Kurdish Studies
November 2022
Volume: 10, No: 02, pp 1273-1301
ISSN: 2051-4883 (Print) | ISSN 2051-4891 (Online)
www.KurdishStudies.net

DOI: 10.67145/ks.v10i2.4157

Repeated Text in the Kurdish (Kurmanji) File of CC-100 and the Calibration of a MinHash-LSH Near-Duplicate Detector

Luis Eduardo Muñoz Guerrero*

*Universidad Tecnológica de Pereira, Facultad de Ingenierías, Pereira, Risaralda, Colombia, lemunozg@utp.edu.co, lemunozg@utp.edu.co · ORCID 0000-0002-9414-6187

Abstract

In post hoc analyses of CC-100's Kurmanji file, extra paragraph copies hold 22.5% of the tokens, extra document copies 5.5% and near-duplicates at Jaccard 0.8 up to 7.3% (levels overlap); 6,004 paragraphs occur over twelve times, whereas one copy per monthly snapshot allows at most twelve. A MinHash-LSH detector finds, averaged over hash draws, 99.2% of the document pairs at that threshold, nearly as predicted. Its frozen calibration test passes only under a later-chosen validity reading; stricter later rules fail, as they often would even for a calibrated detector; the cost hypothesis fails. An exact search takes minutes.

Keywords: web corpora; language resources; deduplication; set similarity join; reproducibility

1. Introduction

Ahmadi describes Kurdish as "a less-resourced language with a notable diversity in dialects and scripts" [1]. The multilingual CC-100 collection, built from web crawls to train language models, includes a Kurdish file. Its distribution page describes the collection as an attempt to recreate the data used to train XLM-R, the multilingual language model of Conneau et al., and states that the file was built with the URLs and paragraph indices of the CCNet repository [2]; Conneau et al. list the Kurdish part of their data as Kurdish (Kurmanji) [3]. The CCNet pipeline removes duplicated paragraphs across web pages, comparing paragraph hashes within one, several or all shards of a snapshot, and identifies the language of documents automatically [4]. Users of the file therefore inherit two open questions: how much of its text is repeated, and how reliable its language label is. On the second, Caswell et al., who trained their own language-identification models on up to 1,629 languages, report that "human-judged LangID accuracy for web-crawl text corpora created using these models is only around 5% for many lower-resource languages" [5], and Kreutzer et al., who audited by hand more than two hundred language-specific corpora from five web-mined datasets, found that those of lower-resource languages have systematic issues [6]. This paper addresses the first question.

Repetition matters to corpus users. A paragraph that occurs thousands of times contributes its words thousands of times to every frequency count, and near-copies of a document make one text count as many. Lee et al. found that existing language-modelling datasets contain many near-duplicate examples and long repetitive substrings, and that deduplication makes trained models emit memorised text less often [7]. Near-duplicates are common among web pages [8]. Exact copies are easy to find. Near-duplicates need a similarity measure, such as the Jaccard similarity of the sets of word n-grams of two documents, and a way to find every pair above a threshold. Henzinger compared two classic methods, the shingling algorithm of Broder et al. [9] and the random projections of Charikar [10], and found that neither works well for near-duplicates within the same site [11].

MinHash with locality-sensitive hashing (LSH) finds such pairs without comparing all of them: it groups documents whose hash values agree on a band of rows, so that only candidate pairs need an exact check (Section 2). In plain terms (Table 1 explains these and the other technical terms), each document is summarised by 128 hash values, and two documents are compared only if, in at least one of 16 fixed blocks of eight of these values, all eight agree. Whether such a detector can be trusted depends on its calibration: the probability that a pair becomes a candidate should match the function its parameters promise, and the detector should miss no more pairs above the threshold than that function predicts. When it is affordable, an exact similarity join finds every pair and needs no calibration. We found no study of repetition in the Kurmanji file of CC-100 (Section 2 describes how we searched).

Table 1. Technical terms of the paper in plain words.

Term	Meaning
Token	A word as the tokeniser cuts it: a maximal run of letters, digits and underscores
Shingle	A run of five consecutive tokens; a document is compared through the set of its shingles
Jaccard similarity (J)	The shingles two documents share, divided by all their distinct shingles: 1 for identical sets, 0 for disjoint ones
Exact copy	A document identical, character for character, to an earlier one

Term	Meaning
Near-duplicate	A document whose Jaccard similarity with another is at least the threshold $\theta = 0.8$
Hash function	A rule that turns a piece of text into a number, so that equal texts give equal numbers
MinHash signature	The smallest value of each of 128 hash functions over a document's shingles; with ideal functions, two documents agree on each value with probability J
Band	A fixed block of eight of the 128 values; the detector uses 16 bands
Candidate pair	Two documents that agree on every value of at least one band; only these are compared exactly
Draw of hash functions	One random choice of the 128 functions; another draw finds slightly different pairs
Recall	The share of the pairs above the threshold that the detector finds
Miss rate	The share of those pairs that it does not find
Calibration	Whether, at each similarity, as many pairs become candidates as the theory predicts
Exact join	Finds every pair above the threshold without comparing all pairs, by indexing each document's rarest shingles
Equivalence test	A test that a difference lies within a margin, here ± 0.03, made of two one-sided tests
Holm's adjustment	A correction of p-values for several tests made together
Wilson interval	A range of plausible values for a proportion estimated from a sample
Bootstrap	Repeating an analysis on resamples of the data to see how much it varies
Clustered variance	A variance that lets pairs built from the same document be correlated
Snapshot	One monthly crawl of the web by Common Crawl; the file comes from those of 2018
Deduplication scope	The texts within which duplicates are looked for: part of a snapshot, a whole snapshot or several

Note. Each is defined formally where it is first used.

We ask three questions.

RQ1: How much of the file is repeated, as whole documents, as near-duplicate documents and as paragraphs, counted in documents and in tokens, and what does the repetition imply about how the file was deduplicated?

RQ2: Can a MinHash-LSH detector be trusted to find the near-duplicates of this file? We test whether its candidate rate follows the theoretical curve, count the pairs above the threshold that it misses, and measure how both depend on the random draw of its hash functions.

RQ3: What does each kind of removal change for frequency-based uses of the corpus, and does a file of this size need an approximate detector at all?

The answers have different standing. The calibration test of RQ2 (hypotheses H1 and H2) and a hypothesis on cost (H3) were fixed in an analysis plan before any calibration or cost result was computed. The frozen plan asked whether the detector is calibrated and what it costs, and added descriptive analyses of the lengths of repeated lines, of word frequencies after removing near-duplicates and of script markers; the RQ1's counts in tokens and bound on deduplication, and RQ3's exact join and cost comparison were added after the primary results were known (Amendments 5, 7 and 8; Section 5). Everything outside H1 to H3 is descriptive or exploratory. The study measures; it does not propose a new hashing scheme, and it does not classify the language of the documents.

2 Background and related work

Haig and Öpengin review the questions of which varieties count as Kurdish and how they relate, and note that there is relatively little controversy about Kurmanji (Northern Kurdish) and Sorani (Central Kurdish) [18]. Kurmanji is mostly written in a Latin-based script and Sorani in an Arabic-based one [1], and the variation of Kurmanji in Turkey had been almost entirely neglected in the literature when Öpengin and Haig proposed a preliminary classification of its dialects [19]. Hassani proposed a basic language resource kit for Kurdish, which faces a paucity of resources and has several dialects [20]; Kurmanji has a dependency treebank [16]; and Ahmadi's toolkit, KLPT, offers text preprocessing, stemming, tokenisation, lemmatisation and transliteration for Sorani and Kurmanji [1]. We do not use the toolkit, because the frozen plan fixed a language-independent tokeniser; Section 6 measures what folding some letters changes. Sheykh Esmaili et al. noted that Kurdish had seen little attention from information retrieval and language processing research, and they built Pewan, the first standard test collection for Kurdish information retrieval [17]. Web text has served such resources, among them a parallel corpus of news, Bianet [15], and some corpora cover a single variety, such as AsoSoft for Sorani [14]; Table 2 gives the variety, script, size and source of these and of the multilingual corpora below, as their sources report them. Sheyholislami

studied the part that satellite television and the Internet play in how Kurds construct their identities [21]. We did not find a prior study of repetition in Kurmanji web text.

Web corpora such as ukWaC, deWaC and itWaC are built by crawling [22], and the construction of web corpora usually includes the removal of boilerplate and of duplicated content [23]. Besides CC-100, multilingual corpora drawn from Common Crawl, such as OSCAR [12] and mC4 [13], have Kurdish parts.

Table 2. Kurdish text in the corpora cited in this paper, as their sources report it.

Corpus	Variety and script	Size	Source of the text
CC-100, ku file (this study) [2]	Labelled Kurmanji; Latin script (Section 3)	236,550 documents; 66,968,643 tokens	Common Crawl, January to December 2018
OSCAR, Kurdish [12]	As labelled	94M bytes and 15,561,003 words; after deduplication 60M and 9,946,440	Common Crawl, November 2018, classified by language
OSCAR, Central Kurdish [12]	As labelled	487M bytes and 48,478,334 words; after deduplication 226M and 18,726,721	The same
mC4, Kurdish [13]	As labelled	0.4 billion tokens; 0.3 million pages	Common Crawl
AsoSoft [14]	Central Kurdish (Sorani)	188 million tokens, normalised	Mostly websites, published books and magazines
Bianet [15]	Northern Kurdish; Latin script	126,350 tokens on the Kurdish side of the English pairs and 142,668 on that of the Turkish pairs	News of a Turkish online outlet, with English and Turkish versions
Dependency treebank [16]	Kurmanji	A little over ten thousand tokens	Syntactically annotated text
Pewan [17]	A Sorani and a Kurmanji collection	—	Test collection for information retrieval

For two sets A and B of shingles, the Jaccard similarity J(A, B) is given by Equation 1. Indyk and Motwani introduced locality-sensitive hashing for approximate nearest-neighbour search, with hash functions under which close objects collide more often than distant ones [24]; Charikar considered schemes in which the probability that two objects collide equals their similarity, $\Pr[h(x)=h(y)] = \mathrm{sim}(x,y)$ [10]. Under a random permutation from a min-wise independent family, two sets have the same minimum with probability equal to their Jaccard similarity [25], and grouping k such minima into b bands of r rows gives the probability that two sets become candidates [26], derived in Section 4. Chaudhuri et al. defined the prefix filter for set-similarity joins [27], after Sarawagi and Kirpal had evaluated such joins with inverted indexes [28]; the filter avoids computing the similarity of all possible pairs [29]. Bayardo et al. find all pairs above a similarity threshold without approximation methods [30], and in an experimental comparison of set-similarity joins the prefix filter was the key technique [31].

$$J(A,B)=|A\cap B|\,|A\cup B| \qquad (1)$$

We searched one bibliographic index of titles and abstracts with twelve queries, which combined the terms MinHash, near-duplicate, locality-sensitive hashing, deduplication, CCNet, cross-lingual representation learning, language identification of web corpora, Kurmanji, Kurdish corpus or language processing, and the exact phrase CC-100 with corpus or with Kurdish; the queries, the date of each search, the number of records it returned and the name of the index are recorded in the repository. For each query we screened up to 15 records with the most citations in the index.

