Rillor

METHODS NOTE

Near-duplicate detection for training data: MinHash, SimHash and embeddings

How near duplicate detection works on training data: shingles and Jaccard similarity, MinHash with banding, SimHash, embeddings, the settings that dataset papers report and how to test a threshold.



Contents

01 Summary

02 Background

03 Method

04 Findings

05 Implications

06 Limits of this analysis

07 Sources

Summary

Near duplicate detection for training data rests on four choices: what counts as similar, at what granularity, with which method and threshold, and what is recorded when a copy is removed.

- Near duplicate detection flags documents that repeat the same text with small edits. Measured as Jaccard similarity over shingles (runs of consecutive words), one pair in Broder's 1997 example scores 60%, 50% or 42.85% at shingle sizes 1, 2 and 3 [1].
- MinHash signatures with locality-sensitive hashing (LSH) catch a pair with a probability that rises along an S-curve as its Jaccard similarity grows. In two web text datasets, FineWeb's 112 hashes catch 56% of pairs at 0.7 similarity and 92% at 0.8, and RefinedWeb's 9,000 hashes catch 99.4% at 0.8 [10, 12].
- SimHash compresses a page into a short bit string and compares the bits. For 8 billion web pages, 64-bit fingerprints within 3 bits gave precision and recall near 0.75 [3]. On 1.6 billion pages, the random-projection method behind SimHash had precision of 0.50, against 0.38 for shingling [4].
- Embedding methods find pairs that say the same thing in different words. SemDeDup removed 37% of the LAION-440M image set with no performance drop [18], and about a third of LAION-2B's images are duplicates [20].
- More removal did not always help. A model trained on FineWeb's data deduplicated across all 96 of its crawl snapshots, each snapshot against the newer ones, did little better than one trained on the data with no deduplication [10]. A model trained on the data deduplicated within each snapshot matched the performance of one trained on RefinedWeb, another web text dataset [10, 12].
- Training data needs its own check against evaluation sets. In one study, 4.6% of C4's and 14.4% of RealNews' validation examples had an approximate duplicate in training [6], and paraphrased test items pass word-sequence (n-gram) checks [24].
- Thresholds are tested on labeled pairs. One web study labeled sampled pairs at each bit distance and chose the setting where precision and recall were both near 0.75 [3]. Another could not determine recall across 1.6 billion pages [4].
- Each removal can be recorded with the group of matching documents it belonged to (its cluster), the copy kept, the rule and the threshold. Datasheets for Datasets asks whether a dataset has redundancies [25], and one curation toolkit finds duplicates in one step and removes them in another [29].

Background

Web-scale collections repeat themselves. In 1997, Broder clustered over 30 million web documents at 50% resemblance and found 3.6 million clusters holding 12.3 million documents, of which 2.1 million clusters held only identical documents [1]. In 2024, What's In My Big Data reported, from exact hashes of each document's text, that about 50% of the documents in RedPajama and LAION-2B-en are duplicates [23]. The Pile, which oversampled some sources on purpose, has 139 million documents in 64.6 million duplicate clusters [23].

Repeated training data shows up in what models produce. Carlini and colleagues describe three log-linear relationships: memorization grows with model capacity, with the number of times an example is duplicated and with the tokens of context in the prompt [8]. Regeneration is superlinear in a sequence's count. A sequence present 10 times in the training data is generated about 1,000 times more often on average than one present once [7]. In a 2022 study of repeated data, an 800-million-parameter model with 0.1% of its data repeated 100 times performed like a 400-million-parameter model, though 90% of its training tokens stayed unique [9].

Three kinds of duplicate recur in the sources. Exact duplicates "are easy to identify by standard checksumming techniques" [3]. Near duplicates are "identical in terms of content but differ in a small portion of the document such as advertisements, counters and timestamps" [3]. Semantic duplicates are "semantically similar, but not exactly identical" [18]. The FineWeb authors list design choices such as "line, paragraph, or document-level deduplication? fuzzy or exact matching?" [10], and Lee and colleagues add exact substrings of a fixed token length [6]. Quality filtering and the removal of personal data are separate steps and outside this note.

This note is for teams that build or buy training data.

Method

Sources. Thirty-one sources, each opened on 8 November 2026: five method papers [1-5], twenty dataset, training and documentation papers that report deduplication settings, results or effects [6-25], and six tool documentation pages and code [26-31].

