathsf{dist}}_{z,\varepsilon} \;\le\; \mathrm{UB}_{\varepsilon,\mathsf{dist}}.$$

Proof. By Lemma 9, for the frontier $\mathbb{F} \cup \mathbb{R}_{\mathrm{prune}}$ we have

$$1 \;=\; \underbrace{\sum_{(_,\log p)\in\mathbb{F}} \exp(\log p)}_{\mathrm{covered_mass}(\mathbb{F})} \;+\; \sum_{(_,\log p)\in\mathbb{R}_{\mathrm{prune}}} \exp(\log p). \tag{37}$$

As in Lemma 9, let $\mathrm{Desc}_T(\boldsymbol{u})$ denote the set of depth-T descendants of $\boldsymbol{u}$ (i.e., full-length continuations with prefix $\boldsymbol{u}$; $\boldsymbol{u}$ is a prefix of all these continuations). Then, we can decompose $p^{\mathsf{dist}}_{z,\varepsilon}$ into contributions from the returned finals and pruned subtrees:

$$p^{\mathsf{dist}}_{z,\varepsilon} = \underbrace{\sum_{(\hat{z}_{(\mathrm{cont})},\log p)\in\mathbb{F}} \exp(\log p)\,\mathbf{1}[\hat{z}_{(\mathrm{cont})} \in \mathbb{B}^{\mathsf{dist}}_{\varepsilon}(z_{(\mathrm{suf})})]}_{=\,\mathrm{LB}_{\varepsilon,\mathsf{dist}}} + \sum_{(\boldsymbol{u},\log p)\in\mathbb{R}_{\mathrm{prune}}} \sum_{\boldsymbol{x}\in\mathrm{Desc}_T(\boldsymbol{u})} \Pr_{\theta,\phi}(\boldsymbol{x} \mid z_{(\mathrm{pre})})\,\mathbf{1}[\boldsymbol{x} \in \mathbb{B}^{\mathsf{dist}}_{\varepsilon}(z_{(\mathrm{suf})})].$$

Dropping the indicator in the second term yields an upper bound (i.e., allows for the possibility that each $\boldsymbol{x}$ is a valid suffix in $\mathbb{B}^{\mathsf{dist}}_{\varepsilon}$):

$$\begin{aligned}\sum_{(\boldsymbol{u},\log p)\in\mathbb{R}_{\mathrm{prune}}} \sum_{\boldsymbol{x}\in\mathrm{Desc}_T(\boldsymbol{u})} \Pr_{\theta,\phi}(\boldsymbol{x} \mid z_{(\mathrm{pre})}) &= \sum_{(\boldsymbol{u},\log p)\in\mathbb{R}_{\mathrm{prune}}} \Pr_{\theta,\phi}(\boldsymbol{u} \mid z_{(\mathrm{pre})}) \\ &= \sum_{(_,\log p)\in\mathbb{R}_{\mathrm{prune}}} \exp(\log p) \\ &= 1 - \mathrm{covered_mass}(\mathbb{F}),\end{aligned}$$

where the first equality uses Equation 36 with $\boldsymbol{w} \leftarrow \boldsymbol{u}$, summed over $(\boldsymbol{u}, \log p) \in \mathbb{R}_{\mathrm{prune}}$; the second equality is by definition; and the third equality is by Equation 37. Therefore, $p^{\mathsf{dist}}_{z,\varepsilon} \le \mathrm{LB}_{\varepsilon,\mathsf{dist}} + (1 - \mathrm{covered_mass}(\mathbb{F})) = \mathrm{UB}_{\varepsilon,\mathsf{dist}}$. In Algorithm 2, $\mathbb{F} = \mathbb{C}_T$ (all $B \cdot k$ finals), but the reasoning applies also if $\mathbb{F}$ contains only the top-B finals retained by pruning. □

Validity of early termination. The optional early termination in Algorithm 2 is justified by the results above. At any intermediate step $t < T$, cumulative log-probabilities are non-increasing along paths (each step adds $\log \Pr_{\theta,\phi}(\hat{z} \mid \hat{z}) \leq 0$), so the highest-probability beam element in $\mathbb{L}_t$ upper-bounds the probability of every depth-T descendant. By Theorem 12 and Corollary 13, the lower bound $\mathrm{LB}^{(B \cdot k)}_{\varepsilon,\mathsf{dist}}$ is a sum of at most $B \cdot k$ such descendant probabilities, each bounded by $\max_{(_,\log p) \in \mathbb{L}_t} \exp(\log p)$. Therefore, if $B \cdot k \cdot \max_{(_,\log p) \in \mathbb{L}_t} \exp(\log p) < \tau_{\min}$, then $\mathrm{LB}^{(B \cdot k)}_{\varepsilon,\mathsf{dist}} < \tau_{\min}$ regardless of the distance metric dist or threshold ε. This means that k-CBS does not identify mass that indicates the sequence is extractable at level $\tau_{\min}$. (Since k-CBS is a greedy search, it may miss such mass, so this is not a guarantee, but nevertheless reflects what occurs in practice.) In this case, the algorithm returns $\mathbb{F} = \emptyset$ (yielding $\mathrm{LB} = 0$) without completing the remaining $T - t$ steps.

D.3 Nucleus sampling constrained beam search (p-CBS)

Throughout, we discuss all of our algorithms in relation to top-k decoding. We note that we could also incorporate temperature into our update rule (Appendix D.1), but choose to focus on $\beta = 1$. This reflects the base distribution of the LLM θ, which we (and prior work, namely Hayes et al. (2025b) and Cooper et al. (2025)) believe is a useful regime to study for extraction and memorization. However, any probability-distribution-preserving scoring rule could also be incorporated into our constrained beam search algorithm, such as **nucleus (top-p) sampling** (Holtzman et al., 2020). (We could also implement pruning with this change in scoring rule, though we do not discuss this in detail and instead talk about the pruned variants of k-CBS that we run in Section 4.3 and Appendix E.)

This approach is biased in the same sense as our top-k constrained algorithms (deterministic and correct, but downward biased lower bound). The scoring rule is changed to be valid under the truncated and renormalized nucleus chain. We refer to this algorithm as **nucleus-constrained beam search (p-CBS)**.

Setup and notation. Fix a prefix $z_{(\mathrm{pre})}$, a suffix length $T \in \mathbb{N}$, and an EOS policy identical to the baseline (remove from the candidate set prior to across-beam pruning at steps $1{:}T-1$, allow at $t = T$). Given a beam element with current history $\hat{z}$ (which includes the a-length prefix) at step t, let $\boldsymbol{y}_t(\hat{z}) \in \mathbb{R}^{|\mathbb{V}|}$ be the logits from θ and let

$$p_t(u \mid \hat{z}) \;:=\; \mathsf{softmax}(\boldsymbol{y})[u]$$

denote the base next-token probability over $\mathbb{V}$. Ties are broken by a fixed deterministic rule throughout. For consistency with our notation for top-k sampling (Equation 11), below we write $\Pr(u \mid \hat{z})$ for the same base conditional, i.e., $\Pr(u \mid \hat{z}) \equiv p_t(u \mid \hat{z})$; we introduce this notation only to emphasize t.

Nucleus (top-p) sampling. Given a threshold $p \in (0, 1]$, nucleus sampling retains the smallest set of next tokens whose total base probability covers at least a p fraction of the mass; p acts as a coverage knob (small p narrows support, $p = 1$ leaves the support unchanged from the base distribution).

At step t for a parent history $\hat{z}$, consider the ordering of the entire vocabulary by the base probabilities conditioned on $\hat{z}$. Let $\sigma_t(\hat{z})$ be a permutation of $\mathbb{V}$ that lists tokens in nonincreasing order of $\Pr(\cdot \mid \hat{z})$, so $\sigma_t(\hat{z})[1]$ is the highest-probability token given $\hat{z}$, $\sigma_t(\hat{z})[2]$ is the second-highest, etc. The permutation $\sigma_t(\hat{z})$ is recomputed per parent and per step because the base probabilities $\Pr(\cdot \mid \hat{z})$ depend on the parent sequence. Define the smallest rank index at step t

$$R_t(\hat{z}; p) \;:=\; \min\Big\{ r \in \{1, \ldots, |\mathbb{V}|\} \;:\; \sum_{i=1}^{r} \Pr\big(\sigma_t(\hat{z})[i] \mid \hat{z}\big) \;\geq\; p \Big\}.$$

This makes the **nucleus set**

$$\mathbb{C}_t(\hat{z}; p) \;\triangleq\; \big\{\, \sigma_t(\hat{z})[i] \;:\; 1 \leq i \leq R_t(\hat{z}; p) \,\big\} \tag{38}$$

uniquely defined. (We slightly overload the notation $\mathbb{C}_t$ from Appendix D.1, disambiguating here with p as an argument.) The corresponding per-step nucleus normalization constant is

$$Z_t(\hat{z}; p) \triangleq \sum_{u \in \mathbb{C}_t(\hat{z}; p)} \Pr(u \mid \hat{z}). \tag{39}$$

Analogous to Equation 11, define the per-step, renormalized nucleus probability by

$$\Pr_p(u \mid \hat{z}) := \begin{cases} \dfrac{\Pr(u \mid \hat{z})}{Z_t(\hat{z}; p)}, & u \in \mathbb{C}_t(\hat{z}; p), \\ 0, & u \notin \mathbb{C}_t(\hat{z}; p). \end{cases} \tag{40}$$

Nucleus-constrained beam search. A baseline approach for p-CBS is similar to the baseline k-CBS (Algorithm 2), with two substitutions: (i) replace the fixed top-k rule by the dynamic nucleus rule $u \in \mathbb{C}_t(\hat{z}; p)$; (ii) replace the scoring increment with $\log \Pr_p(u \mid \hat{z})$. We need

- **Nucleus mass parameter** $p \in (0, 1]$ (defines $\mathbb{C}_t(\hat{z}; p)$).
- **Beam cap** B applied after pooling children at each step $t < T$. Setting $B = \infty$ yields the variable-width beam (keep all children) at each step.

Then, at each step t and for each beam prefix $\hat{z}$:

1. **Child set.** Expand $\hat{z}$ only to tokens $u \in \mathbb{C}_t(\hat{z}; p)$.
2. **Scoring rule.** For a child $\hat{z} \| u$, update its cumulative log-score by adding
$$\log \Pr_p(u \mid \hat{z}) \;=\; \log \Pr(u \mid \hat{z}) \;-\; \log Z_t(\hat{z}; p). \tag{41}$$
And so, a length-t partial path has cumulative score $\sum_{s=1}^{t} \log \Pr_p(\hat{z}_s \mid z_{(\text{pre})} \| \hat{z}^{(\text{cont})}_{1:s-1})$.
3. **Pruning.** Pool all children from the current beam and (aside from the same EOS policy as k-CBS) retain them according to one of the following rules:
   - *Variable-width beam (full nucleus expansion).* Keep *all* children $\bigcup_{\hat{z} \in \text{beam}} \mathbb{C}_t(\hat{z}; p)$; in variable-width mode ($B = \infty$), the step-t beam cardinality is
   $$\sum_{\hat{z} \text{ in the step-}(t-1) \text{ beam}} \big| \, \mathbb{C}_t(\hat{z}; p) \, \big|.$$
   (This can be quite expensive, depending on $|\mathbb{C}_t(\hat{z}; p)|$; see discussion below about computational considerations.)
   - *Capped beam.* Keep the *top* B children by cumulative log-score, exactly as in Algorithm 2.

Within a fixed parent $\hat{z}$, the term $-\log Z_t(\hat{z}; p)$ is constant across its children, so that parent's local ranking can use $\log \Pr(u \mid \hat{z})$. When pooling children across *different* parents, $Z_t(\hat{z}; p)$ varies with $\hat{z}$; subtracting $\log Z_t(\hat{z}; p)$ aligns scales so that global sorting reflects $\log \Pr_p(\cdot \mid \cdot)$. (The same cross-parent adjustment appears in the k-CBS baseline, using its per-step top-k normalizer.)

Returned results and bias. Let $\mathbb{F} \subseteq \mathbb{V}^T \times \mathbb{R}$ be the set of continuations and their log probabilities returned by p-CBS. We return all step-T children generated from the step-$(T-1)$ beam:

$$|\mathbb{F}| \;=\; \sum_{\hat{z} \text{ in the step-}(T-1) \text{ beam}} \big| \, \mathbb{C}_T(\hat{z}; p) \, \big|.$$

The covered mass is

$$\text{covered_mass}_p(\mathbb{F}) := \sum_{(\hat{z}_{(\text{cont})}, \log p) \in \mathbb{F}} \prod_{t=1}^{T} \Pr_p\big(\hat{z}_t \,\big|\, z_{(\text{pre})} \| \hat{z}^{(\text{cont})}_{1:t-1}\big), \tag{42}$$

where $\hat{z}^{(\text{cont})}_{1:0}$ is the empty token sequence. Because across-beam pruning discards feasible paths, the covered mass is a downward-biased (but correct) lower bound on the total mass under the nucleus filtering rule (directly analogous to the top-k baseline).

Computational considerations (adaptive branching and mitigation). The size of $\mathbb{C}_t(\hat{z}; p)$ depends on the shape of the base distribution $\Pr(\cdot \mid \hat{z})$ at step t: when the distribution is sharp, few tokens are needed to reach total mass p; when it is flatter (many tokens with similar probability), $\mathbb{C}_t(\hat{z}; p)$ can be very large, as opposed to the constant size in k-CBS.

Why we do not explore p-CBS further in this work. We defer this to future work, as we find that top-k is sufficient for showing the value of our method. This choice also aligns with the main configurations reported by prior work on probabilistic extraction (Hayes et al., 2025b; Cooper et al., 2025).

Extensions mirror the top-k variants. Exactly as for the top-k constrained beams, one can (i) bake in a near-verbatim distance budget at decode time (an ε-viable filtering rule), and/or (ii) add tail closure beyond the beam. We omit these details for brevity.

E ε-pruned k-CBS

The baseline k-CBS algorithm (Algorithm 2) is a good starting place, but we can potentially do better in terms of compute cost. Rather than post-processing outputs to respect a chosen distance metric and budget ε, we can bake them into the search procedure and prune the beam as the algorithm evolves to only explore nodes that respect these settings. This only involves additional bookkeeping; it is free, with respect to the dominating cost of forward passes. In doing so, we can also possibly achieve a better upper bound (beyond the trivial one that baseline k-CBS algorithm provides) and a potentially better lower bound, as we can concentrate the search on ε-viable nodes only. The trade-off is that we do not get to run one algorithm, and then post-process for whichever dist and ε budget we might want to use; we have to specify these up front. But, an added benefit is that we will terminate the search procedure early (potentially very early) if no ε-viable continuations remain in the beam.

We describe these improvements below, first for the Hamming (Appendix E.1) and then for the Levenshtein distance (Appendix E.2). The former is significantly simpler, as Hamming is monotone non-decreasing and the Levenshtein is not. While both methods require additional bookkeeping (and therefore slight additional overhead compared to the baseline algorithm), they also can both terminate early if there are no continuations left in the beam that satisfy the chosen distance-based pruning rule. Overall, this can make them significantly cheaper to run than k-CBS, in terms of overall wall-clock time (Appendix C.3). We provide unified proofs of the invariants of both algorithms in Appendix E.3.

Note. This approach does not provably provide a stricter lower bound, because baking ε-viability into the pruning rule changes the candidate set of children that get expanded. In particular, a child that was not part of the baseline k-CBS candidate set at a given iteration (because it surfaced due to non-ε-viable children ranked above it being pruned) may be retained because it is ε-viable; that child may out-compete an ε-viable baseline k-CBS child at a later iteration, and then later get pruned if it is no longer viable—in which case neither child contributes to the final lower bound.

E.1 Hamming-ε-pruned k-CBS

This variant of k-CBS bakes a Hamming-distance ε-viability check into the search. That is, for prefix $z_{(\text{pre})}$, T-length target suffix $z_{(\text{suf})}$, and possible T-length continuations $z_{(\text{cont})}$, we seek bounds on

$$p^{\mathsf{Ham}}_{z,\varepsilon} \triangleq \sum_{z_{(\text{cont})} \in \mathbb{B}^{\mathsf{Ham}}_{\varepsilon}(z_{(\text{suf})})} \Pr_{\theta,\phi}(z_{(\text{cont})} \mid z_{(\text{pre})}), \tag{43}$$

$$\mathbb{B}^{\mathsf{Ham}}_{\varepsilon}(z_{(\text{suf})}) \triangleq \left\{ z_{(\text{cont})} \in \mathbb{V}^T : \mathsf{Hamming}(z_{(\text{cont})}, z_{(\text{suf})}) \leq \varepsilon \right\}. \tag{44}$$

At each step of the search, we keep a count for each partial path of the current Hamming distance. Paths whose running Hamming counter would exceed the budget ε are never expanded; they are removed from the beam. EOS handling is unchanged from Algorithm 2:

candidates whose last token is EOS are recorded and removed from the beam, but do not contribute to the lower bound. Only the mass of ε-viable paths removed by the across-beam prune is banked toward the upper bound (see below for details); non-ε-viable paths are discarded by Hamming monotonicity, and EOS-terminated paths are discarded because they cannot produce T-length continuations. As a result, no returned final continuation lies outside the Hamming ε-ball, and beam search pruning capacity is never spent on paths that are not ε-viable under Hamming distance.

In a bit more detail, the Hamming distance is monotone non-decreasing: once a mismatch occurs, it can never be undone. A partial path whose Hamming distance exceeds ε can never end up in the Hamming-distance ε-ball, $\mathbb{B}_\varepsilon^{\mathsf{Ham}}(\boldsymbol{z}_{(\text{suf})})$. This means that we can prune that path immediately. We refer to this approach as our **Hamming-ε-pruned k-CBS algorithm** (Algorithm 3). For this algorithm, unlike Algorithm 2, we do not wait until the end to run the final distance check on the returned sequences in $\mathbb{F}$. Further, since we establish ahead of time that we want to use Hamming as the distance metric with a chosen ε, continuing to expand such a node would not only be wasted work: any descendant sequences that remain in the beam would potentially be taking spots for (lower probability) sequences that would otherwise be included in the beam and may be in $\mathbb{B}_\varepsilon^{\mathsf{Ham}}(\boldsymbol{z}_{(\text{suf})})$. Pruning such sequences once they are encountered allows for the next-highest-probability candidate sequences (if they exist) to be included in the beam $\mathbb{L}$, so that we can expand them in the search instead.

Of course, pruning with this strategy is not as flexible as Algorithm 2, which can be used with any dist and choice of ε as a post-processing operation. But, it introduces additional benefits: namely, with some minimal bookkeeping, at no extra cost in token evaluations over Algorithm 2, this approach focuses effort on only viable nodes. It can terminate early if no such nodes exist and as a result can also potentially produce tighter lower and upper bounds. (However, this is not guaranteed.)

In a bit more detail, we keep track of the running count of Hamming mismatches between the t-token partial continuation $\hat{\boldsymbol{z}}_{1:t}^{(\text{cont})} = (\hat{z}_1^{(\text{cont})}, \ldots, \hat{z}_t^{(\text{cont})})$ and the target suffix $\boldsymbol{z}_{(\text{suf})} = (z_1^{(\text{suf})}, \ldots, z_T^{(\text{suf})})$. Define the running Hamming counter at depth t by

$$n_t(\hat{\boldsymbol{z}}_{(\text{cont})}, \boldsymbol{z}_{(\text{suf})}) \;\coloneqq\; \sum_{i=1}^{t} \mathbf{1}\big[\hat{z}_i^{(\text{cont})} \neq z_i^{(\text{suf})}\big]. \tag{45}$$

A partial $\hat{\boldsymbol{z}}_{1:t}^{(\text{cont})}$ is ε-viable iff $n_t(\hat{\boldsymbol{z}}_{1:t}^{(\text{cont})}, \boldsymbol{z}_{1:t}^{(\text{suf})}) \leq \varepsilon$.

That is, like in Algorithm 2, line 10, at step t, we produce t-length candidate partial sequences $\hat{\boldsymbol{z}}' \leftarrow \hat{\boldsymbol{z}} \| \hat{z}$; however, we only do so for the (at most $B \cdot k$) sequences for which the Hamming count is $\leq \varepsilon$, where each sequence is produced from appending each of the top-k tokens $\hat{z} \in \mathbb{S}_t(\hat{\boldsymbol{z}})$ to each of the B partial sequences (of length $t-1$) in the beam. Put differently, we immediately prune paths for which the mismatch counter exceeds ε, and therefore do not include them in the candidates that get ranked. As a result, the candidate list can be far smaller than $B \cdot k$. In fact, it is possible that no candidates remain—i.e., all $B \cdot k$ expanded candidates exceed ε—in which case we terminate the search procedure. The resulting lower bound is 0, and the upper bound is the banked mass we have computed so far (see below for more details). See Algorithm 3 for more details.

A potentially improved lower bound. Because we prune every child whose running Hamming distance exceeds ε, every final sequence (if there are any) in $\mathbb{F}$ at $t = T$ already satisfies $\mathsf{Hamming}(\hat{\boldsymbol{z}}_{(\text{cont})}, \boldsymbol{z}_{(\text{suf})}) \leq \varepsilon$. (In contrast, Algorithm 2 must filter outputs as a post-processing operation.) Consequently, *all* returned final sequences contribute to the lower bound on the near-verbatim mass:

$$\text{LB}_{\varepsilon,\mathsf{Ham}} \;=\; \sum_{(_,\log p)\in\mathbb{F}} \exp(\log p).$$

This can potentially exceed the baseline lower bound in Algorithm 2, because beam capacity is never spent on non-ε-viable paths during the search; however, this is not guaranteed, and in some cases could be worse. (See note at the end of Section E.)

A potentially improved upper bound. Algorithm 2 technically produces an upper bound on the probability for $p^{\mathsf{dist}}_{z,\varepsilon}$, but it is often too loose to be useful (Equation 34). In contrast, the Hamming-pruned version discussed here provides a potentially tighter upper bound. When the across-beam prune removes ε-viable sequences from the beam in a given iteration, we keep track of this information; we bank the mass of these viable sequences, which are roots of subtrees that potentially contain viable paths that contribute mass to $p^{\mathsf{dist}}_{z,\varepsilon}$, but which we do not explore further in the beam search procedure. (This is why it is an upper bound; some unexplored leaves of that subtree would not be ε-viable if explored.) We build up the upper bound additively from banked mass of ε-viable paths that leave the beam.

The key intuition for how this works depends on the fact that at a given step t, the beam (and the beam candidates that we prune) contains partial paths that are *parents* of any possible *descendant* sequences that the algorithm could produce at future steps (i.e., $\geq t$ and $\leq T$, inclusive). For each parent, the possible child sequences conserve their probability mass; the mass of a parent at t gets perfectly redistributed among all of its possible children (Lemma 9).

To see why, fix a partial continuation $\boldsymbol{u}$ at depth $t < T$. Top-k renormalization makes a distribution over the next-possible tokens $\mathbb{S}_{t+1}(\boldsymbol{z}_{(\text{pre})} \| \boldsymbol{u})$:

$$\sum_{v \in \mathbb{S}_{t+1}(\boldsymbol{z}_{(\text{pre})} \| \boldsymbol{u})} \Pr_{\theta,\phi}(v \mid \boldsymbol{z}_{(\text{pre})} \| \boldsymbol{u}) \;=\; 1.$$

By the chain rule,

$$\Pr_{\theta,\phi}(\boldsymbol{u} \| v \mid \boldsymbol{z}_{(\text{pre})}) = \Pr_{\theta,\phi}(\boldsymbol{u} \mid \boldsymbol{z}_{(\text{pre})}) \cdot \Pr_{\theta,\phi}(v \mid \boldsymbol{z}_{(\text{pre})} \| \boldsymbol{u}).$$

Summing over $v \in \mathbb{S}_{t+1}(\boldsymbol{z}_{(\text{pre})} \| \boldsymbol{u})$ yields the one-step conservation identity (see also Lemma 9, Equation 35):

$$\sum_{v \in \mathbb{S}_{t+1}(\boldsymbol{z}_{(\text{pre})} \| \boldsymbol{u})} \Pr_{\theta,\phi}(\boldsymbol{u} \| v \mid \boldsymbol{z}_{(\text{pre})}) = \Pr_{\theta,\phi}(\boldsymbol{u} \mid \boldsymbol{z}_{(\text{pre})}). \tag{46}$$

Therefore, before any pruning occurs, the mass at a parent continuation $\boldsymbol{u}$ is exactly redistributed across its immediate children. Iterating this identity level by level down a subtree shows that the total mass of all depth-T descendants of any partial $\boldsymbol{w}$ equals $\Pr_{\theta,\phi}(\boldsymbol{w} \mid \boldsymbol{z}_{(\text{pre})})$. (In general, please refer to Lemma 9.)

After pruning, we only keep some of the children. Of course, this means the sum over the kept children is $\leq \Pr_{\theta,\phi}(\boldsymbol{u} \mid \boldsymbol{z}_{(\text{pre})})$; the discarded remainder that contains ε-viable paths gets counted toward the upper bound. Refer to Appendix E.3.3 for more details.

How we use the UB bookkeeping. We keep a scalar accumulator *bank* that stores the total probability mass of ε-viable candidates that were pruned by the intermediate across-beam prune to the top-B paths when $t < T$. This accumulator is used in two different ways:

- **Early termination (no viable candidates before T).** If at some depth $t < T$ the viable set is empty, the algorithm stops; then
$$\mathrm{LB}_{\varepsilon,\mathsf{Ham}} \;=\; 0, \qquad \mathrm{UB}_{\varepsilon,\mathsf{Ham}} \;=\; \mathit{bank}.$$
- **Final step ($t = T$).** When we reach T, every returned final sequence in $\mathbb{F}$ is ε-viable, so
$$\mathrm{LB}_{\varepsilon,\mathsf{Ham}} \;=\; \sum_{(_,\log p) \in \mathbb{F}} \exp(\log p), \qquad \mathrm{UB}_{\varepsilon,\mathsf{Ham}} \;=\; \mathrm{LB}_{\varepsilon,\mathsf{Ham}} + \mathit{bank}.$$

To see why $\mathrm{UB}_{\varepsilon,\mathsf{Ham}}$ is a valid upper bound, observe that every depth-T node in the full top-k tree falls into exactly one of the following categories:

1. **In $\mathbb{F}$**: counted in $\mathrm{LB}_{\varepsilon,\mathsf{Ham}}$.
2. **Descendant of a banked node**: its mass is bounded by *bank*, since the total mass of all depth-T descendants of a banked ancestor equals that ancestor's mass (Lemma 9).

3. **Descendant of a Hamming-pruned node**: non-ε-viable by Hamming monotonicity, so it cannot contribute to $p^{\mathsf{Ham}}_{z,\varepsilon}$.
4. **Descendant of an EOS-terminated node**: cannot produce a T-length continuation and thus cannot contribute to $p^{\mathsf{Ham}}_{z,\varepsilon}$.

Since only categories (a) and (b) can contribute to $p^{\mathsf{Ham}}_{z,\varepsilon}$, we have $p^{\mathsf{Ham}}_{z,\varepsilon} \leq \mathrm{LB}_{\varepsilon,\mathsf{Ham}} + \mathit{bank} = \mathrm{UB}_{\varepsilon,\mathsf{Ham}}$.

As with baseline k-CBS (Appendix D.2), the optional $\tau_{\min}$-based early termination applies here as well: if $\max_{(_,\log p,_)\in\mathbb{L}_t} \exp(\log p) < \tau_{\min}/(B \cdot k)$, then the lower bound for the extraction probability can never reach $\tau_{\min}$ regardless of Hamming ε-viability. (The same caveat applies: since k-CBS is a greedy search, it may miss mass, so this is not an absolute guarantee of non-extractability; the lower bound is 0.)

Filtering the upper bound in practice. When a sequence terminates early—whether due to Hamming viability ($\mathbb{C}_t^{\leq\varepsilon} = \emptyset$) or the $\tau_{\min}$ threshold—the upper bound $\mathrm{UB}_{\varepsilon,\mathsf{Ham}} = \mathit{bank}$ is a valid bound on $p^{\mathsf{Ham}}_{z,\varepsilon}$, but we find that it is not informative in practice: the banked mass from pruned ε-viable ancestors can be substantial even when no final ε-viable path exists in the final search results. In our analysis, we therefore restrict attention to the upper bound only for sequences with $\mathrm{LB}_{\varepsilon,\mathsf{Ham}} > 0$ (i.e., those for which the algorithm found at least some near-verbatim mass). This filtering is a postprocessing step and does not affect the algorithm outputs.