3 Data and corpus audit

The corpus is the Kurdish (ku) file of CC-100, with documents separated by blank lines and paragraphs by line breaks [2]; the copy we downloaded is an xz-compressed plain-text file of 93,664,008 bytes. The distribution page lists the file as Kurdish and says that the data are generated with the open-source CCNet repository [2]; Conneau et al. list the Kurdish part of the data they used as Kurdish (Kurmanji) [3]. The label comes from automatic language identification. CCNet uses the language classifier of fastText, trained on Wikipedia, Tatoeba and SETimes, which supports 176 languages [4]; for their data, Conneau et al. combined an internal identification model with that of fastText and filtered documents with language models trained in each language, as CCNet does [3]; we did not establish which identifier produced the labels of the recreated file. CCNet's filter scores how close a page is to a reference domain such as Wikipedia [4], so it can shape which Kurmanji text

survives. In another web-mined dataset, CCAligned, Kreutzer et al. found that of a sample of English–Kurdish sentence pairs (code ku_TR) 36.52% were correct and 28.70% had a side in the wrong language; for the ku and ckb parts of OSCAR, they note that in the ISO standard ku is a superset of ckb, Central Kurdish [6]. We did not check the label against human judgement, and we call the file Kurmanji in that sense only. Of the paragraphs that occur once, 99.8% contain one of the letters ê, î and û, which are rare in Turkish and English text; Zazaki, whose Latin orthography also writes ê and û, would not be separated by this indicator. The documentation states that the file was built from Common Crawl snapshots from January to December 2018 [2]. The SHA-256 digest of the file is 4c3a23ffbcef88753f95e87ce36f0fca0627c717076653200b7e4343cfe0af6c, and the download script checks size and digest on every run; the server gives 26 October 2020 as the date of its last modification.

Table 3 summarises the audit. Paragraphs and documents are compared as exact character sequences, without normalisation. The file has 236,550 documents, 2,482,825 non-empty paragraphs and 66,968,643 tokens. Of the paragraphs, 1,662,246 are distinct. The excess occurrences, that is occurrences beyond the first of each paragraph, are 820,579 (33.1% of non-empty paragraphs). There are 11,634 copies of whole documents beyond the first (4.9% of documents); 1,613 of them are copies of documents with fewer than five tokens. Section 6 reports what these and the near-duplicate documents amount to in tokens.

correct and 28.70% had a side in the wrong language; for the ku and ckb parts of OSCAR, they note that in the ISO standard ku is a superset of ckb, Central Kurdish [6]. We did not check the label against human judgement, and we call the file Kurmanji in that sense only. Of the paragraphs that occur once, 99.8% contain one of the letters ê, î and û, which are rare in Turkish and English text; Zazaki, whose Latin orthography also writes ê and û, would not be separated by this indicator. The documentation states that the file was built from Common Crawl snapshots from January to December 2018 [2]. The SHA-256 digest of the file is 4c3a23ffbcef88753f95e87ce36f0fca0627c717076653200b7e4343cfe0af6c, and the download script checks size and digest on every run; the server gives 26 October 2020 as the date of its last modification.

Table 3 summarises the audit. Paragraphs and documents are compared as exact character sequences, without normalisation. The file has 236,550 documents, 2,482,825 non-empty paragraphs and 66,968,643 tokens. Of the paragraphs, 1,662,246 are distinct. The excess occurrences, that is occurrences beyond the first of each paragraph, are 820,579 (33.1% of non-empty paragraphs). There are 11,634 copies of whole documents beyond the first (4.9% of documents); 1,613 of them are copies of documents with fewer than five tokens. Section 6 reports what these and the near-duplicate documents amount to in tokens.

Table 3. Corpus audit of the Kurdish (ku) file of CC-100.

Quantity	Value
Documents	236,550
Documents with fewer than five tokens	10,993
Tokens	66,968,643
Non-empty paragraphs	2,482,825
Distinct paragraphs	1,662,246
Excess paragraph occurrences	820,579 (33.1%)
Copies of documents beyond the first	11,634 (4.9%)
Size after decompression (bytes)	420,494,778
Characters	377,868,451
Characters per document: median; mean; maximum	681; 1,607; 596,990
Documents with fewer than fifty characters	26,398

Note. Counts come from streaming passes over the file; paragraphs and documents are compared as exact character sequences.

Text is predominantly in Latin-script letters: 99.98% of letters, and 97.4% of documents contain only letters whose Unicode names are Latin (modifier letters such as ’ count as other scripts). Table 4 counts the documents with letters of other scripts, by Unicode block (Amendment 8) and by their share of each document’s letters (Amendment 9). Arabic-script letters, which can belong to Sorani, to Kurmanji written in Arabic script or to other languages, are the most common, but only a minority of those documents contain the letters of the Arabic-based Kurdish alphabet that Arabic and Persian do not use, no document contains the letters of the Kurdish Cyrillic alphabet, and almost no document is written mostly in another script.

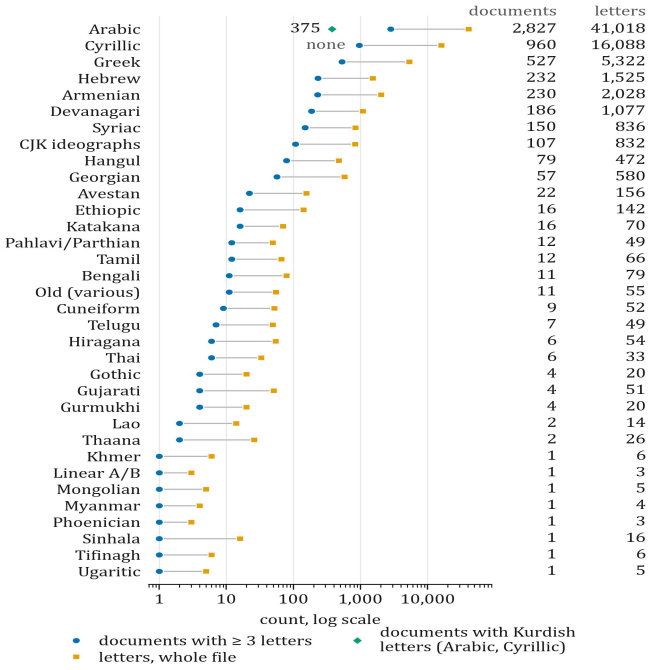
Table 4. Documents with letters of scripts other than Latin (exploratory).

Documents with	Documents
at least three Arabic-script letters	2,827
… and one of ڵ, ڕ or ێ (Arabic-based Kurdish alphabet)	375
any Arabic-script letter	2,981
Arabic-script letters as more than half of their letters	1
at least three Cyrillic letters	960
… and Q or W (Kurdish Cyrillic alphabet)	0
Cyrillic letters as more than half of their letters	0
letters of other scripts as more than a tenth of their letters	577
… as more than half of their letters	1

Documents with	Documents
at least three of ı, İ, ğ, Ğ (Turkish alphabet)	5,959 (2.5%)

Note. Counts of at least three letters are by Unicode block (Amendment 8; the Arabic joining stroke tatweel included) or, for the Turkish letters, from the audit; the other rows count the letters of each document, without the tatweel (Amendment 9). Ԛ (U+051A) and Ԝ (U+051C) look like Latin Q and W.

Figure 1 counts the documents for each of the 34 groups of scripts found; among them, in 9 to 22 documents each, are scripts of ancient Iran and Mesopotamia: Avestan, Inscriptional Pahlavi or Parthian and cuneiform. The findings therefore concern Latin-script Kurmanji web text.



The letters ı, İ, ğ and Ğ belong to the Turkish alphabet but not to the Kurmanji Latin alphabet; they also occur in Turkish proper names, in the Latin orthography of Zazaki and as keyboard artefacts in Kurmanji text, so the documents that contain them are an indicator, not a language classification, and we do not use them to exclude documents. Our tokeniser (Section 4) lower-cases İ to i followed by a combining dot and then splits the word at that dot. The split applies to every copy of a text alike, so it does not hide repetition, but it splits those words in the frequency counts. Lines with the replacement character U+FFFD are kept. Strict decoding cannot detect damage that is valid UTF-8; Table 5 gives the indicators of such damage that Amendment 9 counts, next to what strict decoding finds. Because combining marks are not word characters, the tokeniser also splits Arabic-script words that carry vowel marks.

Table 5. Indicators of encoding damage in the file (exploratory). þ, ý and ð are letters of other alphabets that Kurmanji does not use and what ş, ı and ğ of the Turkish code page become in Latin-1; Ş, with a comma below, and the Cyrillic Ç look like ş and ç; each letter is counted with its capital (Amendment 9).