How. Each figure is quoted from its source. Figures marked "computed for this report" apply a source's formula or divide its own numbers. Collision probabilities use P(s) = 1 - (1 - s^r)^b for b bands of r rows [26], checked against the values FineWeb and RefinedWeb report [10, 12]. No detector was run and no corpus was measured for this note. No product is ranked.

Notation. Lee and colleagues write the same formula with b as hashes per band and r as the number of bands, the reverse of datasketch's letters [6, 26]. RefinedWeb's main text (20 buckets of 450 hashes) and its appendix (b = 20 and r = 450 in P = 1 - (1 - s^b)^r) describe its 9,000 hashes differently [12]. This note follows the appendix and FineWeb's description, 450 bands of 20 rows [10, 12], and writes every setting as bands of rows.

Findings

1. What "near duplicate" means: resemblance, containment and shingles

Broder reduced document similarity to "set intersection problems that can be easily evaluated by a process of random sampling" [1]. A shingle is "a contiguous subsequence" of a document, and its w-shingling is the set, or bag, of all shingles of size w [1]. Resemblance r(A, B), the intersection of two shingle sets over their union, captures "roughly the same", and containment c(A, B) captures "roughly contained" [1]. Resemblance needs a fixed-size sketch per document. Containment needs a sketch proportional to the document [1].

The definition decides the score. Broder's example compares A = "a rose is a rose is a rose" with B = "a rose is a flower which is a rose" [1].

Shingle size	As sets	As bags (repeats counted)
1	60%	70%
2	50%	50%
3	42.85%	30%

Source: [1].

A larger size makes resemblance more sensitive to reordering, but "a large size is possibly over-sensitive to small alterations since the change in one token affects w shingles" [1]. Lee and colleagues, RefinedWeb and FineWeb use 5-grams [6, 10, 12]. Henzinger's comparison used 8-token shingles [4], and one curation toolkit's example uses character 24-grams [29]. The datatrove library's MinHash defaults to 5-grams, with the comment that the n-gram size "should be small enough to catch fuzzy matches but big enough to not have each shingle be too common" [30].

Resemblance is not transitive. Consecutive versions of a paper "might well be 'roughly the same,' but version 100 is probably quite far from version 1" [1]. When matches are grouped into clusters (Finding 3), a chain can place two documents that are not alike in one cluster.

2. Exact deduplication: documents, lines, spans and substrings

Exact methods differ in the unit they hash.

Corpus or survey	Exact unit	Reported result	Source
RealNews	Hash of each document's first 100 characters, in a Bloom filter	Any document whose hash collided was excluded	[6]
C4	Three-sentence spans	All but one copy of each repeated span discarded	[13]
OSCAR-21.09	Lines	About 55% removed	[12]
Dolma, web	URL, then document, then paragraph	53.2% of documents, then 14.9% of the rest, then 18.7% of paragraphs	[15]

Order. Dolma's first two stages "remove copies of the same item, such as re-crawls of the same URL and identical pages with multiple URLs" [15]. Its paragraph stage removes boilerplate such as "the byline under all articles by the same author", and runs last because "paragraph removal risks disrupting content analysis" [15].

Exact substrings. Lee and colleagues' ExactSubstr method finds substrings shared by two examples with a suffix array and removes the substring from one of them when it is at least 50 tokens long, a length chosen "based on statistical analyses" [6]. The share of training tokens inside such duplicated substrings was 7.18% in C4, 19.4% in RealNews, 2.76% in Wiki40B and 0.76% in LM1B, where "90% of all documents are under 50 tokens" [6]. The released code writes out the byte ranges to remove, and its README calls the length threshold "entirely dataset-dependent" [31]. RefinedWeb removed "any match of more than 50 consecutive tokens" with the same implementation, and found that dropping whole documents or masking the loss instead "didn't result in significant changes in zero-shot performance" [12].

3. MinHash and locality-sensitive hashing

Signatures. Spark's documentation describes MinHash as "an LSH family for Jaccard distance" that "applies a random hash function g to each element in the set and take the minimum of all hashed values" [28]. Charikar defines a locality-sensitive hashing scheme as a family of hash functions whose collision probability equals the similarity, with min-wise independent permutations as the construction for sets [2]. MinHash's collision probability "is a function of resemblance similarity" [5]. datasketch's default of 128 permutations has a sampling error of "about 0.03-0.04" [27].

Bands. Comparing every signature with every other is quadratic. For The Pile, "A simple quadratic Minhash comparison of all documents would have taken several hundred thousand years" [14]. LSH splits each signature into b bands of r values. A stored set becomes a candidate when it "agrees on all r values of at least one band", so a pair with Jaccard similarity s is returned with probability P(s) = 1 - (1 - s^r)^b [26]. The curve "jumps from near 0 to near 1 around a similarity determined by (b, r)", and it is "a soft threshold" [26].