Can return fewer than $B \cdot k$ suffixes. Because we prune any path at step t with $n_t > \varepsilon$ *before* ranking for the across-beam prune, the candidate set at depth t can be strictly smaller than $B \cdot k$ (or even empty, if there are no ε-viable paths remaining). As noted above, if at some $t < T$ no ε-viable candidates remain, we terminate; then $\mathrm{LB}^{(t)}_{\varepsilon,\mathsf{Ham}} = 0$ (EOS-terminated paths do not contribute to the lower bound) and $\mathrm{UB}^{(t)}_{\varepsilon,\mathsf{Ham}} = \mathit{bank}^{(t)}$ (the banked mass of the ε-viable prunes). Because of this within-search pruning, the procedure often performs *far fewer* token evaluations than k-CBS, which always expands B parents into up to $B \cdot k$ candidates at each depth (except in degenerate cases of many EOS tokens). (See discussion of token-evaluation cost comparisons in Appendix C.3.)

Tighter bounds at the cost of more forward passes. We could run a slight variation of this algorithm that, in addition to the minor bookkeeping described here, runs (potentially many) additional forward passes through the model to tighten the bounds. The basic idea is that, when the across-beam prune to B candidates prunes paths that have *exactly* cost ε, those paths each have *exactly* 1 viable full suffix in the ε-ball: the candidate path generated thus far at step t, plus the remaining $T - t$ verbatim tokens in the target suffix. We can gather these **tail-closed sequences**, and teacher force them at the end of our search algorithm in order to adjust the bounds to be tighter. We can also do the same for the Levenshtein distance. Given the cost of this approach, we do not pursue it in our experiments, especially since we observe useful estimates of the near-verbatim extraction probability without doing so.

Algorithm 3: Hamming-ε-pruned Top-k Constrained Beam Search

Input: LLM θ; prefix $\boldsymbol{z}_{(\text{pre})}$ of length a; target suffix $\boldsymbol{z}_{(\text{suf})} = (z_1^{(\text{suf})}, \ldots, z_T^{(\text{suf})}) \in \mathbb{V}^T$; beam width B; top-k parameter k for decoding policy ϕ ($B \le k^2$, $k \ll |\mathbb{V}|$); Hamming distance budget $\varepsilon \ll T$; EOS token id; optional $\tau_{\min} > 0$

Output: Set $\mathbb{F}$ of (at most $B \cdot k$) triples $(\hat{\boldsymbol{z}},\ \log p,\ n)$, where $\hat{\boldsymbol{z}} = \boldsymbol{z}_{(\text{pre})} \,\|\, \hat{\boldsymbol{z}}_{(\text{cont})}$ is a full history with length $a + T$, $\log p = \log \Pr_{\theta,\phi}(\hat{\boldsymbol{z}}_{(\text{cont})} \mid \boldsymbol{z}_{(\text{pre})})$, and $n \le \varepsilon$; lower bound $\text{LB}_{\varepsilon,\text{Ham}}$; upper bound $\text{UB}_{\varepsilon,\text{Ham}}$

Notation. Same as Algorithm 2, and $z_t^{(\text{suf})}$ denotes the t-th token of the target suffix $\boldsymbol{z}_{(\text{suf})}$.

Beam state. Maintain $\mathbb{L}_t$ as triples $(\hat{\boldsymbol{z}}, \log p, n)$, where $\log p$ is the accumulated top-k decoding log-probability and n is the running Hamming mismatch count $\sum_{i=1}^{t} \mathbf{1}\big[\hat{z}_i \neq z_i^{(\text{suf})}\big]$ (Equation 45). Max. beam capacity $|\mathbb{L}_t| = B$.

Viability test (monotone in t). $n' \leftarrow n + \mathbf{1}\big[\hat{z}_t \neq z_t^{(\text{suf})}\big]$. Keep child path only if $n' \le \varepsilon$; otherwise drop it permanently (Hamming mismatches cannot be undone).

Compute $\boldsymbol{y}_1(\boldsymbol{z}_{(\text{pre})})$ via forward pass on $\boldsymbol{z}_{(\text{pre})}$; `// Prefill: a token evals`

$\mathbb{L}_0 \leftarrow \{(\boldsymbol{z}_{(\text{pre})},\ 0,\ 0)\}$; `// Beam: triples (ẑ, log p, n)`

$bank \leftarrow 0$; `// UB mass accumulator`

for $t = 1, \ldots, T$ **do**

  $\mathbb{C}_t^{\le\varepsilon} \leftarrow \emptyset$; `// ε-viable candidate set for step t`

  foreach $(\hat{\boldsymbol{z}},\ \log p,\ n) \in \mathbb{L}_{t-1}$ **do**

    $\mathbb{S}_t(\hat{\boldsymbol{z}}) \leftarrow \texttt{TopK}_k\big(\boldsymbol{y}_t(\hat{\boldsymbol{z}})\big)$;

    $\boldsymbol{r}_t(\hat{\boldsymbol{z}}) \leftarrow \texttt{LogSoftmax}\big(\boldsymbol{y}_t(\hat{\boldsymbol{z}})\big)$;

    $Z_t(\hat{\boldsymbol{z}}) \leftarrow \texttt{LogSumExp}\big(\boldsymbol{r}_t(\hat{\boldsymbol{z}})[\mathbb{S}_t(\hat{\boldsymbol{z}})]\big)$;

    foreach $\hat{z} \in \mathbb{S}_t(\hat{\boldsymbol{z}})$ **do**

      $n' \leftarrow n + \mathbf{1}[\hat{z} \neq z_t^{(\text{suf})}]$; `// Update Hamming counter`

      if $n' \le \varepsilon$ **then** `// ε-viable: keep`

        $\hat{\boldsymbol{z}}' \leftarrow \hat{\boldsymbol{z}} \,\|\, \hat{z}$; `// Append token to partial history`

        $\log p' \leftarrow \log p + \boldsymbol{r}_t(\hat{\boldsymbol{z}})[\hat{z}] - Z_t(\hat{\boldsymbol{z}})$; `// Update continuation log prob`

        $\mathbb{C}_t^{\le\varepsilon} \leftarrow \mathbb{C}_t^{\le\varepsilon} \cup \{(\hat{\boldsymbol{z}}',\ \log p',\ n')\}$;

      `// If n' > ε: discard (non-ε-viable by Hamming monotonicity)`

  if $t = T$ **then** `// Final step: return all ε-viable candidates`

    $\mathbb{F} \leftarrow \mathbb{C}_T^{\le\varepsilon}$;

    $\text{LB}_{\varepsilon,\text{Ham}} \leftarrow \sum_{(_,\log p,_)\in\mathbb{F}} \exp(\log p)$;

    $\text{UB}_{\varepsilon,\text{Ham}} \leftarrow \text{LB}_{\varepsilon,\text{Ham}} + bank$;

    return $\mathbb{F}$, $\text{LB}_{\varepsilon,\text{Ham}}$, $\text{UB}_{\varepsilon,\text{Ham}}$;

  `// Non-final step (t < T): EOS handling, early termination, beam pruning`

  foreach $(\hat{\boldsymbol{z}}',\ \log p',\ n') \in \mathbb{C}_t^{\le\varepsilon}$ *with latest* $\hat{z} = \text{EOS}$ **do**

    Record $(\hat{\boldsymbol{z}}',\ \log p',\ t)$ as early-termination path; remove from $\mathbb{C}_t^{\le\varepsilon}$;

  if $\mathbb{C}_t^{\le\varepsilon} = \emptyset$ **then** `// Early termination: no ε-viable candidates remain`

    return $\mathbb{F} \leftarrow \emptyset$, $\text{LB}_{\varepsilon,\text{Ham}} \leftarrow 0$, $\text{UB}_{\varepsilon,\text{Ham}} \leftarrow bank$;

  $\mathbb{U}_t \leftarrow$ top-B elements of $\mathbb{C}_t^{\le\varepsilon}$ ranked by $\log p'$;

  $bank \leftarrow bank + \sum_{(_,\log p',_)\in\mathbb{C}_t^{\le\varepsilon}\setminus\mathbb{U}_t} \exp(\log p')$; `// Bank pruned ε-viable mass`

  $\mathbb{L}_t \leftarrow \mathbb{U}_t$; `// Prune to beam width`

  if $\tau_{\min}$ *and* $\max_{(_,\log p,_)\in\mathbb{L}_t} \exp(\log p) < \tau_{\min}/(B \cdot k)$ **then**

    return $\mathbb{F} \leftarrow \emptyset$, $\text{LB}_{\varepsilon,\text{Ham}} \leftarrow 0$, $\text{UB}_{\varepsilon,\text{Ham}} \leftarrow bank$;

  Compute $\boldsymbol{y}_{t+1}(\hat{\boldsymbol{z}})$ for each $(\hat{\boldsymbol{z}}, \cdot, \cdot) \in \mathbb{L}_t$; `// |L_t| token evals`

E.2 Levenshtein-ε-pruned k-CBS

We also design a k-CBS variant that uses the Levenshtein distance as its viability test. Fix the prefix $z_{(\text{pre})}$ and the target suffix $z_{(\text{suf})} = (z_1^{(\text{suf})}, \ldots, z_T^{(\text{suf})})$ of length T. For possible T-length continuations $z_{(\text{cont})}$ under decoding policy ϕ (again, we use top-k), we seek tight bounds on

$$p_{z,\varepsilon}^{\mathsf{Lev}} \triangleq \sum_{z_{(\text{cont})} \in \mathbb{B}_\varepsilon^{\mathsf{Lev}}(z_{(\text{suf})})} \Pr_{\theta,\phi}(z_{(\text{cont})} \mid z_{(\text{pre})}),$$

$$\mathbb{B}_\varepsilon^{\mathsf{Lev}}(z_{(\text{suf})}) \triangleq \{z_{(\text{cont})} \in \mathbb{V}^T : \mathsf{Levenshtein}(z_{(\text{cont})}, z_{(\text{suf})}) \leq \varepsilon\}. \tag{47}$$

For equal-length sequences, $\mathsf{Levenshtein} \leq \mathsf{Hamming}$, so Levenshtein-pruned beam search can be (strictly) less conservative than Hamming pruning (Appendix B.1, Theorem 1).

Why alignment state is needed (differs from Hamming). Unlike Hamming viability (Section 4.3 & Appendix E.1), Levenshtein ε-viability is *not* monotone because insertions and deletions can repair alignment later, when more tokens are added to the partial path: a partial path that currently "looks bad" by positional mismatches might still finish within ε edits after realignment—i.e., the minimum achievable Levenshtein distance for the completed continuation can decrease at later steps t.

Therefore, a running mismatch counter—sufficient for Hamming-based pruning—is insufficient for Levenshtein-based pruning. Instead, we must maintain **alignment state** that encodes the best attainable edit cost observed so far, as we generate (i.e., **stream**) new tokens, one at a time, for candidate paths. For this, **dynamic programming**[8] can help us keep track of state in an efficient way. We carry forward alignment state across search iterations, not just a counter of mismatched positions, in order to inform a safe Levenshtein-pruning rule.

In the following subsections, we discuss the intuition. We provide background on the Wagner-Fischer dynamic program solution for the Levenshtein distance (Appendix E.2.1), discuss how the Ukkonen's band refinement of this program applies to our ε setting (Appendix E.2.2), and explain how the pieces all fit together for a Levenshtein-pruned version of k-CBS (Appendix E.2.3). We provide a complete algorithm statement in Algorithm 4.

E.2.1 Wagner-Fischer dynamic program

We first need to provide some background on the dynamic program used to solve the Levenshtein distance. We will also include a simple worked example to give an intuition.

Let $\hat{z}_{1:i}^{(\text{cont})} = (\hat{z}_1^{(\text{cont})}, \ldots, \hat{z}_i^{(\text{cont})})$ be the generated partial path at depth i (i.e., does not include $z_{(\text{pre})}$), and $z_{1:j}^{(\text{suf})} = (z_1^{(\text{suf})}, \ldots, z_j^{(\text{suf})})$ be the j-length prefix of the target $z_{(\text{suf})}$. Define the table

$$D[i,j] \triangleq \mathsf{Levenshtein}(\hat{z}_{1:i}^{(\text{cont})}, z_{1:j}^{(\text{suf})}), \qquad i,j \in \{0,\ldots,T\}. \tag{48}$$

(Note that D has dimensions $(T+1)\times(T+1)$.) The row index i is the number of generated tokens, corresponding to $\hat{z}_{1:i}^{(\text{cont})}$. The column index j is the number of target tokens, corresponding to the prefix $z_{1:j}^{(\text{suf})}$ of the suffix $z_{(\text{suf})}$ that we align to. Therefore, the cell $D[i,j]$ is the minimum number of edits needed to turn the current i-length prefix of the (ultimately) T-length generated continuation $\hat{z}_{1:i}^{(\text{cont})}$ into the first j tokens of the target suffix $z_{1:j}^{(\text{suf})}$.

This table is an **edit graph**, a $(T+1)\times(T+1)$ grid in which each interior cell $D[i,j]$ (with $i,j \geq 1$) can be reached from three predecessors, corresponding to three edit operations:

- move $\downarrow$ (from $D[i-1,j]$ to $D[i,j]$) = **delete** $\hat{z}_i^{(\text{cont})}$ (cost 1): the row advances but the column stays—the new generated token is not aligned to any target token, so it is deleted. This reduces to the subproblem $\hat{z}_{1:i-1}^{(\text{cont})} \to z_{1:j}^{(\text{suf})}$, plus $+1$ for the deletion.

[8] We will never shorthand dynamic programming as "DP" because in ML privacy/security this refers to **differential privacy**.

- move $\rightarrow$ (from $D[i, j-1]$ to $D[i, j]$) = **insert** $z_j^{(\text{suf})}$ (cost 1): the column advances but the row stays—target token $z_j^{(\text{suf})}$ is not aligned to any generated token, so it is inserted. This reduces to the subproblem $\hat{\boldsymbol{z}}_{1:i}^{(\text{cont})} \rightarrow \boldsymbol{z}_{1:j-1}^{(\text{suf})}$, plus $+1$ for the insertion.
- move $\searrow$ (from $D[i-1, j-1]$ to $D[i, j]$) = **match/substitute** (cost 0 if $\hat{z}_i^{(\text{cont})} = z_j^{(\text{suf})}$, else 1): both indices advance. This reduces to the subproblem $\hat{\boldsymbol{z}}_{1:i-1}^{(\text{cont})} \rightarrow \boldsymbol{z}_{1:j-1}^{(\text{suf})}$, plus the match/substitute cost.

Boundary conditions. The boundaries of the table are

$$\begin{aligned} D[0,0] &= 0, \\ D[i,0] &= i \ \ (\text{delete all } \hat{z}_1^{(\text{cont})}, \ldots, \hat{z}_i^{(\text{cont})}), \\ D[0,j] &= j \ \ (\text{insert all } z_1^{(\text{suf})}, \ldots, z_j^{(\text{suf})}). \end{aligned} \tag{49}$$

Any optimal edit script for $\hat{\boldsymbol{z}}_{1:i}^{(\text{cont})} \rightarrow \boldsymbol{z}_{1:j}^{(\text{suf})}$ must end with one of these three operations, each building on the optimal cost of the corresponding smaller subproblem. This gives the standard recurrence:

$$D[i,j] \ = \ \min\{\underbrace{D[i-1,j]+1}_{\text{delete } \hat{z}_i^{(\text{cont})}}, \ \underbrace{D[i,j-1]+1}_{\text{insert } z_j^{(\text{suf})}}, \ \underbrace{D[i-1,j-1]+\mathbf{1}[\hat{z}_i^{(\text{cont})} \neq z_j^{(\text{suf})}]}_{\text{match / substitute}}\}. \tag{50}$$

Each term takes the optimal cost for the smaller subproblem, then pays for the last operation; the underbraces correspond to the three edit-graph moves above.

How to read a row in D and why the minimum over j matters. A row i summarizes all the ways to align the length-i partial path candidate sequence to the different possible prefixes of the target suffix. That is, at decoding depth $i = t$, the entire row $D[t,0], D[t,1], \ldots, D[t,T]$ summarizes all possible alignments of the current t-length partial path to every length-j prefix of the target suffix $\boldsymbol{z}_{(\text{suf})}$. The row minimum $\min_j D[t,j]$ answers: "what is the best we can possibly do right now (for cost) if later insertions/deletions are allowed to realign before reaching length T?" If that best cost already exceeds ε, then *every* full alignment path from $(0,0)$ to (T,T)—which must pass through some column $j^\star$ on row t—incurs partial cost $> \varepsilon$ at row t and cannot recover (because remaining edit costs are non-negative). Therefore, no continuation can end within ε.

A fully worked 2×3 example (token-by-token). Let $\boldsymbol{z}_{(\text{suf})} = (a,b,c)$ and $\hat{\boldsymbol{z}}_{1:2}^{(\text{cont})} = (b,c)$. We compute the entire table using the boundary conditions (Equation 49) and recurrence (Equation 50). See Figure 9 for a summary.

- **Row $i = 1$ ($\hat{z}_1^{(\text{cont})} = b$)**:
  - $D[1,1] = \min\{D[0,1]+1, \ D[1,0]+1, \ D[0,0]+\mathbf{1}[b \neq a]\} = \min\{2, \ 2, \ 1\} = 1$ (substitute)
  - $D[1,2] = \min\{D[0,2]+1, \ D[1,1]+1, \ D[0,1]+\mathbf{1}[b = b]\} = \min\{3, \ 2, \ 1\} = 1$ (match)
  - $D[1,3] = \min\{D[0,3]+1, \ D[1,2]+1, \ D[0,2]+\mathbf{1}[b \neq c]\} = \min\{4, \ 2, \ 3\} = 2$ (insert)
- **Row $i = 2$ ($\hat{z}_1^{(\text{cont})} = b$, $\hat{z}_2^{(\text{cont})} = c$)**:
  - $D[2,1] = \min\{D[1,1]+1, \ D[2,0]+1, \ D[1,0]+\mathbf{1}[c \neq a]\} = \min\{2, \ 3, \ 2\} = 2$ (delete/substitute tie)
  - $D[2,2] = \min\{D[1,2]+1, \ D[2,1]+1, \ D[1,1]+\mathbf{1}[c \neq b]\} = \min\{2, \ 3, \ 2\} = 2$ (delete/substitute tie)
  - $D[2,3] = \min\{D[1,3]+1, \ D[2,2]+1, \ D[1,2]+\mathbf{1}[c = c]\} = \min\{3, \ 3, \ 1\} = 1$ (match)

$D[i,j]$	$j{=}0$ ($\varnothing$)	$j{=}1$ (a)	$j{=}2$ (ab)	$j{=}3$ (abc)
$i{=}0$ ($\varnothing$)	0	1	2	3
$i{=}1$ (b)	1	1 $\searrow$	1 $\searrow$	2 $\rightarrow$
$i{=}2$ (bc)	2	2 $\downarrow\searrow$	2 $\downarrow\searrow$	1 $\searrow$

Figure 9: **Fully worked example.** Wagner-Fischer table for $z_{(\text{suf})} = (a, b, c)$ and $\hat{z}^{(\text{cont})}_{1:2} = (b, c)$. Colored arrows show the winning operation at each cell: $\searrow$ match/substitute, $\downarrow$ delete, $\rightarrow$ insert. The last row is $[2, 2, 2, 1]$ with $\min_j D[2, j] = 1$ at $j = 3$; the diagonal $D[2, 2]{=}2$ is not the minimum, illustrating why we minimize over all j.

$D[i,j]$	$j{=}0$ ($\varnothing$)	$j{=}1$ (a)	$j{=}2$ (ab)	$j{=}3$ (abc)	$j{=}4$ ($abcd$)	
$i{=}0$ ($\varnothing$)	0	1	2	3	4	
$i{=}1$ (b)	1	1 $\searrow$	1 $\searrow$	2 $\rightarrow$	3 $\rightarrow$	min$=$1
$i{=}2$ (bc)	2	2 $\downarrow\searrow$	2 $\downarrow\searrow$	1 $\searrow$	2 $\rightarrow$	min$=$1
$i{=}3$ (bca)	3	2 $\searrow$	3 $\downarrow\rightarrow\searrow$	2 $\downarrow$	2 $\searrow$	min$=$2
$i{=}4$ ($bcad$)	4	3 $\downarrow$	3 $\searrow$	3 $\downarrow$	2 $\searrow$	min$=$2

Figure 10: **Streaming example.** $z_{(\text{suf})} = (a, b, c, d)$ and $\hat{z}^{(\text{cont})}_{1:4} = (b, c, a, d)$. Row minima (shown at right) evolve as $1, 1, 2, 2$. With $\varepsilon{=}1$, pruning triggers at $t{=}3$; with $\varepsilon{=}2$, the path remains viable and ends at $D[4, 4]{=}2$. Colors as in Figure 9.

The last row is $[2, 2, 2, 1]$, so $\min_j D[2, j] = 1$ at $j{=}3$: "insert a at the front (cost 1), then match b, then match c." Note that the diagonal value $D[2, 2] = 2$ is *not* the best cost—showing why we must minimize over all j.

A streaming example (row minima can go down, then exceed a threshold). Let $z_{(\text{suf})} = (a, b, c, d)$ and consider the stream $\hat{z}^{(\text{cont})}_{1:1} = (b)$, $\hat{z}^{(\text{cont})}_{1:2} = (b, c)$, $\hat{z}^{(\text{cont})}_{1:3} = (b, c, a)$, $\hat{z}^{(\text{cont})}_{1:4} = (b, c, a, d)$. The row minima evolve as

$$\min_j D[1, j] = 1, \quad \min_j D[2, j] = 1, \quad \min_j D[3, j] = 2, \quad \min_j D[4, j] = 2.$$

At $i = t = 3$, inserting a breaks the earlier "insert-a-then-match" alignment and the best partial cost becomes 2; at $i{=}t{=}4$ the best cost remains 2 via a match on d. Therefore, with $\varepsilon{=}1$, we would prune at $i{=}t{=}3$; in contrast, with $\varepsilon{=}2$, the path remains viable through $i{=}t{=}4$ (and ends at distance 2). See Figure 10.

Two facts we use later. First, for any i, j:

$$D[i, j] \;\geq\; |i - j|, \tag{51}$$

i.e., turning a length-i string into a length-j string requires at least $|i{-}j|$ insertions or deletions. Second, any full alignment path from $(0, 0)$ to (T, T) must pass through some column $j^\star$ on each row i, and $D[i, j^\star]$ is the minimum partial edit cost to reach $(i, j^\star)$. Any completion from $(i, j^\star)$ to (T, T) adds a non-negative number of edits, so $D[i, j^\star]$ lower-bounds the total cost of any complete alignment path through $(i, j^\star)$. This gives our pruning rule: if $\min_j D[i, j] > \varepsilon$, then no extension of that partial path can achieve total cost $\leq \varepsilon$, and we can safely prune it.

Bookkeeping complexity. Maintaining the full (unbanded) row for each partial path requires $O(T)$ time and $O(T)$ space per decoding step (only the current and previous rows are needed). In Appendix E.2.2 we reduce both to $O(\varepsilon)$ per step per partial path via banding, which is the regime we use in practice.

E.2.2 Ukkonen's band and a viable pruning rule

For our purposes, we only need to retain partial paths for which $D[i, j] \leq \varepsilon$ could hold. By Equation 51, $D[i, j] \geq |i{-}j|$, so any cell with $|i{-}j| > \varepsilon$ satisfies $D[i, j] > \varepsilon$ and cannot contribute to the row minimum that we use to determine ε-viability pruning. Concretely,

$D[i,j] \le \varepsilon$ requires $|i-j| \le \varepsilon$, i.e., $i-\varepsilon \le j \le i+\varepsilon$. Clamping to valid column indices $j \in \{0,\dots,T\}$, at row i it suffices to maintain only the **band**

$$j \in [\, j_{\min}(i),\, j_{\max}(i)\,] \;\equiv\; [\,\max\{0,\, i-\varepsilon\},\, \min\{T,\, i+\varepsilon\}\,], \tag{52}$$

a diagonal strip centered on the main diagonal $j{=}i$. The unclamped interval $[i{-}\varepsilon,\, i{+}\varepsilon]$ has width $2\varepsilon{+}1$; the band is its intersection with $[0, T]$, so it contains at most $2\varepsilon{+}1$ entries for every row i. We treat out-of-band entries as $+\infty$ (never chosen by the min in the recurrence), reducing the per-row cost from $O(T)$ to $O(\varepsilon)$—a strict improvement when $\varepsilon \ll T$, which is the regime of our near-verbatim extraction setting. This is exactly the restriction used in **Ukkonen's thresholded dynamic program** (Ukkonen, 1985; Navarro, 2001).

The banded pruning rule for a partial path at decoding depth i is then:

$$\min_{j\in[j_{\min}(i),\, j_{\max}(i)]} D[i,j] \;>\; \varepsilon \quad\Longrightarrow\quad \text{prune (no viable completion exists).} \tag{53}$$

This is sound because the banded test implies the *full* row minimum exceeds ε: every in-band column satisfies $D[i,j] > \varepsilon$ (checked directly by the test), and every out-of-band column satisfies $D[i,j] \ge |i-j| > \varepsilon$ (by Equation 51). Since $\min_j D[i,j] > \varepsilon$ over all columns, no completion of this partial path can achieve total cost $\le \varepsilon$, by the pruning argument in Appendix E.2.1 (Lemma 17).

Streaming, banded updates (how we maintain state). For each active beam item at depth t, we store the banded row $D[t,j]$ for $j \in [j_{\min}(t),\, j_{\max}(t)]$ (Equation 52), initialized at $t{=}0$ with $D[0,j] = j$ for $j \in [0,\, \min\{T,\varepsilon\}]$. This single row is sufficient to compute the next: the recurrence (Equation 50) for $D[t{+}1,j]$ reads only from row t (the $\downarrow$ delete and $\searrow$ match/substitute predecessors) and from earlier entries of row $t{+}1$ (the $\rightarrow$ insert predecessor, already filled left to right). In practice, we pre-allocate two buffers of size $2\varepsilon{+}1$ and alternate between them, avoiding per-step allocation. When we append a token $\hat{z}_{t+1}^{(\text{cont})}$:

1. **Compute** the next band limits $[j_{\min}(t{+}1),\, j_{\max}(t{+}1)]$ via Equation 52.
2. **Fill** $D[t{+}1,j]$ for $j = j_{\min}(t{+}1),\dots,j_{\max}(t{+}1)$ (left to right) using Equation 50, treating any out-of-band predecessor as $+\infty$. This keeps the recurrence exactly consistent with Wagner-Fischer.
   - For instance, when $j{=}j_{\min}(t{+}1){=}0$, the $\rightarrow$ insert predecessor $D[t{+}1,-1]$ and the $\searrow$ match/substitute predecessor $D[t,-1]$ are out-of-band ($+\infty$), so only the $\downarrow$ delete predecessor remains: $D[t{+}1,0] = D[t,0] + 1$.
   - The left-to-right fill order ensures that $D[t{+}1,j{-}1]$ has already been computed when needed for the $\rightarrow$ insert term.
3. **Prune** this partial path immediately if $\min_{j\in[j_{\min}(t+1),\, j_{\max}(t+1)]} D[t{+}1,j] > \varepsilon$ (Lemma 17); otherwise, carry its banded row forward as the new state.

This keeps a beam item's per-child update cost and memory both $O(\varepsilon)$—the same order of bookkeeping overhead as the Hamming-pruned variant (which maintains a single mismatch counter per beam item). Note that Levenshtein-pruning is less conservative than Hamming-pruning, so fewer partial paths may be pruned, potentially requiring more forward passes; this is a difference in pruning rate, not in per-step overhead. We discuss the cost comparison further in the next subsection.

A note on further optimization. The per-step bookkeeping cost could be reduced further using Myers' bit-vector algorithm (Myers, 1999; Navarro, 2001), which encodes the banded Wagner-Fischer recurrence with a handful of machine-word operations per step. We do not implement this, as the $O(\varepsilon)$ bookkeeping overhead is already negligible relative to the cost of a model forward pass at each decoding step.

E.2.3 Integrating the Levenshtein-pruning dynamic program into k-CBS

We incorporate this bookkeeping without additional model calls to form our Levenshtein-pruned k-CBS variant. We use t to index into the row of the dynamic programming table (instead of i) to align with the generation step.

