

How Many Duplicates Are Left? Capture–Recapture and Sampling Inversion for Residual Duplication in FineWeb

Gary Zhang

William A. Shine Great Neck South High School

zhanggary607@gmail.com

Abstract

Pretraining corpus releases report how much duplication their pipelines removed, but we are not aware of one that estimates how much they missed. Measuring the residue is difficult because two recall problems apply at once: detectors miss duplicates they cannot see, and any tractable subsample misses duplicates whose partners were not sampled. In this paper we address both. We 1) run four detectors with distinct failure modes over a corpus and treat them as capture occasions, 2) recover what all four missed through capture–recapture, and 3) invert the sampling rate that produced the subsample, so that the result describes a full 15-trillion-token corpus rather than a convenient slice of it. The design is scored first against a planted population of known size, where it recovers 89.2% of the truth from a 44.5% detected rate while remaining a lower bound. Applied to the 14,868,862-document `sample-10BT` release of FineWeb, 4.50% of probes have a surviving near-duplicate, and capture–recapture raises that to 4.81% (95% CI 4.37–5.26). Inverting a sampling rate of one document in 1,532 then carries the figure to the corpus and gives a lower bound of 68 on the mean near-duplicate degree of a FineWeb document. Near-duplicate here means lexical overlap at Jaccard 0.35 or above. The bound is on a mean and not on a prevalence, since our design leaves the share of documents with any surviving duplicate anywhere between 7% and 73%. The survivors also carry a positional signature. The two copies of an exact-duplicate pair sit a mean 34.4% of the corpus apart against 33.3% expected under uniform sampling, which supports reading them as residue of FineWeb’s deliberate per-snapshot deduplication. A deduplicated corpus, therefore, is not a duplicate-free corpus, and the size of the gap can now be measured in one streaming pass.

1 Introduction

Deduplication is a standard stage of pretraining-data pipelines, because duplicated text distorts training. Duplicates drive memorization (Carlini et al., 2023), they inflate effective epochs on repeated content (Lee et al., 2022), and they leak evaluation data into training (Jacovi et al., 2023). Typically, corpus reports quantify the stage by what it removed. FineWeb, RefinedWeb and Dolma each describe their pipelines and the fraction of text discarded (Penedo et al., 2024, 2023; Soldaini et al., 2024), and readers are left to assume that the remainder is clean. No release we are aware of estimates the duplication that survived. The claim that a corpus is deduplicated is therefore unfalsifiable in practice.

Two distinct recall problems stand between a corpus and a correct residual number, and prior work addresses neither. The first is detector recall. Any similarity detector has a threshold, and duplicates below it are invisible, so reporting what one detector found says nothing about what all detectors would find. The second is observation recall, and it has, to our knowledge, gone unnoticed. Corpora at web scale are studied through subsamples, but duplication is a pairwise property, so a subsample destroys it quadratically: a sample retaining a fraction p of documents retains only about p^2 of the duplicate pairs. A duplication rate measured on a subsample and reported as a property of the corpus is thus too low by a factor of roughly $1/p$, which at the sampling rates web-scale work actually uses means orders of magnitude.

We solve both problems in one design. For detector recall we borrow capture–recapture, the estimator that ecology and census science use when detection is known to be incomplete (Chao, 1987; Fienberg, 1972; Hook and Regal, 1995). Each duplicate detector becomes a capture occasion, each document a unit, and the cell of documents that

every detector misses is what the estimator recovers. For observation recall we invert the sampling rate analytically. The rate at which sampled documents find a partner inside the sample is a first moment of the population degree distribution, so dividing by p recovers the mean number of duplicates a document has in the whole corpus, without any distributional assumption.

The design is applied to FineWeb (Penedo et al., 2024), whose sample-10BT release is small enough to index completely and popular enough that its properties reach many training runs. We 1) fingerprint all 14,868,862 documents in one pass, 2) query the full index with 10,000 random probes and record which of four detectors finds a near-duplicate for each probe, and 3) calibrate the whole stack against synthetic injections at three perturbation strengths.

Our contributions are the following. First, we introduce capture–recapture as a measurement instrument for corpus filters, an application for which we can find no precedent in NLP, and we calibrate it on a population whose size we set ourselves, which establishes both what the estimator recovers and where it stops working. Second, we show that sampling inversion is necessary, and that omitting it understates residual duplication in FineWeb by a factor above 1,500. Third, we report a corpus-level result: an average FineWeb document has at least 68 near-duplicates elsewhere in the corpus, a bound that holds without any assumption about how duplication is distributed. The bound is on a mean degree, however, and says nothing about what share of documents carries a duplicate, which we treat throughout as unmeasured. Fourth, we read the mechanism off a positional signature rather than assuming it; the signature supports the interpretation that the survivors are predominantly cross-snapshot pairs, left by FineWeb’s documented decision to deduplicate each snapshot independently after cross-snapshot deduplication had measurably hurt downstream models. Finally, we release fingerprints, capture histories and estimation code, so that any corpus that can be streamed once may be audited the same way.¹

2 Related Work

Deduplication and its measurement. Deduplication pipelines are built from four families of

method: exact substring removal (Raffel et al., 2020), document-level MinHash with locality-sensitive hashing (LSH) (Lee et al., 2022; Broder, 1997), SimHash (Charikar, 2002), and embedding-based semantic deduplication (Abbas et al., 2023). The downstream value is established: removing duplicates lowers memorization and leaves accuracy intact or better (Lee et al., 2022). Evaluation of these pipelines, however, measures yield rather than residue. Elazar et al. (2024) probe what large corpora contain, duplication included, and report rates found by exact n -gram matching, but without estimating what that matching misses. Corpus documentation studies such as Dodge et al. (2021) reconstruct a pipeline’s effects in depth, and pipeline ablation studies such as Parmar et al. (2024) and Longpre et al. (2024) measure how construction choices change downstream models. None of them estimates detector recall, which is the quantity that converts a found rate into a corpus property.