Curves for published settings. The values below are computed for this report from that formula [26].

Jaccard similarity	14 bands of 8 (FineWeb)	450 bands of 20 (Lee and colleagues, RefinedWeb)	40 bands of 20 (Lee and colleagues, second run)
0.6	21.1%	1.6%	0.1%
0.7	56.5%	30.2%	3.1%
0.75	77.2%	76.1%	11.9%
0.8	92.4%	99.46%	37.1%
0.85	98.8%	above 99.99%	79.4%
0.9	99.96%	above 99.99%	99.4%

The first two columns match FineWeb's published 56%, 77%, 92% and 98.8% [10] and RefinedWeb's 76% and 99.4% [12]. Under FineWeb's setting, about one pair in five at 0.6 becomes a candidate (computed for this report from [10, 26]). Its authors chose fewer hashes because "the compute and storage savings make up for the higher uncertainty on documents near the threshold" [10]. RefinedWeb's 9,000 hashes allow "a steeper, more well-defined cut off" and require "a substantially larger amount of compute resources" [10]. Lee and colleagues chose 450 bands of 20 "so as to make sure a collision at the desired Jaccard index threshold of 0.8 had a high probability of occurring", and their second run at 0.9 used 800 hashes [6].

datasketch picks bands and rows itself. At 128 permutations it chooses 37 bands of 3 for a threshold of 0.3, 25 of 5 for 0.5, 14 of 9 for 0.7 and 5 of 25 for 0.9, since "fewer, longer bands are more selective" [26]. With 14 bands of 9, a pair at exactly 0.7 is returned 43.8% of the time and a pair at 0.8 86.7% of the time (computed for this report from [26]). The threshold and permutation count "cannot be changed afterwards" [26].

Settings in published corpora.

Corpus	Shingles and signature	Rule	Reported result	Source
Training data of a 2020 175-billion-parameter model	Spark MinHashLSH, 10 hashes	Documents with high overlap removed within each dataset	Dataset size down about 10% on average	[11]

Corpus	Shingles and signature	Rule	Reported result	Source
The Pile (OpenWebText2, Pile-CC)	datasketch, 10 hashes	Approximate Jaccard 0.5	Duplicate rates of 28% and 26%	[14]
C4, RealNews, LM1B, Wiki40B (NearDup)	5-grams, 9,000 hashes	Jaccard 0.8, then edit similarity 0.8	3.04%, 13.63%, 4.86% and 0.39% of training examples had a near duplicate	[6]
RefinedWeb	5-grams, 9,000 hashes, text normalized first	Candidates kept without a false-positive check	About 50% removed by MinHash and exact substrings together	[12]
FineWeb	5-grams, 112 hashes	75% similarity, each snapshot apart	20 trillion tokens kept	[10]
The Stack	256 permutations	Jaccard above 0.85	38.6% of files and 53.7% of volume removed	[16]

RefinedWeb normalized text before hashing: punctuation removed, lowercased, NFD Unicode normalization, accents removed and whitespace normalized [12]. It reports that "using less aggressive settings, such as the 10 hashes of The Pile (Gao et al., 2020), resulted in lower deduplication rates and worsened model performance" [12].

Clusters. Lee and colleagues turned matches into a graph and took its connected components as clusters [6]. In RefinedWeb, "if documents A and B match in one bucket and B and C in another, A-B-C becomes a cluster", and all but one are removed at random [12]. FineWeb clusters the same way and keeps one randomly chosen document [10]. The Stack reduced each cluster "by ensuring that each file in the original cluster is similar to at least one other file in the reduced cluster" [16].

Verification. Lee and colleagues computed each candidate's actual Jaccard similarity and kept it above 0.8, then required edit similarity above 0.8 [6]. RefinedWeb skipped this false-positive check because "our main concern is improving recall" [12].

4. SimHash, and MinHash vs SimHash

Construction. Charikar's random-hyperplane hash gives one bit per hyperplane. Two vectors collide with probability $1 - \theta(u, v)/\pi$, where θ is their angle, which is "very closely related to the cosine similarity measure" [2]. Concatenated, t bits form a fingerprint, and "Estimating similarity between vectors amounts to measuring

the Hamming distance between the corresponding t-bit hash vectors" [2]. In Manku and colleagues' version, a page becomes a set of weighted features, and each feature's hash bits raise or lower an f-dimensional vector by its weight [3].