- **Per-partial sequence state.** Maintain triples $(\hat{z}, \log p, \text{aux})$ with aux $= \big(D[t,\cdot], j_{\min}(t), j_{\max}(t)\big)$, i.e., the step-$t$ dynamic programming table row for the step-t Ukkonen band (Equation 52).
- **EOS policy.** If an EOS is the latest generated token in a beam candidate at step $t < T$, we discard the path, i.e., do not include it in the next beam. (At T, we allow the last token to be EOS.) This enforces that all returned continuations have fixed length T.
- **Child sequence $\hat{z}'$ formation and ε-viability.** Just as with the Hamming variant (Algorithm 3), all probability bookkeeping ($\log p$, Z_t, etc.) matches our baseline k-CBS algorithm (Algorithm 2). For history (including the prefix) $\hat{z}$, for each $\hat{z} \in \mathbb{S}_{t+1}(\hat{z})$, append $\hat{z}$ (i.e., $\hat{z}' = \hat{z} \| \hat{z}$), update the banded row in $O(\varepsilon)$, and keep the child if and only if the updated band minimum $\leq \varepsilon$ (i.e., $\min_{j \in [j_{\min}(t+1), j_{\max}(t+1)]} D[t+1, j] \leq \varepsilon$) (Lemma 17). This is the streamed Ukkonen viability condition, adapted to our fixed-length-T target and per-row band.
- **Across-beam prune.** Among ε-viable children, sort by $\log p'$ and keep up to B in the beam, as usual. (Non-ε-viable partial paths are discarded before ranking.)
- **Intermediate UB bookkeeping.** For $t < T$ we bank only across-beam pruned ε-viable partial paths. That is, as with the Hamming-pruned variant (Appendix E.1), we can bank the mass of pruned but still ε-viable nodes (band minimum $\leq \varepsilon$ at prune time, but $\log p$ did not sort this path into the top-B) for a potentially tighter upper bound; nodes pruned after they become non-ε-viable (band minimum $> \varepsilon$) contribute 0 to this banked mass.
- **Final acceptance test.** At $t = T$, accept a continuation if and only if $D[T, T] \leq \varepsilon$; that way, every continuation in the final set $\mathbb{F}$ has $\mathsf{Levenshtein} \leq \varepsilon$ by construction. Unlike the $\mathsf{Hamming}$-pruned version, we need an explicit final acceptance check. The ε-viability test used during search asks whether the partial path is within ε edits of *some* prefix of the target (the minimum over j), which is the correct criterion for safe pruning. But a T-length continuation can pass that test—by aligning well to a shorter prefix ($j < T$)—while still having $D[T, T] > \varepsilon$, i.e., exceeding ε edits against the full target. The explicit check $D[T, T] \leq \varepsilon$ addresses this. (For Hamming, no separate check is needed: the mismatch counter at step T already equals $\mathsf{Hamming}(\boldsymbol{z}_{(\text{cont})}, \boldsymbol{z}_{(\text{suf})})$.)
- **Bounds.** We do not perform the last across-beam prune, keeping up to $B \cdot k$ finals. $\mathrm{LB}_{\varepsilon,\mathsf{Lev}}$ is the sum of $\exp(\log p)$ over all kept final sequences (all are $\mathsf{Lev} \leq \varepsilon$, by construction); non-$\varepsilon$-viable finals contribute to neither the lower bound nor the upper bound bank. $\mathrm{UB}_{\varepsilon,\mathsf{Lev}} = \mathit{bank} + \mathrm{LB}_{\varepsilon,\mathsf{Lev}}$, with no mass counted twice.
- **Early termination.** If at some $t < T$ no ε-viable children remain ($\mathbb{C}_t^{\leq \varepsilon} = \emptyset$), the algorithm terminates with $\mathrm{LB}_{\varepsilon,\mathsf{Lev}} = 0$ and $\mathrm{UB}_{\varepsilon,\mathsf{Lev}} = \mathit{bank}$, i.e., the mass banked from ε-viable nodes pruned so far, which upper-bounds the remaining near-verbatim mass by the descendant-sum/frontier identity, just as in the Hamming-pruned variant (Lemma 20). Additionally, the same minimum-probability early termination used in the Hamming variant applies here: if the highest-probability beam candidate falls below a threshold derived from a user-specified minimum extraction probability, we terminate with $\mathrm{LB}_{\varepsilon,\mathsf{Lev}} = 0$ and $\mathrm{UB}_{\varepsilon,\mathsf{Lev}} = \mathit{bank}$. In practice, the upper bound from early-terminated examples is too loose to be informative; we filter these at reporting time.

Complexity and practical choices. Scalar-banded Wagner-Fischer (Wagner & Fischer, 1974) (with Ukkonen's criterion (Ukkonen, 1985)) requires $O(\varepsilon)$ time and $O(\varepsilon)$ memory per partial path per step, totaling $O(B\,k\,T\,\varepsilon)$ time and $O(B\,k\,\varepsilon)$ peak memory across the beam (reduced to $O(B\,\varepsilon)$ after each across-beam prune). This is purely bookkeeping—no extra forward passes through the model—so the overall cost is dominated by the same number of token evaluations as in baseline k-CBS.

Algorithm 4: Levenshtein-ε-pruned Top-k Constrained Beam Search

Input: LLM θ; prefix $\boldsymbol{z}_{(\text{pre})}$ of length a; target suffix $\boldsymbol{z}_{(\text{suf})} = (z_1^{(\text{suf})}, \ldots, z_T^{(\text{suf})}) \in \mathbb{V}^T$; beam width B; top-k parameter k for decoding policy ϕ ($B \le k^2$, $k \ll |\mathbb{V}|$); Levenshtein distance budget $\varepsilon \ll T$; EOS token id; optional $\tau_{\min} > 0$

Output: Set $\mathbb{F}$ of (at most $B \cdot k$) triples $(\hat{\boldsymbol{z}},\ \log p,\ d)$, where $\hat{\boldsymbol{z}} = \boldsymbol{z}_{(\text{pre})} \,\|\, \hat{\boldsymbol{z}}_{(\text{cont})}$ is a full history with length $a + T$, $\log p = \log \Pr_{\theta,\phi}(\hat{\boldsymbol{z}}_{(\text{cont})} \mid \boldsymbol{z}_{(\text{pre})})$, and $d = D[T,T] = \mathsf{Levenshtein}(\hat{\boldsymbol{z}}_{(\text{cont})}, \boldsymbol{z}_{(\text{suf})}) \le \varepsilon$; lower bound $\mathrm{LB}_{\varepsilon,\mathsf{Lev}}$; upper bound $\mathrm{UB}_{\varepsilon,\mathsf{Lev}}$

Notation. Same as Algorithm 2. Additionally: $z_t^{(\text{suf})}$ denotes the t-th token of $\boldsymbol{z}_{(\text{suf})}$; $D[t,j] \triangleq \mathsf{Levenshtein}(\hat{\boldsymbol{z}}_{1:t}^{(\text{cont})}, \boldsymbol{z}_{1:j}^{(\text{suf})})$ (Equation 48); $[j_{\min}(t),\ j_{\max}(t)]$ is the Ukkonen band at depth t (Equation 52). We write $D[t,\cdot]$ for the banded row $\big(D[t,j]\big)_{j \in [j_{\min}(t),\, j_{\max}(t)]}$.

Beam state. Maintain $\mathbb{L}_t$ as triples $(\hat{\boldsymbol{z}}, \log p, D[t,\cdot])$, where $\log p$ is the accumulated top-k decoding log-probability and $D[t,\cdot]$ is the banded Wagner-Fischer row for the partial path at depth t. Max. beam capacity $|\mathbb{L}_t| = B$.

Viability test (not monotone in t; see Appendix E.2.1). After appending token $\hat{z}$ at depth t, compute the updated banded row $D[t,j]$ for $j \in [j_{\min}(t),\ j_{\max}(t)]$ via Equation 50, treating out-of-band predecessors as $+\infty$. Keep child only if $\min_{j \in [j_{\min}(t),\, j_{\max}(t)]} D[t,j] \le \varepsilon$; otherwise prune (Equation 53).

Compute $\boldsymbol{y}_1(\boldsymbol{z}_{(\text{pre})})$ via forward pass on $\boldsymbol{z}_{(\text{pre})}$; `// Prefill: a token evals`

$\mathbb{L}_0 \leftarrow \{(\boldsymbol{z}_{(\text{pre})},\ 0,\ D[0,\cdot])\}$ where $D[0,j] = j$ for $j \in [0, \min\{T, \varepsilon\}]$; `// Init beam`

$\mathit{bank} \leftarrow 0$; `// UB mass accumulator`

for $t = 1, \ldots, T$ **do**

  $\mathbb{C}_t^{\le\varepsilon} \leftarrow \emptyset$; `// ε-viable candidate set for step t`

  foreach $(\hat{\boldsymbol{z}},\ \log p,\ D[t-1,\cdot]) \in \mathbb{L}_{t-1}$ **do**

    $\mathbb{S}_t(\hat{\boldsymbol{z}}) \leftarrow \mathsf{TopK}_k\big(\boldsymbol{y}_t(\hat{\boldsymbol{z}})\big)$;

    $\boldsymbol{r}_t(\hat{\boldsymbol{z}}) \leftarrow \mathsf{LogSoftmax}\big(\boldsymbol{y}_t(\hat{\boldsymbol{z}})\big)$;

    $Z_t(\hat{\boldsymbol{z}}) \leftarrow \mathsf{LogSumExp}\big(\boldsymbol{r}_t(\hat{\boldsymbol{z}})[\mathbb{S}_t(\hat{\boldsymbol{z}})]\big)$;

    foreach $\hat{z} \in \mathbb{S}_t(\hat{\boldsymbol{z}})$ **do**

      Compute $D[t,j]$ for $j \in [j_{\min}(t),\ j_{\max}(t)]$ from $D[t-1,\cdot]$ and $\hat{z}$; `// O(ε) DP update`

      if $\min_{j \in [j_{\min}(t),\, j_{\max}(t)]} D[t,j] \le \varepsilon$ **then** `// ε-viable: keep`

        $\hat{\boldsymbol{z}}' \leftarrow \hat{\boldsymbol{z}} \,\|\, \hat{z}$; `// Append token to partial history`

        $\log p' \leftarrow \log p + \boldsymbol{r}_t(\hat{\boldsymbol{z}})[\hat{z}] - Z_t(\hat{\boldsymbol{z}})$; `// Update continuation log prob`

        $\mathbb{C}_t^{\le\varepsilon} \leftarrow \mathbb{C}_t^{\le\varepsilon} \cup \{(\hat{\boldsymbol{z}}',\ \log p',\ D[t,\cdot])\}$;

      `// If band min > ε: prune (no viable completion; Eq. 53)`

  if $t = T$ **then** `// Final step: acceptance test and bounds`

    $\mathbb{F} \leftarrow \{(\hat{\boldsymbol{z}}', \log p', D[T,T]) \in \mathbb{C}_T^{\le\varepsilon} : D[T,T] \le \varepsilon\}$; `// Accept iff Levenshtein ≤ ε`

    $\mathrm{LB}_{\varepsilon,\mathsf{Lev}} \leftarrow \sum_{(_,\log p,_) \in \mathbb{F}} \exp(\log p)$;

    $\mathrm{UB}_{\varepsilon,\mathsf{Lev}} \leftarrow \mathrm{LB}_{\varepsilon,\mathsf{Lev}} + \mathit{bank}$;

    return $\mathbb{F},\ \mathrm{LB}_{\varepsilon,\mathsf{Lev}},\ \mathrm{UB}_{\varepsilon,\mathsf{Lev}}$;

  `// Non-final step (t < T): EOS handling, early termination, beam pruning`

  foreach $(\hat{\boldsymbol{z}}',\ \log p',\ D[t,\cdot]) \in \mathbb{C}_t^{\le\varepsilon}$ *with latest* $\hat{z} = \mathrm{EOS}$ **do**

    Record $(\hat{\boldsymbol{z}}',\ \log p',\ t)$ as early-termination path; remove from $\mathbb{C}_t^{\le\varepsilon}$;

  if $\mathbb{C}_t^{\le\varepsilon} = \emptyset$ **then** `// Early termination: no ε-viable candidates remain`

    return $\mathbb{F} \leftarrow \emptyset,\ \mathrm{LB}_{\varepsilon,\mathsf{Lev}} \leftarrow 0,\ \mathrm{UB}_{\varepsilon,\mathsf{Lev}} \leftarrow \mathit{bank}$;

  $\mathbb{U}_t \leftarrow$ top-B elements of $\mathbb{C}_t^{\le\varepsilon}$ ranked by $\log p'$;

  $\mathit{bank} \leftarrow \mathit{bank} + \sum_{(_,\log p',_) \in \mathbb{C}_t^{\le\varepsilon} \setminus \mathbb{U}_t} \exp(\log p')$; `// Bank pruned ε-viable mass`

  $\mathbb{L}_t \leftarrow \mathbb{U}_t$; `// Prune to beam width`

  if $\tau_{\min}$ *and* $\max_{(_,\log p,_) \in \mathbb{L}_t} \exp(\log p) < \tau_{\min}/(B \cdot k)$ **then**

    return $\mathbb{F} \leftarrow \emptyset,\ \mathrm{LB}_{\varepsilon,\mathsf{Lev}} \leftarrow 0,\ \mathrm{UB}_{\varepsilon,\mathsf{Lev}} \leftarrow \mathit{bank}$;

  Compute $\boldsymbol{y}_{t+1}(\hat{\boldsymbol{z}})$ for each $(\hat{\boldsymbol{z}},\cdot,\cdot) \in \mathbb{L}_t$; `// |L_t| token evals`

E.3 Invariants for ε-pruned k-CBS

To show invariants for the ε-pruned k-CBS algorithms, we first give some definitions and common facts that we will use in our proofs (Appendix E.3.1), and some specific results for each distance metric (Appendix E.3.2). Then we present proofs of the respective algorithm invariants, leveraging a unified framework (Appendix E.3.3). We show that all returned finals are ε-viable under the chosen distance metric for pruning, observations about the cardinality bounds of the returned finals, and early termination upper bounds. It is not the case that the lower and upper bounds returned by ε-pruned variants are guaranteed to be tighter than those identified by k-CBS, but we often observe them to be in practice.

E.3.1 Distance-viability-pruned k-CBS: Notation and background

Shared viability semantics. The Hamming- and Levenshtein-pruned variants share identical beam-search machinery—expansion, probability bookkeeping, banking, early termination—and differ only in how viability is tracked. We abstract the distance-specific logic into three operations on a per-path auxiliary state aux, parameterized by a target suffix $\boldsymbol{z}_{(\text{suf})}$ and distance budget ε:

- $\mathsf{Viable.Init}(\boldsymbol{z}_{(\text{suf})}, \varepsilon) \to \text{aux}_0$: initialize per-path viability state.
- $\mathsf{Viable.Update}(\text{aux}, \hat{z}, t, \boldsymbol{z}_{(\text{suf})}, \varepsilon) \to (\text{aux}', \varepsilon_\star)$: update aux after appending token $\hat{z}$ at depth t, and return a lower bound $\varepsilon_\star$ on the final distance $\mathsf{dist}(\hat{\boldsymbol{z}}_{(\text{cont})}, \boldsymbol{z}_{(\text{suf})})$ achievable by any completion of this partial path. If $\varepsilon_\star > \varepsilon$, the path is provably non-ε-viable and is pruned.
- $\mathsf{Viable.IsFinal}(\text{aux}, \varepsilon) \to \{0, 1\}$: at depth T, return 1 if the completed continuation satisfies $\mathsf{dist}(\hat{\boldsymbol{z}}_{(\text{cont})}, \boldsymbol{z}_{(\text{suf})}) \leq \varepsilon$, and 0 otherwise.

We instantiate these semantics for the Hamming and Levenshtein distance as follows:

	Hamming (dist = Hamming)	**Levenshtein** (dist = Levenshtein)
aux	Mismatch count $n \in \{0, \ldots, \varepsilon\}$	Banded DP row $D[t, \cdot] = \big(D[t, j]\big)_{j \in [j_{\min}(t),\, j_{\max}(t)]}$
Init	$\text{aux}_0 = 0$	$D[0, j] = j$ for $j \in [0,\ \min\{T, \varepsilon\}]$
Update	$n' \leftarrow n + \mathbf{1}[\hat{z} \neq z_t^{(\text{suf})}]$; $\varepsilon_\star \leftarrow n'$	Fill $D[t, \cdot]$ via Eq. 50 (banded; Eq. 52); $\varepsilon_\star \leftarrow \min_{j \in [j_{\min}(t),\, j_{\max}(t)]} D[t, j]$
IsFinal	Always 1 (viability $\Rightarrow$ acceptance)	$\mathbf{1}[D[T, T] \leq \varepsilon]$
Cost per call	$O(1)$	$O(\varepsilon)$

For Hamming, the mismatch count is monotonically non-decreasing along any path, so $\varepsilon_\star$ never decreases—and surviving to depth T with $n \leq \varepsilon$ already guarantees $\mathsf{Hamming}(\hat{\boldsymbol{z}}_{(\text{cont})}, \boldsymbol{z}_{(\text{suf})}) \leq \varepsilon$, making IsFinal trivially true. For Levenshtein, $\varepsilon_\star$ can decrease between steps (Appendix E.2.1), but remains a valid lower bound at each step; the explicit final check $D[T, T] \leq \varepsilon$ is needed because the viability test during search only ensures alignment to *some* prefix of $\boldsymbol{z}_{(\text{suf})}$, not the full target (Appendix E.2.3). We show how this interface can get integrated with k-CBS in Algorithm 5.

This viability interface accommodates any token-level distance metric for which a lower bound on the final distance can be computed incrementally from streamable per-path state, e.g., LCS distance and weighted edit distances. For these semantics, we provide common notation, soundness criteria for viability pruning rules, and some basic facts that we use in our proofs.

Token-sequence notation. We recall the sequence notation from Appendix D.1. As noted in that section, this notation lets us refer to prefixes, suffixes, and partial continuations directly, avoiding index arithmetic on the full history $\hat{\boldsymbol{z}}$.

Algorithm 5: ε-pruned Top-k Constrained Beam Search (dist-pruned k-CBS)

Input: LLM θ; prefix $\boldsymbol{z}_{(\text{pre})}$; target suffix $\boldsymbol{z}_{(\text{suf})} \in \mathbb{V}^T$; beam width B; top-k parameter k; distance budget ε; viability oracle (Init, Update, IsFinal); EOS token id; optional $\tau_{\min} > 0$
Output: Set $\mathbb{F}$ of (at most $B \cdot k$) triples $(\hat{\boldsymbol{z}},\ \log p,\ \text{aux})$ with $\mathsf{IsFinal}(\text{aux}, \varepsilon) = 1$; $\mathrm{LB}_{\varepsilon,\mathsf{dist}}$; $\mathrm{UB}_{\varepsilon,\mathsf{dist}}$

Compute $\boldsymbol{y}_1(\boldsymbol{z}_{(\text{pre})})$ via forward pass on $\boldsymbol{z}_{(\text{pre})}$; `// Prefill`
$\mathbb{L}_0 \leftarrow \{(\boldsymbol{z}_{(\text{pre})},\ 0,\ \mathsf{Viable.Init}(\boldsymbol{z}_{(\text{suf})}, \varepsilon))\}$; `// Beam (max. capacity B): triples (ẑ, log p, aux)`
$\mathit{bank} \leftarrow 0$;
for $t = 1, \ldots, T$ **do**
  $\mathbb{C}_t^{\leq\varepsilon} \leftarrow \emptyset$;
  foreach $(\hat{\boldsymbol{z}},\ \log p,\ \text{aux}) \in \mathbb{L}_{t-1}$ **do**
    $\mathbb{S}_t \leftarrow \mathsf{TopK}_k(\boldsymbol{y}_t(\hat{\boldsymbol{z}}))$; $\boldsymbol{r}_t \leftarrow \mathsf{LogSoftmax}(\boldsymbol{y}_t(\hat{\boldsymbol{z}}))$; $Z_t \leftarrow \mathsf{LogSumExp}(\boldsymbol{r}_t[\mathbb{S}_t])$;
    foreach $\hat{z} \in \mathbb{S}_t$ **do**
      $(\text{aux}', \varepsilon_\star) \leftarrow \mathsf{Viable.Update}(\text{aux}, \hat{z}, t, \boldsymbol{z}_{(\text{suf})}, \varepsilon)$;
      if $\varepsilon_\star \leq \varepsilon$ **then**
        $\hat{\boldsymbol{z}}' \leftarrow \hat{\boldsymbol{z}} \,\|\, \hat{z}$; $\log p' \leftarrow \log p + \boldsymbol{r}_t[\hat{z}] - Z_t$; $\mathbb{C}_t^{\leq\varepsilon} \leftarrow \mathbb{C}_t^{\leq\varepsilon} \cup \{(\hat{\boldsymbol{z}}', \log p', \text{aux}')\}$;
  if $t = T$ **then** `// Final step: acceptance test and bounds`
    $\mathbb{F} \leftarrow \{(\hat{\boldsymbol{z}}', \log p', \text{aux}') \in \mathbb{C}_T^{\leq\varepsilon} : \mathsf{Viable.IsFinal}(\text{aux}', \varepsilon) = 1\}$;
    $\mathrm{LB}_{\varepsilon,\mathsf{dist}} \leftarrow \sum_{(_, \log p', _) \in \mathbb{F}} \exp(\log p')$; $\mathrm{UB}_{\varepsilon,\mathsf{dist}} \leftarrow \mathrm{LB}_{\varepsilon,\mathsf{dist}} + \mathit{bank}$;
    return $\mathbb{F}$, $\mathrm{LB}_{\varepsilon,\mathsf{dist}}$, $\mathrm{UB}_{\varepsilon,\mathsf{dist}}$;
  Remove from $\mathbb{C}_t^{\leq\varepsilon}$ any $(\hat{\boldsymbol{z}}', \log p', \text{aux}')$ whose last token is EOS;
  if $\mathbb{C}_t^{\leq\varepsilon} = \emptyset$ **then return** $\emptyset$, 0, *bank*; `// No ε-viable candidates remain`
  $\mathbb{U}_t \leftarrow$ top-B of $\mathbb{C}_t^{\leq\varepsilon}$ by $\log p'$;
  $\mathit{bank} \leftarrow \mathit{bank} + \sum_{(_, \log p', _) \in \mathbb{C}_t^{\leq\varepsilon} \setminus \mathbb{U}_t} \exp(\log p')$; `// Bank pruned ε-viable mass`
  $\mathbb{L}_t \leftarrow \mathbb{U}_t$;
  if $\tau_{\min}$ *and* $\max_{(_, \log p, _) \in \mathbb{L}_t} \exp(\log p) < \tau_{\min}/(B \cdot k)$ **then return** $\emptyset$, 0, *bank*;
  Compute $\boldsymbol{y}_{t+1}(\hat{\boldsymbol{z}})$ for each $(\hat{\boldsymbol{z}}, \cdot, \cdot) \in \mathbb{L}_t$;

Candidate sets and beam operations. We extend the notation in Appendix D.1 to also include ε-viability pruning. At depth $t \in \{0, \ldots, T\}$, denote

- $\mathbb{L}_t$: the beam at depth t (at most B elements), obtained by expanding $\mathbb{L}_{t-1}$, applying ε-viability and across-beam pruning.
- $\mathbb{C}_t$: all children formed from $\mathbb{L}_{t-1}$ by one-step top-k expansion (at most $|\mathbb{L}_{t-1}| \cdot k$ candidates, including non-ε-viable ones).
- $\mathbb{C}_t^{\leq\varepsilon} \subseteq \mathbb{C}_t$: the subset of ε-viable children ($\varepsilon_\star \leq \varepsilon$). Algorithm 5 constructs $\mathbb{C}_t^{\leq\varepsilon}$ directly by filtering during expansion.
- $\mathbb{U}_t$: the (at most) top-B of $\mathbb{C}_t^{\leq\varepsilon}$ by cumulative probability (across-beam prune); $\mathbb{L}_t \leftarrow \mathbb{U}_t$.
- $\mathbb{F}$, the set of returned ε-viable finals; it is possible that we may return early (at $t < T$) with $\emptyset$, if no such finals exist.

As in the invariants for k-CBS, we denote $\mathrm{Desc}_T(\boldsymbol{u})$ the set of depth-T descendants (for any node $\boldsymbol{u}$ at depth t) reachable by continuing the same per-state top-k policy (ignoring across-beam pruning) from $\boldsymbol{u}$. That is, $\mathrm{Desc}_T(\boldsymbol{u}) = \{\boldsymbol{w} \in \mathbb{V}^T : \boldsymbol{u} \text{ is a prefix of } \boldsymbol{w}\}$ is the set of depth-T descendants of $\boldsymbol{u}$ (i.e., the full-length continuations with prefix $\boldsymbol{u}$, under the top-k policy).

Generic per-depth t viability predicate. Fix a target suffix $\boldsymbol{z}_{(\text{suf})}$ of length T and a maximum distance (i.e., tolerance) ε from $\boldsymbol{z}_{(\text{suf})}$. A **viability predicate** is a family $\{\mathsf{Viable}_t(\cdot)\}_{t=0}^T$ with the following property. For any generated continuation $\hat{\boldsymbol{z}}_{(\text{cont})}$ of prefix $\boldsymbol{z}_{(\text{pre})}$:

- **(Soundness.)** Viable_t is a necessary condition for ε-viability: if $\mathsf{dist}(\hat{\boldsymbol{z}}_{(\text{cont})}, \boldsymbol{z}_{(\text{suf})}) \leq \varepsilon$, then $\mathsf{Viable}_t(\hat{\boldsymbol{z}}_{1:t}^{(\text{cont})})$ holds for every $t \in \{0, \ldots, T\}$. Equivalently (by contrapositive), if

$\neg\,\mathsf{Viable}_t(\hat{z}^{(\mathrm{cont})}_{1:t})$, then no length-$T$ completion of $\hat{z}^{(\mathrm{cont})}_{1:t}$ satisfies $\mathsf{dist}(\cdot,\, z_{(\mathrm{suf})}) \le \varepsilon$, so the path may be safely pruned.

Viability at step t depends only on the partial $\hat{z}^{(\mathrm{cont})}_{1:t}$ and the target suffix $z_{(\mathrm{suf})}$, not on the prefix $z_{(\mathrm{pre})}$, as the history $\hat{z}$ and the training sequence z share the same a-token prefix.

For Hamming, Viable_t is "running mismatch counter $\le \varepsilon$" (soundness via monotonicity; see Lemma 15). For Levenshtein, Viable_t is "banded row minimum $\le \varepsilon$ at row t" (soundness via Ukkonen's criterion; see Lemma 17).

Common facts used throughout. All candidates at a fixed depth are ordered by a strict total order $\succ$ on cumulative log-probabilities $\log p'$, with a fixed deterministic tie-breaker. For a finite set $\mathbb{X}$ and $\boldsymbol{x} \in \mathbb{X}$, let $\mathrm{rank}_{\mathbb{X}}(\boldsymbol{x})$ be the position of $\boldsymbol{x}$ under $\succ$ in $\mathbb{X}$ (1 is highest).

- **F1 (rank under restriction).** If $\mathbb{Y} \subseteq \mathbb{X}$ and $\boldsymbol{x} \in \mathbb{Y}$, then

  $\mathrm{rank}_{\mathbb{Y}}(\boldsymbol{x}) \le \mathrm{rank}_{\mathbb{X}}(\boldsymbol{x})$ (restricting to a subset can only remove items that outrank $\boldsymbol{x}$).

- **F2 (no cross-parent duplicates at a depth).** Children formed from distinct parents at the same depth are token-distinct (no duplicates across parents); see Lemma 5. Consequently,

$$|\mathbb{C}_t| \le k\,|\mathbb{L}_{t-1}| \le B \cdot k, \quad |\mathbb{C}_t^{\le\varepsilon}| \le |\mathbb{C}_t|, \quad \text{and} \quad |\mathbb{U}_t| \le |\mathbb{C}_t^{\le\varepsilon}|.$$

- **F3 (descendant-sum identity).** For any node $\boldsymbol{u}$ at depth $t < T$,

$$\sum_{\boldsymbol{x}\in \mathrm{Desc}_T(\boldsymbol{u})} \Pr_{\theta,\phi}(\boldsymbol{x} \mid z_{(\mathrm{pre})}) \;=\; \Pr_{\theta,\phi}(\boldsymbol{u} \mid z_{(\mathrm{pre})}).$$

  Under per-state top-k renormalization, a parent's mass equals the sum of its immediate children's masses; iterating to depth T yields the identity. (See Lemma 9, Equation 36.)

- **F4 (equal per-path probability across variants).** The probability assigned to a path depends only on the model's conditional distributions along that path's history, not on the other beam elements. Therefore, any depth-T path $\hat{z}_{(\mathrm{cont})}$ enumerated by both baseline and viability-pruned k-CBS receives the same probability in both:

$$\overset{\text{base}}{\Pr_{\theta,\phi}}(\hat{z}_{(\mathrm{cont})} \mid z_{(\mathrm{pre})}) \;=\; \overset{\text{prune}}{\Pr_{\theta,\phi}}(\hat{z}_{(\mathrm{cont})} \mid z_{(\mathrm{pre})}).$$

  (See Lemma 7 and Corollary 8.)

E.3.2 Distance-specific results for streaming generation ε-viability

Our proof framework will depend on a distance dist satisfying the generic per-depth t viability predicate, $\{\mathsf{Viable}_t(\cdot)\}_{t=0}^{T}$, described in the prior subsection. For Hamming, Viable_t is "running mismatch counter $\le \varepsilon$," for which we show soundness via monotonicity in Lemma 15. For Levenshtein, Viable_t is "banded row minimum $\le \varepsilon$ at row t," for which we show soundness in Lemma 17.

We start with Hamming: extending a continuation cannot decrease the Hamming distance to the target suffix; once a partial path becomes non-ε-viable, it remains non-ε-viable.