Contamination as the adjacent problem. Work on benchmark contamination shares our structure, since it also asks how much of something a detector failed to see (Jacovi et al., 2023; Deng et al., 2024; Riddell et al., 2024). Chang et al. (2023) infer membership at scale from an indirect signal. That literature has converged on better detectors; the complementary move, we argue, is to estimate the recall of whatever detector is in hand.

Capture–recapture. The estimator has a long history in census undercount and epidemiological prevalence work (Fienberg, 1972; Chao, 1987), reviewed with its failure modes by Hook and Regal (1995). The closest precedent outside those fields is software inspection, where multiple imperfect inspectors serve as capture occasions to estimate undetected defects (Petersson et al., 2004). To our knowledge the method has not been applied to text corpora. Deduplicated corpora are the difficult case for this transfer. The duplicates that remain are the ones a competent detector already failed to catch, which is exactly the situation an estimator built for undetected populations addresses.

3 Two Recall Problems

Let the corpus contain M documents. Build the near-duplicate graph on them, joining two documents when their similarity exceeds the threshold, and write $K - 1$ for the degree of a uniformly chosen document, so that K counts the document to-

¹Code and artifacts: <https://github.com/garyzhang1006/residual-duplication-audit>

gether with its near-duplicate neighbors. We work with degree rather than with cluster membership, and the choice is deliberate. Near-duplication at a fixed Jaccard threshold is not transitive, so the corpus does not partition into duplicate families, and any quantity defined through such a partition would depend on how one closed the relation. Degree needs no such choice. The quantity a corpus card should report is thus $\mathbb{E}[K - 1]$, the mean number of near-duplicates a document has. A related quantity is $\Pr(K \geq 2)$, the share of documents with at least one. Exact-hash identity, by contrast, is a genuine equivalence relation, so exact duplicates do partition into clusters, and we use cluster language only there.

A study of the corpus through a uniform sample of $m = pM$ documents observes neither. A sampled document sees a partner only if at least one of its neighbors was also sampled, which happens with probability $1 - (1 - p)^{K-1}$. Writing r for the observed within-sample rate,

$$r = \mathbb{E}[1 - (1 - p)^{K-1}] \leq p \mathbb{E}[K - 1]. \quad (1)$$

The inequality is Bernoulli’s and it holds for every p and every K , so r/p is a lower bound on $\mathbb{E}[K - 1]$ unconditionally rather than in a small- p limit. Equation 1 is the observation-recall correction, and it is model free: only the first moment enters, and nothing is assumed about the shape of the degree distribution. Documents with degree large enough to saturate the exponential contribute less than proportionally, which is what makes the direction of the bound safe.

The second correction sits inside r itself. A study measures $\hat{r}$, the rate at which its detectors flag a partner, and $\hat{r} \leq r$ whenever detection is imperfect. Capture–recapture recovers r from the overlap structure of several detectors. The two corrections compose, since the first acts on the sampling design and the second on the instrument, and neither is identifiable from the other’s data.

4 Method

Corpus and fingerprints. All 14,868,862 documents of `sample-10BT` are streamed once. Per document we compute an exact hash of the first 8,000 normalized characters, a 64-permutation MinHash signature over word 5-grams, and a 64-bit SimHash over character 4-grams. Indexes are rebuilt in memory from the fingerprints, so the design needs one pass and stores no text beyond the 500-character heads used for verification. Peak memory

stays under 16 GB and the full run completes in 6.3 hours on commodity CPU. Every random choice derives from fixed seeds recorded in the released code, so the run reproduces exactly.

Sampling rate. `sample-10BT` is a uniform random sample of whole documents from the FineWeb release described by [Penedo et al. \(2024\)](#), which draws on 96 Common Crawl snapshots running from CC-MAIN-2013-20 to CC-MAIN-2024-18. Fixing the denominator needs care, however, because the public dataset has since grown to 110 snapshots and its headline row count now covers dumps that postdate the sample. Summing the per-snapshot row counts over exactly those first 96 snapshots gives $M = 2.277 \times 10^{10}$ documents, against 2.589×10^{10} for the corpus as it now stands. We confirmed that `sample-10BT` contains no document from a post-2024-18 snapshot, so the smaller figure is the correct denominator, and it yields $p = 6.529 \times 10^{-4}$, or one document in 1,532. Using the current row count instead would inflate every corpus-level number in this paper by 13.7%. The entire inversion rests on the uniformity of the sample as well as on M , and we therefore test uniformity empirically in §5.5 rather than relying on the dataset card.

Four capture occasions. For each probe document we record which detectors find at least one other document that qualifies as a near-duplicate. Detector E is exact-hash identity. Detectors A and B split the MinHash signature into two disjoint 32-permutation families, each with its own 16-band LSH index, and each verifies candidates against its own half-signature at estimated Jaccard ≥ 0.35 . Sixteen bands over 32 permutations puts two rows in a band, which places the soft LSH threshold near $(1/16)^{1/2} = 0.25$, below the verification threshold of 0.35. The band step therefore proposes candidates a little below the estimand, and the verification step does the cutting. The interval between the two is the boundary region characterized in §5.8. Because the families use disjoint random permutations, they are independent draws of the same detection process conditional on a document’s true similarity to its nearest neighbor, and that conditional independence is what identifies the unobserved cell. Detector S buckets documents by the top 16 bits of the SimHash, accepts a candidate within Hamming distance 8, restricts itself to documents of at least 300 characters, and verifies every hit with a character-shingle Jaccard of at least

0.2 so that SimHash collisions cannot inflate captures. The single-block bucketing is a deliberate weakness rather than an oversight. A guaranteed Hamming-8 search would need nine blocks by the pigeonhole argument, whereas one 16-bit block finds a pair at distance d only when all d differing bits fall outside it, which happens for 8.5% of pairs at $d = 8$. Capture-recapture tolerates such a channel, since an occasion may have any recall so long as its misses are accounted for, and the consequence is the small S-only cell in Table 1 rather than bias.