The web crawling setting. For a web search engine's crawler, Manku and colleagues report that "for a repository of 8B webpages, 64-bit simhash fingerprints and k = 3 are reasonable", with k the Hamming distance in bits [3]. Each 64-bit fingerprint takes 8 bytes, 64GB for 8 billion pages, against 24 bytes per fingerprint for Broder's shingle-based method [3].

A large-scale comparison. Henzinger compared shingling (Alg. B) with Charikar's random projection (Alg. C) on 1.6 billion distinct web pages, each method storing 48 bytes per page [4]. Shingling used 84 min-values in 6 supershingles of 14 and 8-token shingles, and called pages near duplicates when at least 2 supershingles matched. Random projection used 384 bits, with a threshold of 372 agreeing bits [4].

Measure	Shingling (Alg. B)	Random projection (Alg. C)
Overall precision	0.38	0.50
Share of its near-duplicate pairs on the same site	92%	74%

Source: [4].

Both "perform about the same for pairs on the same site (low precision) and for pairs on different sites (high precision)", where precision was 0.86 and 0.9 [4]. For a combined algorithm, "The precision of one tradeoff is 0.79 with 79% of the recall of Alg. B" [4]. The two count different things. Random projection "ignores the order of the tokens" and takes term frequency into account, while shingling takes order into account and "ignores the frequency of shingles" [4].

What the comparison covers. Shrivastava and Li found that "MinHash virtually always outperforms SimHash when the data are binary, as common in practice such as search", and that it "significantly outperforms SimHash in high similarity region" [5]. These results come from web pages and retrieval. None of the sources reviewed tests both methods on one training corpus and trains models on the outputs. MinHash is the fuzzy method behind The Pile, RefinedWeb, FineWeb, The Stack, Lee and colleagues' NearDup and the training data of the 2020 175-billion-parameter model [6, 10, 11, 12, 14, 16]. Allamanis compared code files by exact Jaccard over token sets [17], SemDeDup used embeddings on C4 [18], and FineWeb notes that ROOTS used SimHash [10].

5. Embeddings and semantic deduplication

Method. SemDeDup embeds each item with a pre-trained model, clusters the embeddings into k clusters with k-means (50,000 for images, 11,000 for text), computes pairwise cosine similarity inside each cluster and treats pairs above a threshold as semantic duplicates. It keeps the item "with the lowest cosine similarity to the cluster centroid" [18]. For images it "considers only the images and ignores the captions" [18].

Cost. Comparing every pair in LAION-440M "would require ≈ 1.9x10^17 similarity computations". Clustering cuts that to about 4.6x10^12, which the authors call "a 5-order of magnitude improvement" [18]. With 50,000 clusters, the method needs "only 0.29 of the GPU hours needed for one epoch of training on 100% of the data", or 0.0091 of the complete training cost in GPU hours [18].

What it removed. In LAION-440M, 30% of images have a semantic duplicate at a cosine distance threshold of 0.00095 and 50% at 0.03 [18]. SemDeDup "can remove up to 37% of LAION440M with no performance drop, and 50% with minimal performance drop (< 0.5%)", while random removal "results in much larger drops" [18]. At low thresholds on text it finds "templated text, where typically few words (e.g. a geographic location or a name) is changed", which "successfully evades exact-string deduplication methods" [18]. On C4 at 4% pruning, "NearDup and SemDeDup have comparable performance", and with several epochs over pruned data the authors reached a full single-epoch result with 10% to 15% less compute [18].

Image near duplicates. About 700 million, or a third, of LAION-2B's images are duplicates, found with compressed image embeddings by a chain that "runs with modest compute" [20]. Carlini and colleagues compared training images as 512-dimensional embeddings [21]. From "the 350,000 most-duplicated examples" and 175 million generated images, 94 images were extracted under their definition and 109 with manual review [21]. Extraction happened "most often when they have been duplicated at least k = 100 times" [21]. They note that in the image domain "simple deduplication is common, where images with identical URLs and captions are removed, but most datasets do not compute other inter-image similarity metrics" [21].

6. What deduplication changed in published training runs

Copies in output. Lee and colleagues measured the share of tokens in 100,000 unprompted generations of a 1.5-billion-parameter model that were part of a 50-token copy from training [6].

Training data	1 epoch	2 epochs
Original	1.926%	1.571%
NearDup	0.189%	0.264%
ExactSubstr	0.138%	0.168%

Source: [6].