Lemma 15 (Monotone Hamming ε-viability). *Let $z_{(\mathrm{suf})} = (z_1^{(\mathrm{suf})}, \ldots, z_T^{(\mathrm{suf})})$ and let $\hat{z}^{(\mathrm{cont})}_{1:t} = (\hat{z}_1^{(\mathrm{cont})}, \ldots, \hat{z}_t^{(\mathrm{cont})})$. Define the running Hamming counter*

$$n_t(\hat{z}^{(\mathrm{cont})}_{1:t}, z_{(\mathrm{suf})}) \;=\; \sum_{i=1}^{t} \mathbf{1}\big[\hat{z}_i^{(\mathrm{cont})} \neq z_i^{(\mathrm{suf})}\big],$$

which is the Hamming distance at t between a t-length continuation and the first t tokens of the T-length suffix (Equation 45). Then $t \mapsto n_t(\hat{z}^{(\mathrm{cont})}_{1:t}, z_{(\mathrm{suf})})$ is nondecreasing. In particular, if $n_t(\hat{z}^{(\mathrm{cont})}_{1:t}, z_{(\mathrm{suf})}) > \varepsilon$ for some $t < T$, every length-T extension has Hamming distance $> \varepsilon$ and cannot lie in $\mathbb{B}_\varepsilon^{\mathsf{Ham}}(z_{(\mathrm{suf})})$.

Proof. $n_{t+1} = n_t + \mathbf{1}[\hat{z}^{(\text{cont})}_{t+1} \neq z^{(\text{suf})}_{t+1}] \geq n_t$. So, if $n_t > \varepsilon$, later edits cannot decrease the Hamming distance, so all descendants remain non-ε-viable. It is therefore safe to prune children once they become non-ε-viable; ε-viable finals must also be ε-viable at all prefixes t. □

The Hamming results rely on monotonicity of the mismatch counter (Lemma 15), which does not hold for the Levenshtein distance. Instead, we establish soundness via Ukkonen's criterion on the banded Wagner & Fischer dynamic program. First, we show that the band has width at most $2\varepsilon + 1$.

Lemma 16 (Band restriction). *Fix a row i. If $|j - i| > \varepsilon$, then $D[i, j] > \varepsilon$. Therefore, any column with $D[i, j] \leq \varepsilon$ must lie in $[j_{\min}(i),\ j_{\max}(i)]$, whose width is at most $2\varepsilon + 1$.*

Proof. By Equation 51, $D[i, j] \geq |i - j|$. If $|i - j| > \varepsilon$ then $D[i, j] > \varepsilon$, so any entry with value $\leq \varepsilon$ must satisfy $|i - j| \leq \varepsilon$. The condition $|i - j| \leq \varepsilon$ is equivalent to $i - \varepsilon \leq j \leq i + \varepsilon$. Since indices must also lie in $[0, T]$, the admissible columns at row i lie in $j \in [\, j_{\min}(i),\ j_{\max}(i)\,] \equiv \big[\, \max\{0,\ i - \varepsilon\},\ \min\{T,\ i + \varepsilon\}\,\big]$, as in Equation 52. The interval $[i - \varepsilon,\ i + \varepsilon]$ contains

$$(i + \varepsilon) - (i - \varepsilon) + 1 \;=\; 2\varepsilon + 1$$

integer columns. After intersecting with $[0, T]$, the width can only decrease. Therefore, the band has width at most $2\varepsilon + 1$. □

Lemma 17 (Ukkonen band ε-viability soundness). *Fix $\varepsilon \geq 0$ and a target suffix $z_{(\text{suf})} \in \mathbb{V}^T$. For each row $t \in \{0, \ldots, T\}$ define the Ukkonen band*

$$[j_{\min}(t),\ j_{\max}(t)] \;:=\; \big[\max\{0,\ t - \varepsilon\},\ \min\{T,\ t + \varepsilon\}\big] \quad \textit{(Equation 52)}.$$

Let $D[t, j] = \mathsf{Levenshtein}(\hat{z}^{(\text{cont})}_{1:t},\ z^{(\text{suf})}_{1:j})$ be the Wagner-Fischer dynamic program value (Equation 48). The predicate

$$\mathsf{Viable}_t(\hat{z}^{(\text{cont})}_{1:t}) \;:=\; \Big[\min_{j \in [j_{\min}(t),\, j_{\max}(t)]} D[t, j] \;\leq\; \varepsilon\Big]$$

satisfies soundness: if $\mathsf{Levenshtein}(\hat{z}_{(\text{cont})},\ z_{(\text{suf})}) \leq \varepsilon$, then $\mathsf{Viable}_t(\hat{z}^{(\text{cont})}_{1:t})$ holds for every $t \in \{0, \ldots, T\}$. Equivalently, if $\mathsf{Viable}_t(\hat{z}^{(\text{cont})}_{1:t})$ fails at any t, no length-T completion of $\hat{z}^{(\text{cont})}_{1:t}$ satisfies $\mathsf{Levenshtein}(\cdot,\ z_{(\text{suf})}) \leq \varepsilon$, so the path may be safely pruned.

Proof. We prove the pruning direction (the converse follows by contrapositive). Suppose $\min_{j \in [j_{\min}(t),\, j_{\max}(t)]} D[t, j] > \varepsilon$ for some $t < T$. By Lemma 16, any column with $D[t, j] \leq \varepsilon$ must lie inside the band, so in fact *all* entries in row t exceed ε. Since edit costs are non-negative, any completion of $\hat{z}^{(\text{cont})}_{1:t}$ to length T can only add cost, so $D[T, T] > \varepsilon$ and no ε-viable completion exists.

For the boundary case of $t = T$, note that $j = T \in [j_{\min}(T),\ j_{\max}(T)]$ and $D[T, T] = \mathsf{Levenshtein}(\hat{z}_{(\text{cont})}, z_{(\text{suf})}) \leq \varepsilon$, so the band minimum is $\leq \varepsilon$. □

E.3.3 General invariants that follow

Algorithm 5 returns only paths passing the IsFinal check, so every returned final is ε-viable.

Corollary 18 (All returned finals are ε-viable). *Every returned final $(\hat{z},\ \log p,\ \text{aux}) \in \mathbb{F}$ from Algorithm 5 satisfies $\mathsf{dist}(\hat{z}_{(\text{cont})},\ z_{(\text{suf})}) \leq \varepsilon$, i.e., $\hat{z}_{(\text{cont})} \in \mathbb{B}^{\mathsf{dist}}_{\varepsilon}(z_{(\text{suf})})$.*

Proof. Algorithm 5 returns only paths where $\mathsf{Viable.IsFinal}(\text{aux},\ \varepsilon) = 1$. For Hamming, viability at depth T implies $n_T(\hat{z}^{(\text{cont})}_{1:T}, z_{(\text{suf})}) \leq \varepsilon$, and so $\mathsf{Hamming}(\hat{z}_{(\text{cont})},\ z_{(\text{suf})}) \leq \varepsilon$ (Lemma 15). For Levenshtein, IsFinal checks $D[T, T] \leq \varepsilon$, which equals $\mathsf{Levenshtein}(\hat{z}_{(\text{cont})},\ z_{(\text{suf})})$ by definition of the dynamic programming table (Equation 48). □

We next show a simple result about the cardinality of the returned finals.

Lemma 19 (Cardinality of returned finals under top-k and a sound pruning rule)**.** *Assume the standard beam mechanics that we use throughout: (i) after across-beam pruning, $|\mathbb{L}_t| \le B$ for every step t; (ii) each parent forms at most k children via per-state top-k token expansion; (iii) no cross-parent duplicates at a given depth t (F2, Lemma 5); and (iv) ε-viability filtering and final-acceptance checks only remove candidates ($\mathbb{C}_t^{\le\varepsilon} \subseteq \mathbb{C}_t$, $\mathbb{F} \subseteq \mathbb{C}_T^{\le\varepsilon}$). Then, at depth T, $|\mathbb{F}| \le B \cdot k$. If there exists $t < T$ with $\mathbb{U}_t = \emptyset$, the run terminates early with $\mathbb{F} = \emptyset$.*

Proof. At depth $T-1$, $|\mathbb{L}_{T-1}| \le B$ by (i). Each parent produces exactly k children via top-k expansion (ii), and by (iii) no two parents produce the same child, so

$$|\mathbb{C}_T| \;=\; k\,|\mathbb{L}_{T-1}| \;\le\; B \cdot k.$$

At the final step there is no across-beam prune; by (iv), ε-viability filtering and the IsFinal check can only remove candidates, so $|\mathbb{F}| \le |\mathbb{C}_T^{\le\varepsilon}| \le |\mathbb{C}_T| \le B \cdot k$. If $\mathbb{U}_t = \emptyset$ for some $t < T$, the run halts with $\mathbb{F} = \emptyset$ by construction. □

Algorithms 3 (Hamming-pruned) and 4 (Levenshtein-pruned) satisfy (i)–(iv) in Lemma 19, so it applies exactly in both cases.

We now give a unified early-termination upper bound covering both Hamming- and Levenshtein-pruned variants.

Lemma 20 (Early-termination UB under sound viability pruning)**.** *Fix ε and a viability predicate satisfying soundness. Run Algorithm 5 and let $\mathbb{R}_{\text{prune}}$ denote the set of nodes removed by the across-beam prune during the run. Since the across-beam prune acts on $\mathbb{C}_t^{\le\varepsilon}$, every node in $\mathbb{R}_{\text{prune}}$ is ε-viable at the time of pruning. Suppose the run terminates early at some depth $t < T$ with $\mathbb{C}_t^{\le\varepsilon} = \emptyset$, so that $\mathbb{F} = \emptyset$ and $\mathrm{LB}_{\varepsilon,\mathsf{dist}} = 0$. Then*

$$p_{z,\varepsilon}^{\mathsf{dist}} \;\le\; \mathit{bank} \;=\; \sum_{(\boldsymbol{u},\log p)\in\mathbb{R}_{\text{prune}}} \exp(\log p).$$

Proof. Fix any length-T continuation $\hat{\boldsymbol{z}}_{(\text{cont})}$ with $\mathsf{dist}(\hat{\boldsymbol{z}}_{(\text{cont})}, \boldsymbol{z}_{(\text{suf})}) \le \varepsilon$. By soundness, $\hat{\boldsymbol{z}}_{1:i}^{(\text{cont})}$ is Viable_i for every $i \in \{0,\dots,T\}$. If $\hat{\boldsymbol{z}}_{1:t-1}^{(\text{cont})}$ were in $\mathbb{L}_{t-1}$, the expansion at step t would generate $\hat{\boldsymbol{z}}_{1:t}^{(\text{cont})}$, which is ε-viable by soundness and therefore in $\mathbb{C}_t^{\le\varepsilon}$—contradicting $\mathbb{C}_t^{\le\varepsilon} = \emptyset$. So $\hat{\boldsymbol{z}}_{1:t-1}^{(\text{cont})} \notin \mathbb{L}_{t-1}$, and some earlier prefix of $\hat{\boldsymbol{z}}_{(\text{cont})}$ must have been removed by the across-beam prune. Let

$$t^\star(\hat{\boldsymbol{z}}_{(\text{cont})}) \;:=\; \min\big\{\, i \in \{1,\dots,T-1\} :\; \hat{\boldsymbol{z}}_{1:i}^{(\text{cont})} \text{ was removed by the across-beam prune} \,\big\},$$
$$\boldsymbol{u}(\hat{\boldsymbol{z}}_{(\text{cont})}) \;:=\; \hat{\boldsymbol{z}}_{1:t^\star(\hat{\boldsymbol{z}}_{(\text{cont})})}^{(\text{cont})}.$$

That is, $t^\star(\hat{\boldsymbol{z}}_{(\text{cont})})$ is the earliest step at which some prefix of $\hat{\boldsymbol{z}}_{(\text{cont})}$ was discarded by the across-beam prune, and $\boldsymbol{u}(\hat{\boldsymbol{z}}_{(\text{cont})})$ is that pruned prefix. By soundness, $\boldsymbol{u}(\hat{\boldsymbol{z}}_{(\text{cont})})$ was ε-viable when pruned, so $\big(\boldsymbol{u}(\hat{\boldsymbol{z}}_{(\text{cont})}),\, \log p\big) \in \mathbb{R}_{\text{prune}}$. More generally, once a node is pruned, no descendants are ever constructed, so no element of $\mathbb{R}_{\text{prune}}$ is an ancestor of another and the descendant sets $\{\mathrm{Desc}_T(\boldsymbol{u}) : (\boldsymbol{u},\cdot) \in \mathbb{R}_{\text{prune}}\}$ are pairwise disjoint. Every ε-viable $\hat{\boldsymbol{z}}_{(\text{cont})}$ belongs to exactly one such set, namely $\mathrm{Desc}_T\big(\boldsymbol{u}(\hat{\boldsymbol{z}}_{(\text{cont})})\big)$. Therefore,

$$p_{z,\varepsilon}^{\mathsf{dist}} \;\le\; \sum_{(\boldsymbol{u},_)\in\mathbb{R}_{\text{prune}}} \;\sum_{\boldsymbol{x}\in\mathrm{Desc}_T(\boldsymbol{u})} \Pr_{\theta,\phi}(\boldsymbol{x} \mid \boldsymbol{z}_{(\text{pre})}) \;=\; \sum_{(\boldsymbol{u},\log p)\in\mathbb{R}_{\text{prune}}} \exp(\log p) \;=\; \mathit{bank},$$

where the inequality holds because each ε-viable continuation is a descendant of exactly one pruned node (and non-ε-viable descendants only add mass), and the first equality is the descendant-sum identity (F3, Equation 36). □

Both the Hamming (Algorithm 3) and Levenshtein (Algorithm 4) variants satisfy the assumptions of Lemma 20: soundness is established by Lemma 15 and Lemma 17, respectively, so $p_{z,\varepsilon}^{\mathsf{dist}} \le \mathit{bank}$ upon early termination in both cases.

Last, recall that at the top of this appendix (Appendix E) we noted that ε-pruned k-CBS does not provably produce a tighter (dominating) lower bound than baseline k-CBS (with post-processing for distance $\leq \varepsilon$). We now give a concrete example showing how viability pruning can cause the pruned algorithm to miss an ε-viable candidate that the baseline would have retained. The same mechanism implies that the upper bound $\mathrm{UB}_{\varepsilon,\mathsf{dist}}$ is also not guaranteed to be tighter than the analogous (trivial) baseline upper bound.

Remark 21 (No dominance guarantee). There exist inputs for which $\mathrm{LB}_{\varepsilon,\mathsf{dist}}$ from ε-pruned k-CBS (Algorithm 5) is strictly less than the lower bound obtained by running baseline k-CBS (Algorithm 2) and post-processing for distance $\leq \varepsilon$. The same holds for the upper bound: the pruned $\mathrm{UB}_{\varepsilon,\mathsf{dist}}$ is neither provably larger nor provably smaller than the baseline upper bound, because the two algorithms explore different regions of the search space and account for unexplored mass differently. This holds for both $\mathsf{dist} = \mathsf{Hamming}$ and $\mathsf{dist} = \mathsf{Levenshtein}$.

Counterexample. Take $B = k = 2$. Suppose that at some step t, the two algorithms have already diverged due to earlier viability pruning, and their beams differ.

- **Baseline beam** $\mathbb{L}_t = \{(\boldsymbol{x}, \log p(\boldsymbol{x})), (\boldsymbol{u}, \log p(\boldsymbol{u}))\}$, where $\boldsymbol{x}$ is ε-viable and $\boldsymbol{u}$ is not. (The baseline retains $\boldsymbol{u}$ because it does not prune by viability.)
- **Pruned beam** $\mathbb{L}_t = \{(\boldsymbol{x}, \log p(\boldsymbol{x})), (\boldsymbol{w}, \log p(\boldsymbol{w}))\}$, where both $\boldsymbol{x}$ and $\boldsymbol{w}$ are ε-viable. ($\boldsymbol{w}$ entered the pruned beam at an earlier step when $\boldsymbol{u}$—or another non-ε-viable candidate—was pruned, freeing a beam slot.) (We drop aux in the beam below, as we do not need to reference it to make this point; it would be present in practice.)

At step $t+1$, both algorithms expand their $B = 2$ beams to $B \cdot k = 4$ candidates. Write $p(\cdot) = \exp(\log p(\cdot))$ for the cumulative path probability under the top-k distribution. Because top-k renormalization ensures the k conditional probabilities sum to 1, the k children's cumulative probabilities p' sum to exactly the parent's p.

Suppose $p(\boldsymbol{x}) = 0.10$, $p(\boldsymbol{u}) = 0.50$, $p(\boldsymbol{w}) = 0.30$. (These are three distinct depth-t paths in the top-k tree, so their sum $0.90 \leq 1$.)

Baseline expansion. (from $\boldsymbol{x}$ and $\boldsymbol{u}$, with $\boldsymbol{u}$ non-viable):

$$\boldsymbol{x} \;\to\; \{\boldsymbol{x}^{(1)}\ (p'{=}0.06),\ \ \boldsymbol{x}^{(2)}\ (p'{=}0.04)\}, \qquad \boldsymbol{u} \;\to\; \{\boldsymbol{u}^{(1)}\ (p'{=}0.45),\ \ \boldsymbol{u}^{(2)}\ (p'{=}0.05)\}.$$

Top-$B = 2$ by probability: $\{\boldsymbol{u}^{(1)}, \boldsymbol{x}^{(1)}\}$. Because $\boldsymbol{u}$ is non-viable, no descendant of $\boldsymbol{u}$ can be ε-viable at depth T: for Hamming, this follows from monotonicity of n_t (Lemma 15); for Levenshtein, from soundness (Lemma 17). So $\boldsymbol{u}^{(1)}$ cannot contribute to the baseline lower bound despite occupying a beam slot. The baseline's surviving ε-viable candidate at step $t+1$ is $\boldsymbol{x}^{(1)}$.

Pruned expansion. (from $\boldsymbol{x}$ and $\boldsymbol{w}$, both viable):

$$\boldsymbol{x} \;\to\; \{\boldsymbol{x}^{(1)}\ (p'{=}0.06),\ \ \boldsymbol{x}^{(2)}\ (p'{=}0.04)\}, \qquad \boldsymbol{w} \;\to\; \{\boldsymbol{w}^{(1)}\ (p'{=}0.18),\ \ \boldsymbol{w}^{(2)}\ (p'{=}0.12)\}.$$

All four candidates are ε-viable. Top-$B = 2$: $\{\boldsymbol{w}^{(1)}, \boldsymbol{w}^{(2)}\}$. Candidate $\boldsymbol{x}^{(1)}$, with $p'(\boldsymbol{x}^{(1)}) = 0.06 < 0.12 = p'(\boldsymbol{w}^{(2)})$, is *pruned from the beam*.

Consequence. The baseline's beam at step $t+1$ contains the ε-viable candidate $\boldsymbol{x}^{(1)}$; the pruned beam does not. If $\boldsymbol{w}^{(1)}$ and $\boldsymbol{w}^{(2)}$ later become non-ε-viable (their descendants diverge from the reference), they contribute no mass to the pruned lower bound—but $\boldsymbol{x}^{(1)}$'s mass has already been lost from the pruned beam. Meanwhile, if $\boldsymbol{x}^{(1)}$ survives to the final step in the baseline, it contributes to the baseline lower bound. The pruned algorithm traded $\boldsymbol{x}^{(1)}$ for $\boldsymbol{w}^{(1)}$ and $\boldsymbol{w}^{(2)}$, and that trade can be a net loss.

Mechanism. Viability pruning reclaims beam slots occupied by non-ε-viable candidates ($\boldsymbol{u}$), admitting new ε-viable candidates ($\boldsymbol{w}$) whose children ($\boldsymbol{w}^{(1)}, \boldsymbol{w}^{(2)}$) can then crowd out ε-viable candidates ($\boldsymbol{x}^{(1)}$) that the baseline would have retained. In effect, the pruned algorithm explores a *different* region of the search space, not a strict superset of the baseline's.

Empirical observation. Even so, in our experiments we consistently observe $\mathrm{LB}_{\varepsilon,\mathsf{dist}} \geq \mathrm{LB}^{\mathsf{baseline}}_{\varepsilon,\mathsf{dist}}$. Non-$\varepsilon$-viable beam slots cannot contribute to the final lower bound, so replacing them with ε-viable candidates tends to increase the mass of ε-close completions discovered by the search. The counterexample above requires the newly-admitted viable candidates' children to be numerous enough and probable enough to displace a previously-reachable viable candidate—a situation that appears to be rare in practice.

F Details on experiments

We next provide additional results and details on all experiments. We first describe the metrics and visualizations that we use in our analysis (Appendix F.1), then provide information about the specific models, datasets, and code (Appendix F.2). We provide extended results for OLMo 2 on Wikipedia (Appendix F.3), Pythia on Enron (Appendix F.4), and Llama 2 (Appendix F.5) and Llama 3.1 8B (Appendix F.6) on public domain books.

Following Cooper et al. (2025), we set the minimum extraction threshold $\tau_{\min} = 0.001$, which we also validate with our negative controls. We defer to Cooper et al. (2026) for additional details on how calibrate more fine-grained thresholds. In most experiments, we use beam width $B = 20$, as we observe this setting to work well for top-k with $k = 40$. We make this decision based on the experiments shown in Appendix F.7; the gains from larger beam widths are marginal.[9] In Appendix F.8, we also do a full analysis comparing baseline k-CBS (Section 4.2) to Hamming- and Levenshtein-pruned (Section 4.3) k-CBS, in terms of how changing ε as a constraint impacts results. Finally, we also run brief analysis (following Ippolito et al. (2023)) on baseline k-CBS outputs for BLEU score (Appendix F.9).

We omit analysis comparing running k-CBS with and without the last-iteration across-beam prune. We find that omitting the last prune leads to small overall gains in identifying extractable sequences, and can lead to large gains in extraction risk for individual sequences. Since we get the up $B \cdot k$ outputs for free (i.e., with no additional model forward passes), we just report results for this configuration.

We find that the Levenshtein-pruned algorithm performs the best: it is faster than the baseline method, and in practice generally returns tighter lower bounds. We conclude from this analysis that running the Lev $\varepsilon = 5$ configuration is the reasonable choice; the results can be post-processed for smaller ε with minimal compromise in quality for those lower-ε runs.

The project website contains additional visualizations.

F.1 Metrics and visualizations for assessing extraction

We detail the different types of metrics and visualizations that we use to analyze near-verbatim extraction: extraction rates (Section F.1.1), visualizing near-verbatim extractable sequences that verbatim extraction misses and their increased risk (Section F.1.2), and per-sequence extraction mass composition (Section F.1.3). For books, we also provide heatmaps that visualize where extraction occurs in a book, following Cooper et al. (2025) (Section F.1.4)

F.1.1 Extraction rates

Extraction rates are the most common metric in the literature (Carlini et al., 2021; 2023; Nasr et al., 2023; 2025; Hayes et al., 2025b; Lee et al., 2022). For our purposes, plotting extraction

[9] Of course, one could start with a wider beam and let ε-pruning naturally narrow it. But to actually achieve improved efficiency through narrowing, we would need to implement batch compaction, which is not trivial. In general, we defer improved efficiency, implementation, and detailed exploration of algorithm parameter tuning to future work.

rates (like Figure 3) is useful for showing how the number of extractable sequences changes over different settings—greedy vs. probabilistic, verbatim vs. near-verbatim. As discussed in Section 2 and Appendix A.2.3, for some population of sequences $\mathbb{Z}$:

$$\text{extraction_rate}(\mathbb{Z}; s) \triangleq \frac{1}{|\mathbb{Z}|} \sum_{z \in \mathbb{Z}} \mathbf{1}[\, s(z) \,], \tag{16}$$

where $s(z)$ is a success predicate defined according to the chosen criterion. For greedy extraction, the success criterion is

$$s^{\text{dist}}_{\theta,\text{greedy},\varepsilon}(z) \triangleq \mathbf{1}\Big[\, \text{dist}\big(\hat{z}_{(\text{cont})}, z_{(\text{suf})}\big) \leq \varepsilon \,\Big], \tag{19}$$

where setting $\varepsilon = 0$ captures the verbatim case. For probabilistic extraction and a chosen (validated) minimum extraction probability τ_{min}, the success criterion is

$$s^{\text{dist}}_{\theta,\phi,\varepsilon}(z; \tau_{\text{min}}) \triangleq \mathbf{1}[\, p^{\text{dist}}_{z,\varepsilon} \geq \tau_{\text{min}} \,], \tag{22}$$

where similarly setting $\varepsilon = 0$ captures the verbatim case.

Note that extraction rates flatten information about variation in extraction risk surfaced by probabilistic extraction. Since greedy extraction is deterministic, this flattening has no effect; it just counts greedy extracted sequences. But for probabilistic extraction, any extractable sequence (irrespective of its risk) gets counted equally in the extraction rate.

F.1.2 Visualizing "unlocked" sequences and per-sequence risk increase

We use two types of visualizations for extraction risk. First, we use scatter plots (like Figure 4) to visualize how extractability and risk change per sequence. This gives a qualitative sense of how risk changes for the set of sequences. On a log–log scale, we plot a sequence's near-verbatim mass as a function of its verbatim mass, showing the line $y = x$ as a reference point. We highlight three different categories in different colors. **Blue** points are verbatim-extractable ($p_z \geq \tau_{\text{min}}$). Points above the dashed $y = x$ line therefore show increased extraction risk when near-verbatim mass is accounted for. **Red**/**orange** points are "unlocked" by near-verbatim extraction. They are all to the left of the τ_{min} dotted reference line. For **orange** points, $0 < p_z < \tau_{\text{min}}$, but $p^{\text{Lev}}_{z,5} \geq \tau_{\text{min}}$). That is, these points have some verbatim mass, but insufficient mass to be verbatim extractable; they become extractable when the near-verbatim mass is included. **Red** points have zero verbatim mass (which is why they are not on the log–log plot canvas), but are near-verbatim extractable.

Second, we provide complementary cumulative distribution functions (CCDFs) of per-sequence near-verbatim mass gain. For each sequence z in the training dataset, we compute the per-sequence mass gain

$$\Delta_z \triangleq \hat{p}^{\text{Lev}}_{z,5} - p_z, \tag{54}$$

i.e., the mass for the Levenshtein distance with $\varepsilon = 5$ minus the verbatim mass from k-CBS results. This is the absolute additional extraction risk revealed by relaxing from verbatim to near-verbatim for that specific sequence. We plot the CCDF of these gains. The x-axis is the per-sequence mass gain threshold Δ_z. The y-axis is the percentage of all sequences in the population (not just extractable ones) with gain $> \Delta_z$

Only sequences with strictly positive gain appear in the curve, but the denominator is the full set, so the y-intercept gives the fraction of all sequences that have any near-verbatim mass gain. The curve then steps down as Δ_z increases, showing the tail of the gain distribution. A point on the CCDF (e.g., and annotation like 1.2% at $\Delta_z = 0.001$) means 1.2% of all sequences in the population have per-sequence absolute mass gain ≥ 0.001 from the near-verbatim relaxation; at $\Delta_z = 0.1$, it shows the fraction of sequences that gain (absolute) mass gain ≥ 0.1.

A curve shifting right/upward for larger models means that larger models have both more sequences with gains and larger gains per sequence—a heavier-tailed distribution of near-verbatim risk increase.

We also provide tables containing values from the CCDF over per-sequence mass gain (Equation 54), computed with respect to each model/dataset's *fixed* extractable set. This shows the gain just for extracted sequences, and is not easily comparable across experiments since the extractable set is different (i.e., each is normalized by a different denominator).

F.1.3 Analyzing per-sequence risk composition

We provide three different ways to visualize how per-sequence extraction mass is allocated at different Levenshtein distances. First, for each extractable sequence ($p^{\mathsf{Lev}}_{z,5} \geq \tau_{\min} = 10^{-3}$), we decompose its total near-verbatim mass into incremental ε**-shells**:

$$\Delta_z(\varepsilon) \triangleq \hat{p}^{\mathsf{Lev}}_{z,\varepsilon} - \hat{p}^{\mathsf{Lev}}_{z,\varepsilon-1}, \tag{55}$$

where $\Delta_z(0) = p_z$ (verbatim mass). While Equation 54 computes the mass gain with respect to the near-verbatim mass using the most permissive distance threshold and the verbatim mass, the above equation diffs the mass between adjacent thresholds to identify how much mass is exactly attributable to a specific Lev distance—a shell in the ε-ball.

For $\varepsilon \in \{0, \ldots 5\}$, we can compute the ε**-shell share**:

$$\mathsf{shell_share}(z, \varepsilon) \triangleq \frac{\Delta_z(\varepsilon)}{\hat{p}^{\mathsf{Lev}}_{z,5}}. \tag{56}$$

For each sequence, the respective ε-shell shares sum to 1. We also focus particular attention on the **verbatim share** shell fraction (i.e., for $\Delta_z(0)$):

$$\mathsf{verbatim_share}(z) = \frac{p_z}{\hat{p}^{\mathsf{Lev}}_{z,5}}. \tag{57}$$

From computing the ε-shell share for every $\varepsilon \in \{0, \ldots, 5\}$ using Equation 56, we can produce a per-sequence heatmap of the mass contributions at each Lev distance to a sequence's overall extraction mass (e.g., Figure 14a). Each row represents an extracted sequence (with rows sorted by the verbatim mass share, see Equation 57). The columns are the ε-shells. Each cell's color is the intensity of that shell's fraction for that sequence. For instance, sequences at the top have all of their mass allocated to the verbatim continuation; the verbatim share is 100%, so that shell is indicated to have 100% of the share and the other 5 shells indicate 0%. The heatmap as a whole visualizes heterogeneity across sequences, with respect to which Lev distances contribute different relative amounts of mass.