Indicator	Count	Documents	Documents with three or more
Lines with invalid UTF-8 (strict decoding)	0 lines	—	—
Lines with the replacement character U+FFFD	389 lines	128	—
þ, ý or ð	7,503	951	348
Ş	8,700	838	536
Cyrillic Ç	1,157	274	84
Byte-order mark at the start of a paragraph	2,797 paragraphs	2,790	—
İ, dotted capital I	24,500	11,159	—

Note. Our exact counts treat a byte-order mark as content; once it is removed, only 2 of the paragraphs that begin with one are identical to a paragraph without it. The tokeniser splits words at the dot of İ (Section 4). Count: occurrences, or lines or paragraphs where stated; documents: those with at least one.

Let D be the set of distinct documents with at least five tokens (exact copies removed). Tokens come from Unicode normalisation to form NFC, lower-casing with Python's locale-independent case mapping, and splitting into maximal runs of word characters (letters, digits and the underscore; combining marks are not word characters), all as defined by the Unicode 13.0 database of Python 3.10. Each document d has a shingle set S(d): the word 5-grams of its token stream, each hashed to 64 bits with BLAKE2b (the five tokens joined by the unit separator character and encoded in UTF-8, with an 8-byte digest read as a little-endian integer), so a shingle can span a paragraph boundary. The plan gives no reason for word 5-grams in Kurmanji; one of its sensitivity analyses (S1, Section 5) repeats the calibration with 3-grams. The problem is to find, for $\theta = 0.8$, the set $E^*(\theta)$ of Equation 2: the pairs of distinct documents whose Jaccard similarity is at least θ, the edges of the threshold graph (a similarity self-join). Algorithm 1 verifies each of its candidate pairs C exactly, so it returns $E = C \cap E^*(\theta)$, with no false positives; its recall $R(\theta)$ is the share of $E^*(\theta)$ that it finds, and its miss rate $1 - R(\theta)$. Table 6 lists the notation.

$$E^*(\theta) = \{\{x,y\} \subseteq D : x \neq y, \; J(S(x),S(y)) \geq \theta\} \quad E = C \cap E^*(\theta) \quad R(\theta) = |E| \,/\, |E^*(\theta)| \qquad (2)$$

Algorithm 1. MinHash-LSH near-duplicate detection
Input: documents D; k = 128; b = 16; r = 8; threshold θ = 0.8
Output: verified near-duplicate pairs E and their connected components
1 for each d in D: S(d) ← BLAKE2b hashes of the 5-grams of its tokens
2 for each d: sig(d)_i ← min over v in S(d) of (a_i·v + c_i) mod 2^64, i = 1..k
3 for each band j = 1..b: key_j(d) ← (sig(d)_{(j−1)r+1}, ..., sig(d)_{jr})
4 C ← all distinct pairs within each group of documents with equal keys, over all bands
5 for each (x, y) in C: compute J(x, y) exactly; if J ≥ θ, add (x, y) to E
6 components ← connected components of E (union–find)

Algorithm 1 gives the detector. The signature uses k = 128 linear hash functions $h_i(v) = a_i \cdot v + c_i$ of a shingle hash v, with arithmetic modulo 2^{64}. The a_i are odd 64-bit integers and the c_i integers below 2^{63}, both drawn with NumPy's default generator (PCG64) from the seed fixed in the configuration file of the frozen plan. We use modulo 2^{64} instead of a prime modulus because 64-bit integer overflow performs that reduction natively. This choice changes the hash family, and Amendment 1, written before any calibration result was computed, records it. The keys of the linear maps are themselves BLAKE2b values, so the calibration test checks the composition of the shingle hash and the linear maps. A band key keeps all r values of its band, so two documents share a key only if all r rows agree; candidate pairs are collected over all bands without repetition. Figure 2a and b apply these steps to a toy example.

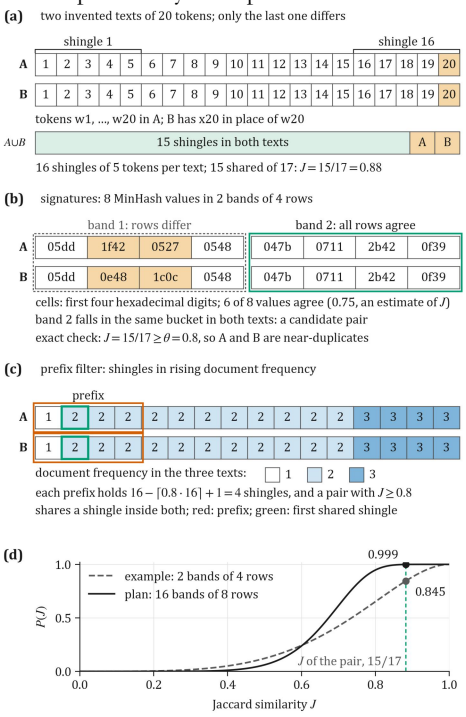


Figure 2. The method step by step on a toy example with invented tokens (Amendment 10).
(a) Two texts of 20 tokens that differ only in the last one (cells: positions of the tokens; shaded: the token that differs); the bar below is the union of their sets of shingles of five tokens: 15 in both texts and one proper to each (J = 0.88). (b) Their signatures of eight values (first four hexadecimal digits; shaded where they differ) fall into two bands of four rows; the second band agrees in every row, so the pair is a candidate, and the exact check keeps it. The detector uses 128 values in 16 bands of 8 rows. (c) The shingles of the same texts in the global order of the prefix filter, by increasing document frequency in a collection of three texts (number and shade of each cell), ties broken by hash value: with θ = 0.8 each set indexes its first 4 elements (red frame), and the first shared shingle (green) lies in both prefixes, as the argument of Section 4 guarantees for every pair with J ≥ θ. (d) Candidate probability P(J) (Equation 3) with the two bands of four rows of the example (dashed) and with the plan's 16 bands of 8 rows (solid); the dots give its value at the J of the pair.

The algorithm has four invariants, checked by property-based tests with Hypothesis [32] and unit tests: adding a shingle cannot increase any signature value (the plan's first invariant, independence from the order of the 5-grams, holds by construction, because S(d) is a set); identical sets have identical signatures, and an empty set has none; two documents share a band key if and only if all of its rows agree; and union–find returns the same components as a breadth-first search on random graphs. Two further tests draw 600 pairs of random sets of about a hundred 64-bit keys, with J near 0.5 and near 0.8, each pair with its own draw of the hash functions, and require the candidate rate to lie within 0.03 of Equation 3; the tolerance is about three binomial standard errors, so these tests catch only gross deviations.

If the k hash functions are drawn independently from a min-wise independent family [25], a pair with similarity s agrees on each signature coordinate with probability s, independently across coordinates, so it agrees on all r rows of a band with probability s^r and becomes a candidate with the probability P(s) of Equation 3. Since $1 - P(J)$ decreases with J, Equation 4 bounds the probability that a pair above the threshold is missed: for $\theta = 0.8$, $r = 8$ and $b = 16$, every such pair is missed with probability at most 0.0530, above the level 0.05, and the bound falls below a level γ only for $J \geq J_\gamma$, which is 0.80177 for $\gamma = 0.05$. The linear family used here is not min-wise independent in the strict sense, and simple linear and multiplicative hash functions can fail badly as min-wise hash functions [33]. Mitzenmacher and Vadhan show that 2-universal hash functions perform essentially as truly random ones when each key carries enough entropy [34]; we have not shown that the linear maps here meet their conditions, but their keys are BLAKE2b values, and the argument would not carry over to pipelines that feed raw token or n-gram identifiers to such maps. Equation 3 is therefore a reference curve, and the calibration test (Section 5) measures how far the detector departs from it.

$$\Pr[\text{sig}_i(x) = \text{sig}_i(y)] = s \qquad \Pr[\text{band } j \text{ agrees}] = s^r \qquad P(s) = 1 - (1 - s^r)^b \tag{3}$$

$$1 - P(J) \approx 1 - J^{rb} \leq 1 - \theta^{rb} \quad \text{if } J \geq \theta \iff 1 - P(J) \leq \gamma \iff J \geq J_\gamma, J_\gamma = 1 - \gamma^{1/b}{}^{1/r} \tag{4}$$

H2 tests a pooled miss rate over the generated pairs with $J \geq 0.8$, so it does not bound the miss probability at each J; Equation 4 gives that value. The curve P(s) is steepest at the point s^* of Equation 5, where its second derivative vanishes, that is, where $(r - 1)(1 - s^r) = (b - 1) r s^r$; for $b = 16$ and $r = 8$ it is 0.696, and s^*, the usual approximation of the threshold [26], is 0.707. The parameters therefore place the steep part of the curve below the threshold. The choice of b and r trades missed pairs against candidates [26]. Fewer rows per band would raise the candidate probability on both sides of the threshold: with 126 of the 128 values, 18 bands of 7 rows would give a miss probability of 0.014 at $J = 0.8$ and a candidate probability of 0.132 at $J = 0.5$, against 0.053 and 0.061 for 16 bands of 8 rows. Section 8 measures that trade-off on the corpus (Amendment 8), and Figure 3 draws P(J) for the bandings it compares.