Deduplicated models also "require fewer training steps to achieve the same or better accuracy" [6]. The authors caution that the two techniques "used separate duplicate thresholds and a different choice of thresholds could change the results" [6].

Privacy. After deduplication, "language models are considerably more secure against these types of privacy attacks", and existing methods for detecting memorized sequences "have near-chance accuracy on non-duplicated training sequences" [7].

Model quality. For source code, "near-deduplicating the data significantly boosts performance across all experiments" [16]. RefinedWeb found that "MinHash alone is insufficient, as it doesn't match the zero-shot performance of exact deduplication", and that combining it with exact deduplication "doesn't improve performance further" [12].

Where more removal did not help. FineWeb's global MinHash over all 96 snapshots "removed as much as 90% of the original base filtered data" from the oldest snapshots and "showed little improvement over a model trained on the non-deduplicated data" [10]. In the 2013-48 snapshot, the data kept "was actually of worse quality" than the data removed, with "more ads, incoherent lists of keywords and generally badly formatted text" [10]. Per-snapshot deduplication "resulted in 20 trillion tokens" and "matched RefinedWeb's performance" [10]. The authors' hypothesis is that the gain "lies in the removal of large clusters of duplicates with hundreds of thousands of documents", while deduplicating clusters of fewer than about 100 duplicates "can harm performance" [10].

Repetition of different kinds. Training "with up to 4 epochs of repeated data yields negligible changes to loss compared to having unique data", across 400 runs of up to 9 billion parameters [22]. That repetition covers a whole dataset for several epochs. The 2022 study's damage came from 0.1% of the data repeated 100 times [9], and D4, which selects data by embeddings on top of deduplicated data, found that "repeating random data performs worse than baseline training" while repeating selected data outperformed it [19]. For images, removing CIFAR-10 images with similarity above 0.85 removed 5,275 of 50,000 examples, and the retrained model regenerated 986 training examples against 1,280 for the original, so "deduplication is not a perfect solution" [21].

7. Train-test overlap

All four validation sets in Lee and colleagues' study "contained text duplicated in the training set" [6]. In C4 and RealNews, 4.6% and 14.4% of validation examples had an approximate duplicate in training, which "could cause evaluation metrics to be unfairly inflated" [6]. The Pile's authors state that "it is still possible that some documents are duplicated across the train/validation/test splits" [14]. For source code, reported metrics were "sometimes inflated by up to 100%" on duplicated corpora [17]. Allamanis treated two files as near duplicates when the Jaccard similarities of two token sets exceeded 0.8 and 0.7, and skipped files with fewer than 20 identifier tokens [17].

The 13-gram method. The 2020 paper on a 175-billion-parameter model removed "the colliding 13-gram as well as a 200 character window around it", and removed whole any document split into more than 10 pieces [11]. Removing a whole document on one collision "overly penalized long documents such as books for false positives", and 13-grams matching more than 10 training documents were ignored [11]. A bug left the removal partial, so the authors built clean benchmark subsets without the examples that overlapped the pretraining set [11].

Other checks. Dolma's Bloom filter can be seeded "with test examples, then flagging any detected duplicates when mixing the pretraining data" [15]. What's In My Big Data found contaminated benchmarks "including the Winograd Schema Challenge and parts of GLUE and SuperGLUE" [23]. Yang and colleagues show that n-gram methods are insufficient because "simple variations of test data (e.g., paraphrasing, translation) can easily bypass these decontamination measures" [24]. Their method takes the training samples closest to each test item by embedding and has a model judge each pair, and they found that 8% to 18% of the HumanEval benchmark overlaps with RedPajama-Data-1T and a public code corpus [24].

8. Choosing a threshold and measuring the detector

Labeled pairs at each distance. For each Hamming distance from 1 to 10, Manku and colleagues "randomly sampled an equal number of pairs of fingerprints that are at Hamming distance exactly k" and tagged each as true positive, false positive or unknown [3]. Pages "that differ slightly, such as only in counters, ads, or timestamps" counted as true positives [3]. "A very low value misses near-duplicates (false negatives), and a very high value tags incorrect pairs as near-duplicates (false positives)", and k = 3 was chosen "because both precision and recall are near 0.75" [3].

Histograms of colliding pairs. Lee and colleagues plotted the Jaccard and edit similarity of all colliding pairs: "If the threshold chosen has few comparisons around the chosen threshold, then we've likely captured the majority of actual near duplicates above that threshold" [6].