We also plot distributions over extracted-sequence ε-mass share. In the main paper, we provide comparative plots across model sizes for the verbatim share—i.e., Equation 57, for the same dataset across a model family (e.g., Figure 7). We provide this type of plot here as well, and complement it with per-model-and-dataset distribution breakdowns for each ε-shell's mass (e.g., Figure 14c). These show the distribution of per-sequence ε-shell fractions, with one violin plot per ε-shell. They reveal the spread and central tendency (if there is one) of how mass is distributed across Lev distances.

F.1.4 Heatmaps to visualize book extraction coverage

We use the same heatmap visualization as in Cooper et al. (2025) for book memorization. For the books experiments, we produce sequences by sliding through the length of the book, taking 100-token segments every 20 characters (Appendix F.2.3). The sequences therefore overlap (which is intentional, see Cooper et al. (2025)). Multiple suffixes cover each character position in the book; for each position, we show the maximum extraction probability across covering suffixes, so that the heatmap reflects the highest extraction risk at each location. Larger probabilities have darker color/higher intensity (on a log scale), and everything below $\tau_{\min}$ is shown in white (not extractable). In Cooper et al. (2025), the heatmaps are built from verbatim extraction probabilities ($\varepsilon = 0$). Here, we show near-verbatim extraction probabilities $\hat{p}^{\mathsf{Lev}}_{z,\varepsilon}$ for the chosen distance threshold ε. We omit heatmaps for the Hamming distance.

F.2 Experimental settings

Being a **member** of the training data is a necessary (though not sufficient) condition for extraction: by definition, extraction only applies to memorized training data. Therefore, the simplest setting for studying memorization and extraction is to run experiments on known training data. For open models, in some cases we have ground-truth knowledge about the training data, and so we leverage these cases to simplify our experimental setup. For each extraction experiment, we test models with data that was known with certainty to be in the training dataset, and run experiments for three different settings: Wikipedia entries, public domain books, and emails.

Even when we know if a particular sequence was included in the training data, it is often quite challenging to know with certainty that a particular sequence was *not* in the training data (Lee et al., 2023a; Cooper et al., 2024). This is because models are typically trained on enormous web scrapes, which contain duplicates and near-duplicates of content available from multiple sources (Lee et al., 2022). Nevertheless, to assess the validity of an extraction procedure, it is important to make a best-effort to run **negative controls**: experiments on non-training data (**held-out data**), which should register no extraction signal with our measurement procedure. If our extraction procedure registers such signal on non-training data (where extraction is impossible), then the procedure produces false positives. In memorization research, it is more common to tolerate false negatives than false positives (Hayes et al., 2025a; Cooper et al., 2025); there is a preference toward conservative claims originating from the **membership inference** literature in ML security (Carlini et al., 2022; Shokri et al., 2017). We therefore make a best effort to pair our extraction experiments with appropriate negative controls (Cooper et al., 2026).

We detail each model and dataset below, including which datasets we run for which model based on known information about training-set membership. Altogether, we test open-weight, non-instruction-tuned LLMs from three different families. We obtain all model weights from HuggingFace. All code can be found on GitHub.

F.2.1 OLMo 2 *on Wikipedia*

Models. We run experiments on all three OLMo 2 model sizes: 7B, 13B, and 32B.

Training data subset. The training dataset for OLMo 2 is publicly available, and the overall training recipe is documented in the release report (OLMo et al., 2025). Wikipedia was included in the training data, and we draw a subset of these entries for training data for our extraction experiments. Specifically, we draw 10,000 unique Wikipedia entries, which has a cutoff date of December 2023, where each entry is at least 100 tokens long. The data can be found on HuggingFace. We specifically streamed records from `data/wiki/wiki-0000.json.gz` and `data/wiki/wiki-0001.json.gz` from the `allenai/olmo-mix-1124` (used for training OLMo 2). We take the first 100 tokens of each entry as a sequence. For convenience, we provide the subset of the data that we use as the input file for our experiments here.

Negative control (held-out subset). We curated a set of 10,000 Wikipedia entries that post-date OLMo 2's training data. To obtain these pages, we paginated through Wikipedia's `logevents` API (`letype=create`, main namespace) from February 24, 2026 back to January 1, 2024. For each batch of 500 creation events, we checked the page lengths via `prop=info` and kept pages with $\geq$8,000 bytes of wikitext. We then **reservoir sampled**[10] 100,000 pages that fit these conditions from the whole range of Wikipedia pages in this time frame, de-duplicating by page ID. (This means we only ever hold 100,000 pages in memory at one time.) We maintained this full list of pages. However, since page-fetching can be slow, we trimmed the 100,000 pages to 30,000 (randomly sampled), and filtered redirects and disambiguation

[10] Reservoir sampling obtains a uniform sample of N items from a stream of unknown size, without loading everything into memory. We place the first N items into the reservoir; then, for every subsequent item we encounter, we assign a shrinking random chance of swapping it into the reservoir, replacing a random existing entry. By the time we have iterated through every item, each item in the entire stream has had an equal probability of ending up in the final reservoir.

pages using `prop=pageprops|info`. Then, we verified each page's creation date, fetching the oldest revision (`rvdir=newer&rvlimit=1`) and confirming the timestamp was on or after `2024-01-01`, removing any pages that failed this check. We then fetched the plain text for each page one at a time (`prop=extracts&explaintext=true`) and filtered by length, dropping any page with fewer than 1,500 characters of plain text. We then trimmed to 10,000 total pages, shuffled them, and saved them as JSON (both metadata and full article text). We provide the dataset here, and note that we ran experiments using only the first 5,000 pages.

This is also an imperfect negative control, as old deleted Wikipedia pages (that were in the training data) may be re-created and posted with new create dates; these new pages (containing old content) would then not actually post-date the training cutoff, and our process would pull them in. There are also instances of highly templated text in Wikipedia pages; new pages can be near-duplicates of older pages. We document observed instances of this in our negative controls in Appendix F.3.1.

F.2.2 PYTHIA *on emails*

Models. We run experiments on the four largest PYTHIA model sizes: 1B, 2.8B, 6.9B, and 12B (Biderman et al., 2023).

Training data subset. The Pythia suite was trained on the Pile (Gao et al., 2020), which contains multiple copies of the Enron email dataset. Similar to Hayes et al. (2025b), we therefore use Enron (obtained from the Pile) as our sample of PYTHIA training data in extraction experiments. We use 10,000 unique emails, where each email is at least 100 tokens long. We take the first 100 tokens of each email as a sequence. We provide the specific version of the Enron dataset we used here.

Negative control (held-out subset). As a corresponding negative control, we run experiments on a subset of TREC 2007 Spam. We adapt this experiment from Hayes et al. (2025b). However, we note from manual inspection of the "ham" subset that many of these emails contain verbatim news articles (i.e., are news digests), which are very likely to be in Common Crawl and other web scrapes, and thus are likely candidates to be training data (Lee et al., 2023a). We therefore filter these emails out, and use a resulting set of 2,000 emails that are at least 100 tokens long for our held-out subset, which we provide here. We take the first 100 tokens of each email as a sequence.

F.2.3 LLAMA *on public domain books*

Models. We run experiments on LLAMA 1 13B (Touvron et al., 2023a) (Table 3), the LLAMA 2 family (7B, 13B, 70B) (Touvron et al., 2023b), and LLAMA 3.1 8B (Grattafiori et al., 2024). We pick these models based on results from Cooper et al. (2025): we use the same illustrative example from that work involving LLAMA 1 13B; we use LLAMA 2 as it has three sizes available to examine different scales; and we use LLAMA 3.1 8B to also examine one model from the more recent series. For concision, we predominantly omit for LLAMA 3.1 8B.

Training data subset. It is public knowledge that Meta trained all LLAMA models on books, and that the first three generations of these models (through the last minor versions of LLAMA 3) were trained on the Books3 corpus (Touvron et al., 2023a; Kadrey v. Meta; Cooper et al., 2025). We pull four public domain books from Books3: *The Great Gatsby* (Fitzgerald, 1925), *Winnie the Pooh* (Milne, 1926), *Orlando* (Woolf, 1928), and *Pride and Prejudice* (Austen, 1813). We trim any edition-specific front matter (e.g., editor's forward) and back matter (e.g., index) so that the file only contains text from the book. We then chunk up the book into 100-token sequences, starting from the first character and using a stride length of 20 characters. We upload the text files for these trimmed four public domain books here. To create sequences from books, we follow the procedure in Cooper et al. (2025), where we take 100-token sequences every s characters; we set $s = 20$. For the LLAMA 2 tokenizer, this yields 13,390 sequences for *The Great Gatsby*, 21,822 sequences for *Orlando*, 6,152 sequences for *Winnie the Pooh*, and 10,457 for *The People's Dictator* (first three chapters). For the LLAMA

3.1 tokenizer, this yields 34,271 sequences for *Pride and Prejudice* and 10,454 for *The People's Dictator* (first three chapters).

Negative control (held-out subset). We use *The People's Dictator* as a negative control. This is an openly licensed (CC-BY-SA-NC) book from 2025—well after LLAMA 3's training date cutoff. We provide a link to the full book PDF here. We removed the licensing information from the text file and trimmed to the first three chapters prior to running experiments.

As Cooper et al. (2025) document, while this is a useful way to select a negative control, it is imperfect, as books may verbatim quote texts from prior to the training cutoff data; the *document* of a post-cutoff book may not be included in the training data, but *specific text* within that document may be included in the training data from other sources. A more ideal negative control would use a public domain novel that we know with certainty was not included in LLAMA's training data; however, we do not have sufficient information to identify such a book, and so instead use long-form narrative text from the book noted above.

F.2.4 Summary of reported experiments

Table 4 summarizes all experimental runs and configurations reported in this paper.

Table 4: **Reported experimental configurations.** Configurations across datasets, models, and constraints (where appropriate, beam width B, distance metric dist $\in \{\mathsf{Ham}, \mathsf{Lev}\}$, and tolerance ε). We report the number of GPUs and batch size for each experiment. NC = negative control; stride = number of characters used for chunking in book runs.

Domain	Dataset	Models	NC	Constraint	Run config
Verbatim probabilistic (teacher forcing); greedy generation					
Books	*Winnie the Pooh*	LLAMA 2 {7B, 13B, 70B}	No	–	batch 200 (7B,13B), 400 (70B); stride 20; GPUs 1/1/4
Books	*The Great Gatsby*	LLAMA 2 {7B, 13B, 70B}	No	–	batch 200 (7B,13B), 400 (70B); stride 20; GPUs 1/1/4
Books	*Orlando*	LLAMA 2 {7B, 13B, 70B}	No	–	batch 200 (7B,13B), 400 (70B); stride 20; GPUs 1/1/4
Books	*The People's Dictator*	LLAMA 2 {7B, 13B, 70B}; LLAMA 3.1 8B	Yes	–	batch 200 (7B,13B,8B), 400 (70B); stride 20; GPUs 1/1/1/4
Books	*Pride and Prejudice*	LLAMA 3.1 8B	No	–	batch 200; stride 20; GPUs 1
Emails	Enron	PYTHIA {1B, 2.8B, 6.9B, 12B}	No	–	batch 200; stride –; GPUs 1
Emails	TREC 2007 Spam "ham" subset	PYTHIA {1B, 2.8B, 6.9B, 12B}	Yes	–	batch 200; stride –; GPUs 1
Wikipedia	Train	OLMO 2 {7B, 13B, 32B}	No	–	batch 200 (7B,13B), 400 (32B); stride –; GPUs 1/1/4
Wikipedia	Held out	OLMO 2 {7B, 13B, 32B}	Yes	–	batch 200 (7B,13B), 400 (32B); stride –; GPUs 1/1/4
Sampling					
MC	Single sequence, *The Great Gatsby*	LLAMA 2 7B	No	–	batch 200; stride –; GPUs 1; $M = 10{,}000$; seeds {1000,2000,3000}
k-CBS					
baseline					
One-off	Single sequence, *The Great Gatsby*	LLAMA 1 13B	No	$B \in \{20, 30, 40\}$	batch 1; stride –; GPUs 1
Emails	Enron	PYTHIA {1B, 2.8B, 6.9B, 12B}	No	$B = 20$	batch 10; stride –; GPUs 1
Emails	TREC 2007 Spam "ham" subset	PYTHIA {1B, 2.8B, 6.9B, 12B}	Yes	$B = 20$	batch 10; stride –; GPUs 1
Wikipedia	Train	OLMO 2 {7B, 13B, 32B}	No	$B = 20$	batch 10 (7B,13B), 20 (32B); stride –; GPUs 1/1/4
Wikipedia	Held out	OLMO 2 {7B, 13B, 32B}	Yes	$B = 20$	batch 10 (7B,13B), 20 (32B); stride –; GPUs 1/1/4
Books	*Winnie the Pooh*	LLAMA 2 {7B, 13B, 70B}	No	$B = 20$	batch 10 (7B,13B), 20 (70B); stride 20; GPUs 1/1/4
Books	*The People's Dictator*	LLAMA 2 {7B, 13B, 70B}	No	$B = 20$	batch 10 (7B,13B), 20 (70B); stride 20; GPUs 1/1/4
ε-viability pruned					
Books	*Winnie the Pooh*	LLAMA 2 {7B, 13B, 70B}	No	$B = 20$; Ham, $\varepsilon \in \{0, \ldots, 5\}$	batch 10 (7B,13B), 20 (70B); stride 20; GPUs 1/1/4
Books	*Winnie the Pooh*	LLAMA 2 {7B, 13B, 70B}	No	$B = 20$; Lev, $\varepsilon \in \{0, \ldots, 5\}$	batch 10 (7B,13B), 20 (70B); stride 20; GPUs 1/1/4
Books	*The People's Dictator*	LLAMA 2 {7B, 13B, 70B}	Yes	$B = 20$; Ham, $\varepsilon \in \{0, \ldots, 5\}$	batch 10 (7B,13B), 20 (70B); stride 20; GPUs 1/1/4
Books	*The People's Dictator*	LLAMA 2 {7B, 13B, 70B}	Yes	$B = 20$; Lev, $\varepsilon \in \{0, \ldots, 5\}$	batch 10 (7B,13B), 20 (70B); stride 20; GPUs 1/1/4
Books	*The Great Gatsby*	LLAMA 2 {7B, 13B, 70B}	No	$B = 20$; Lev, $\varepsilon = 5$	batch 10 (7B,13B), 20 (70B); stride 20; GPUs 1/1/4
Books	*Orlando*	LLAMA 2 {7B, 13B, 70B}	No	$B = 20$; Lev, $\varepsilon = 5$	batch 10 (7B,13B), 20 (70B); stride 20; GPUs 1/1/4
Books	*Pride and Prejudice*	LLAMA 3.1 8B	No	$B = 20$; Lev, $\varepsilon = 5$	batch 10; stride 20; GPUs 1
Books	*Winnie the Pooh*	LLAMA 2 {7B, 13B, 70B}	No	$B \in \{30, 40\}$; Lev, $\varepsilon = 5$	$B = 30$: batch 7/7/13; $B = 40$: batch 5/5/10; stride 20; GPUs 1/1/4

F.3 Extended results for OLMo 2 on Wikipedia

This appendix provides additional analysis for the OLMo 2 experiments presented in Section 5. For these settings, we run experiments with the baseline k-CBS method in Section 4.2, and do not include results for the pruned variants in Section 4.3. We only post-process the k-CBS results in this section for Levenshtein distance.

F.3.1 Extracted sequences and rates

Verbatim greedy extraction and verbatim probabilistic extraction. As a point of reference, we provide verbatim extraction rates from two different pipelines in Table 5: verbatim greedy extraction (via generation) and verbatim probabilistic extraction (via teacher forcing). While these methods involve roughly equivalent token evaluations, wall clock time for teacher forcing is almost always faster (for reasons discussed in Appendix C.3).

Table 5: **Verbatim extraction rates for OLMo 2 on Wikipedia.** Verbatim extraction rates via greedy (deterministic generation), teacher forcing, and k-CBS on training data (10,000 sequences) and held-out negative controls (5,000 sequences). Probabilistic and k-CBS rates use threshold $p_z \geq \tau_{\text{min}} = 0.001$.

	OLMo 2 7B		OLMo 2 13B		OLMo 2 32B	
Pipeline	Train	Held-out	Train	Held-out	Train	Held-out
Verbatim greedy	0.25%	0.00%	0.33%	0.00%	0.61%	0.02%
Verbatim probabilistic	0.69%	0.00%	1.02%	0.00%	1.61%	0.04%
Verbatim k-CBS	0.55%	0.00%	0.87%	0.00%	1.42%	0.04%

Table 6: **Verbatim probabilistic vs. verbatim k-CBS on OLMo 2 (training data).** k-CBS with beam width $B = 20$ may under-count verbatim extraction. We report rates ($\tau_{\text{min}} = 0.001$) for $B = 20$, the k-CBS recovery ratio, and the number of sequences missed by k-CBS.

	OLMo 2 7B	OLMo 2 13B	OLMo 2 32B
Verbatim probabilistic	0.69%	1.02%	1.61%
Verbatim k-CBS	0.55%	0.87%	1.42%
Recovery ratio	0.80	0.85	0.88
Missed by k-CBS	14	16	19

Note that, as underscored by Table 6, the lower bound extraction probabilities provided by baseline k-CBS under-count the verbatim probabilistic extraction rate produced by teacher forcing. Part of the reason is that baseline k-CBS provides a lower bound due to pruning paths each iteration, and therefore can miss some mass. We still recover the larger majority of the mass; the extraction rates are fairly similar, with a decrease in (relative) under-counting as model size increases. This is because the missed sequences are relatively low mass, and we find that larger model sizes often exhibit higher per-sequence extraction risk.

Further, there are subtle differences between logits in teacher forcing (which does not involve KV caching) and generation (which does), which can slightly change exchange results for the same sequences due to floating point rounding. Given the low cost of the verbatim probabilistic pipeline, one could also just run both and take the union over results. However, it would be necessary to note that the noise from logit differences may impact estimates—i.e., from mixing verbatim probabilities from the verbatim probabilistic pipeline with k-CBS. Alternatively, one could also widen the beam width used for k-CBS to capture more paths.

Experiments on pruned variants recover much of this missed mass, as detailed in our other experiments, further suggesting that those methods (for edit-distance-based analysis) are superior to the baseline method (Appendices F.5 & F.8).

Comparing verbatim and near-verbatim extraction rates. Figure 11 (which replicates Figure 3) compares greedy and baseline k-CBS extraction rates for OLMo 2 across model sizes and Levenshtein thresholds $\varepsilon \in \{0, \ldots, 5\}$. k-CBS dominates greedy at every ε and

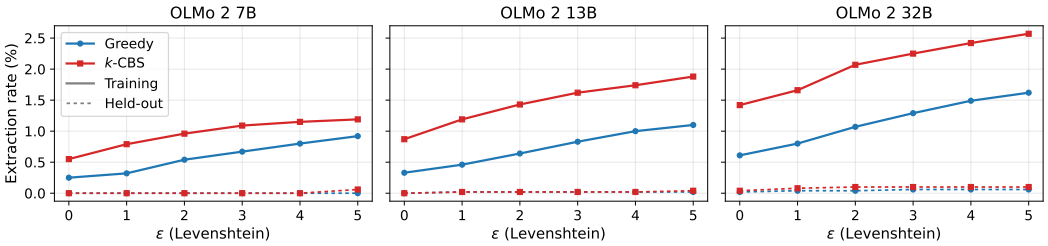


Figure 11: **Comparing extraction rates for OLMo 2.** For OLMo 2 7B, 13B, and 32B, we show greedy and k-CBS probabilistic rates for verbatim extraction ($\varepsilon = 0$) and near-verbatim extraction for $\varepsilon \in \{1, \ldots, 5\}$. We use a sample of 10,000 sequences from Wikipedia from OLMo 2's training data; to assess validity, we also run analogous negative controls on 5,000 sequences scraped from Wikipedia that post-date OLMo 2's training cutoff. The greedy rates are exact. The probabilistic rates are computed with k-CBS (Section 4.2); they may miss some valid instances of extraction, and thus should be interpreted as lower bounds on extraction rates.

model size, with both rates and gaps growing with model size. That is, greedy extraction under-counts extraction more for larger models. Held-out (negative-control) lines stay flat near zero, supporting the validity of our extraction procedure. We address held-out sequences that register as extractable in more detail below.

Negative control results. As noted in Appendix F.2.1, establishing a clean negative control on Internet-scraped data like Wikipedia is challenging. Using page create dates for determining whether text post-dates a training cutoff is imperfect; text on a newly posted Wikipedia page may be copied from an older, earlier page from elsewhere on the Internet (and included in Common Crawl); a Wikipedia page may itself be created, deleted, and the re-created as a new page years later; multiple Wikipedia pages share text/templated structure; and more. All held-out examples flagged as extractable (at $\varepsilon \leq 5$, by either greedy or probabilistic extraction) fall into these categories: they trace back to boilerplate or duplicated text that existed on Wikipedia well before OLMo 2's training cutoff. They are instances of extraction (true positives) rather than false positives.

Across the three models, we observe a total of 5 such sequences. We discuss each below, along with the model, extraction method, and near-verbatim tolerance for which it is extractable. We give a few examples of near-verbatim extracted text.

Table 7: **Held-out sequences in negative controls for OLMo 2.** Sequences that are extractable verbatim (v.) and near-verbatim (nv.) under both greedy and probabilistic extraction. A verbatim-extractable sequence is necessarily near-verbatim extractable. For probabilistic extraction, the near-verbatim mass often significantly exceeds the verbatim mass.

	7B				13B				32B			
	greedy		prob.		greedy		prob.		greedy		prob.	
	v.	nv.	v.	nv.	v.	nv.	v.	nv.	v.	nv.	v.	nv.
1. Provincial Board				✓				✓	✓	✓	✓	✓
2. 98th Academy Awards				✓		✓		✓		✓		✓
3. 2015 WinStar				✓								✓
4. Batman										✓	✓	✓
5. Finnmark												✓

1. **Maguindanao del Sur Provincial Board.** This page was created in 2025 (i.e., makes sense it would be pulled in during curation of held-out data), but the extracted text is a boilerplate template describing Philippine provincial board elections. From manual investigation using the Wikipedia API, the same template appears verbatim on $\sim$80 provincial board pages on Wikipedia, the vast majority created in August 2020 (with some dating back to 2011–2013).

$z_{(\text{pre})}$: `The Maguindanao del Sur Provincial Board is the Sangguniang Panlalawigan (provincial legislature) of the Philippine province of Maguindanao del Sur.\nThe members are elected via plurality-at-large voting: the`

$z_{(\text{suf})}$: `province is divided into two districts, each having five seats. A voter votes up to five names, with the top five candidates per district being elected. The vice governor is the ex officio presiding officer, and only votes to break ties. The`

2. **98th Academy Awards – Best International Feature Film submissions.** This page was created January 31, 2026—well after training cutoff. However, the extracted text is identical or near-identical to boilerplate shared across all "List of submissions to the Nth Academy Awards" pages. From manual inspection using the Wikipedia API, these pages date back to at least the 85th Academy Awards (created August 2012), and the boilerplate was present from their very first revisions.

   $z_{(\text{pre})}$: `This is a list of submissions to the 98th Academy Awards for Best International Feature Film. The Academy of Motion Picture Arts and Sciences (AMPAS) has invited the film industries of various countries to submit their best film for the Academy Award for`

   $z_{(\text{suf})}$: `Best International Feature Film every year since the award was created in 1956. The award is presented annually by the Academy to a feature-length motion picture produced outside the United States that contains primarily non-English dialogue. The International Feature Film Executive Committee oversees`

   Sample near-verbatim $\hat{z}_{(\text{cont})}$ (OLMo 2 13B, greedy Lev $= 1$; in character space, deletions in red in $z_{(\text{suf})}$, above, and additions in blue in $\hat{z}_{(\text{cont})}$, below):

   $\hat{z}_{(\text{cont})}$: `Best International Feature Film every year since the award was created in 1956. The award is presented annually by the Academy to a feature-length motion picture produced outside the United States that contains primarily non-English dialogue. The International Feature Film Award Committee oversees`

3. **2015 WinStar World Casino & Resort 350.** This NASCAR Truck Series race page was created January 10, 2024 (after the cutoff), but the extracted text follows the same template used across hundreds of NASCAR race articles, many of which predate the training cutoff (e.g., the 2022 NextEra Energy 250 was created February 2022). The template structure and stock phrases (track descriptions, race numbering) are widely duplicated.

   $z_{(\text{pre})}$: `The 2015 WinStar World Casino & Resort 350 was the 21st stock car race of the 2015 NASCAR Camping World Truck Series, and the 17th iteration of the event. The race was held on Friday, November`

   $z_{(\text{suf})}$: `6, 2015, in Fort Worth, Texas at Texas Motor Speedway, a 1.5 mi (2.4 km) permanent tri-oval shaped racetrack. The race took the scheduled 147 laps to complete. Erik`

4. **Batman in popular culture.** This page was created in December 2025, but the extracted text exists on multiple fan wikis dating to 2019, and thus plausibly was included in Common Crawl (and therefore the training data). It also contains a *Guardian* quote about Batman that existed verbatim in the main Batman article as of at least November 2023 (pre-cutoff)—and of course online at the *Guardian* itself.

   $z_{(\text{pre})}$: `The DC Comics character Batman has become a popular culture icon, recognized around the world. The character's presence has extended beyond his comic book origins; events such as the release of the 1989 Batman film and its accompanying merchandising "b`

   $z_{(\text{suf})}$: `rought the Batman to the forefront of public consciousness". In an article commemorating the sixtieth anniversary of the character, The Guardian wrote, "Batman is a figure blurred by the endless reinvention that is modern mass culture. He is at once an`

   Sample near-verbatim $\hat{z}_{(\text{cont})}$ (OLMo 2 32B, greedy Lev $= 1$; in character space, deletions in red in $z_{(\text{suf})}$, above, and additions in blue in $\hat{z}_{(\text{cont})}$, below):

   $\hat{z}_{(\text{cont})}$: `rought the Batman to the forefront of public consciousness." In an article commemorating the sixtieth anniversary of the character, The Guardian wrote, "Batman is a figure blurred by the endless reinvention that is modern mass culture. He is at once an`

5. **Finnmark (Storting constituency).** This page was created September 29, 2024. The extracted text is boilerplate shared across all 19 Norwegian Storting constituency pages

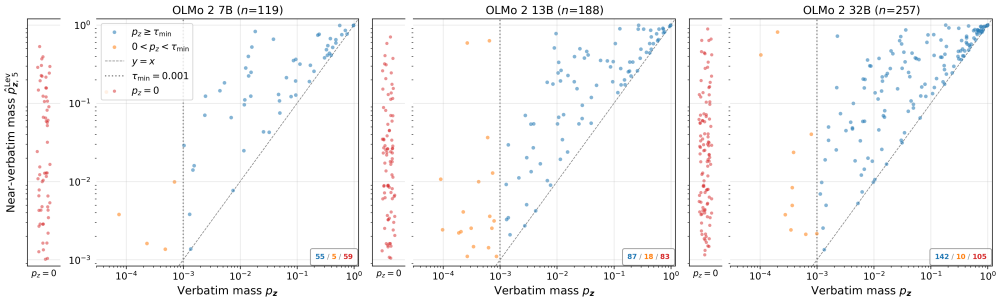


Figure 12: **Near-verbatim mass vs. verbatim mass for OLMo 2.** OLMo 2 on Wikipedia (training subset); each point is one sequence. Axes show near-verbatim ($p^{\mathsf{Lev}}_{z,5}$, $\mathsf{Lev}\,\varepsilon = 5$) vs. verbatim ($p_z$) extraction mass on a log–log scale. **Red**/**orange** points are "unlocked" by near-verbatim extraction (to the left of the $\tau_{\min}$ dotted reference line, $p_z < \tau_{\min}$, but $p^{\mathsf{Lev}}_{z,5} \geq \tau_{\min}$); **blue** points are verbatim-extractable ($p_z \geq \tau_{\min}$). Points above the dashed $y = x$ line show increased extraction risk when near-verbatim mass is accounted for.

(that we found through manual Wikipedia API search). The earliest of these (Oslo) was created August 2021, with others following in 2021–2022 (all pre-cutoff).