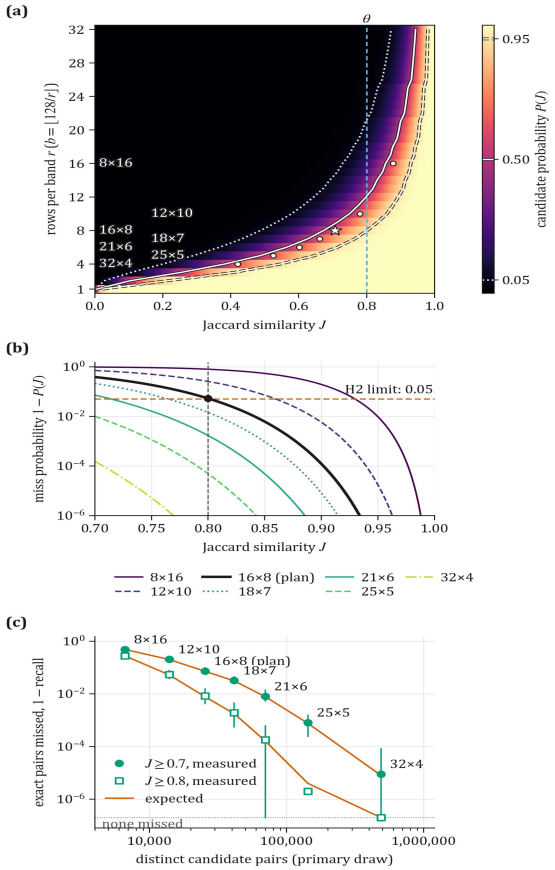


Figure 3. (a) Candidate probability $P(J) = 1 - (1 - J^r)^b$ (Equation 3) for every number of rows per band r from 1 to 32, with $b = \lfloor 128/r \rfloor$ bands, as a heat map, with the curves on which P(J) is 0.05, 0.5 and 0.95, θ, and the usual approximation of the threshold, $s^* \approx (1/b)^{(1/r)}$, of the seven bandings that Section 8 compares (dots; star: the plan's 16 bands of 8 rows). (b) Miss probability $1 - P(J)$ above $J = 0.7$ for those bandings, on a logarithmic scale, with the 0.05 limit

of H2 and θ; the dot marks the plan's banding at θ. (c) Their trade-off on the natural pairs of the file: distinct candidate pairs of the primary draw against the share of the exact pairs with $J \geq 0.7$ (dots) and $J \geq 0.8$ (squares) that the LSH stage misses, as the mean over 221 draws of the linear family, with bars from the 2.5% to the 97.5% quantile (exploratory; Amendment 8), and the share that P(J) predicts (lines); a banding that misses no pair is drawn on the dotted floor.

$$s^* = \left(\frac{r-1}{rb-1}\right)^{1/r}, \qquad \tilde{s} = \left(\frac{1}{b}\right)^{1/r} \tag{5}$$

Comparing all pairs exactly takes $O(n^2)$ pair comparisons. The exact baseline of H3 computes the Jaccard similarity of all pairs as in Equation 6, with M the sparse $|V| \times n$ matrix that marks which of the distinct shingles V each of the n documents contains and $w_x = |S(x)|$; I and J are dense $n \times n$ arrays, so its memory grows with n^2; it is the baseline that the plan fixed, not a competitive exact method. The candidate stage costs $O(k \Sigma |S(d)|)$ for the signatures, with the sum over the documents d, which is linear in the total number of shingles, and $O(nb)$ for the band tables, while the number of verified pairs depends on the data. We do not claim a lower bound for the exact problem. Choosing a minimum set of representatives that covers all near-duplicates is a different covering question, and we do not address it.

$$I = M^{\top}M, \qquad J_{xy} = \frac{I_{xy}}{w_x + w_y - I_{xy}} \tag{6}$$

To measure recall on the natural pairs of the file, we also compute the exact set of pairs with $J \geq \theta$ over all distinct documents with the prefix filter of Section 2. Shingles follow a strict total order: increasing document frequency over the distinct documents, with ties broken by hash value. Each set X is indexed by its first $\lambda(X)$ elements in that order (Equation 7). If $J(X, Y) \geq \theta$, the two sets share at least θ times the size of their union, so Equation 7 holds: all the shared elements lie at or after the first shared one, so the first shared element lies within the indexed prefix of X, and likewise of Y (Figure 2c), and a pair that fails the size condition is skipped. Verifying every other pair that shares an indexed element with the exact J, compared in integer arithmetic, therefore finds every pair above θ among the 64-bit shingle hashes; unit tests compare it with a comparison of all pairs. The prefixes are computed for $\theta = 0.7$, and the pair sets for 0.8 and 0.9 are subsets of that result.

$$\begin{aligned} &\lambda(X) = |X| - \lceil \theta |X| \rceil + 1 \\ &J(X,Y) \geq \theta \;\Rightarrow\; |X \cap Y| \geq \lceil \theta |X| \rceil \\ &\qquad\qquad \text{and } |Y| \geq \theta |X| \end{aligned} \tag{7}$$

5 Study design

Table 3. Corpus audit of the Kurdish (ku) file of CC-100.

Quantity	Value
Documents	236,550
Documents with fewer than five tokens	10,993
Tokens	66,968,643
Non-empty paragraphs	2,482,825
Distinct paragraphs	1,662,246
Excess paragraph occurrences	820,579 (33.1%)
Copies of documents beyond the first	11,634 (4.9%)
Size after decompression (bytes)	420,494,778
Characters	377,868,451
Characters per document: median; mean; maximum	681; 1,607; 596,990
Documents with fewer than fifty characters	26,398

Note. Counts come from streaming passes over the file; paragraphs and documents are compared as exact character sequences.

The analysis plan was written and frozen before any similarity, candidate or cost result was computed; the corpus audit of Table 3, except the token total and the count of documents with fewer than five tokens, which were computed after the primary run, was known at that point, as were the statistics of repeated paragraphs and the hypothesis that the file repeats more than deduplication across whole snapshots allows; the letter counts by script were recomputed and the count of lines with invalid UTF-8 added under Amendment 4. The plan is an internal, version-controlled record: its frozen version is tagged in the repository, and later changes are recorded as dated amendments; Amendment 2 was rewritten once in place (Section 7) and later received a dated note on its history, Amendment 7 received a dated addendum before the analyses it describes were run, and the registration lines of Amendments 5 to 9 received the identifier and time of their commit after registration, and in Amendment 10 the file read by one check was corrected before its script was written and the output narrowed after its first run; no rule or definition changed. There is no public registration, so the order of events rests on the repository history. After the tag, the plan file received a dated log entry and a pointer to Amendment 1; its rules did not change.

Three hypotheses are confirmatory.

H1: In each similarity bin, the expected candidate proportion is equivalent to its theoretical mean with margin δ = 0.03.

For a bin with m pairs, Equation 8 gives the mean residual $\bar{e}$, the observed minus the predicted proportion, where X_j is the candidate indicator of pair j and P_j its probability P(J) from Equation 3; its standard error σ, computed from the Bernoulli variances of the pairs, that is, under exact calibration; the p-value p of the two one-sided normal tests (TOST) [35] at $\alpha = 0.05$, with Φ the standard normal distribution function; and the p-values of the five bins adjusted by Holm's method [36], with the p-values ordered from the smallest. A bin passes if its adjusted p-value is below α; before the adjustment, it passes exactly when the 90% interval of $\bar{e}$ lies inside $\pm\delta$. The plan chose the margin as about 3.3 times the largest standard error of a bin with 3,000 pairs, the bound 0.009 of Equation 8, for power, not from the consequences of a deviation (Section 9). The plan's text calls this test two-sided at $\alpha = 0.05$; with half that level for each one-sided test, every bin would still pass (the largest Holm-adjusted p-value is 0.008). Table 7 lists the notation of this and the later tests and measures.

$$\bar{e} = \frac{1}{m}\sum_j \left(X_j - P_j\right)$$

$$\sigma = \frac{1}{m}\left(\sum_j P_j\left(1 - P_j\right)\right)^{1/2} \leq \frac{1}{2\sqrt{m}} \qquad (8)$$

$$p = \max\left\{1 - \Phi\left(\frac{\bar{e} + \delta}{\sigma}\right), \Phi\left(\frac{\bar{e} - \delta}{\sigma}\right)\right\}$$

$$\tilde{p}_{(i)} = \max_{l \leq i} \min\{1, (6 - l)\, p_{(l)}\}$$

Equation 9 gives the Wilson bounds U and L of a proportion p observed among m pairs, with $z = 1.960$ for a two-sided 95% interval and $z = 1.645$ for a one-sided 95% bound; its last two lines are the bounds when no pair, or every pair, is counted. Because of an arithmetic error, the frozen formulation gave a theoretical miss probability of 0.0486, below 0.05, at $J = 0.8$, so this limit was set in the belief that the parameters met it at the threshold; Amendment 3 corrects the value to 0.0530.

$$U, L = \frac{\hat{p} + \frac{z^2}{2m} \pm z\sqrt{\frac{\hat{p}(1 - \hat{p})}{m} + \frac{z^2}{4m^2}}}{1 + z^2/m} \qquad (9)$$

$$U|_{\hat{p}=0} = \frac{z^2}{m + z^2}$$

$$L|_{\hat{p}=1} = \frac{m}{m + z^2}$$

H3: The log-log slope of the candidate stage lies in [0.8, 1.3] and that of the exact baseline in [1.8, 2.2], both with 95% confidence intervals inside those ranges.

Each slope $\varkappa$ is a least-squares fit of Equation 10 to the logarithms of the median times T_n, with the interval of its second line, where t is the 97.5% quantile of Student's t with $q - 2$ degrees of freedom for q points: 9 for the candidate stage and 4 for the exact baseline.

$$\ln T_n = \kappa_0 + \kappa \ln n + \varepsilon_n$$

$$\kappa \in \hat{\kappa} \pm t\,\mathrm{se}(\hat{\kappa}) \qquad 10)$$

Similarity is controlled as follows. Two thousand base documents with at least 200 tokens are chosen with a fixed seed, and variants are generated for them in turn. In each variant a fraction f, drawn uniformly from [0, 0.6], of the token positions is chosen at random, and each chosen token is replaced by a token drawn at random from one random document of the base pool, which may be the base itself. The plan gives no reason for this range, and 131,163 of the 153,162 variants fell outside the bins. Base documents are drawn from all documents with at least 200 tokens, so two bases can be copies of each other. The Jaccard similarity of each base–variant pair is computed exactly, and the pair joins one of the five bins [0.45, 0.55), [0.55, 0.65), [0.65, 0.75), [0.75, 0.85) and [0.85, 0.95), until each bin has 3,000 pairs or 300,000 variants have been generated. The plan sets a limit of 60,000 variants per bin and declares a bin short if it does not reach 3,000 pairs within it; the code, written before the run, pooled the five limits into one budget, because a single stream of variants feeds all the bins. With 60,000 variants in all, every bin would have stayed short, as in the replica described in Section 7 (1,100 to 2,477 pairs per bin). Pairs outside the bins are counted and those arriving after their bin is full are discarded; neither enters H1; H2 uses every generated pair with $J \geq 0.8$. Each base document received 76 or 77 variants, and the pairs of each bin come from 1,544 to 1,583 distinct base documents. Scattered single-token replacement at a fixed length is a benign model of near-duplication, a point we return to in Section 9.