Candidate caps. Both index lookups cap a bucket at 2,000 candidates. Boilerplate collides heavily in LSH, and without a cap a handful of probes would dominate the runtime of the whole pass. A capped bucket may exclude a genuine partner, but it can only remove captures and never create them, which makes every rate we report a lower bound for this reason as well as the others. The original run did not count how often a cap was reached, and we report that as an instrumentation gap rather than as a threat, since the direction of the effect is fixed. The frequency still matters for judging whether the resulting bias is negligible, however, so the released pipeline now counts cap hits per detector and writes them alongside the capture histories. Any rerun will report the number this run does not.

Estimation. The 10,000 capture histories populate a 2^4 table (Table 1). Three estimators of r are reported. The naive rate counts probes flagged by any detector. The Chapman-corrected Lincoln-Petersen estimator uses only detectors A and B, the pair whose conditional independence follows from the construction rather than from an assumption. The Chao lower-bound estimator uses singleton and doubleton capture counts and requires no model selection (Chao, 1987), following the Good-Turing argument that the frequency of once-seen units carries the information about unseen ones (Good, 1953). For completeness we also fit Poisson log-linear models over the observed cells, with all subsets of up to two pairwise interactions, and read the unobserved cell from the fitted intercept (Fienberg, 1972). Among these we select both by the Akaike information criterion (AIC) and by AIC weighting (Buckland et al., 1997). Every estimator carries a 2,000-draw bootstrap over probes that reruns the full estimation, model selection included, on each resample.

History	n	History	n
none	9,550	A+B	240
A only	72	A+B+S	82
B only	48	B+S	1
S only	3	E+A+B	1
		E+A+B+S	3

Table 1: Capture histories for 10,000 probes. E = exact hash, A/B = the two disjoint MinHash families, S = verified SimHash. The large A+B overlap and the near-symmetric A-only and B-only cells are the pattern independent channels should produce.

Injection calibration. A 6,000-document pool is reserved during streaming. We generate synthetic near-duplicates by deleting 5%, 15% or 30% of words with occasional local reordering, and run the full detection stack on 2,000 injections per strength. The recovery rates define the instrument’s scope.

5 Results

5.1 What the sample shows

Direct detection flags 450 of 10,000 probes, a rate of 4.50% with bootstrap 95% CI 4.10–4.90. The Chapman estimator on the two MinHash families gives 4.58% (4.16–4.98), and Chao gives 4.81% (4.37–5.26). Table 2 collects the estimators with their intervals. The three agree to within a third of a percentage point. Such agreement indicates that the detector stack is close to saturation within its similarity scope, so the residual uncertainty about the sample is small.

The detector marginals explain where that agreement comes from. MinHash family A captures 398 probes and family B captures 375, and 326 probes are captured by both. Exact hashing captures 4 and verified SimHash captures 89, of which only 3 are captured by SimHash alone. The design therefore has two strong, symmetric occasions and two weak ones, which is enough for Chao and Lincoln-Petersen, but not enough for a four-way log-linear fit.

5.2 Validation on a known population

Before trusting the estimator on a population we cannot see, we ran it on one we constructed. Each injection batch contains exactly 2,000 planted near-duplicates, so the true population is known and the estimators can be scored against it (Table 3).

At 5% word deletion the detectors find 1,990 of 2,000 and nothing is left to estimate, so every method returns the same answer. The informative

Estimator	Rate	Bootstrap 95% CI	Corpus $\mathbb{E}[K-1] \geq$
Detected (naive)	4.50%	4.10–4.90	68.9
Lincoln–Petersen	4.58%	4.16–4.98	70.1
Chao (primary)	4.81%	4.37–5.26	73.7
Chao (bias-corr.)	4.81%	4.37–5.26	73.7
Chao, dep.-adjusted	5.40%	—	82.7

Table 2: Within-sample near-duplicate rate by estimator, with 2,000-draw bootstraps that rerun the full estimation on each resample, and the corpus-level bound each rate implies through Equation 1. The upper block is what we report. The final row divides the Chao rate by the 89.2% recovery it achieved on the planted population of §5.2, which is our best available handle on detector dependence and heterogeneity; §5.7 explains why we treat it as a sensitivity rather than as the headline. Two log-linear variants are unstable under resampling and appear in Appendix B.

case is 15% deletion, where direct detection finds 890 and misses more than half of the planted population. Chao recovers 1,784, which is 89.2% of the truth, and Lincoln–Petersen recovers 1,332, or 66.6%. Both stay below the true value, as the theory requires of a lower bound under heterogeneous catchability, and Chao doubles the naive count in the right direction. At 30% deletion the instrument is blind, only six injections are found at all, and no estimator can repair a design that captured almost nothing.

One table thus establishes both the credibility of the corpus estimates and their scope. Where detection is partial, Chao closes most of the gap; where detection has collapsed, it correctly cannot. The scope of that claim, however, is the perturbation family we planted. Real web duplicates also arise through templating, boilerplate edits, partial copying, truncation and markup change, so the 89.2% figure calibrates the estimator under word deletion with local reordering rather than measuring its recall on FineWeb’s own duplication process. What the table licenses is the direction and the rough magnitude of the correction, not a transferable recall figure.