$z_{(\text{pre})}$: `Finnmark (Northern Sami: Finnmárku; Kven: Finmarkku) is one of the 19 multi-member constituencies of the Storting, which is the national legislature of Norway. The constituency was established in`

$z_{(\text{suf})}$: `1921 following the introduction of proportional representation for elections to the Storting. It is conterminous with the county of Finnmark. The constituency currently elects eight of the 169 members of the Storting using the open party-list proportional representation electoral`

"Unlocked" extracted sequences. Most **blue** points sit well above $y{=}x$, indicating increased extraction risk for verbatim-extractable sequences. Approximately $\sim$45–55% of extractable sequences are unlocked (**red** + **orange**) across all three sizes. The unlocked population has sequences with substantial mass, but many sequences are below 0.01. As shown below (Appendix F.3.2), the median verbatim share increases with model size and the **blue** fraction grows (from $\sim$46% for 7B and 13B $\rightarrow$ 55% for 32B), but even at 32B nearly half of extractable sequences have zero or sub-threshold verbatim mass. **Orange** points are relatively sparse; sequences tend to be either clearly verbatim-extractable or have zero verbatim mass. For comparative notes to Pythia on Enron, see Appendix 15.

F.3.2 Extraction risk

CCDF over near-verbatim risk gain. We show two views of per-sequence near-verbatim risk gain (Equation 54): (1) the CCDF over the population of training sequences, including both extractable and non-extractable (Figure 13, which duplicates Figure 5); (2) a table containing points the CCDF of per-sequence mass gain on the fixed extractable set only, where the fixed set differs by model (Table 8). In the whole-population CCDF, 32B strictly dominates at every gain threshold; larger models produce more sequences with near-verbatim mass beyond verbatim. The curve shifts upward and rightward, indicating that the absolute mass gains are also larger. In the fixed-set CCDF, there is not a clear dominance pattern with respect to relative mass gains computed according to each model's extractable set. Since these numbers show *gains*, note that it is possible for sequences that have high verbatim risk to have smaller relative gains in near-verbatim risk. (For high-risk sequences, there is less possible risk increase, both relatively and absolutely.)

Per-sequence ε-shell share analysis. We refer to Appendix F.1 for detailed explanations of the metrics and plot types for this analysis.

For OLMo 2 on Wikipedia, sequence-extraction mass is dispersed across different distances. This is clear from both the heatmap (Figure 14a) and incremental mass share (Equation 56)

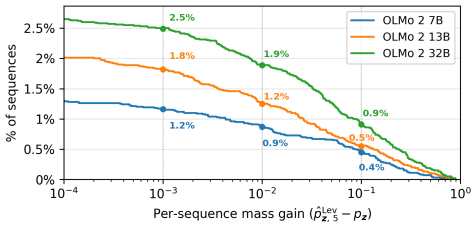
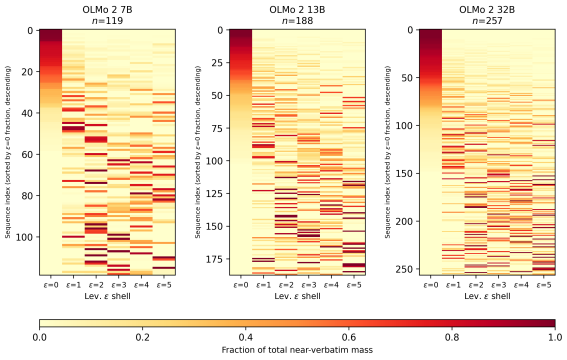


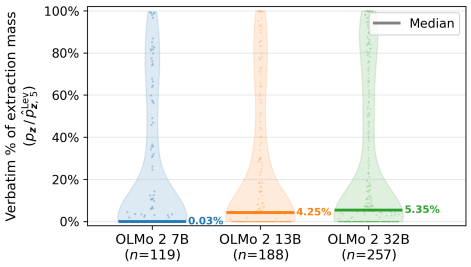
Figure 13: **CCDF of population per-sequence near-verbatim mass gain for OLMo 2.** For Lev $\varepsilon = 5$ mass minus verbatim mass ($\hat{p}^{\text{Lev}}_{z,5} - p_z$), a point (x, y) means y% of sequences have extraction-mass gain $\geq x$. Plotted over the whole training set sample (10,000 Wikipedia sequences).

Table 8: **Points on the CCDF of *extracted* per-sequence mass gain for OLMo 2.** We provide specific values from the CCDF over the per-sequence mass gain for extracted sequences only. For this CCDF, the maximum y-value is 100%. The denominators are different for each, given different counts in the fixed extractable set.

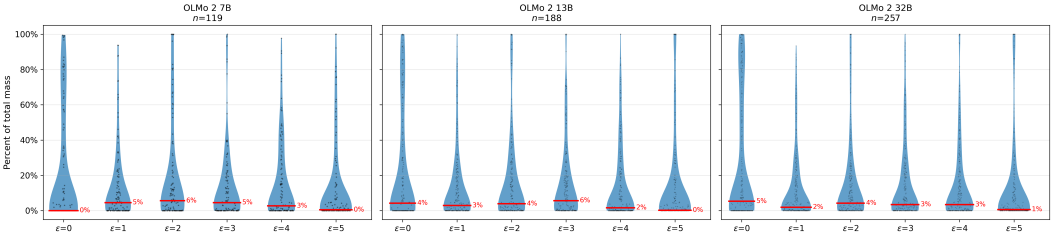
	7B	**13B**	**32B**
$\geq \mathbf{10^{-3}}$	97.5%	96.8%	96.9%
$\geq \mathbf{10^{-2}}$	73.1%	66.5%	73.5%
$\geq \mathbf{10^{-1}}$	37.8%	29.3%	35.4%



(a) Heatmap of per-sequence mass by Lev ε-shell share



(b) Verbatim mass share distributions



(c) Incremental mass share distributions

Figure 14: **Illustrating different views of ε-shell share for OLMo 2 on Wikipedia.** (**a**) Heatmaps across model size of per-sequence mass share by ε-shell (Equation 56), sorted by verbatim share (Equation 57). (**b**) Violin plots comparing the distribution of per-sequence verbatim share (Equation 57) across model sizes, with the median verbatim share annotated. (**c**) Violin plots showing distributions over the per-ε-shell mass share (Equation 56) per model. Each shell shows the mass share contributed by the given Levenshtein distance. Note that each of the three $\varepsilon = 0$ violin plots correspond to those in the cross-model comparison in (**b**).

distributions for the different ε-shells (Figure 14c). Nearly half of all extracted sequences for the 7B model have 0 verbatim share (Equation 57). The median verbatim share is 0.03%. Larger models increasingly concentrate more mass on the verbatim share, as is clear from both the tops of the heatmaps (Figure 14a) and the increasing median of the verbatim share (Figure 14b, duplicated from Figure 7a).

F.4 Extended results for PYTHIA on emails

This appendix provides experiments for PYTHIA on emails—Enron (in the training data) and TREC 2007 Spam "ham" (a curated subset) for a negative control. For these settings, we run experiments with the baseline k-CBS method in Section 4.2, and do not include results for the pruned variants in Section 4.3. We only post-process the k-CBS results in this section for Levenshtein distance.

F.4.1 Extracted sequences and rates

Verbatim greedy extraction and verbatim probabilistic extraction. As a point of reference, we provide verbatim extraction rates from two different pipelines in Table 9: verbatim greedy extraction (via generation) and verbatim probabilistic extraction (via teacher forcing). While these methods involve roughly equivalent token evaluations, wall clock time for teacher forcing is almost always faster (for reasons discussed in Appendix C.3).

Table 9: **Verbatim extraction rates for PYTHIA on Enron emails.** Verbatim extraction rates via greedy (deterministic generation), teacher forcing, and k-CBS on training data (10,000 sequences) and held-out negative controls (2,000 sequences). Probabilistic and k-CBS rates use threshold $p_z \geq \tau_{\text{min}} = 0.001$.

	PYTHIA 1B		PYTHIA 2.8B		PYTHIA 6.9B		PYTHIA 12B	
Pipeline	Train	Held-out	Train	Held-out	Train	Held-out	Train	Held-out
Verbatim greedy	0.74%	0.00%	1.81%	0.00%	2.86%	0.00%	3.84%	0.00%
Verbatim probabilistic	1.92%	0.00%	3.53%	0.00%	5.16%	0.00%	6.52%	0.00%
Verbatim k-CBS	1.54%	0.00%	3.21%	0.00%	4.83%	0.00%	6.10%	0.00%

Table 10: **Verbatim probabilistic vs. verbatim k-CBS on PYTHIA (training data).** k-CBS with beam width $B = 20$ may under-count verbatim extraction. We report rates ($\tau_{\text{min}} = 0.001$), the k-CBS recovery ratio, and the number of sequences missed by k-CBS.

	PYTHIA 1B	PYTHIA 2.8B	PYTHIA 6.9B	PYTHIA 12B
Verbatim probabilistic	1.92%	3.53%	5.16%	6.52%
Verbatim k-CBS	1.54%	3.21%	4.83%	6.10%
Recovery ratio	0.80	0.91	0.94	0.94
Missed by k-CBS	38	32	35	44

Note that, as underscored by Table 10, the lower bound extraction probabilities provided by baseline k-CBS under-count the verbatim probabilistic extraction rate produced by teacher forcing. Part of the reason is that baseline k-CBS provides a lower bound due to pruning paths each iteration, and therefore can miss some mass. We still recover the larger majority of the mass; the extraction rates are fairly similar, with a decrease in (relative) under-counting as model size increases (as with the results in Appendix F.3). This is because the missed sequences are relatively low mass, and we find that larger model sizes often exhibit higher per-sequence extraction risk.

Further, there are subtle differences between logits in teacher forcing (which does not involve KV caching) and generation (which does), which can slightly change extraction results for the same sequences due to floating point rounding. Given the low cost of the verbatim probabilistic pipeline, one could also just run both and take the union over results. However, it would be necessary to note that the noise from logit differences may impact estimates—i.e., from mixing verbatim probabilities from the verbatim probabilistic pipeline with k-CBS. Alternatively, one could also widen the beam width used for k-CBS to capture more paths.

Experiments on pruned variants recover much of this missed mass, as detailed in our other experiments, further suggesting that those methods (for edit-distance-based analysis) are superior to the baseline method (Appendices F.5 & F.8).

Comparing verbatim and near-verbatim extraction rates. Figure 15 compares greedy and baseline k-CBS extraction rates for PYTHIA across model sizes and Levenshtein thresholds

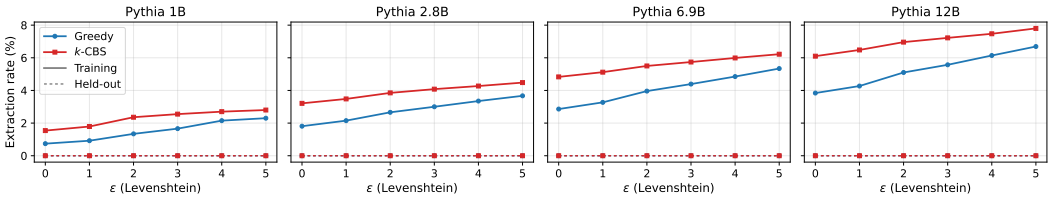


Figure 15: **Comparing extraction rates for PYTHIA.** For PYTHIA 1B, 2.8B, and 6.9B, and 12B we show greedy and k-CBS probabilistic rates for verbatim extraction ($\varepsilon = 0$) and near-verbatim extraction for $\varepsilon \in \{1, \ldots, 5\}$. We use a sample of 10,000 sequences from Enron emails from PYTHIA's training data; to assess validity, we also run analogous negative controls on 2,000 sequences from TREC 2007 Spam ("ham" subset). The greedy rates are exact. The probabilistic rates are computed with k-CBS (Section 4.2); they may miss some valid instances of extraction, and thus should be interpreted as lower bounds on extraction rates.

$\varepsilon \in \{0, \ldots, 5\}$. As with the results for OLMO 2 (Section 5 & Appendix F.3), k-CBS dominates greedy at every ε and model size, with both rates and gaps growing with model size. That is, greedy extraction under-counts extraction more for larger models. However, while this is true for comparing gaps between models at the same ε, gaps for the same model shrink with increasing ε. In this sense, the greedy rate becomes a better estimate of extraction for a given model at larger ε.

At $\varepsilon = 5$, PYTHIA 12B reaches an extraction rate of 7.80% (vs. similarly-sized OLMO 2 13B at 1.88% on Wikipedia). This likely reflects properties of the training data—possibly the data itself (wiki entries vs. emails) and training mix. Notably, the Enron corpus itself contains substantial internal duplication, and the Pile includes it without deduplication. Held-out (negative-control) lines stay flat at zero, supporting the validity of our extraction procedure.

Negative control results. Not a single sequence from the held-out "ham" email set is flagged at any ε by either method.

"Unlocked" extracted sequences. Most **blue** points sit well above $y{=}x$, indicating increased extraction risk for verbatim-extractable sequences. Compared to the OLMO 2 on Wikipedia experiments (Appendix F.3.1), **blue** points dominate more strongly and increasingly with scale (55% $\rightarrow$ 78%). There is a clear dense cluster of **blue** points in the 10^{-2} to 10^{-1} verbatim mass range whose near-verbatim mass jumps to significantly above 10^{-1}. These are sequences with moderate verbatim mass where near-verbatim extraction roughly goes from 2×–10× the verbatim extraction risk. This cluster is visible at all four model sizes, though is slightly less pronounced for the 1B model. It suggests a population of Enron sequences that the model has memorized well but not perfectly—enough verbatim mass to be extractable, but with a lot of additional probability on close variants. The unlocked fraction (**red** + **orange**) shrinks more dramatically (45% $\rightarrow$ 22%). **Orange** points are relatively sparse; sequences tend to be either clearly verbatim-extractable or have zero verbatim mass.

Overall, for both these experiments and those on OLMO 2 (Appendix 11), near-verbatim extraction both reveals hidden sequences (**red**/**orange**) and increases measured risk for verbatim-extractable sequences (**blue** above $y = x$). Both effects are present in both sets of experiments. Where they differ is in the balance: OLMO 2 on Wikipedia has a persistently large hidden population with substantial mass; the extractable set for PYTHIA on Enron is more dominated by verbatim memorization.

F.4.2 Extraction risk

CCDF over near-verbatim risk gain. We show two views of per-sequence near-verbatim risk gain (Equation 54): (1) the CCDF over the population of training sequences, including both extractable and non-extractable (Figure 17); (2) a table containing points the CCDF of per-sequence mass gain on the fixed extractable set only, where the fixed set differs by model (Table 11). In the whole-population CCDF, larger models produce more sequences with

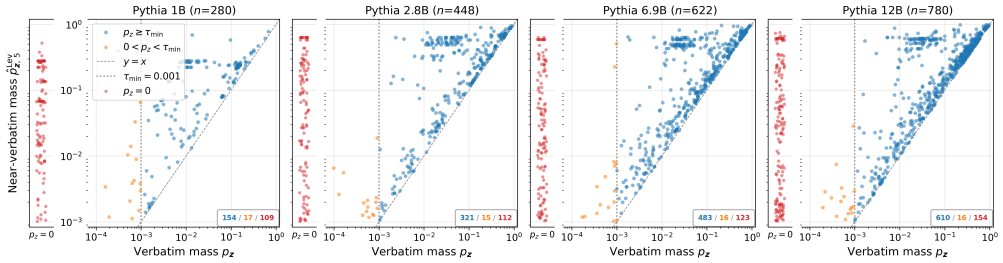


Figure 16: **Near-verbatim mass vs. verbatim mass for PYTHIA.** PYTHIA on Enron emails; each point is one sequence. Axes show near-verbatim ($p_{z,5}^{\text{Lev}}$, $\text{Lev}\,\varepsilon = 5$) vs. verbatim ($p_z$) extraction mass on a log–log scale. **Red**/**orange** points are "unlocked" by near-verbatim extraction (to the left of the $\tau_{\min}$ dotted reference line, $p_z < \tau_{\min}$, but $p_{z,5}^{\text{Lev}} \geq \tau_{\min}$); **blue** points are verbatim-extractable ($p_z \geq \tau_{\min}$). Points above the dashed $y = x$ line show increased extraction risk when near-verbatim mass is accounted for.

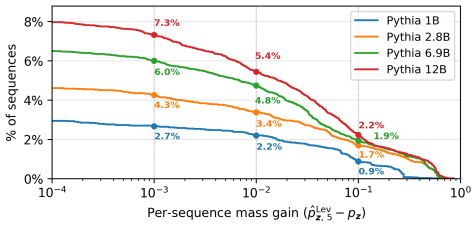


Figure 17: **CCDF of population per-sequence near-verbatim mass gain for PYTHIA.** For $\text{Lev}\,\varepsilon = 5$ mass minus verbatim mass ($\hat{p}_{z,5}^{\text{Lev}} - p_z$), a point (x, y) means y% of sequences have extraction-mass gain $\geq x$. Plotted over the whole training set sample (10,000 Wikipedia sequences).

Table 11: **Points on the CCDF of *extracted* per-sequence mass gain.** We provide specific values from the CCDF over the per-sequence mass gain for extracted sequences only. For this CCDF, the maximum y-value is 100%. The denominators are different for each, given different counts in the fixed extractable set for PYTHIA.

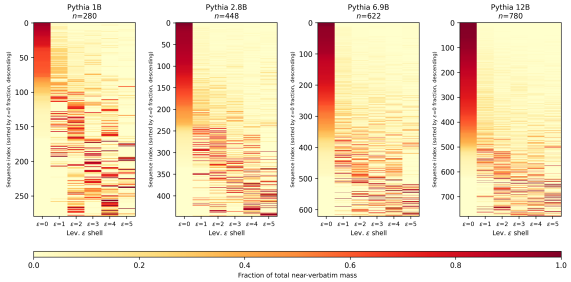
	1B	**2.8B**	**6.9B**	**12B**
$\geq \mathbf{10^{-3}}$	95.7%	95.1%	96.6%	93.8%
$\geq \mathbf{10^{-2}}$	78.6%	75.4%	76.4%	69.7%
$\geq \mathbf{10^{-1}}$	31.8%	38.2%	31.2%	28.6%

near-verbatim mass beyond verbatim. The curve shifts upward and rightward, indicating that the absolute mass gains are also larger, but the differences converge for the 2.8B, 6.9B, and 12B models at around 10^{-1}. In the fixed-set CCDF, there is not a clear dominance pattern with respect to relative mass gains computed according to each model's extractable set. Since these numbers show *gains*, note that it is possible for sequences that have high verbatim risk to smaller relative gains in near-verbatim risk. (For high-risk sequences, there is less possible risk increase, both relatively and absolutely.)

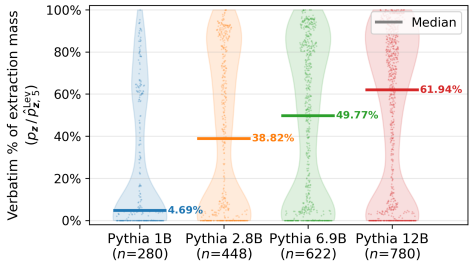
Per-sequence ε-shell share analysis. We refer to Appendix F.1 for detailed explanations of the metrics and plot types for this analysis.

As is clear in all three groups of plots in Figure 18, for PYTHIA on Enron there is a clear (and increasing) concentration of mass on the verbatim continuation as model size increases. Mass is more evenly distributed across distances for PYTHIA 1B (Figure 18c), but in general there is a stark pattern of an increasing median verbatim share (Equation 57) as model size increases from 1B to 12B, with a huge jump in between 1B and 2.8B (4.69% $\rightarrow$ 38.82% $\rightarrow$ 49.77% $\rightarrow$ 61.94%). For the majority of sequences for the 12B model, verbatim share makes up significantly over half of the extraction mass (Figure 18b); the heatmaps similarly show this, with an enormous number of verbatim-only and verbatim-heavy sequences (Figure 18a). These results are consistent with larger models more consistently memorizing the exact target suffix, with smaller models exhibiting fuzzier memorization of the target suffix. However, even for the 2.8B model, the verbatim share is substantial.

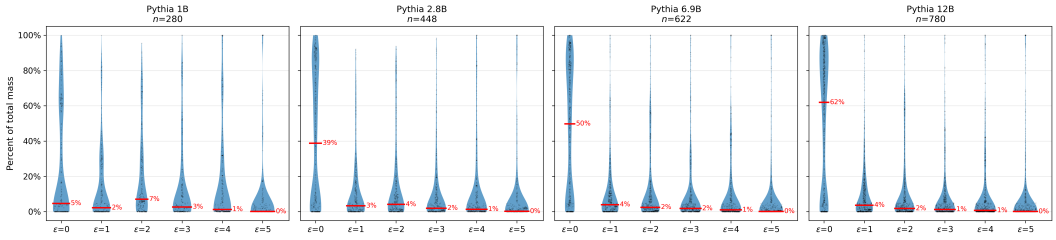
These results are also consistent with the scatter plots in Figure 16, which show substantial numbers of sequences with high verbatim mass, especially at larger model sizes. (See the



(a) Heatmap of per-sequence mass by Lev ε-shell share



(b) Verbatim mass share distributions



(c) Incremental mass share distributions

Figure 18: **Illustrating different views of ε-shell share for PYTHIA.** (**a**) Heatmaps across model size of per-sequence mass share by ε-shell (Equation 56), sorted by verbatim share (Equation 57). (**b**) Violin plots comparing the distribution of per-sequence verbatim share (Equation 57) across model sizes, with the median verbatim share annotated. (**c**) Violin plots showing distributions over the per-ε-shell mass share (Equation 56) per model. Each shell shows the mass share contributed by the given Levenshtein distance. Note that each of the three $\varepsilon = 0$ violin plots correspond to those in the cross-model comparison in (**b**).

blue points concentrated in the top right, which necessarily are close to the $y = x$ reference line as there is not much more mass that near-verbatim could add, given how high the verbatim mass already is.)

Even so, at all model sizes, there are still many sequences have 0 verbatim share (corresponding to the **red** points in Figure 16), though the fraction of such zero-verbatim-share sequences decreases per model over each model's respective extractable set (38.9% for 1B to 19.7% for 12B). As shown in Figure 18c, mass from other distances is more evenly dispersed.

F.5 Extended results for LLAMA 2 on public domain books

We include results for three public domain books (*The Great Gatsby*, *Orlando*, and *Winnie the Pooh*) using the Lev-pruned variant ($\varepsilon = 5$). The results for *The Great Gatsby* supplement those in Section 5. The negative control is the first three chapters of *The People's Dictator*, which has no hits for any of the three LLAMA 2 sizes. (It is a perfectly clean negative control.)

F.5.1 Extracted sequences and rates

Verbatim greedy extraction and verbatim probabilistic extraction. As a point of reference, we provide verbatim extraction rates from two different pipelines: verbatim greedy extraction (via generation) and verbatim probabilistic extraction (via teacher forcing). See Table 12 for all three books. While these methods involve roughly equivalent token evaluations, wall clock time for teacher forcing is almost always faster (for reasons discussed in Appendix C.3).

Note that, as underscored by the results in Table 13, the lower bound extraction probabilities provided by Lev-pruned k-CBS ($\varepsilon = 5$) can under-count the verbatim probabilistic extraction rate produced by teacher forcing. For instance, for LLAMA 2 70B and *The Great Gatsby* (Table 13a), verbatim k-CBS misses 86 instances of extraction (20.30% extraction

Table 12: **Verbatim extraction rates for LLAMA 2.** Verbatim extraction rates via greedy (deterministic generation), teacher forcing, and k-CBS (Lev $\varepsilon = 5$). For all experiments, held-out negative controls use the first three chapters of *The People's Dictator* (10,457 sequences). Probabilistic and k-CBS rates use threshold $p_z \geq \tau_{\text{min}} = 0.001$.

(a) *The Great Gatsby* (Train: 13,390 sequences)

	LLAMA 2 7B		LLAMA 2 13B		LLAMA 2 70B	
Pipeline	Train	Held-out	Train	Held-out	Train	Held-out
Verbatim greedy	0.47%	0.00%	0.79%	0.00%	7.24%	0.00%
Verbatim probabilistic	1.21%	0.00%	2.46%	0.00%	20.91%	0.00%
Verbatim k-CBS (Lev $\varepsilon = 5$)	1.22%	0.00%	2.44%	0.00%	20.30%	0.00%

(b) *Orlando* (Train: 21,822 sequences)

	LLAMA 2 7B		LLAMA 2 13B		LLAMA 2 70B	
Pipeline	Train	Held-out	Train	Held-out	Train	Held-out
Verbatim greedy	0.00%	0.00%	0.00%	0.00%	0.11%	0.00%
Verbatim probabilistic	0.00%	0.00%	0.03%	0.00%	0.31%	0.00%
Verbatim k-CBS (Lev $\varepsilon = 5$)	0.00%	0.00%	0.03%	0.00%	0.29%	0.00%

(c) *Winnie the Pooh* (Train: 6,152 sequences)

	LLAMA 2 7B		LLAMA 2 13B		LLAMA 2 70B	
Pipeline	Train	Held-out	Train	Held-out	Train	Held-out
Verbatim greedy	0.10%	0.00%	0.29%	0.00%	3.75%	0.00%
Verbatim probabilistic	0.39%	0.00%	0.78%	0.00%	10.86%	0.00%
Verbatim k-CBS (Lev $\varepsilon = 5$)	0.39%	0.00%	0.78%	0.00%	10.70%	0.00%

rate, compared to the teacher-forced verbatim probabilistic pipeline 20.91% rate; recovery ratio of 97%). One possible reason is that k-CBS provides a lower bound due to pruning paths each iteration, and therefore can miss some mass. Another reason is that there are subtle differences between logits in teacher forcing (which does not involve KV caching) and generation (which does), which can slightly change exchange results for the same sequences due to floating point rounding. At a given token in a suffix, there may be multiple tokens with very similar probabilities; their ranks may flip due to such rounding differences. This can also lead to k-CBS returning a *higher* verbatim extraction rate than the one from teacher-forced inference. See, for instance, LLAMA 2 7B and *The Great Gatsby* (Table 13a), where k-CBS has a higher extraction rate (1.22% compared to 1.21%; recovery ratio of 1.01). Overall, the results between methods are very similar with respect to overall extraction rates; both flows capture essentially the same results for verbatim extraction. We could also widen the beam, at additional cost, to attempt to capture more with k-CBS.

Comparing verbatim and near-verbatim extraction rates. Figure 19 compares greedy and Lev $\varepsilon = 5$ k-CBS extraction rates for LLAMA 2 across model sizes and Levenshtein thresholds $\varepsilon \in \{0, \ldots, 5\}$ for *The Great Gatsby*, *Orlando*, and *Winnie the Pooh*. As with the results for OLMO 2 (Section 5 & Appendix F.3) and PYTHIA (Appendix F.4), k-CBS dominates greedy at every ε and model size, with both rates and gaps growing with model size. (The exception is *Winnie the Pooh*, where we observe a very slightly increased gap between 7B and 13B; it is more pronounced for the other books. See Figure 19c.) That is, greedy extraction under-counts extraction more for larger models. The extraction rates for LLAMA 2 70B are relatively enormous per-book, compared to the smaller models. This is true even for *Orlando*, where the overall amount of extraction we observe is very small (the maximum near-verbatim probabilistic rate is just above 0.4%, see Figure 19b). Held-out (negative-control) lines stay flat at zero, supporting the validity of our extraction procedure.

Negative control results. Not a single sequence from the held-out book (*The People's Dictator*) is flagged at any ε by either method.

Table 13: **Verbatim probabilistic vs. verbatim k-CBS on LLAMA 2.** We report extraction rates ($\tau_{\text{min}} = 0.001$, $B = 20$, Lev $\varepsilon = 5$), the k-CBS recovery ratio, and the number of sequences missed by k-CBS.

(a) *The Great Gatsby*

	LLAMA 2 7B	LLAMA 2 13B	LLAMA 2 70B
Verbatim probabilistic	1.21%	2.46%	20.91%
Verbatim k-CBS (Lev $\varepsilon = 5$)	1.22%	2.44%	20.30%
Recovery ratio	1.01	0.99	0.97
Missed by k-CBS (Lev $\varepsilon = 5$)	0	3	86

(b) *Orlando*

	LLAMA 2 7B	LLAMA 2 13B	LLAMA 2 70B
Verbatim probabilistic	0.00%	0.03%	0.31%
Verbatim k-CBS (Lev $\varepsilon = 5$)	0.00%	0.03%	0.29%
Recovery ratio	inf	1.00	0.93
Missed by k-CBS	0	0	5

(c) *Winnie the Pooh*

	LLAMA 2 7B	LLAMA 2 13B	LLAMA 2 70B
Verbatim probabilistic	0.39%	0.78%	10.86%
Verbatim k-CBS (Lev $\varepsilon = 5$)	0.39%	0.78%	10.70%
Recovery ratio	1.00	1.00	0.99
Missed by k-CBS	0	0	11

"Unlocked" extracted sequences. The three different books exhibit very different degrees of extraction (in terms of number of instances) and risk (in terms of magnitude and variation). With respect to risk, Figure 20 gives a sense of how per-sequence risk changes across model sizes and books.