As a validity check, the plan requires 3,000 pairs of distinct documents with $J < 0.05$ (the code drew them at random from the base pool) to produce candidates at a rate of at most 0.001 (upper 95% Wilson bound, Equation 9). Equation 3 gives these pairs a candidate probability below 10^{-9}, so the check can detect only gross faults. Apart from counting the documents with at least 200 tokens, no check of whether the plan's rules could be met was run before the freeze. Amendment 1 and the calibration code were written before a short test run of the calibration, with 60 pairs per bin and 2,000 variants, which checked that the code runs (with that number of variants the bins with $J \geq 0.65$ did not fill), and were committed together half a minute after it; the cap on variants was already in that code, which has not changed since, and a clean clone reproduces the confirmatory run (Section 10). The plan foresaw a calibration pilot that would compute no candidate proportion; the test run computed the statistics of the analysis on its small sample. Its numbers are not results and did not modify the plan.

orthographic variants. The levels overlap: a copied document also adds paragraph occurrences.

The repetition also bounds how the file was deduplicated. CCNet removes duplicated paragraphs across the web pages of a snapshot and can compare paragraph hashes as far as the whole snapshot [4], and the file comes from monthly snapshots of one year (Table 12). If each month contributed one snapshot, deduplication across each whole snapshot would leave at most 12 occurrences of any paragraph, $n_p \leq 12$, because paragraphs that are identical in the file are also identical after CCNet's normalisation, so the occurrences beyond the twelfth, O_{12} in Equation 11, would be zero; O_1 is the count of excess occurrences of Section 3, and the second sum, which weights each paragraph by its tokens ℓ_p, counts their tokens. Yet thousands of distinct paragraphs occur more than 12 times (Figure 5b), and Table 13 gives O_{12} and its tokens, which no such deduplication would leave; they weigh less in tokens than in occurrences, because these paragraphs are short, and the most repeated one occurs 10,757 times. The bound does not rest on short lines alone: it also holds for the paragraphs with at least 100 characters (Amendment 8). Under that premise, the file is not deduplicated across whole snapshots: either CCNet deduplicated within a narrower scope, or rebuilding the file from the URLs and paragraph indices added repetition. We cannot tell which, because the intermediate CCNet data are not part of the public file. The file carries no URLs, and we did not try to recover them from the CCNet repository, so the sites behind the repetition are unknown.

$$O_u = \sum_p \left(n_p - u\right)^+, \qquad \sum_p \ell_p \left(n_p - u\right)^+ \qquad 11)$$

Table 8. Amendments to the frozen analysis plan, in the order written.

No.	When written, and content	Rule
1	After the freeze: modulo 2^{64} instead of a prime modulus	Hash family
2	After the primary run: bound relaxed to 0.002; rewritten after the replica as a one-sided reading of 0.001, with the two-sided one also reported	Validity check
3	After the primary run and the replica: miss probability at $J = 0.8$; the corpus analysis uses the 225,557 documents with at least five tokens (215,536 distinct), not 236,550	None
4	After an internal audit: wording; the last cost point re-labelled with $n = 225{,}557$, which moves the slope of H3 from 1.053 to 1.058 and keeps its verdict; letters by script; S3 replaced by a clustered variance and the approximation ratio dropped, both restored by Amendment 5	None
5	After the primary results: S1; S3; recall in subsamples; second seed; exact S2 and recall; clean-clone reproduction	Adds rules
6	After the results of 5: twenty hash draws; recall expected from Equation 3	Adds a 95% rule
7	After an internal review: miss rates by stratum; pair-level calibration; agreement by coordinate and band; low-similarity pairs; power of S3; 90% intervals; repetition in tokens and the bound of 12 occurrences; frequencies over a fixed vocabulary; profile of repeated paragraphs and scripts; a dated addendum: the bound in tokens and the combined removal	None
8	After a second internal review: power of the robustness rules; more hash draws and a fully random control; other bands; cost on the same input; corpus checks	None
9	After a third internal review: encoding damage; script by document	None
10	After the reviews: data of four figures (word frequencies, document lengths, power of H1, a toy example of the method)	None

Note. Rule = the decision rule that the amendment changes or adds.

Table 8 lists the ten amendments. Amendments 1 to 3 concern how the plan is computed or read; Amendment 2, which decides the validity check, was written after its result was known and rewritten once (Section 7). Amendments 4 to 10 change no confirmatory rule, but Amendments 5 and 6 add decision rules for later analyses. S1, S2 and S3 are sensitivity analyses of the plan: S1 uses shingles of three words, S2 varies the threshold over 0.7, 0.8 and 0.9 for the corpus counts, and S3 resamples base documents with replacement, a bootstrap [38] by clusters [39]. Amendment 5 computes the S2 counts exactly and adds the recall of the detector. Amendments 5 to 10 were written after the primary results were known and before the analyses they describe were run; Amendments 7 to 9 follow internal reviews of drafts of this paper; the last review had already reported two of the counts of Amendment 9, and Amendment 10 adds the data of four figures. None of

these analyses replaces the confirmatory run. Two outputs registered in Amendment 8 were not computed: the standard error and t interval of the recall by range of J over the twenty draws of Amendment 6 (its result file has the means), and the rank of the primary draw among those 21 draws in the range 0.70–0.80. In their place, Section 7 plots the mean and the 2.5% and 97.5% quantiles of the recall in each range over the 220 further draws and gives the rank of the primary draw among 221.

Exploratory analyses are marked as such: the repetition counts and lengths, the change in word-frequency ranking after removing near-duplicates, the script markers and the year mentions. For the frequency ranking, the comparison as first coded, over the common 5,000 most frequent word types (a choice of the analysis code, not of the plan), gives a Spearman correlation of 0.9968; Amendment 7 replaced it with a fixed vocabulary (Section 6). The only formal test reported outside H1 to H3 is the Wald test of Section 7, which is not adjusted; Sections 7 and 8 also state how many standard errors some means over hash draws lie from Equation 3 or from each other, comparisons that were not registered and are descriptive. The plan's Benjamini–Hochberg correction for exploratory tests is not used.

6 Repetition in the corpus

Table 9 counts the repeated text at each level, in documents and in tokens (RQ1; exploratory). Paragraph occurrences beyond the first of each paragraph hold 22.5% of the 66,968,643 tokens of the file, far more than any other level. Copies of whole documents beyond the first hold 5.5%. After exact copies and documents with fewer than five tokens are set aside, 215,536 distinct documents remain. Among them, the exact pairs with J ≥ 0.8 form 3,744 groups of near-duplicates, the largest with 47 documents; 2,823 of them are pairs (Figure 4a). Keeping the first document of each group would remove 5,213 documents, 2.4% of the distinct ones but 7.2% of the tokens of the file: the removed documents have 923 tokens on average, against 294 for all distinct documents (Figure 4b).

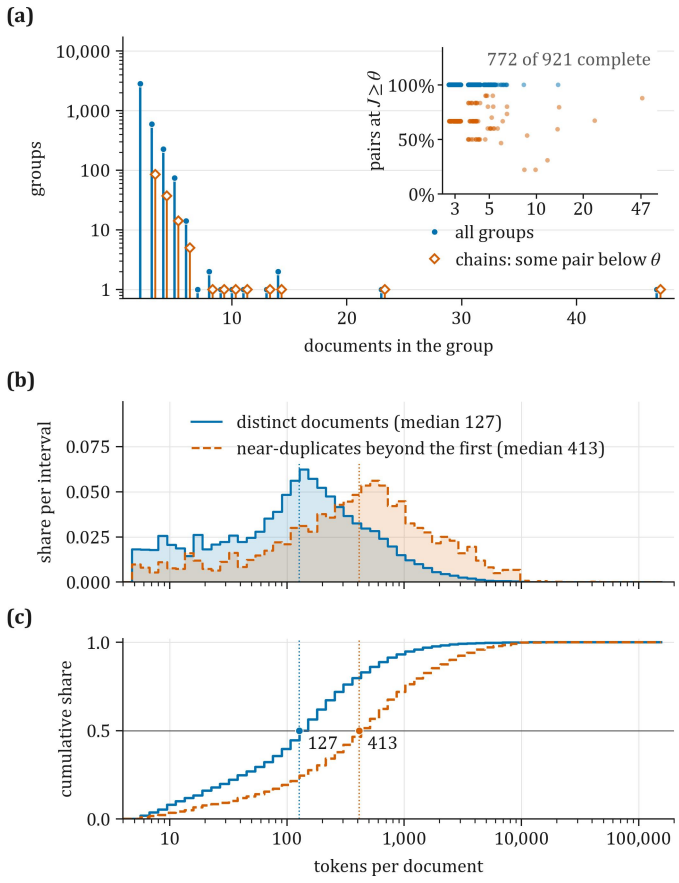


Figure 4.

Figure 4. (a) Groups of near-duplicates at J ≥ 0.8 by number of documents (3,744 groups; exploratory), and among them the chains (diamonds): groups in which some pair of documents has J below θ and is joined only through other members. Inset: share of the pairs of documents of each group of three or more with J ≥ θ, from the exact pairs, against the size of the group (sizes jittered). (b) Tokens per document, as the share of each set in intervals of a quarter of a power of two (Amendment 10): all 215,536 distinct documents with at least five tokens (median 127) and the 5,213 near-duplicates beyond the first of each group (median 413). (c) The same sets as cumulative shares; dots and dotted lines mark the medians.