5.3 Which estimator to trust

The log-linear route fails this test, and Figure 1 shows why we exclude it. AIC selection gives 4.64% at the point estimate, but a bootstrap interval of 4.26–8.85, because five cells hold fewer than five probes and the selected model changes under resampling. AIC weighting, the standard remedy for

Deletion	Truth	Detected	Lincoln–P.	Chao
5%	2,000	1,990 (99.5%)	1,990 (99.5%)	1,991 (99.5%)
15%	2,000	890 (44.5%)	1,332 (66.6%)	1,784 (89.2%)
30%	2,000	6 (0.3%)	15 (0.8%)	21 (1.1%)

Table 3: Estimators scored against a known population of 2,000 planted near-duplicates per batch. At 15% deletion, where detection misses more than half, Chao recovers 89.2% of the truth from a 44.5% detected rate while remaining a lower bound.

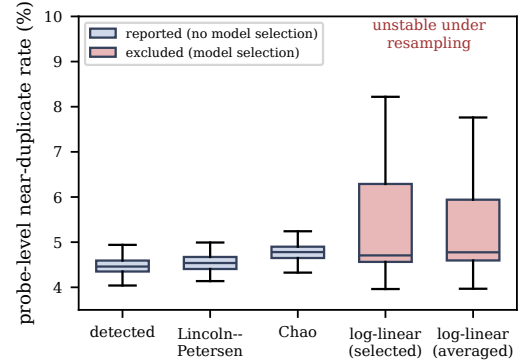


Figure 1: Bootstrap distributions over 2,000 probe resamples. The three estimators on the left are stable and mutually consistent. Both log-linear routes, including AIC model averaging, inherit the instability of sparse multi-detector cells and are excluded from the headline.

selection instability, does not help; Appendix B gives both variants with their intervals and the across-model spread. Reporting a log-linear headline here would be reporting an artifact of sparse overlap cells. We therefore take Chao as the primary estimator, since it needs no model selection and is a lower bound under heterogeneous catchability, and Lincoln–Petersen as a check whose independence assumption can at least be derived from the construction. That derivation is conditional on a document’s latent similarity to its nearest neighbor. Marginal dependence across documents survives it, so both estimators should be read as conservative lower bounds rather than as unbiased estimates.

5.4 The survivors are cross-snapshot

Scanning the full corpus, 51,972 documents share their first 8,000 normalized characters byte-for-byte with an earlier document, and 103,552 documents, or 0.70% of the corpus, sit in an exact-duplicate cluster of size two or more. The cluster spectrum is almost pure pairing: 51,279 clusters of size two,

281 of size three, and a tail reaching a single cluster of 36.

Where those pairs sit in the stream carries information about their origin, but how much it carries depends on how the stream is laid out. We measured the layout directly, by reading the snapshot label of 40,000 documents in runs of 100 at 400 positions spaced evenly through `sample-10BT`. The label changes 266 times in 39,600 adjacent pairs, so one snapshot holds the stream for a mean run of 149 documents, and 139 of the 400 runs stay inside a single snapshot. Snapshots interleave at that fine grain, and they do not run the length of the corpus. The sampled documents of a given snapshot span a median 6.3% of the stream and at most 8.8%, and the regions appear in an order unrelated to crawl date. Position is therefore a proxy for snapshot at the scale of a few percent of the stream, not at the scale of a block of twenty. A within-snapshot duplicate pair would fall inside one of those regions, contributing separations under 9%, so failed within-snapshot deduplication would pile mass into the left tail of the histogram. Figure 2 shows no such pile. Over the 51,279 clusters of size exactly two, the observed mean separation is 34.4% of the corpus span against 33.3% expected under uniform document sampling, and the observed median is 30.3% against 29.3% expected. Only 1.4% of pairs fall within 1% of the span.

Two readings follow. First, the histogram supports the interpretation that the surviving exact duplicates are predominantly cross-snapshot residue, with the 3% excess over the uniform curve reading as depletion of the same-snapshot diagonal by a deduplication stage that worked. This matches FineWeb’s documented design, in which each snapshot was deduplicated independently after global deduplication had been tried and rejected for harming downstream performance. Second, and needed for everything in §5.6, the agreement with the uniform curve is evidence that `sample-10BT` samples documents rather than files or shards. Shard-level sampling would concentrate pairs and produce a visibly different histogram.

The measured region width also bounds how much within-snapshot residue the histogram tolerates. Mixing a share f of within-snapshot pairs, each uniform inside a region of width 0.09 and so averaging 0.03, with cross-snapshot pairs averaging the uniform 0.333, gives an overall mean of $0.333 - 0.30f$. The observed mean of 0.344 lies above the uniform value rather than below it, so the

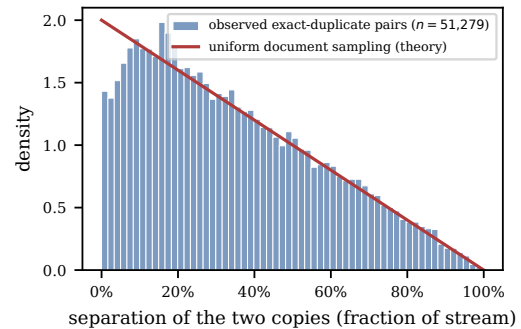


Figure 2: Separation between the two copies of each exact-duplicate pair, as a fraction of the corpus stream. The observed distribution tracks the triangular density expected under uniform document sampling, with a slight shift away from zero. One snapshot spans at most 8.8% of the stream (§5.4), so surviving within-snapshot pairs would show as mass in the left tail, and the tail is if anything thinner than uniform.

model returns no positive f at all. The margin is wide: over 51,279 pairs the standard error of the mean separation is 0.0010, and even a one percent within-snapshot share would sit thirteen standard errors below what we see.

Position nevertheless remains a proxy, and a direct measurement exists. FineWeb rows carry a `dump` field naming the snapshot each document came from, so storing one label per fingerprint would replace the separation histogram with the fraction of exact-duplicate pairs whose two members disagree on that label. The run reported here did not store it, but the released pipeline now does, so the next application of this audit can report the direct fraction. We prefer that number to ours, and we report the proxy as a proxy.