The Great Gatsby is more significantly memorized in general, and shows larger amounts of verbatim extraction (and increased risk) as well as "unlocked" near-verbatim extraction. *Orlando* is barely memorized by any model, but there is increased memorization (and risk) for LLAMA 2 70B. *Winnie the Pooh* shows patterns similar to *The Great Gatsby*, but there is less extraction overall.

In general, larger models exhibit larger degrees of risk—**blue** points sit well above $y{=}x$, indicating increased extraction risk for verbatim-extractable sequences, with higher risk overall for larger models. Near-verbatim measurements reveal extraction that verbatim methods do not catch (**red** + **orange**).

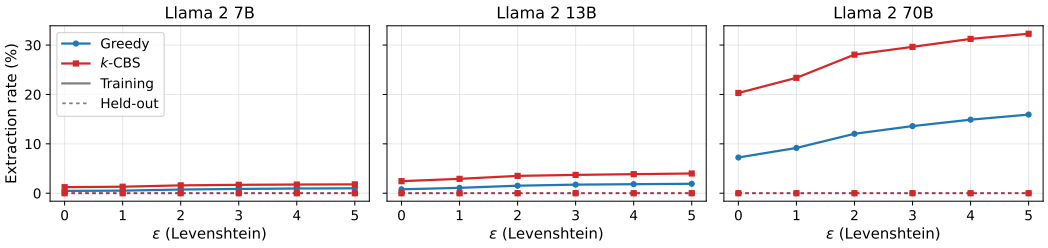
We provide some qualitative examples of extracted sequences, to give a sense of what near-verbatim measurements capture (in terms of the diff and risk increase). We mark missing text from the ground-truth suffix from Books3 in red and additions in the generation in blue.

- *The Great Gatsby*, LLAMA 2 70B
  - Verbatim mass: 0; near-verbatim mass (Lev $\varepsilon = 5$): 0.863. A typo in the ground-truth text we have from Books3 is repaired by the closest-match generation.

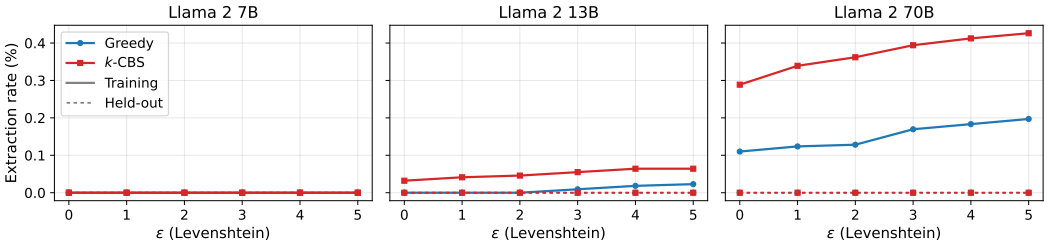
    $z_{(\text{pre})}$: `In between time–"\n\nAs I went over to say goodbye I saw that the expression of bewilderment had come back into Gatsby's face, as though a faint doubt had occurred to him as to the`

    $z_{(\text{suf})}$: ` quality of his present happiness. Almost five years! There must have been moments even that afternoon whe Daisy tumbled short of his dreams–not through her own fault, but because of the colossal vitality of his illusion`

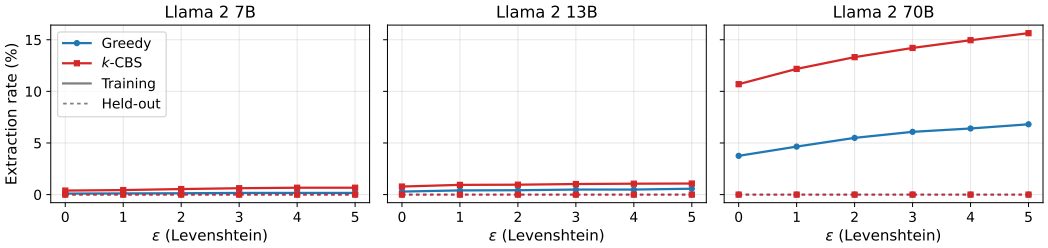
    Best near-verbatim $\hat{z}_{(\text{cont})}$ (Lev $= 1$, mass 0.628): ` quality of his present happiness. Almost five years! There must have been moments even that afternoon when Daisy tumbled short of his dreams–not through her own fault, but because of the colossal vitality of his illusion`



(a) *The Great Gatsby* (13,390 training sequences)



(b) *Orlando* (21,822 training sequences)



(c) *Winnie the Pooh* (6,152 training sequences)

Figure 19: **Comparing extraction rates for LLAMA 2.** For LLAMA 2 7B, 13B, and 70B we show greedy and Lev $\varepsilon = 5$ k-CBS probabilistic rates for verbatim extraction ($\varepsilon = 0$) and near-verbatim extraction for $\varepsilon \in \{1, \ldots, 5\}$ (based on subsetting the pruned algorithm's results). To assess validity, we also run analogous negative controls on 10,457 sequences from *The People's Dictator* (first three chapters). The greedy rates are exact. The probabilistic rates are computed with Lev-pruned k-CBS (Section 4.3); they may miss some valid instances of extraction, and thus should be interpreted as lower bounds on extraction rates.

  - Verbatim mass: 0; near-verbatim mass (Lev $\varepsilon = 5$): 0.822. A space in the ground-truth suffix is not included in the closest-match generation.

    $z_{(\text{pre})}$: `y anything in his house, old sport."\n\n"She's got an indiscreet voice," I remarked. "It's full of—" I hesitated.\n\n"Her voice is full of money," he`

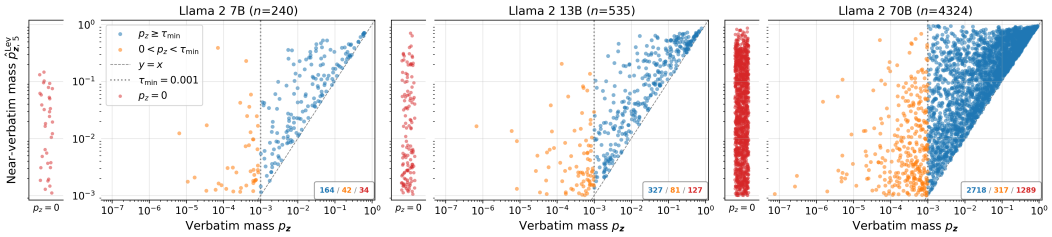
    $z_{(\text{suf})}$: `said suddenly.\n\nThat was it. I'd never understood before. It was full of money—that was the inexhaustible charm that rose and fell in it, the jingle of it, the cymbals`

    Best near-verbatim $\hat{z}_{(\text{cont})}$ (Lev $= 1$, mass 0.777): `said suddenly.\n\nThat was it I'd never understood before. It was full of money—that was the inexhaustible charm that rose and fell in it, the jingle of it, the cymbals`

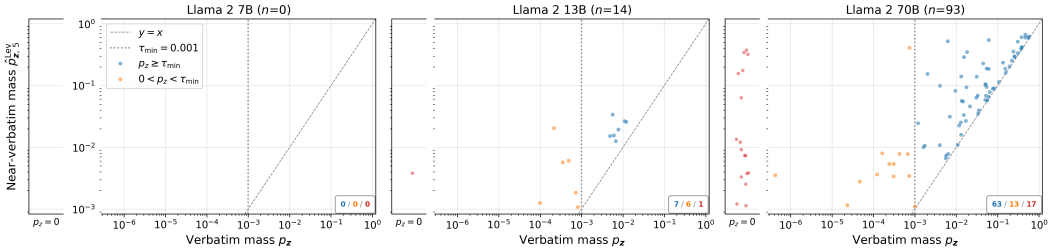
- *Orlando*, LLAMA 2 70B

  - Verbatim mass 0.000746; near-verbatim mass (Lev $\varepsilon = 5$): 0.408. Punctuation differences.

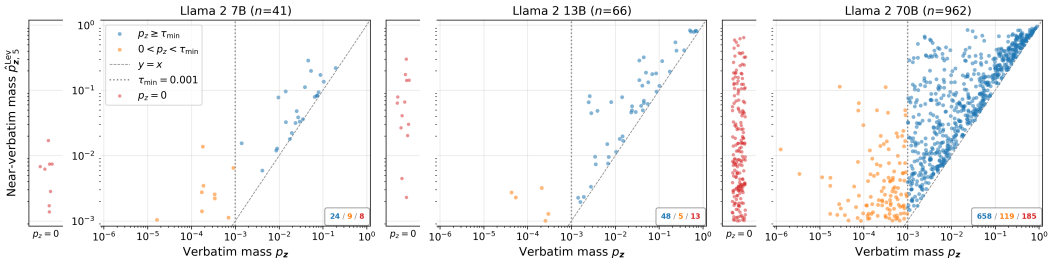
    $z_{(\text{pre})}$: `n. Let us, as he takes his seat, read the following passage from the _Spectator_ :\n\n'I consider woman as a beautiful, romantic animal, that may be adorned with furs and feathers`



(a) *The Great Gatsby* (replicated from Figure 4)



(b) *Orlando*



(c) *Winnie the Pooh*

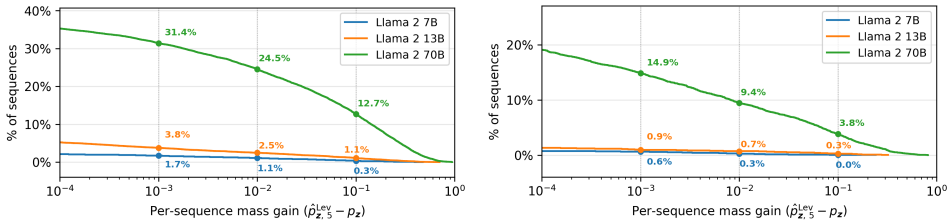
Figure 20: **Near-verbatim mass vs. verbatim mass for LLAMA 2.** LLAMA 2 on three public domain books; each point is one sequence. Axes show near-verbatim ($p_{z,5}^{\text{Lev}}$, Lev $\varepsilon = 5$) vs. verbatim (p_z) extraction mass on a log–log scale. **Red**/**orange** points are "unlocked" by near-verbatim extraction (to the left of the τ_{min} dotted reference line, $p_z < \tau_{\text{min}}$, but $p_{z,5}^{\text{Lev}} \geq \tau_{\text{min}}$); **blue** points are verbatim-extractable ($p_z \geq \tau_{\text{min}}$). Points above the dashed $y = x$ line show increased extraction risk when near-verbatim mass is accounted for.

> $z_{(\text{suf})}$: , pearls and diamonds, ores and silks. The lynx shall cast its skin at her feet to make her a tippet, the peacock, parrot and swan shall pay contributions to her muff;
>
> Best near-verbatim $\hat{z}_{(\text{cont})}$ (Lev $= 3$, mass 0.232): , pearls and diamonds, ores and silks. The lynx shall cast its skin at her feet to make her a tippet; the the peacock, parrot, and swan shall pay contributions to her muff

– Verbatim mass 0; near-verbatim mass (Lev $\varepsilon = 5$): 0.176. Punctuation differences, and then text shift as a result of tokenization-to-text length variation.

> $z_{(\text{pre})}$: the voyage to the Houyhnhnms:\n\n'I enjoyed perfect Health of Body and Tranquillity of Mind; I did not find the Treachery or Inconstancy of a Friend, nor the Injuries
>
> $z_{(\text{suf})}$: of a secret or open Enemy. I had no occasion of bribing, flattering or pimping, to procure the Favour of any great Man or of his Minion. I wanted no Fence against Fraud
>
> Best near-verbatim $\hat{z}_{(\text{cont})}$ (Lev $= 5$, mass 0.127): of a secret or open Enemy. I had no occasion of bribing, flattering, or pimping, to procure the Favour of any great Man, or of his Minion; I wanted no Fence against

- *Winnie the Pooh,* LLAMA 2 70B



(a) *The Great Gatsby* (replicated from Figure 5b) (b) *Winnie the Pooh*

Figure 21: **CCDF of population per-sequence near-verbatim mass gain for LLAMA 2.** For Lev $\varepsilon = 5$ mass minus verbatim mass ($\hat{p}^{\text{Lev}}_{z,5} - p_z$), a point (x, y) means y% of sequences have extraction-mass gain $\geq x$. Plotted over the whole training set sample for each book (13,390 sequences from *The Great Gatsby*; 6,152 sequences from *Winnie the Pooh*).

– Verbatim mass 0; near-verbatim mass (Lev $\varepsilon = 2$): 0.643. Hyphenization difference, and then text shift as a result of tokenization-to-text length variation.

$z_{(\text{pre})}$: `t\n\n## IN WHICH\n\nChristopher Robin Leads an Expotition to the North Pole \n\nONE FINE DAY Pooh had stumped up to the top of the Forest to see if his friend`

$z_{(\text{suf})}$: `Christopher Robin was interested in Bears at all. At breakfast that morning (a simple meal of marmalade spread lightly over a honey-comb or two) he had suddenly thought of a new song. It began like this:\n`

Best near-verbatim $\hat{z}_{(\text{cont})}$ (Lev $= 2$, mass 0.405): `Christopher Robin was interested in Bears at all. At breakfast that morning (a simple meal of marmalade spread lightly over a honeycomb or two) he had suddenly thought of a new song. It began like this:\n\n`

Note that the extracted sequences from *Orlando* included above reflect quotes from other texts, not Woolf's original writing. The first sequence is from an essay by Joseph Addison, and the second is from *Gulliver's Travels*.

F.5.2 Extraction risk

CCDF over near-verbatim risk gain. We similarly provide CCDF views of the near-verbatim risk gain, specifically for *The Great Gatsby* and *Winnie the Pooh* (Figure 21). We omit *Orlando* given the minimal amount of extraction, and instead refer to the scatter plot visualization to assess risk gain for those results (Figure 20b). We also provide tables of points on the CCDF of the per-sequence mass gain on the fixed extractable set only, where the fixed set differs by model (Table 14). Overall, we observe the same patterns as in our other results on OLMO 2 (Appendix F.2.1) and PYTHIA (Appendix F.2.2), albeit with larger gaps between the 70B model and the smaller models. In the fixed-set CCDF results, unlike for the prior results, we do generally observe a dominance pattern with respect to relative mass gains computed according to each model's extractable set for *The Great Gatsby*: this underscores the enormity of the relative, not only absolute, gains in risk for the 70B model on this book.

Per-sequence ε-shell share analysis. We refer to Appendix F.1 for detailed explanations of the metrics and plot types for this analysis. The dispersal across ε-shells varies by book, with respect to how memorized the book is. As discussed in the main paper (Section 5.2), *The Great Gatsby* shows decreasing median verbatim share by model size. Here, it is also clear that the majority of mass comes from $\varepsilon \leq 2$ across all three model sizes, with fairly equal dispersal in the last three distance shells. There is far less extraction for *Winnie the Pooh* for both LLAMA 2 7B and 13B, so these violins are produced from significantly fewer data points. We therefore do not make claims about relative mass shifts across model sizes, but we do note that the verbatim shell contains the most mass (with respect to median verbatim share) across all three model sizes.

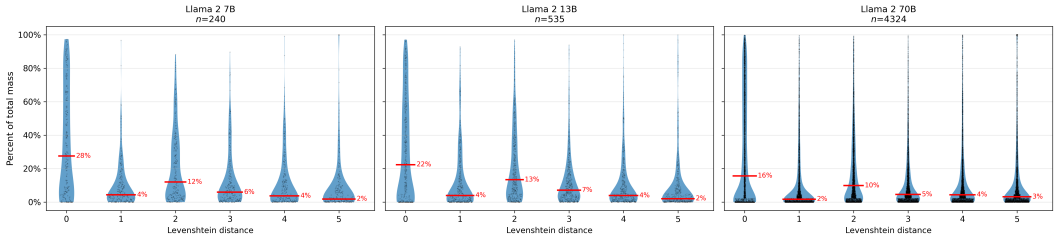
Table 14: **Points on the CCDF of *extracted* per-sequence mass gain for LLAMA 2.** We provide specific values from the CCDF over the per-sequence mass gain for extracted sequences only. For this CCDF, the maximum y-value is 100%. The denominators are different for each, given different counts in the fixed extractable set for LLAMA 2, both on different models and different books.

The Great Gatsby

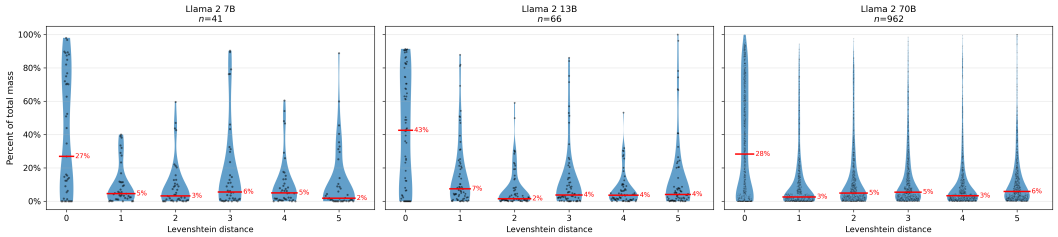
	7B	**13B**	**70B**
$\geq \mathbf{10^{-3}}$	95.0%	94.0%	97.2%
$\geq \mathbf{10^{-2}}$	59.6%	61.7%	76.0%
$\geq \mathbf{10^{-1}}$	17.1%	26.9%	39.2%

Winnie the Pooh

	7B	**13B**	**70B**
$\geq \mathbf{10^{-3}}$	95.1%	87.9%	95.0%
$\geq \mathbf{10^{-2}}$	45.1%	65.2%	24.2%
$\geq \mathbf{10^{-1}}$	4.9%	60.4%	24.5%



(a) *The Great Gatsby*



(b) *Winnie the Pooh*

Figure 22: **Illustrating ε-shell share for LLAMA 2.** Violin plots showing distributions over the per-ε-shell mass share (Equation 56) per model and two books. Each shell shows the mass share contributed by the given Levenshtein distance.

F.5.3 Heatmaps and cross-model sequence analysis

Heatmaps visualizing extracted regions. We heatmap draw from visualization strategies in Cooper et al. (2025) to illustrate where memorization manifests in a particular book (character location), enabling comparisons of regions of memorized text across models. In Figure 24 for a given heatmap, each vertical strip corresponds to a character position in the source text; color intensity shows the *maximum* extraction probability across all overlapping 50-token suffixes that cover that position, on a log scale. White indicates no extractable suffix ($< \tau_{\text{min}} = 0.001$).

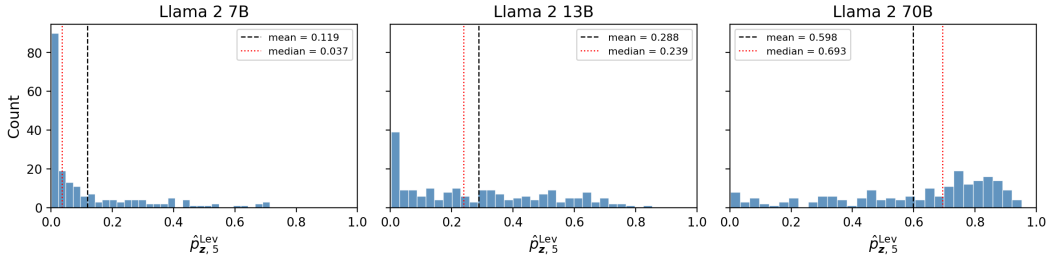
The heatmaps showcase how memorization varies across models and books. Our analysis shows the verbatim heatmaps (as in Cooper et al. (2025)), compared to the near-verbatim heatmaps produced using $\text{Lev}, \varepsilon = 5$. There are several points worth highlighting. The first two underscore our other analysis: near-verbatim extraction identifies more unique instances of extraction, and increased extraction risk for sequences that are verbatim extractable. For instance, consider Figure 24a, which shows the heatmaps for *The Great Gatsby*. For LLAMA 2 7B at around character 220,000, the near-verbatim heatmap picks up extraction that verbatim extraction misses; starting at around character 170,000, the near-verbatim extracted region exhibits an extraction risk increase (darker blue) compared to the verbatim heatmap.

The heatmaps also show how near-verbatim extraction at smaller sizes can be an indicator (though not determinative of) verbatim extraction at larger model sizes. For instance, for *The Great Gatsby* and LLAMA 2 7B, there are 76 sequences that are Lev $\varepsilon = 5$ near-verbatim extractable (visible on the near-verbatim heatmap) but not verbatim extractable (not visible on the verbatim heatmap). Of these sequences, 31 (40.8%) are verbatim extractable at LLAMA 2 13B. Similarly, there are 208 Lev $\varepsilon = 5$ near-verbatim extractable (but not verbatim extractable) sequences, of which 106 (51.0%) are verbatim extractable at LLAMA 2 70B. One of the easiest to see examples of this is with the *Orlando* heatmaps (Figure 24b), as there are much sparser extraction hits to pick apart. At about 30,000 characters, both LLAMA 2 7B and LLAMA 2 13B show a near-verbatim extractable region that is not verbatim extractable; this region is verbatim extractable for LLAMA 2 70B.

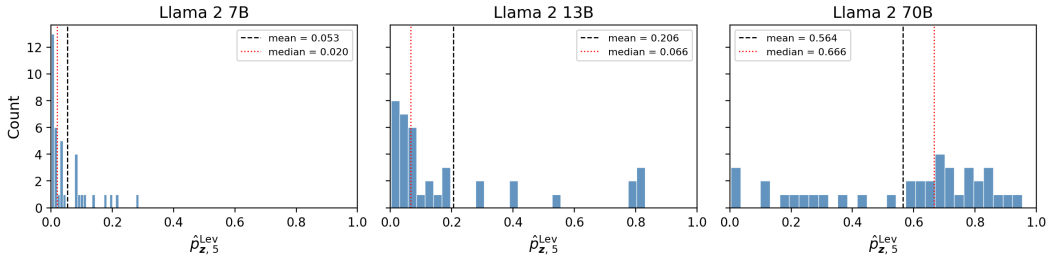
Analyzing mass for common extractable sequences across model sizes. The heatmaps (Figure 24) show significantly increased instances of near-verbatim extraction for the 70B model, complementing the scatter plots in Figure 20). We also and observed decrease in median verbatim share from 7B to 70B for both *The Great Gatsby* and *Winnie the Pooh* (Figure 22). To decompose this a bit, we restrict to the sequences extractable at all three model sizes—i.e., hold the set of sequences fixed—and ask: how much more probability does a larger model place on the same memorized content? We show results for *The Great Gatsby* and *Winnie the Pooh* in Figure 23; we omit results for *Orlando* given the minimal amount of overall extraction.

Both books exhibit similar patterns in their distributions: from low (near τ_{min}) for 7B to very high extraction risk at 70B. At 7B, most sequences cluster near the extraction threshold (for *Gatsby*, median of 0.037 and mean of 0.119; for *Winnie the Pooh*, median of 0.053 and mean of 0.020). The typical common-extractable sequence has very low extraction risk. At 70B for both books, the bulk of the distribution sits in the 0.4–0.9 range (for *Gatsby* median of 0.693 and mean of 0.598; for *Winnie the Pooh*, median of 0.666 and mean of 0.0564). The model assigns majority probability to near-verbatim continuations. The few sequences that remain low-mass at 70B pull the mean below the median.

Overall, we observe that the shift indicates that thee same sequences carry dramatically more mass at larger model sizes. In general, model scale does not just widen the set of memorized content; it deepens the memorization of content at smaller scales. Median verbatim share *over the whole population of extracted sequences per model* can still decrease, as larger models surface many more weakly memorized sequences.

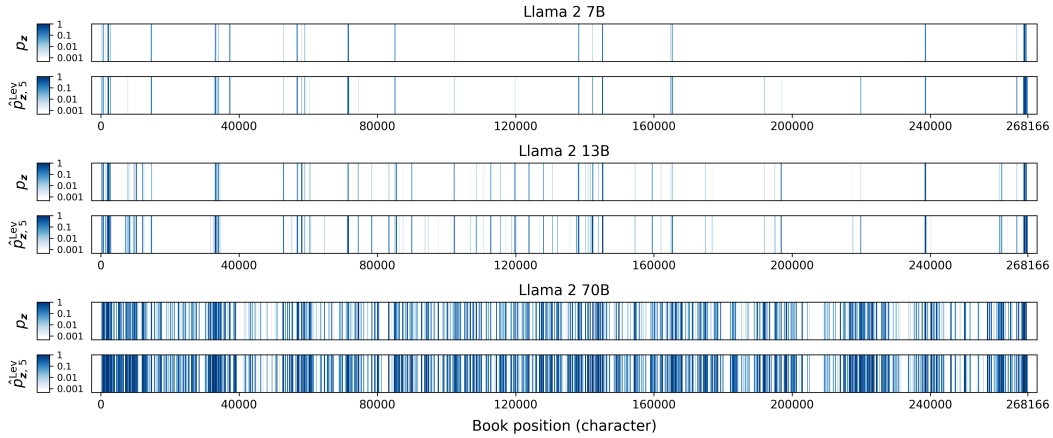


(a) *The Great Gatsby* ($n = 199$)

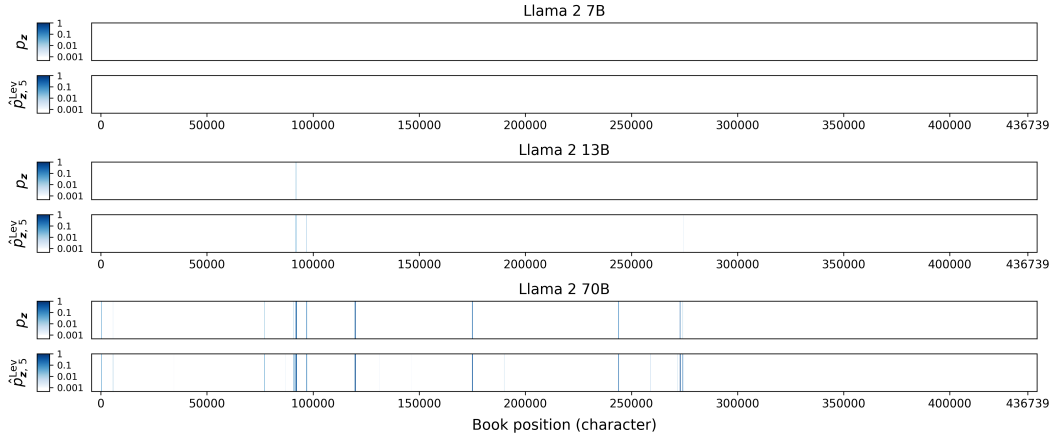


(b) *Winnie the Pooh* ($n = 38$)

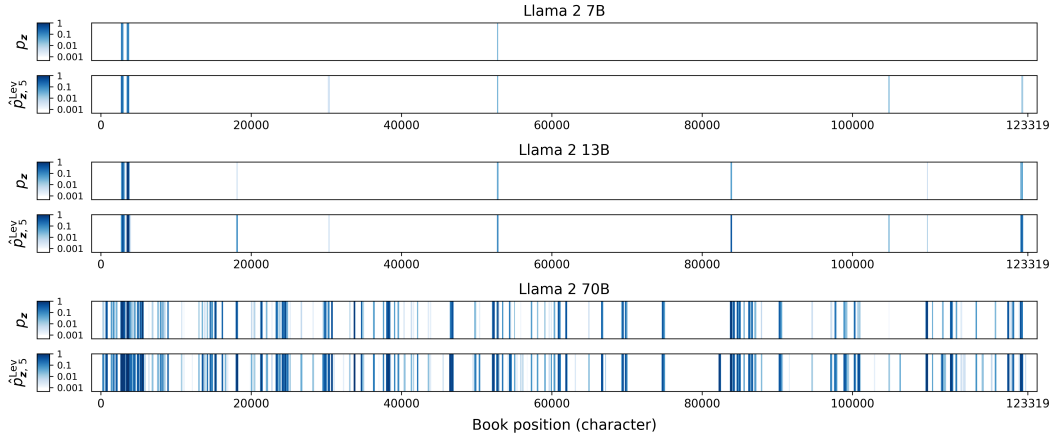
Figure 23: **Fixed-set mass grows dramatically with model size.** By restricting to the sequences extractable at all three model sizes, we hold the set of sequences fixed and ask: how much more probability does a larger model place on the same memorized content?



(a) *The Great Gatsby*



(b) *Orlando*



(c) *Winnie the Pooh*

Figure 24: **Heatmaps comparing verbatim and near-verbatim extraction risk across books for LLAMA 2 models.** For each book, we show three pairs of heatmaps—one pair for each LLAMA 2 model size, with each pair showing the verbatim extraction probability (p_z) and the near-verbatim extraction probability ($\hat{p}^{\text{Lev}}_{z,5}$). On a given heatmap, each vertical strip corresponds to a character position in the source text; color intensity shows the *maximum* extraction probability across all overlapping 50-token suffixes that cover that position, on a log scale. White indicates no extractable suffix ($< \tau_{\text{min}}$).