The groups are connected components, so a group can join documents whose own similarity is below θ: in four of the five largest groups two members have J below θ, the lowest 0.676. Keeping one document per group is the largest removal in documents that the pairs support. Which document is kept, whether the pairs come from the exact join or from the LSH stage, and the draw of hash functions change the removal little (Table 10; Amendment 8). Folding ê, î, û, ç and ş to e, i, u, c and s and repairing İ merges spellings with and without these letters and adds pairs at both thresholds (Table 11); folding

can also merge distinct words, so the extra pairs only roughly indicate orthographic variants. The levels overlap: a copied document also adds paragraph occurrences.

Table 10. Near-duplicate documents removed at J ≥ 0.8 under different choices (Amendment 8; exploratory).

Pairs and choice	Documents	Tokens	Share
Exact pairs, keeping the first of each group (Table 9)	5,213	4,810,440	7.18%
Exact pairs, keeping the longest	5,213	4,740,365	7.08%
Exact pairs, keeping the shortest	5,213	4,870,998	7.27%
Exact pairs, greedy policy	5,091	4,753,709	7.10%
LSH pairs of the primary draw, keeping the first	5,195	4,800,564	7.17%
LSH pairs, mean of 221 linear draws	5,188 (5,173 to 5,200)	—	—
LSH pairs, mean of 200 fully random draws	5,188 (5,168 to 5,201)	—	—

Table 11. Spelling variation among near-duplicate pairs (Amendment 8; exploratory).

Pairs among distinct documents	Without folding	With folding
Pairs with J ≥ 0.7	12,076	12,220
Documents in them	12,358	12,522
Pairs with J ≥ 0.8	8,119	8,219
Documents in them	8,957	9,080
Pairs with identical shingle sets	415	446
… differing only in letter case	81	—
… also in spacing or line breaks	26	—
… also in punctuation or other non-word characters	307	—
… with the same 5-grams in another order	1	—

The repetition also bounds how the file was deduplicated. CCNet removes duplicated paragraphs across the web pages of a snapshot and can compare paragraph hashes as far as the whole snapshot [4], and the file comes from monthly snapshots of one year (Table 12).

Table 12. Deduplication in the pipelines behind the corpora discussed here, as their sources describe it, and in this study. —: not described in the passages we verified.

Pipeline or study	What is compared	Normalisation before hashing	Scope
CCNet [4]	Paragraphs, by a hash of each	Lower-casing; numbers replaced by a placeholder; Unicode punctuation and accent marks removed	A shard, N shards or the whole snapshot of 1,600 shards; duplicated paragraphs are 70% of the text
CC-100 [2, 3]	Rebuilt from the URLs and paragraph indices of the CCNet repository	That of CCNet	Twelve Common Crawl dumps for the languages other than English, January to December 2018
mC4 [13]	Lines duplicated across documents	—	—
This study	Exact documents and paragraphs; near-duplicate documents, J ≥ 0.8 of word 5-grams	None for the exact comparisons; Unicode NFC and lower-casing for the 5-grams	The whole ku file

If each month contributed one snapshot, deduplication across each whole snapshot would leave at most 12 occurrences of any paragraph, $n_p \leq 12$, because paragraphs that are identical in the file are also identical after CCNet's normalisation, so the occurrences beyond the twelfth, O_{12} in Equation 11, would be zero; O_1 is the count of excess occurrences of Section 3, and the second sum, which weights each paragraph by its tokens ℓ_p, counts their tokens. Yet thousands of distinct paragraphs occur more than 12 times (Figure 5b), and Table 13 gives O_{12} and its tokens, which no such deduplication would leave; they weigh less in tokens than in occurrences, because these paragraphs are short, and the most repeated one occurs

10,757 times. The bound does not rest on short lines alone: it also holds for the paragraphs with at least 100 characters (Amendment 8). Under that premise, the file is not deduplicated across whole snapshots: either CCNet deduplicated within a narrower scope, or rebuilding the file from the URLs and paragraph indices added repetition. We cannot tell which, because the intermediate CCNet data are not part of the public file. The file carries no URLs, and we did not try to recover them from the CCNet repository, so the sites behind the repetition are unknown.

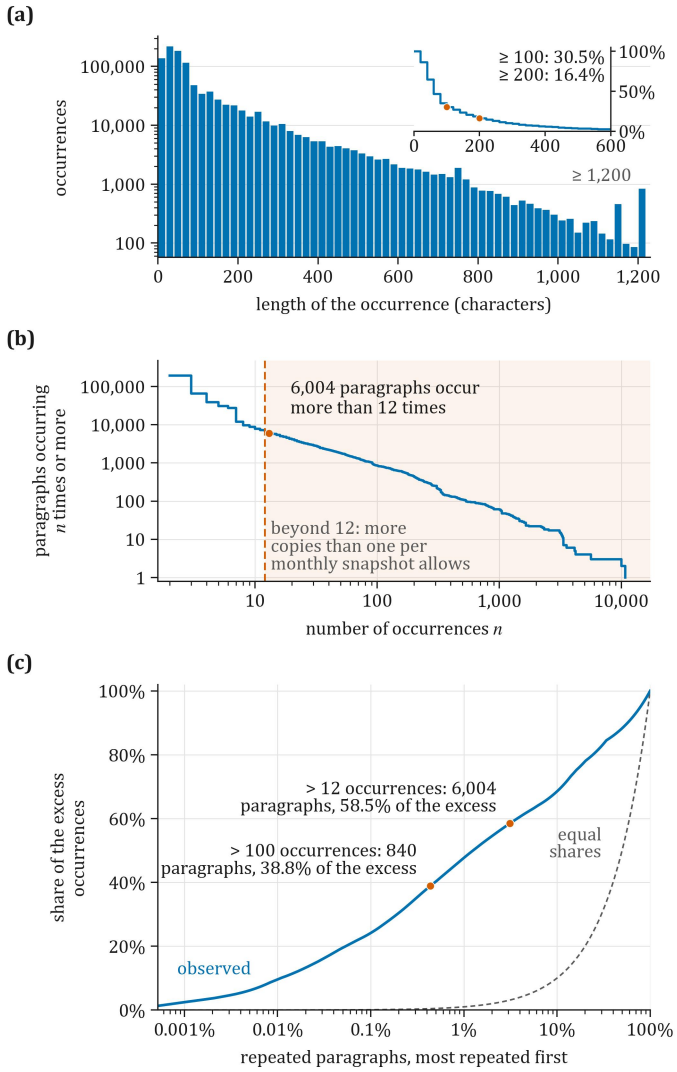


Figure 5.

Figure 5. (a) Lengths of all occurrences of repeated paragraphs, the first included (exploratory): number of occurrences in each interval of 20 characters (Unicode code points of the line as stored, without normalisation), on a logarithmic scale; the last bar collects the occurrences of 1,200 characters or more. Inset: share of those occurrences at least as long as each length, with the shares at 100 and 200 characters. (b) Number of distinct paragraphs that occur n times or more, for the 192,466 paragraphs that occur more than once, on logarithmic axes; the dashed line marks 12 occurrences, the most that deduplication within each monthly snapshot allows, and the shaded region lies beyond it. (c) Share of the excess occurrences (beyond the first of each paragraph) held by the most repeated paragraphs, against the share of the repeated paragraphs that they are (logarithmic axis), with equal shares (dashed) and the paragraphs that occur more than 100 and more than 12 times (dots; 38.8% and 58.5%).

Table 13. Paragraphs that occur more often than one copy per monthly snapshot allows (exploratory).

Paragraphs	Distinct	Excess occurrences	Beyond the twelfth	Tokens beyond the twelfth
Occurring more than 12 times	6,004	479,804 (58.5%)	413,760 (50.4%)	3,723,457 (24.8%)
… with at least 100 characters	730	50,757 (6.2%)	42,727 (5.2%)	—
Occurring more than 100 times	840	318,793 (38.8%)	309,553 (37.7%)	2,611,306 (17.4%)

Note. Excess occurrences are those beyond the first of each paragraph; shares are of all excess occurrences of paragraphs or, for tokens, of their tokens. The row of long paragraphs is from Amendment 8.

$$O_u = \sum_p \left(n_p - u\right)^+, \qquad \sum_p \ell_p \left(n_p - u\right)^+ \qquad (11$$

The most repeated paragraphs are short (Table 14). Compared with the paragraphs that occur once, more of them have fewer than three tokens, a URL or e-mail marker or the copyright sign, and fewer of them contain digits; the letters ê, î and û and the letters of the Turkish alphabet are as common in both, so they do not distinguish repeated text, and none of the most repeated paragraphs contains Arabic-script letters. The most repeated paragraph has 14 characters and 2 tokens; the most repeated document, which occurs 870 times among the documents with at least five tokens, has 2 paragraphs and 7 tokens. Counting every occurrence of a repeated paragraph, the first included, the lengths have a long tail (Figure 5): 30.5% of these occurrences have at least 100 characters and 16.4% at least 200. These are descriptors computed from the text; we did not read the paragraphs to classify them.

Table 14. Paragraphs that occur at least 100 times and paragraphs that occur once (exploratory).

Paragraphs	100 occurrences or more	One occurrence
Number	850	1,469,780
Median length (characters)	40	129
With fewer than three tokens	8.8%	0.4%
With a URL or e-mail marker	0.7%	0.3%
With the copyright sign	1.2%	0.01%
With a digit	23.4%	29.2%
With ê, î or û	99.5%	99.8%
With ı, İ, ğ or Ğ	1.5%	1.6%
With Arabic-script letters	0.0%	0.3%

Note. Shares of the paragraphs of each group; lengths in Unicode code points of the line as stored.

Removing repeated text changes the frequency ranking of common words little at the level of documents and more at the level of paragraphs (RQ3; exploratory; Table 15 and Figure 6).

Table 15. Word frequencies after each kind of removal (RQ3; exploratory).

Removal	Tokens left	Spearman	Top 1,000
Copies of documents beyond the first	63,304,525	0.9961	98.7%
Copies and near-duplicates, among the distinct documents	58,463,725	0.9899	98.2%
Paragraph occurrences beyond the first	51,933,233	0.9584	95.0%

The types whose share falls most when repeated paragraphs are removed (Table 16) include words of licence notices, of web pages and of what reads like a news site's description of itself; each of these 20 types loses more than 95% of its share, so they occur almost only in repeated paragraphs. They suggest that the licence notices of copied pages and the self-descriptions of news sites contribute to the repetition; we did not check this by reading.