5.5 Is the sample uniform?

The inversion multiplies by $1/p$, a factor above 1,500, so uniformity of the sample carries more weight here than a design assumption usually does. The separation histogram of §5.4 tests it positionally, which leaves open any non-uniformity by document length, domain or quality that keeps the separations intact. The release publishes the snapshot label and the token count for every row, however, and the 40,000 labelled documents described above supply both.

Composition by snapshot is the sharper test, since crawl date carries much of what varies across the corpus. We group the 96 snapshots by crawl year and compare each year’s share of the sample

against its share of the release. Ten of the twelve years agree within a 95% interval from a bootstrap that resamples whole runs rather than documents, and neither of the two that miss is off by as much as a factor of two (Appendix E). Length agrees more closely: the sampled documents average 700 GPT-2 tokens against a release mean of 659, so the sample runs about 6% long, which we treat as an upper estimate of the length skew because the published token total is rounded. Neither check covers domain or quality score, and 400 runs resolve a single snapshot only coarsely. What they do rule out is a sampler distorted by a factor rather than by a tenth, the regime that would matter against a correction of 1,500.

5.6 From the sample to the corpus

Applying Equation 1 with $p = 6.529 \times 10^{-4}$ converts the within-sample rates into corpus quantities: the naive rate gives $\mathbb{E}[K - 1] \geq 68.9$, Lincoln-Petersen gives ≥ 70.1 , and Chao gives ≥ 73.7 . An average FineWeb document, then, has at least 68 near-duplicates elsewhere in the corpus. That should be read as a lower bound on a mean degree and not as a statement about a typical document, since a mean over a heavy-tailed degree distribution can be carried by a minority of documents in large clusters. The same inversion applied to exact duplicates gives $\mathbb{E}[K - 1] \geq 10.7$, so the average document also has at least ten byte-identical copies of its first 8,000 characters spread across the corpus.

The mean does not by itself pin down how many documents are affected. Since $\Pr(K \geq 2) \cdot \mathbb{E}[K - 1 \mid K \geq 2] = \mathbb{E}[K - 1]$, converting a mean of 68 into a prevalence would require the mean degree among documents that have any neighbor at all, and our binary capture design does not measure that quantity. The resulting range is wide. A conditional mean of 95, which one surviving copy per snapshot across 96 snapshots would give, implies a prevalence of 72.5%, while conditional means of 300 and 1,000 imply 23.0% and 6.9%. The whole range is compatible with our data, so we report $\mathbb{E}[K - 1]$ as the result and treat prevalence as unmeasured.

Two pieces of evidence favor the lower end of that range without settling it. The exact-duplicate spectrum puts at most 5% of its own first moment in clusters with three or more co-sampled members, so the observable heavy tail is not what carries the mean. Independently, the corpus authors report

that deduplicating across snapshots removed more than 90% of the base filtered data in the oldest snapshots. Community measurements on FineWeb-Edu find that roughly 75% of documents share text with another document.² A high prevalence with moderate degree fits both observations, whereas a low prevalence with degree in the thousands fits neither. Recording neighbor counts rather than binary capture would settle it in one additional pass.

5.7 How far the inversion can be pushed

Three quantities control the corpus-level numbers. The sampling rate enters linearly, so an error of 10% in the corpus size moves $\mathbb{E}[K - 1]$ by 10% in the same direction. It is also the one sensitivity we can check rather than assume, and the check matters: summing per-snapshot row counts over the 96 snapshots the sample was drawn from, rather than over the corpus as it stands today, changes the answer by 13.7%.

Saturation makes Equation 1 a bound, and the size of the gap depends on where the degree mass sits. At the per-snapshot ceiling of 96 the linear approximation overstates the detection probability by 3.0%, so the bound is nearly tight for exactly the degrees the mechanism predicts, while the gap widens to 26.5% at degree 1,000 and to 97.2% at degree 55,000, the largest cluster our spectrum implies. The true mean therefore sits above our bound by an amount that grows with the weight of the tail.

Detector dependence is the third quantity, and the planted populations of §5.2 let us put a number on it rather than only a direction. At 15% deletion, where the same four detectors face the same latent-similarity correlation they face on the corpus, Chao recovers 89.2% of a known population and Lincoln-Petersen recovers 66.6%. Read the Chao shortfall as an estimate of the combined dependence and heterogeneity bias, and divide through by it. That moves 4.81% to 5.40% and $\mathbb{E}[K - 1]$ from 73.7 to 82.7, which Table 2 carries as its final row. We do not adopt it as the headline, however, since a planted population of single twins is neither the mixture of degrees the corpus presents nor a sample of more than one perturbation family. It does bound the discussion: the direction is up, and the

²Reported in the public discussion of the dataset repository (fineweb-edu, discussion 7 on the HuggingFace hub), where a dataset author confirms that the release deduplicates each snapshot individually.

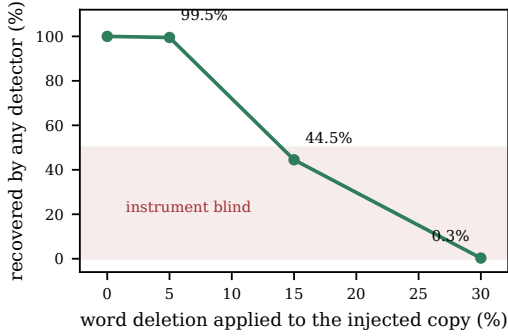


Figure 3: Recovery of synthetic near-duplicates as a function of perturbation strength, 2,000 injections per point. Sensitivity collapses between 15% and 30% word deletion, which fixes the similarity scope of every estimate we report.

magnitude where we can check it is 11%, not a factor.