Boilerplate and templates. Henzinger counted "two shopping pages with common boilerplate text but a different product in the page center" as an incorrect pair [4]. RefinedWeb notes that "MinHash excels at finding templated documents: licenses with only specific entities differing, placeholder SEO text repeated across websites" [12]. The 2020 paper ignored 13-grams matching many documents because "inspection showed the majority of these to contain common cultural phrases, legal boilerplate, or similar content" [11].

Recall. Across 1.6 billion pages, Henzinger writes that "we cannot determine the recall" [4]. Manku and colleagues' recall is measured on sampled pairs [3]. Small samples mislead too: in FineWeb's simulation, with each document repeated 100 times in 20 trillion tokens, a 1-billion-token sample shows almost all documents as unique, so "Small ablations are ineffective for deduplication analysis" [10].

Errors by design. datasketch warns of both false positives and false negatives and lets the user weight the two [26]. Before any labeling, one minus the S-curve gives the chance that a pair at each similarity is never returned as a candidate (Finding 3) [26].

9. Running it at scale

Pairs. One million documents give 499,999,500,000 pairs (computed for this report as n(n-1)/2). The Pile ran in-memory LSH "on a machine with enough RAM for Common Crawl, taking several days", and "Due to memory constraints" did no deduplication across the whole Pile [14].

Suffix arrays. Lee and colleagues built suffix arrays on "a single VM on the cloud with 96 cores and 768GB of memory". C4 (350GB) took "under 12 hours (wall-clock)", about 1,000 CPU-hours, and deduplicating it then took under an hour. The suffix array "requires an 8x space overhead", 1.5TB for C4 [6]. The released code's README advises as many cores as possible (the authors used 96), more than 600GB of RAM and more than 1TB of disk for a C4-sized dataset of about 300GB [31].

MinHash. NearDup on English C4 "consumed approximately 41.5 kWh of energy", with settings designed for very high recall, and the authors write that other parameters could be "much more energy efficient while producing similar results" [6]. RefinedWeb's pipeline ran on 100 to 250 cloud machines of 72 vCPUs and 144 GiB each, usually 10,000 to 20,000 vCPUs [12]. It split Common Crawl into 100 parts for URL deduplication [12] and ran MinHash before exact substrings "as it is easier to scale", which cut the exact stage's input by nearly 40% [12].

Order of stages. One curation toolkit recommends "Exact → Fuzzy → Semantic: Start with fastest methods", requires GPU acceleration for its deduplication workflows, and its exact and fuzzy examples only identify duplicates (`perform_removal=False`), leaving removal to a separate `TextDuplicatesRemovalWorkflow` step [29].

10. What to record

The copy kept. FineWeb and RefinedWeb keep a random document per cluster [10, 12], SemDeDup the item with the lowest cosine similarity to the centroid [18] and The Stack a reduced cluster [16].

Identification and removal apart. One toolkit separates the two steps [29], RedPajama v2 released deduplication labels [10], and the ExactSubstr code outputs byte ranges [31]. Each produces a list of what a rule flags, which a reviewer can check before anything is removed.

Documentation. Datasheets for Datasets asks, under composition, "Are there any errors, sources of noise, or redundancies in the dataset? If so, please provide a description" [25]. A description backed by cluster ids, the copy kept, the rule, the threshold and the stage order answers that question with evidence.

Implications

For a buyer or a builder. A defensible deduplication record:

1. States the similarity measure and the shingle type and size [1, 30].
2. States each stage's granularity and the order of stages [10, 15].
3. Gives the method's parameters: bands and rows, bits and distance, or clusters and cosine threshold [3, 6, 18, 26].
4. Shows the collision curve at the chosen setting, so a reader sees how often true pairs near the threshold are missed [10, 12, 26].
5. Reports precision on a labeled sample of pairs at several similarity levels, with pairs from one source and across sources apart [3, 4].
6. Checks evaluation splits against training data with an n-gram method and an embedding method [6, 11, 24].
7. Keeps the removal record: cluster, copy kept, rule and threshold, with identification apart from removal [10, 25, 29].
8. Judges the effect with training runs large enough to show duplication [10].

A deduplication rate is comparable to another only when both state the measure, the granularity, the threshold and whether the run was global or per part [6, 10, 12].

Rillor's training data cleaning deduplicates, filters and de-identifies data to the client's rules, with exact and near duplicates found and removed.