Table 16. The 20 word types whose relative frequency falls most when the occurrences of repeated paragraphs beyond the first are removed (exploratory).

Type	Gloss	Occurrences	Kept
awrosoft	—	1,541	0.08%
a3yê	—	1,337	0.10%
sharealike	—	3,244	0.16%
bêterefî	'impartiality'	3,312	0.23%
attribution	—	3,252	0.44%
mîsogerkirî	—	2,187	1.30%
parola	'password'	2,348	1.87%
blogerê	'blogger'	2,384	2.06%
ferîbot	—	1,288	2.50%

Type	Gloss	Occurrences	Kept
maîlê	‘mail’	1,324	2.63%
sida	—	2,734	2.74%
ensîklopediya	‘encyclopedia’	3,289	3.06%
copyright	—	4,138	3.12%
ipsyê	—	2,770	3.21%
dihaber	—	1,316	3.23%
(personal name)	—	3,725	3.50%
feed	—	2,403	3.76%
(personal name)	—	1,310	4.13%
duristî	‘honesty’	3,428	4.55%
mîrzengî	—	1,697	4.64%

7. The detector against the exact pairs and in the calibration test

After removing documents with fewer than five tokens and exact copies, the file has 215,536 distinct documents. Among them, the exact join finds 12,076 pairs with $J \geq 0.7$, 8,119 with $J \geq 0.8$ and 4,434 with $J \geq 0.9$ (S2, exploratory); the prefix filter proposes 32,057 pairs and verifies the 25,301 that pass the size condition. With the primary draw of hash functions, the LSH stage yields 25,426 distinct candidate pairs and finds 8,084 of the pairs with $J \geq 0.8$ (Table 17).

Table 17. Exact pairs among distinct documents by Jaccard threshold (prefix filtering; S2, exploratory) and the pairs that the LSH stage (b = 16, r = 8) finds: with the primary draw of hash functions, and as a recall over 220 further draws of the same family (Amendments 6 and 8; mean, with the 2.5% and 97.5% quantiles rounded outwards).

Threshold	Exact pairs	Found (primary draw)	Recall over draws	Expected
$J \geq 0.7$	12,076	11,434 (0.947)	0.926 (0.901 to 0.943)	0.928
$J \geq 0.8$	8,119	8,084 (0.996)	0.992 (0.984 to 0.996)	0.992
$J \geq 0.9$	4,434	4,434 (1.000)	1.000 (1.000 to 1.000)	1.000

Note. Expected: mean of P(J) over the exact pairs (Equation 3).

That draw was generated in the same way as the others, from the seed fixed in the plan. Amendment 6 compared it with twenty further draws of the same family, and Amendment 8 with 200 more and with 200 draws of fully random hashing, which gives every distinct shingle an independent random value in each of the 128 functions and so meets the assumption of Equation 3 exactly. Over the 220 further draws of the linear family, the recall at 0.8 has mean 0.9917, with 95% of the draws between 0.984 and 0.996, against 0.9921 for the random family and 0.9922 from Equation 3, the mean of P(J) over the exact pairs; at 0.7 the three values are 0.9263, 0.9275 and 0.9279, and at 0.9 nearly every pair is found in every draw. In comparisons that were not registered, the two families differ by at most 1.5 standard errors of their means, and the recall of the linear family is slightly below what Equation 3 predicts, by 0.0016 at 0.7 and 0.0005 at 0.8 (2.3 and 2.5 standard errors), while the random family lies 0.7 and 0.5 standard errors below.

On the natural pairs of the file, the primary draw is favourable at lower similarities: in the range 0.70–0.80 it finds 0.847 of the pairs, against a mean of 0.792 over the linear draws, and no draw of the random family reaches it (the best finds one pair fewer). Its excess there comes mostly from one group of near-duplicates, here a connected component of the pairs with $J \geq 0.7$: the group with the largest excess, with 72 documents and 1,014 pairs in the range, accounts for 58.1% of it, and the ten with the largest excess, with 197 documents, for 81.3%; the pairs of one group tend to be found or missed together. Figure 7 shows the ranges with the spread of both families. The ratio ϱ of Equation 12 between the standard deviation σ of the recall at 0.8 over the draws and the standard deviation ω that independent pairs, each found with its own probability P_j, would give is 3.1 for the linear draws and 2.8 for the random ones, so most of this over-dispersion does not come from the linear maps: pairs that share documents or shingles tend to be found or missed together. The spread over draws, not a binomial interval, therefore describes the uncertainty of one draw.

$$\varrho = \tilde{\sigma}/\omega$$

$$\omega = \frac{1}{|E^*(0.8)|}\left(\sum_{j \in E^*(0.8)} P_j\,(1 - P_j)\right)^{1/2} \qquad (12)$$

At θ = 0.8, what a draw removes varies little between draws and between the two families of hashing (Table 10). Identical shingle sets, which share every band in every draw, are few: exact matching after normalising letter case, spacing and punctuation would find only 5.1% of the pairs with J ≥ 0.8 (Table 11).

A second implementation of the exact join, which reuses our tokenisation, shingle hashing and corpus reader, obtained the same three counts, both with its own prefix filter and with a comparison without filters of the 128,877,820 pairs that share at least one shingle; Section 8 gives its times.

Table 18. Checks fixed in the frozen analysis plan and their outcome in the primary run.

Check	Criterion (frozen plan)	Observed	Status
H1, calibration	TOST, margin 0.03, Holm	five of five bins	Supported (Am. 2)
H2, miss rate for J ≥ 0.8	Upper 95% bound ≤ 0.05	0.0090 two-sided; 0.0086 one-sided	Supported (Am. 2)
H3, cost scaling	Both slopes in range, 95% intervals	Slopes 1.06 (1.03–1.09) and 1.08 (0.94–1.23)	Not supported
Validity check	Upper 95% Wilson bound ≤ 0.001	0 of 3,000; 0.00128 two-sided, 0.00091 one-sided	Fails; holds one-sided (Am. 2)

Note. H1 and H2 count as supported only under the one-sided reading of the validity check adopted after its result (Amendment 2). Sensitivity and robustness analyses are reported later in this section.

Table 19. Calibration by similarity bin (H1), with 3,000 pairs per bin and the primary draw of hash functions.

Bin of J	Obs.	Pred.	Diff. [90% interval]	p (Holm)
[0.45, 0.55)	0.055	0.064	−0.009 [−0.016, −0.001]	<0.001
[0.55, 0.65)	0.234	0.242	−0.008 [−0.020, +0.005]	0.003
[0.65, 0.75)	0.590	0.599	−0.009 [−0.023, +0.005]	0.008
[0.75, 0.85)	0.930	0.929	+0.001 [−0.007, +0.008]	<0.001
[0.85, 0.95)	0.999	0.999	0.000 [−0.001, +0.001]	<0.001

Note. Obs. = observed proportion; Pred. = mean theoretical proportion; Diff. = Obs. minus Pred. ($\bar{e}$ of Equation 8), with its 90% interval in brackets (before Holm's adjustment, the two one-sided tests pass exactly when it lies inside ±0.03); p = Holm-adjusted p-value of the equivalence test.

Table 18 summarises the checks of the plan, and Table 19 reports the five bins. The equivalence test with margin 0.03 is passed in five of the five bins. The validity check gives 0 candidates among 3,000 unrelated pairs. The plan's rule requires an upper 95% Wilson bound of at most 0.001 and declares the result not valid otherwise. The two-sided bound, which the code implemented before the run, is 0.00128, so the rule fails; with 3,000 pairs no outcome could have met it, since that is the bound $z^2/(m + z^2)$ of Equation 9 for zero candidates. Amendment 2, written after the result, first relaxed the bound to 0.002. After the second implementation of the calibration pointed out that the plan does not say whether the bound is one- or two-sided, the amendment was rewritten as a one-sided reading of the original 0.001, under which the bound, with z = 1.645, is 0.00091, rounded up, and the rule holds. Both remedies were chosen after the result. H1 and H2 are therefore supported only under that reading. For H2 the code reports the two-sided bound; the rule holds under both readings. In the primary draw the three lower bins fall below the curve by 0.008 to 0.009, and the 90% interval of the lowest bin excludes zero (Table 19); the robustness analyses below indicate that this deficit is consistent with chance variation of the draw on this sample. Figure 8 shows the observed proportions against the theoretical curve.

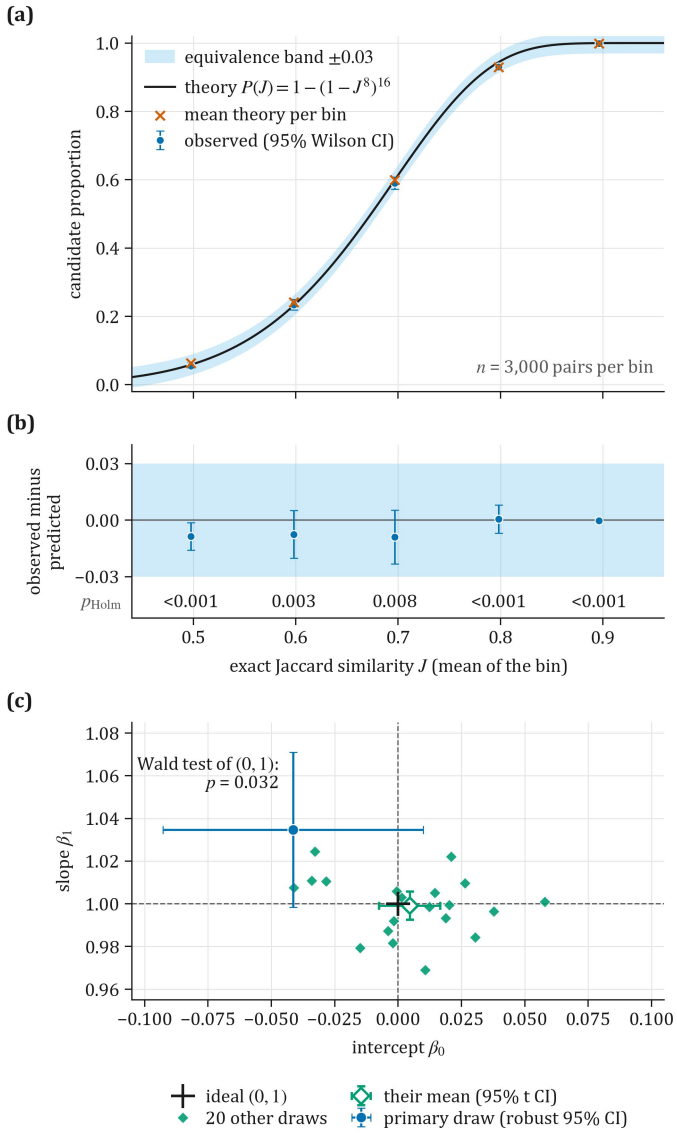


Figure 8.