The tail is real. Fitting a single cluster size to the exact-duplicate spectrum through the S_2/S_1 ratio gives $\kappa = 11.6$. That model then underpredicts the higher cells badly, by a factor of 2.6 at three co-sampled members and 59 at four, and it puts clusters of five and six at effectively zero probability (Appendix D). Cluster sizes are heavy-tailed. That is what web duplication should look like, and it is what makes $\mathbb{E}[K - 1] \geq 68$ conservative.

5.8 What the instrument can see

Injection recovery is 99.5% at 5% word deletion, 44.5% at 15%, and 0.3% at 30% (Figure 3). The estimand is therefore near-duplication at Jaccard similarity of roughly 0.35 and above, and the instrument says nothing about heavier paraphrase. Every claim above is conservative as a result, since duplicates too heavily rewritten for any of our detectors are counted as absent.

Recovery is only half of a calibration, and the other half matters here because capture fires on a noisy statistic estimated from 32 permutations. Appendix C works through the false-positive side and finds it negligible: threshold noise on unrelated documents does not manufacture captures. What it does instead is blur the boundary between roughly 0.25 and 0.35 in true similarity, which moves where the estimand sits by a little.

6 Discussion

The result changes what a deduplication report can be read to mean. FineWeb’s pipeline is not broken, and the duplicates that remain follow from

a deliberate choice: the authors tried deduplicating across snapshots, found that it removed high-quality text and hurt downstream performance, and switched to per-snapshot deduplication. The corpus card, however, does not state the size of the residue that choice leaves. A practitioner who reads that FineWeb is deduplicated has no way to anticipate an average of 68 near-duplicates per document.

The two corrections differ enormously in size. Detector recall moves the number by seven percent relative, from 4.50% to 4.81%, for which four detectors and a 2^4 table are the price. Observation recall, by contrast, moves it by a factor above 1,500.

That asymmetry suggests a reporting convention that would cost almost nothing. A release reporting residual duplication should give four things: the rate its detectors found, the fraction p of the corpus they ran on, the exact release the denominator M came from, and the similarity threshold at which a pair counts as duplicate. With those four numbers a reader can apply Equation 1 without access to the corpus. Without them a published rate cannot be compared across corpora, or across versions of one corpus. Pinning the release matters: FineWeb grew from 96 to 110 snapshots between the paper and the dataset card, and taking the denominator from the card would have inflated every corpus number here by 13.7%. A corpus that ships a residual estimate this way makes a falsifiable claim about its own cleanliness, which is the property none of the releases we surveyed has.

For a practitioner the finding has immediate uses. Anyone auditing duplication in a subsample now has a reason to expect their own rate to understate the corpus, and the sampling rate they already know converts one into the other. Anyone choosing between per-snapshot and cross-snapshot deduplication can weigh the downstream loss the FineWeb authors reported against the residue their alternative leaves, where previously only the first had a number. And our released fingerprints identify which `sample-10BT` documents have a surviving near-duplicate, so memorization work can compare the two halves without recomputing anything.

The design should generalize, although one corpus and one deduplication strategy leave the broader claim resting on a single case. It needs one streaming pass, no stored text and no GPU. Running it across RefinedWeb and Dolma, which both deduplicated globally, would turn residual duplication into a property of pipeline design.

Limitations

The corpus-level claims in §5.6 rest on the sampling inversion, and the inversion rests on two conditions. The first is uniformity. `sample-10BT` must be a uniform sample of documents, which we test two ways and find supported both times. The first test, in §5.4, is positional. It would not detect a non-uniformity in sampling probability by document length, domain or quality that leaves the separation distribution intact. This matters more than a usual design assumption, because p multiplies the headline by a factor above 1,500. A sampler favoring longer or higher-scoring documents, even by a modest margin, would move the bound further than any other quantity reported here. The stronger test compares document covariates between the sample and the full release, and §5.5 reports what that comparison shows on the snapshot label and the token count. Those are the two covariates the release publishes for every row. Domain and quality score are not covered, and they remain untested. The second condition is the corpus size, since p enters every corpus-level number linearly. M is computed by summing published per-snapshot row counts over the 96 snapshots the sample was drawn from, which is verifiable, but not something our own data can confirm. Taking the current headline row count of the still-growing dataset instead would inflate every corpus number here by 13.7%.

Prevalence is the claim we cannot make. As §5.6 sets out, $\Pr(K \geq 2)$ follows from $\mathbb{E}[K - 1]$ only through the conditional mean degree, which our binary capture design never observes, and the plausible range spans 7% to 73%. Readers should take $\mathbb{E}[K - 1] \geq 68$ as the result of this paper. It should not be converted into a share of the corpus. An earlier version of this work reported such a share, derived from a ceiling of one surviving copy per snapshot. Our own spectrum contains a cluster of 36 co-sampled members, implying roughly 55,000 copies corpus-wide, which violates that ceiling and invalidates the derivation.

The capture–recapture side has its own limits. Detectors A and B are described here as independent capture occasions, but that holds only conditionally, given a document’s true similarity to its nearest neighbor. Across documents, all content-based detectors correlate through that latent similarity. The marginal dependence is therefore positive, and both Chao and Lincoln–Petersen are bi-

ased downward, which compounds with the lower-bound direction of Equation 1. Detector S contributes little independent information, with only three probes captured by SimHash alone, so the effective design has closer to two occasions than four.

Our specificity check in §5.8 runs on the 6,000-document held-out pool rather than on the corpus index. A probe against the full index draws candidates that already share an LSH band, so their similarity distribution is shifted upward relative to a random pool pair. The pool measurement bounds noise on unrelated text. It does not characterize the boundary region between roughly 0.25 and 0.35, which is where the threshold does its real work. Repeating the check against the full index, which our released fingerprints permit, would close that gap.