Limits of this analysis

- Every duplicate rate here is for public web, news, code or image corpora. None describes private, enterprise or commissioned datasets.
- No source reviewed compares MinHash, SimHash and embeddings on one training corpus with model results. The closest are SemDeDup against NearDup at 4% pruning on C4 [18], two studies of web pages [3, 4] and a retrieval study [5].
- Recall is not known for any whole corpus [4]. Manku and colleagues' recall is measured on a sample [3].
- No source reviewed measures what legitimate content embedding deduplication removes, beyond SemDeDup's examples [18].
- Settings and results hold for the corpus and model scale each paper used. Several papers have authors at companies that build models [9, 11, 12, 18].
- Tool documentation can change after its viewing date [26-31].
- Video near duplicates and the legal effects of memorized copies are outside this note.

Sources

1. On the resemblance and containment of documents. Andrei Z. Broder, Proceedings of Compression and Complexity of SEQUENCES 1997, IEEE, pp. 21-29, 1997 (preliminary version), viewed 8 November 2026. https://www.cs.princeton.edu/courses/archive/spring13/cos598C/broder97resemblance.pdf

2. Similarity Estimation Techniques from Rounding Algorithms. Moses S. Charikar, Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC 2002), pp. 380-388, 2002, viewed 8 November 2026. https://www.cs.princeton.edu/courses/archive/spring04/cos598B/bib/CharikarEstim.pdf

3. Detecting Near-Duplicates for Web Crawling. Gurmeet Singh Manku, Arvind Jain and Anish Das Sarma, WWW 2007, ACM, pp. 141-150, 2007, viewed 8 November 2026. https://research.google.com/pubs/archive/33026.pdf

4. Finding Near-Duplicate Web Pages: A Large-Scale Evaluation of Algorithms. Monika Henzinger, SIGIR 2006, ACM, pp. 284-291, August 2006, viewed 8 November 2026. https://www3.cs.stonybrook.edu/~cse692/papers/henzinger_sigir06.pdf

5. In Defense of Minhash over Simhash. Anshumali Shrivastava and Ping Li, Proceedings of the 17th International Conference on Artificial Intelligence and Statistics, PMLR 33:886-894, 2014, viewed 8 November 2026. https://proceedings.mlr.press/v33/shrivastava14.html

6. Deduplicating Training Data Makes Language Models Better. Katherine Lee, Daphne Ippolito, Andrew Nystrom and others, ACL 2022, arXiv 2107.06499, version 2, 24 March 2022, viewed 8 November 2026. https://arxiv.org/abs/2107.06499

7. Deduplicating Training Data Mitigates Privacy Risks in Language Models. Nikhil Kandpal, Eric Wallace and Colin Raffel, ICML 2022, arXiv 2202.06539, version 3, 20 December 2022, viewed 8 November 2026. https://arxiv.org/abs/2202.06539

8. Quantifying Memorization Across Neural Language Models. Nicholas Carlini, Daphne Ippolito, Matthew Jagielski and others, ICLR 2023, arXiv 2202.07646, version 3, 6 March 2023, viewed 8 November 2026. https://arxiv.org/abs/2202.07646

9. Scaling Laws and Interpretability of Learning from Repeated Data. Danny Hernandez and others, arXiv 2205.10487, version 1, 21 May 2022, viewed 8 November 2026. https://arxiv.org/abs/2205.10487

10. The FineWeb Datasets: Decanting the Web for the Finest Text Data at Scale. Guilherme Penedo and others, arXiv 2406.17557, version 2, 31 October 2024, viewed 8 November 2026. https://arxiv.org/abs/2406.17557

11. Language Models are Few-Shot Learners. Tom B. Brown and others, NeurIPS 2020, arXiv 2005.14165, version 4, 22 July 2020, viewed 8 November 2026. https://arxiv.org/abs/2005.14165

12. The RefinedWeb Dataset for Falcon LLM: Outperforming Curated Corpora with Web Data, and Web Data Only. Guilherme Penedo and others, arXiv 2306.01116, version 1, 1 June 2023, viewed 8 November 2026. https://arxiv.org/abs/2306.01116

13. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. Colin Raffel and others, Journal of Machine Learning Research 21 (2020), arXiv 1910.10683, version 4, 19 September 2023, viewed 8 November 2026. https://arxiv.org/abs/1910.10683

14. The Pile: An 800GB Dataset of Diverse Text for Language Modeling. Leo Gao and others, arXiv 2101.00027, version 1, 31

December 2020, viewed 8 November 2026. https://arxiv.org/abs/2101.00027

15. Dolma: an Open Corpus of Three Trillion Tokens for Language Model Pretraining Research. Luca Soldaini and others, arXiv 2402.00159, version 2, 6 June 2024, viewed 8 November 2026. https://arxiv.org/abs/2402.00159