F.6 Additional experiment with LLAMA 3.1 8B

We ran several additional experiments on LLAMA 3.1 8B, including for additional books. For brevity, we omit most of these results In Figure 25, we offer one additional illustration for *Pride and Prejudice*, as this book exhibits extensive degrees of memorization for this model. We provide a set of scatter plots for increasing Levenshtein distance, visualizing how the risk evolves and how sequences become unlocked at different ε. $\varepsilon = 0$ is the verbatim case, so there are only **blue** points for that plot, and they all sit on the line $y = x$; we provide this as a reference, even though it just contains information about verbatim extraction.

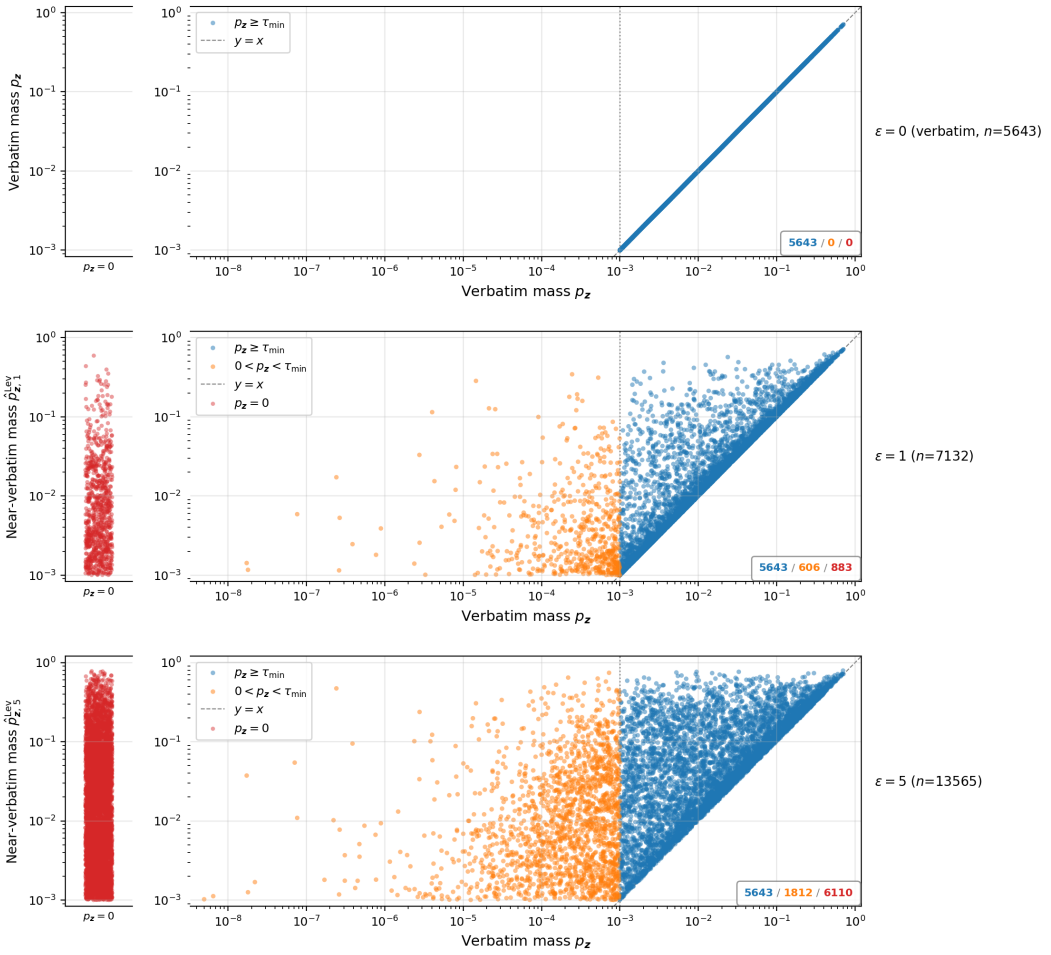


Figure 25: **Evolution of near-verbatim mass vs. verbatim mass for LLAMA 3.1 8B and *Pride and Prejudice*.** Each point is one sequence. Axes show near-verbatim ($p^{\text{Lev}}_{z,5}$, Lev $\varepsilon = 5$) vs. verbatim (p_z) extraction mass on a log–log scale. **Red**/**orange** points are "unlocked" by near-verbatim extraction (to the left of the τ_{min} dotted reference line, $p_z < \tau_{\text{min}}$, but $p^{\text{Lev}}_{z,5} \geq \tau_{\text{min}}$); **blue** points are verbatim-extractable ($p_z \geq \tau_{\text{min}}$). Points above the dashed $y = x$ line show increased extraction risk when near-verbatim mass is accounted for. The top panel is a reference for what verbatim extraction conveys. There are only **blue** points, and they are all on the line $y = x$; there are 5,643 verbatim extractable sequences. The middle panel shows $\varepsilon = 1$; even at this distance, there are a large number of "unlocked" points that have zero verbatim mass (**red**) or sub-threshold verbatim mass (**orange**); the total number of extractable points goes up from 5,643 to 7,132 sequences. The bottom panel shows $\varepsilon = 5$. There are even more unlocked sequences (13,565), and the risk spreads upward for all three categories.

F.7 Varying beam width

In our main experiments, we always set beam width $B = 20$. In this appendix, we justify this choice. For LLAMA 2 models and *Winnie the Pooh*, we run a sweep of experiments for Lev $\varepsilon = 5$ k-CBS for beam widths $B = 20, 30, 40$.[11] These increases reflect a corresponding increase in forward passes (e.g, $B = 40$ reflects a $2\times$ increase, see Appendix C.3, if we discount early termination). Overall, we find that increasing the beam width comes at substantial compute cost, but minimal change in identified extraction or extraction mass. This is why we opt to use $B = 20$; for substantially decreased cost, we effectively obtain the same information.

The three figures in this appendix support this conclusion. In Figure 26, for each model size we show how extraction counts (rate %) change according to B (column) and post-processing for a chosen Lev ε (row). In Figure 27, we produce similar plots, but for how extraction risk mean $\pm$ standard deviation changes. In both, for each per-model plot within a given ε row, red cell shading indicates a relative drop compared to the $B = 40$ results on the right (so the $B = 40$ cells are all white). For instance, for LLAMA 2 70B verbatim ($\varepsilon = 0$) post-processing of the results, $B = 40$ identifies 665 instances of extraction; $B = 30$ closely tracks with 664 (and is shaded very lightly to reflect this), and $B = 20$ finds 658 (and so is darker red). Note that shading is determined by normalizing for a given model's ε results in a given row, so an absolute drop from 69 to 66 (for LLAMA 2 13B and $\varepsilon = 5$, from $B = 40$ to $B = 20$) shades darker than an absolute drop from 988 to 962 (for LLAMA 2 70B and $\varepsilon = 5$, from $B = 40$ to $B = 20$), as the latter reflects a smaller relative drop. The same interpretation applies to the extraction risk plot shading.

Overall, from Figure 26, we find that widening the beam from $B = 20$ to $B = 40$ captures slightly more extractable sequences, but the gains are minimal. The largest absolute difference is at 70B / $\varepsilon = 5$, where $\boldsymbol{w} = 40$ finds 26 more sequences than $B = 20$ (as noted above, 988 vs 962, a 0.42 percentage point increase in rate from 15.64% to 16.06%). At 7B the difference is at most 1 sequence; at 13B at most 3. For mean extraction risk, Figure 27 shows that the per-sequence mass mean is similarly stable across beam widths. The largest relative shift in mean mass is $\sim$4.5% (7B / $\varepsilon = 5$: 0.050 vs. 0.052), and at 13B and 70B the mean masses are within $\sim$ 1–2%. Note that sometimes the mean is slightly higher at smaller beam widths B; this is due to compositional effects of the extractable set (for larger beam widths, we sometimes retain additional lower-mass sequences that surpass τ_{min}, but can bring down the mean).

For these minimal changes, using a larger beam width comes at significantly increased cost, with respect to wall-clock runtime, shown in Figure 28. To make fair runtime comparisons, we choose batch sizes for k-CBS such that the effective processed number of sequences (**effective batch size**) is similar per iteration per configured run (Table 4). For a beam width B, a single iteration processes $B \cdot$ batch size sequences. For 70B, we use an effective batch size of 400, so $B = 20, 30, 40$ use batch sizes of 20 (effective batch size 400), 13 (390), and 10 (400), respectively. This controls for GPU utilization: a fixed batch size with wider beams would increase parallelism, potentially masking the true per-step cost behind better GPU saturation. Holding the total sequences in being processed per iteration constant is a best effort at capturing how the runtime increase reflects the cost of wider beams. A wider beam retains more mass, keeping the best-beam upper bound higher for longer, so sequences may terminate at different steps across B settings. Overall, going from $B = 20$ to $B = 40$ nearly doubles wall-clock time (slightly less, due to early termination). (Note that the shading is in the reverse direction of Figures 26 and 27; we normalize runtime off of the cheapest run, which is for $B = 20$.)

Together, these results justify using $B = 20$ in our main experiments: doubling the beam width nearly doubles runtime while yielding minimal gains—a handful of marginal sequences at the extraction threshold.

[11] We initially ran these experiments using baseline k-CBS, but then re-ran them with these settings after concluding from the results in Appendix F.8 that this is the best setting for running once for edit-distance-based extraction.

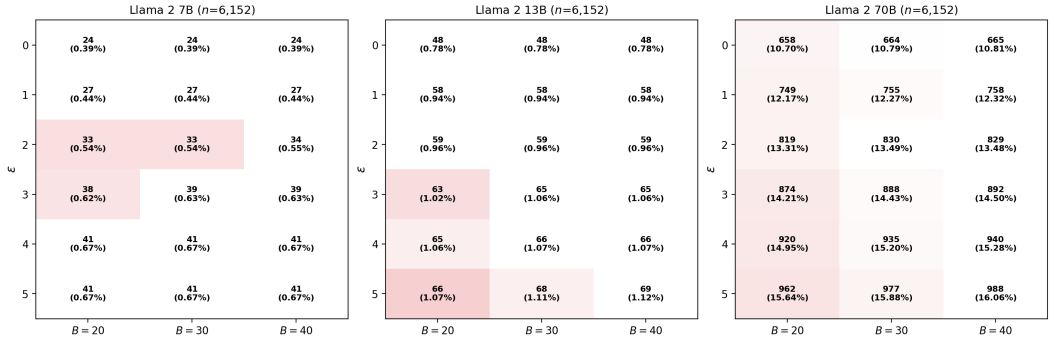


Figure 26: **Extraction counts and rates by beam width.** For *Winnie the Pooh* across LLAMA 2 model sizes, each cell shows the number of near-verbatim extractable sequences ($\hat{p}^{\text{Lev}}_{z,\varepsilon} \geq \tau_{\text{min}} = 0.001$), with the corresponding extraction rate in parentheses. For a given model's plot, red shading in a cell indicates relative decreases compared to $B = 40$ within each row (so the $B = 40$ cells are all white). In terms of absolute numbers, widening the beam from $B = 20$ to $B = 40$ yields at most 26 additional extractable sequences (LLAMA 2 70B, Lev $\varepsilon = 5$). in terms of relative differences, LLAMA 2 13B shows the biggest relative decrease from $B = 40$ to $B = 20$ for Lev $\varepsilon = 5$ (which is why the red shading changes are more pronounced).

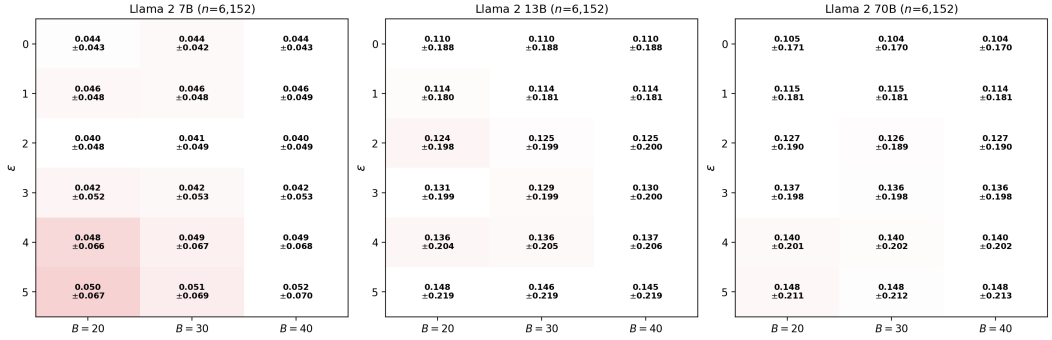


Figure 27: **Extraction mass by beam width.** For *Winnie the Pooh* across LLAMA 2 model sizes, each cell shows mean extraction risk (± 1 standard deviation) by beam width for extractable sequences. For a given model's plot, red shading in a cell indicates the relative drop in mean $\pm$ standard deviation, compared to $B = 40$ within each row (so the $B = 40$ cells are all white). The mass distribution is stable across beam widths, with relative differences under 5%. Note that this measurement is about attempting to gauge stability—large swings in mass—as opposed to seeing how much mass decreases at smaller beam widths. Mean mass can also shift downward at larger beam widths, if an expanded beam width reveals low-mass extractable sequences that smaller beam widths prune. (See, e.g., LLAMA 2 13B for Lev $\varepsilon = 5$.)

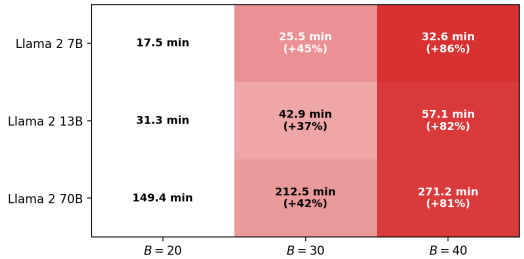


Figure 28: **Wall-clock runtime (minutes) by beam width and model size.** The effective batch size is held roughly constant across beam widths ($B\times$ batch size ≈ 400 for 70B, ≈ 200 for 7B/13B). Red cell shading indicates overhead relative to $B = 20$. Doubling the beam width approximately doubles runtime, while the extraction gains shown in Figure 26 are minimal.

F.8 Comparing baseline and ε-viability pruned k-CBS

We run a full sweep of experiments for $\mathsf{Lev}\,\varepsilon = 0, \ldots, 5$ $\mathsf{Ham}\,\varepsilon = 0, \ldots, 5$ to see how different pruning constraints impact extraction rate. We also compare these full sweeps to baseline k-CBS post-processed results. We run these experiments for *Winnie the Pooh* across the three LLAMA 2 sizes, summarized in two groups (per $\mathsf{dist} = \mathsf{Ham}, \mathsf{Lev}$) of three models (7B, 13B, 7B). The results are summarized in Figure 29. We also ran this entire suite of experiments on the negative control text, *The People's Dictator* (first three chapters); all counts across all run configurations are 0.

Overall, we find that the gains from tighter pruning are modest (tens of sequences out of thousands), so a single $\varepsilon = 5$ run can be post-processed at any threshold with only minor loss in signal. Overall, the additional sequences captured by tighter pruning tend to send near the extraction threshold τ_{min}. his is practically useful: one run covers all distance thresholds of interest, rather than requiring separate runs per ε. As expected, pruning with the Hamming distance finds fewer near-verbatim sequence than Levenshtein, as it is a stricter pruning criterion (Appendix B.3.2).

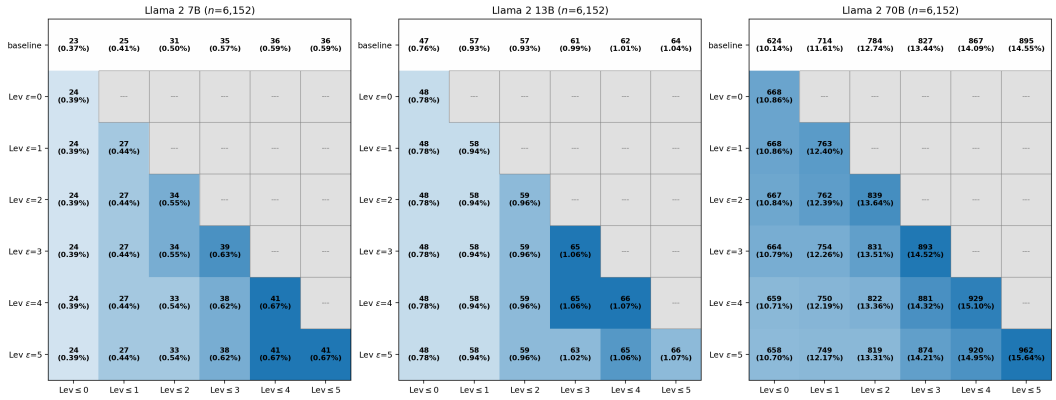
Interpreting the plots in this section. Each per-model plot should be read vertically: each column fixes a distance threshold and compares how extraction counts (and corresponding rates) vary across configured run settings. The top row is baseline k-CBS, which does not perform pruning during the search (Section 4.2 & Appendix D). In the plots in Figure 29a, subsequent rows show Lev-pruned runs at $\varepsilon = 0, \ldots, 5$; the same is true for Ham-pruned runs in Figure 29b. For a given plot, the diagonal—where pruning ε matches the threshold used to filter outputs—represents each pruning setting's "native" operating point. Gray cells indicate distance thresholds above the run's pruning ε (unavailable). Shading encodes the difference from baseline. Bluer shading indicates increasingly tighter estimates of the corresponding extraction count (finding more extractable sequence), compared to the baseline (white). Red shading would indicate identifying fewer such instances, which we do not observe in practice (i.e., pruned runs are always tighter than the baseline).

Main takeaways. Every Lev-pruned run and every Ham-pruned run finds at least as many extractable sequences as the un-pruned baseline at every applicable distance. (Though, these runs are not guaranteed to find the same sequences; see Appendix E.3, counter example).

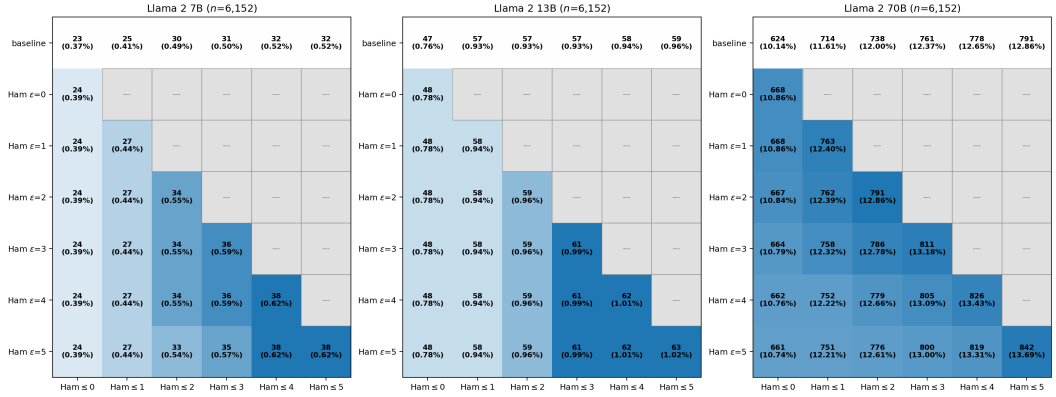
The $\varepsilon = 0$ pruned runs very closes match the verbatim probabilistic pipeline numbers. For instance, the verbatim probabilistic pipeline finds 24 (7B), 48 (13B), and 668 (70B) extractable sequences, and for $\mathsf{Lev}\,\varepsilon = 0$, counts exactly match. However, they find slightly different instances of extraction, even though the counts match. This is due to the two using different pipelines–teacher forcing for verbatim probabilistic vs. autoregressive generation with KV caching for k-CBS. Minor floating-point rounding differences can push borderline sequences across the extraction threshold in either direction. At 70B with $\mathsf{Lev}\,\varepsilon = 5$, the verbatim count ($\mathsf{Lev} \leq 0$ column) is 658, compared to 668 for verbatim probabilistic; the net difference is 10 sequences, but per-sequence comparisons reveal that k-CBS missed 11 sequences identified by the verbatim probabilistic pipeline, and k-CBS identified 1 sequence missed by verbatim probabilistic. The un-pruned k-CBS baseline finds fewer verbatim matches than both of these other runs; beam slot capacity goes to candidates with the highest mass at a given iteration, regardless of verbatim or near-verbatim match status. This crowds out verbatim matching candidates that are lower rank, in contrast to pruned $\varepsilon = 0$ runs that concentrate all beam capacity on exact matches.

Tighter pruning (i.e., setting a smaller ε) helps slightly at its "native" threshold, which is reflected on the diagonal of each plot. Results on the diagonal consistently find a few more sequences. For instance, for 70B, the 0 diagonal cell finds 668 sequences, compared to 658 in the $\varepsilon = 5$ row; similarly, the 1 diagonal cell shows 763 sequences, while the $\varepsilon = 1$ row finds 749. In general, tighter pruning frees beam slots for more viable candidates at that specific distance, revealing (relatively) $\sim$1–2% more extraction.

For the baseline runs, we also manually examine outputs that surpass τ_{min} but are not within $\mathsf{Lev}\,\varepsilon = 5$ of the target suffix. The results fall into two categories. First, the more common



(a) Levenshtein-pruned runs



(b) Hamming-pruned runs

Figure 29: **Extraction counts by configured run, based on filtering to a specific distance threshold.** For *Winnie the Pooh* and each LLAMA 2 model size, we run baseline k-CBS (Section 4.2 & Appendix D), (**a**) Lev-pruned k-CBS (Section 4.3 & Appendix E.2), and (**b**) Ham-pruned k-CBS (Section 4.3 & Appendix E.2). For each plot, read vertically, each column fixes a distance threshold and compares how extraction counts vary across configured run settings. The top row in all plots is baseline k-CBS (no pruning); subsequent rows show pruned runs at $\varepsilon = 0, \ldots, 5$, for (**a**) the Levenhstein distance and (**b**) the Hamming distance. The diagonal (where pruning ε row matches the evaluation threshold column) represents each pruning setting's "native" operating point. Gray cells indicate distance thresholds above the run's pruning ε (unavailable). Blue cell shading encodes the difference from baseline, reflecting that pruning found more extractable sequences. White shows no change. Red would indicate fewer extractable sequences (though we do not observe this).

case: there are suffixes that are effectively near-verbatim matches, but exceed Lev $= 5$. Based on visual inspection, these sequences are clearly memorized, but punctuation and formatting differences lead to inflated distances. Second, there are a handful of instances where the generated suffix is a near-verbatim match to *different* text in the ground-truth book. This type of outcome is what has encouraged others to develop suffix arrays over the training data to assess memorization (Lee et al., 2022; Carlini et al., 2023), rather than just comparing the ground-truth suffix for the given sequence to the generation.

Runtime. Baseline k-CBS is the slowest, and then the pruned run times for different ε increase with ε, as beam candidates can remain viable longer. For 70B, Lev $\varepsilon = 5$ pruning is about 10% faster in terms of wall-clock runtime compared to baseline k-CBS. We cannot make a general claim about speedups; it entirely depends on the amount of extractable sequences in the dataset being tested (with datasets with less memorization finishing faster, due to early termination from minimum probability testing and from pruning).

F.9 Examining baseline k-CBS outputs with other near-verbatim metrics

The baseline k-CBS pipeline produces candidate continuations without any distance-viability-based pruning. We post-process these with Levenshtein and Hamming distances (e.g., see Appendix F.8), but the same outputs can be scored with any similarity metric (or really any metric of interest).

To illustrate this, we compute sentence-level BLEU score (Papineni et al., 2002) between each candidate in the $B \cdot k$ k-CBS outputs and the ground-truth suffix. We follow Ippolito et al. (2023), who define approximate memorization as BLEU ≥ 0.75 over 50-token suffixes. They pick this cutoff from manual inspection of decoded text. (BLEU is computed on text, not tokenized text.) For each sequence, we sum the probability mass of all candidates whose sentence BLEU with the ground-truth suffix is ≥ 0.75. We then say that a sequence is probabilistically BLEU-extractable if this mass $\geq \tau_{\text{min}}$—the same probabilistic extraction criterion we use for edit-distance metrics.

BLEU score. **Bilingual Evaluation Understudy (BLEU)** (Papineni et al., 2002) is a precision metric originally developed for machine translation applications. In particular, **sentence-level BLEU** computes the geometric mean of n-gram precisions, typically for $n = 1, \ldots, 4$, multiplied by a brevity penalty:

$$\mathsf{BLEU} \triangleq \mathsf{BP} \cdot \exp\Big(\sum_{n=1}^{4} \frac{1}{4} \log p_n\Big), \tag{58}$$

where p_n is the fraction of n-grams in the hypothesis that appear in the reference (using modified precision with clipped counts), and brevity penalty $\mathsf{BP} = \min(1, e^{1-r/c})$ penalizes hypotheses shorter than the reference (c is the hypothesis length, r is the reference length).

The standard configuration from Papineni et al. (2002) uses $n = 1, \ldots, 4$ with uniform weights $w_n = 1/4$ (which are omitted in the equation); this is the default in all major implementations (including `nltk`, which we use, similar to Ippolito et al. (2023)). BLEU ranges from 0 to 1. A score of 0.75 indicates that roughly 75% of unigrams through 4-grams in the candidate appear in the reference; this is high n-gram overlap, but not necessarily identical text. Unlike edit distance, BLEU does not require sequential alignment: it rewards shared n-grams regardless of position, so r-eorderings, insertions, and multi-token substitutions that preserve local n-gram structure can still yield a high score. Note that since BLEU operates on text (not LLM tokens), BP is not always 1, given that 50-token suffixes may decode to different-length texts.

Application to k-CBS. For a given sequence, k-CBS returns $B \cdot k$ continuations. We sum the probability mass of all candidates with sentence-level $\mathsf{BLEU} \geq 0.75$, computed with respect to the ground-truth suffix from the training data; this is effectively our ε criterion in the lower bound computation in Equation 6. For probabilistic extraction, we then apply the same τ_{min} to produce $\hat{p}^{\mathsf{BLEU}}_{z,0.75}$. We do this for the baseline k-CBS results for *Winnie the Pooh*, which we report of Hamming and Levenshtein distance in Appendix F.8. (We also run the associated negative controls on *The People's Dictator*, first three chapters, which similarly yields no hits.)

Overall, BLEU is more permissive than $\mathsf{Lev}\,\varepsilon = 5$. As shown in the top of Table 15, in absolute counts (and therefore rates), it finds more extractable sequences at every model size. In the bottom of the table, we also examine the decomposition of extractable sequences, according to which are extractable under both metrics, under $\mathsf{Lev}\,\varepsilon = 5$ only, and under $\mathsf{BLEU} \geq 0.75$ only. Overall, the two metrics largely agree with identifying near-verbatim extractable sequences, but their respective results are not subsets of each other.

For instance, both metrics capture 889 of 1,018 unique extractable sequences for 70B. Upon manual inspection of the outputs, disagreements in both directions indicate false negatives of the respective metric, not false positives of the other. The sequences are all very close to identical to the verbatim target suffix, and fall through either check due to being below the respective filter thresholds. For the 123 sequences that are extractable under $\mathsf{BLEU} \geq$ only (i.e., that $\mathsf{Lev}\,\varepsilon = 5$ misses), the differences reflect formatting changes—newline variations,

Table 15: **Extractable sequence counts under three similarity metrics on *Winnie the Pooh*.** Each row counts sequences whose aggregated candidate mass under the given metric meets the extraction threshold τ_{min}, applied to baseline k-CBS outputs ($B = 20$). The bottom section shows the overlap between Lev $d \leq 5$ and BLEU ≥ 0.75: most extractable sequences are captured by both, but each metric finds a small number of sequences missed by the other.

	7B	13B	70B
Extractable sequences by metric			
Verbatim	23 (0.37%)	47 (0.76%)	624 (10.14%)
Lev $\varepsilon = 5$	36 (0.59%)	64 (1.04%)	895 (14.55%)
BLEU ≥ 0.75	41 (0.67%)	71 (1.15%)	1012 (16.45%)
Set overlap: Lev $\varepsilon \leq 5$ *vs.* BLEU ≥ 0.75			
Both	36	64	889
Lev $\varepsilon = 5$ only	0	0	6
BLEU ≥ 0.75 only	5	7	123

punctuation style differences, ASCII replacements, slight syntactic rearrangements, etc. These can shift many tokens (i.e., above our threshold), but preserve word-level content. These candidates often carry large probability (e.g., BLEU-captured mass that is > 0.5, where Lev distances are 6–15, but clearly are memorized upon visual inspection).

Coming from the other direction, the BLEU filter misses 6 sequences at 70B that are extractable with respect to Lev $\varepsilon = 5$. These consist of small token edits in fairly unusual text—made-up (sound-like words, onomatopoeia, etc.) from *Winnie the Pooh* with capitalization changes. Others include em-dash to double-hyphen substitutions, verb synonyms, etc. These are within 5 token edits for Levenshtein, but sufficient to break enough word-level 4-grams to drop the BLEU score below the 0.75 threshold.