The exact-hash detector keys on the first 8,000 normalized characters, which creates a threat to §5.4 beyond simple miscounting. Templated sites can share a long prefix across unrelated articles. Such pairs would also scatter across snapshots and reproduce the positional signature we read as evidence of cross-snapshot survival. Among the 6,000 held-out pool documents, prefix agreement decays fast: three pairs share the first 100 characters, two share 200, and none share 500 or more out of nearly 18 million pairs. The evidence is real and weak at once, however, because a random sample of a corpus separates template siblings just as efficiently as it separates duplicates, so the pool cannot contain many of either. Settling the question needs a check against the full index on document text, and our released fingerprints do not carry it.

The scope, finally, is one corpus at one time, and the deduplication design studied here is a single one. The general claim that deduplication reports understate residual duplication is therefore supported by one case rather than by a comparison across pipelines. We measure near-duplication at Jaccard 0.35 and above and say nothing about paraphrase, which a semantic detector added as a fifth occasion could reach, and which matters more each year as generated text enters web crawls. Duplicates are planted by word deletion with local reordering, so the 89.2% recovery describes the estimator under that family and not under templating, boilerplate edits, truncation or markup change. The calibration also has a resolution gap. Recovery falls from 44.5% at 15% deletion to 0.3% at 30%, and no point in between was measured, so

the sensitivity boundary that scopes every claim in this paper is located only to within that interval. Residual duplication is not connected here to any downstream training outcome either. The paper measures a corpus property without establishing what it costs.

Ethics Statement

This work analyzes a public dataset and releases no text beyond what FineWeb already publishes. The released artifacts are hashes, capture histories and code. Identifying which documents are duplicated could in principle assist reconstruction of corpus membership, but the fingerprints are one-way and the underlying corpus is already open.

The quantity we measure has downstream stakes beyond documentation practice. Repeated text drives verbatim memorization, the mechanism by which personal information and copyrighted passages reappear in model output (Carlini et al., 2023; Lee et al., 2022). It is also the route by which evaluation material leaks into training sets and inflates reported capability (Jacovi et al., 2023). A residue the corpus card does not mention is a risk that no model builder can price. Measuring it, as this paper does, is the step that makes the risk visible. Reducing it is a separate decision with its own cost, since the FineWeb authors found that removing cross-snapshot duplication hurt the models they trained.

Acknowledgments

I thank the four WaC-13 reviewers, whose reports prompted the sampling-uniformity analysis in §5.5, the dependence-adjusted row of Table 2, and a good deal of tightening elsewhere.

References

- Amro Abbas, Kushal Tirumala, Dániel Simig, Surya Ganguli, and Ari S. Morcos. 2023. SemDeDup: Data-efficient learning at web-scale through semantic deduplication. In *ICLR 2023 Workshop on Multimodal Representation Learning*.
- Andrei Z. Broder. 1997. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences*, pages 21–29.
- Steven T. Buckland, Kenneth P. Burnham, and Nicole H. Augustin. 1997. Model selection: An integral part of inference. *Biometrics*, 53(2):603–618.
- Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramèr, and Chiyuan Zhang. 2023. Quantifying memorization across neural language models. In *The Eleventh International Conference on Learning Representations*.
- Kent K. Chang, Mackenzie Cramer, Sandeep Soni, and David Bamman. 2023. Speak, memory: An archaeology of books known to ChatGPT/GPT-4. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7312–7327.
- Anne Chao. 1987. Estimating the population size for capture-recapture data with unequal catchability. *Biometrics*, 43(4):783–791.
- Moses S. Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pages 380–388.
- Chunyu Deng, Yilun Zhao, Xiangru Tang, Mark Gestein, and Arman Cohan. 2024. Investigating data contamination in modern benchmarks for large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8706–8719.
- Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. 2021. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1286–1305.
- Yanai Elazar, Akshita Bhagia, Ian Magnusson, Abhilasha Ravichander, Dustin Schwenk, Alane Suhr, Pete Walsh, Dirk Groeneveld, Luca Soldaini, Sameer Singh, et al. 2024. What’s in my big data? In *The Twelfth International Conference on Learning Representations*.
- Stephen E. Fienberg. 1972. The multiple recapture census for closed populations and incomplete 2^k contingency tables. *Biometrika*, 59(3):591–603.
- Irving J. Good. 1953. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3-4):237–264.
- Ernest B. Hook and Ronald R. Regal. 1995. Capture-recapture methods in epidemiology: Methods and limitations. *Epidemiologic Reviews*, 17(2):243–264.
- Alon Jacovi, Avi Caciularu, Omer Goldman, and Yoav Goldberg. 2023. Stop uploading test data in plain text: Practical strategies for mitigating data contamination by evaluation benchmarks. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5075–5084.
- Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2022. Deduplicating training data makes language models better. In *Proceedings*

of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 8424–8445.

Shayne Longpre, Gregory Yauney, Emily Reif, Katherine Lee, Adam Roberts, Barret Zoph, Denny Zhou, Jason Wei, Kevin Robinson, David Mimno, and Daphne Ippolito. 2024. A pretrainer’s guide to training data: Measuring the effects of data age, domain coverage, quality, and toxicity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3245–3276.

Jupinder Parmar, Shrimai Prabhumoye, Joseph Jennings, Bo Liu, Aastha Jhunjhunwala, Zhilin Wang, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. 2024. Data, data everywhere: A guide for pretraining dataset construction. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 10671–10695.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben Al-lal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro von Werra, and Thomas Wolf. 2024. The fineWeb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. 2023. The refinedWeb dataset for falcon LLM: Outperforming curated corpora with web data, and web data only. *Advances in Neural Information Processing Systems*, 36.

Håkan Petersson, Thomas Thelin, Per Runeson, and Claes Wohlin. 2004. Capture-recapture in software inspections after 10 years research: Theory, evaluation and application. *Journal of Systems and Software*, 72(2):249–264.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Martin Riddell, Ansong Ni, and Arman Cohan. 2024. Quantifying contamination in evaluating code generation capabilities of language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14116–14137.

Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, et al. 2024. Dolma: An open corpus of three trillion tokens for language model pretraining research. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15725–15788.

A Reproducibility

Stage A streams `sample-10BT` once and writes fingerprints; stages B through D read only those fingerprints. The full run takes 6.3 hours on a single commodity CPU with peak resident memory under 16 GB, and it needs no GPU and no network access after the corpus is available locally. Two seeds fix every random choice, 20260811 for the corpus pass and 20260812 for estimation. The first governs the MinHash permutation family, the 10,000-probe sample, the 6,000-document injection reservoir, and the perturbations applied to injected copies; the second governs the 2,000-draw bootstrap. `PYTHONHASHSEED` is pinned to 0 by a re-exec at process start, so string hashing is stable across machines, which the stage-resume logic depends on. Re-running `estimate.py` against the released capture histories reproduces every number in Tables 1, 2, 3 and 4 without touching the corpus, and `figures.py` regenerates the figures from the same artifacts. The repository holds the pipeline, the capture histories, the injection results, the bootstrap draws, and the estimation code.

B Log-linear estimators

Table 4 gives the two log-linear variants that §5.3 excludes. We fit Poisson log-linear models over the observed cells of Table 1 with all subsets of up to two pairwise interactions, reading the unobserved cell from the fitted intercept (Fienberg, 1972), and we select among them both by AIC and by AIC weighting (Buckland et al., 1997). Both point estimates sit inside the range the stable estimators cover, and both bootstrap intervals run to roughly 8.7%, nearly double the point estimate. The cause is sparsity rather than a defect of the models: five cells of the 2^4 table hold fewer than five probes, so a resample can move the selected model between specifications whose implied unobserved cells differ by an order of magnitude. AIC weighting averages over that instability instead of removing it, and the across-model standard deviation of the unobserved cell is 51.5 against a point estimate of 38.8. A design with three strong occasions rather than two would support this family of estimators; ours does not.

C Specificity of the capture rule

Recovery is only half of a calibration, and the other half matters here because capture fires on a noisy statistic. A 32-permutation half-signature

Estimator	Rate	95% CI	$\mathbb{E}[K-1]$
Log-linear, AIC-sel.	4.64%	4.26–8.85	71.1
Log-linear, AIC-avg.	4.89%	4.28–8.64	74.9

Table 4: Log-linear variants, excluded from the headline. Bootstrap intervals reach nearly twice the point estimate because model selection is unstable over sparse overlap cells.

estimates Jaccard with a standard error near 0.084, so a document with many candidates gets many chances to clear 0.35 by chance. We measured the resulting false-positive rate two ways. Analytically, capture requires at least 12 of 32 minima to agree, which for a pair with true Jaccard J has probability $\Pr(\text{Bin}(32, J) \geq 12)$: exactly zero at $J = 0$, 2.1×10^{-8} at $J = 0.05$, and 3.3×10^{-5} at $J = 0.10$. Empirically, we compared all 17,997,000 pairs among the 6,000 held-out pool documents under both MinHash families, 36 million comparisons in total, and the rule captured none of them. Threshold noise on unrelated documents does not manufacture captures. It blurs the boundary between roughly 0.25 and 0.35 in true similarity, which shifts where the estimand sits by a little and leaves unrelated text outside it.

D Exact-duplicate cluster spectrum

Table 5 gives the spectrum behind the heavy-tail claim of §5.7.

Co-sampled members j	Observed clusters	Single-size prediction	Ratio
1	14,765,310	14,765,310	1.00
2	51,279	51,279	1.00
3	281	108	2.6
4	9	0.152	59
5	4	1.5×10^{-4}	2.6×10^4
6	2	1.1×10^{-7}	1.8×10^7

Table 5: Exact-duplicate cluster spectrum in the sample against a single-cluster-size model calibrated on S_2/S_1 ($\kappa = 11.6$). The model is fitted through the first two cells and fails from the third onward, which identifies a heavy tail in cluster size.

E Composition of the sample by crawl year

Table 6 compares the share of `sample-10BT` contributed by each crawl year against the share the 96-snapshot release contributes, using the 40,000 documents whose snapshot label we read at 400 positions through the stream. Intervals come from

a 2,000-draw bootstrap that resamples the 400 runs rather than the 40,000 documents, since documents inside a run share a snapshot and are not independent. The comparison is the covariate check §5.5 describes.

The two years whose release share falls outside the bootstrap interval are 2021, where the sample holds 7.1% against 10.2% in the release, and 2024, where it holds 1.2% against 2.3%. The larger gap falls in the year contributing 2% of the corpus. Over all 14,868,862 documents the sampled mean of 700 GPT-2 tokens implies 1.04×10^{10} tokens, within 4% of the ten billion the release name promises; the release reports roughly 1.5×10^{13} tokens over 2.277×10^{10} documents.

Year	Release	Sample	95% CI	Ratio
2013	1.91%	1.78%	0.8–3.0	0.93
2014	7.36%	7.43%	5.3–9.7	1.01
2015	8.25%	8.70%	6.8–11.0	1.05
2016	7.45%	8.19%	6.1–10.4	1.10
2017	14.46%	15.86%	12.9–19.1	1.10
2018	13.04%	12.13%	9.4–15.0	0.93
2019	11.63%	11.23%	8.6–13.9	0.97
2020	9.16%	10.32%	7.7–12.9	1.13
2021	10.20%	7.08%	5.0–9.3	0.69
2022	7.42%	9.37%	6.8–12.1	1.26
2023	6.82%	6.74%	4.6–9.1	0.99
2024	2.30%	1.17%	0.4–2.1	0.51

Table 6: Share of documents by crawl year, in the release and in `sample-10BT`. Bold marks the two years whose release share falls outside the bootstrap interval.