16. The Stack: 3 TB of permissively licensed source code. Denis Kocetkov and others, arXiv 2211.15533, version 1, 20 November 2022, viewed 8 November 2026. https://arxiv.org/abs/2211.15533

17. The Adverse Effects of Code Duplication in Machine Learning Models of Code. Miltiadis Allamanis, arXiv 1812.06469, version 6, 11 August 2019, viewed 8 November 2026. https://arxiv.org/abs/1812.06469

18. SemDeDup: Data-efficient learning at web-scale through semantic deduplication. Amro Abbas, Kushal Tirumala, Dániel Simig and others, arXiv 2303.09540, version 3, 22 March 2023, viewed 8 November 2026. https://arxiv.org/abs/2303.09540

19. D4: Improving LLM Pretraining via Document De-Duplication and Diversification. Kushal Tirumala, Daniel Simig, Armen Aghajanyan and Ari S. Morcos, arXiv 2308.12284, version 1, 23 August 2023, viewed 8 November 2026. https://arxiv.org/abs/2308.12284

20. On the De-duplication of LAION-2B. Ryan Webster, Julien Rabin, Loic Simon and Frederic Jurie, arXiv 2303.12733, version 1, 17 March 2023, viewed 8 November 2026. https://arxiv.org/abs/2303.12733

21. Extracting Training Data from Diffusion Models. Nicholas Carlini and others, USENIX Security 2023, arXiv 2301.13188, version 1, 30 January 2023, viewed 8 November 2026. https://arxiv.org/abs/2301.13188

22. Scaling Data-Constrained Language Models. Niklas Muennighoff and others, NeurIPS 2023, arXiv 2305.16264, version 5, 28 June 2025, viewed 8 November 2026. https://arxiv.org/abs/2305.16264

23. What's In My Big Data? Yanai Elazar and others, ICLR 2024, arXiv 2310.20707, version 2, 5 March 2024, viewed 8 November 2026. https://arxiv.org/abs/2310.20707

24. Rethinking Benchmark and Contamination for Language Models with Rephrased Samples. Shuo Yang, Wei-Lin Chiang, Lianmin Zheng and others, arXiv 2311.04850, version 2, 11 November 2023, viewed 8 November 2026. https://arxiv.org/abs/2311.04850

25. Datasheets for Datasets. Timnit Gebru and others, Communications of the ACM 64(12), December 2021, arXiv 1803.09010, version 8, 1 December 2021, viewed 8 November 2026. https://arxiv.org/abs/1803.09010

26. MinHash LSH. datasketch 2.0.0 documentation, Eric Zhu, viewed 8 November 2026. https://ekzhu.com/datasketch/lsh.html

27. MinHash. datasketch 2.0.0 documentation, Eric Zhu, viewed 8 November 2026. https://ekzhu.com/datasketch/minhash.html

28. Extracting, transforming and selecting features. Spark 4.3.0 documentation, The Apache Software Foundation, viewed 8 November 2026. https://spark.apache.org/docs/latest/ml-features.html

29. Deduplication. NeMo Curator documentation, v1.4.0 (26.09), NVIDIA, viewed 8 November 2026. https://docs.nvidia.com/nemo/curator/latest/curate-text/process-data/deduplication/index.html

30. datatrove, src/datatrove/pipeline/dedup/minhash.py. Hugging Face on GitHub, main branch, file last changed 27 October 2026, viewed 8 November 2026. https://github.com/huggingface/datatrove/blob/main/src/datatrove/pipeline/dedup/minhash.py

31. deduplicate-text-datasets, README. google-research on GitHub, viewed 8 November 2026. https://github.com/google-research/deduplicate-text-datasets

About this report

Published by Rillor on 9 November 2026. Superintelligence (SI) tools assist research and drafting, and every figure cites its source. Rillor builds and secures SI systems, supplies the data they learn from and delivers the infrastructure they run on. We apply SI to financial and compute markets.

Cite as: Rillor. Near-duplicate detection for training data: MinHash, SimHash and embeddings. 9 November 2026. https://rillor.com/insights/near-duplicate-detection-for-training-data

Notices

NVIDIA, AMD, Ceph, Kubernetes and related product names are trademarks of their owners and are used only to identify products. No affiliation or endorsement is implied.

Work with Rillor

Send the question the data should answer, the sources you know of, the coverage you need and how often it should update. Contact us at rillor.com/contact?type=custom-data or contact@rillor.com. More on data services: rillor.com/data