Among the 6,052 generated pairs with J ≥ 0.8, 40 were missed. The pooled miss rate is 0.0066, with a 95% Wilson interval from 0.0048 to 0.0090, and H2 is supported under its pooled criterion. The pooled rate depends on the mix of similarities in the pool: Equation 4 predicts 43.5 misses for these pairs, and over the twenty draws the count ranges from 36 to 52. Near the threshold the miss rate is far higher than the pooled one: 24 of the 688 pairs with J from 0.80 to 0.82 were missed (0.035; Wilson interval 0.023 to 0.052), where Equation 4 predicts 0.038; over the twenty draws, the misses in that stratum average 28.1 (21 to 36) against 26.4 expected. The observed rate there agrees with Equation 4; whether the data exclude 0.05 depends on the reading of the bound, since the two-sided upper bound, 0.052, is above 0.05 and the one-sided one, 0.049, below it. The criterion of H2 concerns the pool, not this stratum. Averaged over the draws the miss rate there is 0.041, and Equation 4 gives 0.038 for the stratum and 0.053 at J = 0.8. The strata were fixed in Amendment 7, after the primary results. In the next two strata, up to J = 0.90, 12 and 4 pairs were missed against 13.9 and 3.2 expected, and above 0.90 none. Figure 9 shows the observed rate in each stratum against the curve. The detector misses about what the curve predicts at each similarity, and the pooled rate is not the miss rate at the threshold, which is 0.0530 in theory.

Table 20. Sensitivity, robustness and replication analyses of the primary results.

Analysis (basis)	Estimate	Reading
S1, shingles of three words (plan; rule in Am. 5)	five of five bins; H2 bound 0.009	Verdicts kept
Second sampling seed (Am. 5)	five of five bins; H2 bound 0.010	Verdicts kept
S3 bootstrap, H1 (plan; rule in Am. 5)	All bins pass in 65.9% of replicates	95% rule not met; out of reach (power below 95%)
S3 bootstrap, H2 (plan; rule in Am. 5)	H2 met in 100.0% of replicates	Robust

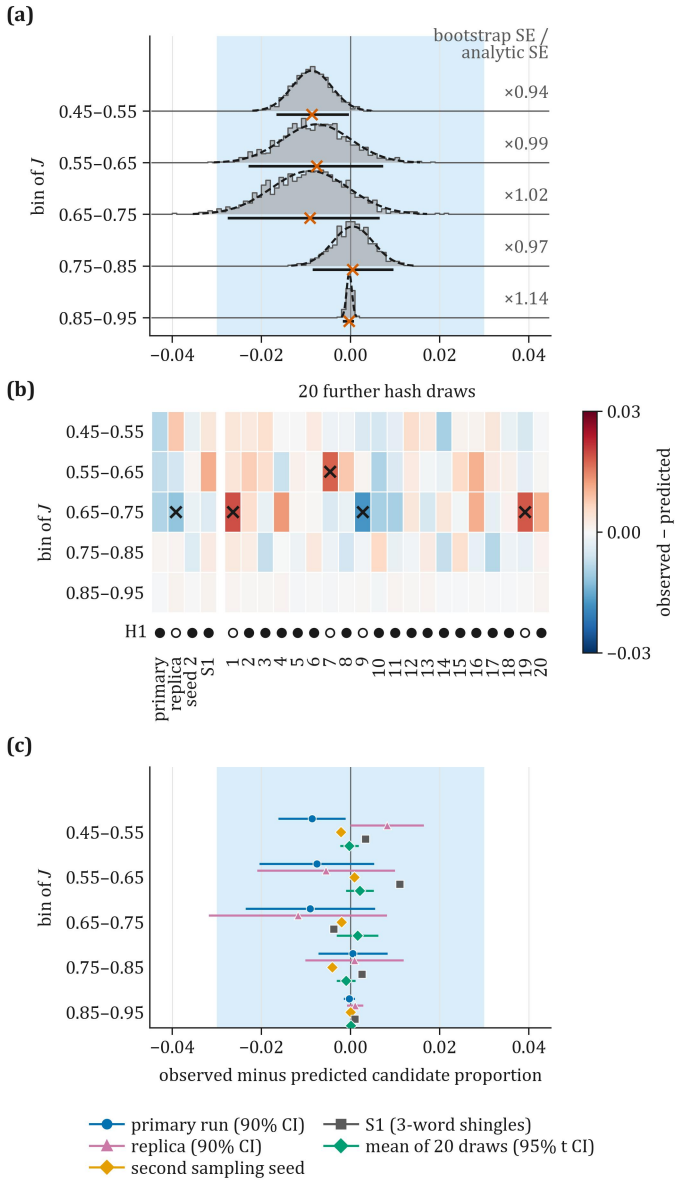


Figure 10.

Figure 10. (a) Bootstrap distribution (S3, 1,000 replicates by base document) of the observed minus predicted candidate proportion in each bin (histogram), with the normal distribution of the analytic standard error around the primary value (dashed), the range from the 2.5% to the 97.5% percentile (bar), the value of the primary run (cross), the ratio of the bootstrap to the analytic standard error, and the equivalence margin ±0.03 (shaded). (b) The same difference in each bin and in each of the 24 calibration analyses: the primary run, the independent replica, the second sampling seed, S1 (shingles of three words) and the 20 further hash draws, on a diverging scale saturated at the margin; crosses mark the bins that fail their equivalence test, and the dots below whether H1 holds (filled) or not (open). (c) The estimates with their intervals: the primary run and the replica with 90% intervals, the second seed, S1, and the mean of the 20 draws with its 95% t interval (Amendments 5, 6 and 8).

Table 20 collects the robustness and sensitivity analyses. S1 and the second sampling seed keep the primary draw of hash functions; S1 changes the shingles, the second seed draws new pairs, and both keep the verdicts of H1 and H2; Figure 10b shows the difference of each bin in these and the other calibration analyses. The bootstrap by base document (S3) is less favourable to H1: all five bins pass in 65.9% of the 1,000 replicates (Figure 10a), below the 95% criterion of Amendment 5, while H2 holds in 100.0%. This criterion could not be met: in a normal approximation with the analytic standard errors of Table 19, Holm's adjustment over the five bins and 100,000 replicates, a detector whose five differences were exactly zero would pass in all bins in only 90.7% of repetitions, so the bootstrap fraction cannot be expected to reach 95% whatever the detector. With the observed differences the figure is 69.2%, close to the bootstrap fraction; being a function of the observed differences, it adds no evidence beyond Table 19 [40]. The bootstrap standard errors are close to the analytic ones in every bin (Table 21), so resampling base documents adds little variance.

Table 21. Standard error of the observed minus predicted candidate proportion in each bin, estimated in four ways, and how often each bin passes over the twenty draws (exploratory).

Bin of J	[0.45, 0.55)	[0.55, 0.65)	[0.65, 0.75)	[0.75, 0.85)	[0.85, 0.95)
Analytic (plan)	0.0044	0.0077	0.0087	0.0046	0.0006
Clustered (Equation 13)	0.0042	0.0076	0.0088	0.0046	0.0007
Bootstrap (S3)	0.0042	0.0076	0.0088	0.0044	0.0007
Over the twenty draws	0.0039	0.0061	0.0094	0.0040	0.0005
Draws in which the bin passes	20	19	17	20	20
Replica: pairs	2,477	2,060	1,557	1,430	1,100
Replica: analytic	0.0049	0.0092	0.0120	0.0066	0.0010

Note. Analytic: the formula of the plan, with independent pairs; clustered: Equation 13, by base document; bootstrap: S3, replicates by base document; over the draws: standard deviation of the difference over the twenty further hash draws (Amendment 6), with the number of those draws in which the bin passes its test. Replica: pairs and analytic standard error of the second implementation (Amendment 8).

Twenty further draws of the hash functions (Amendment 6) give the same picture. The rule of H1 holds in 16 of the 20 draws, short of the 95% that Amendment 6 requires, and the bin 0.65–0.75 fails most often; H2 holds in all of them. That rule could hardly be met either: with a pass probability of 90.7% per draw, an exactly calibrated detector would meet it with probability 0.433, and 16 or fewer passes have probability 0.109 (Amendment 8), so the observed count is compatible with such a detector. Both power calculations were defined after these rules had failed.

Averaged over the draws, the differences of the five bins lie between −0.0010 and +0.0022 (extremes rounded outward), and their 95% t intervals lie within ±0.006; the intervals condition on the pairs of the sample. The spread between draws is of the size of the analytic standard error, and the bins that fail are those with the largest error (Table 21). The deficit of the primary draw in the lower bins is consistent with chance variation of one draw on one sample: the second sampling seed, which keeps that draw and draws new pairs, gives −0.002, +0.001 and −0.002 in those bins, and among the 21 draws the primary one has the largest standardised deficit there (Table 22).

Table 22. Where the primary draw of hash functions stands among the other draws (Amendments 6 to 8; exploratory).

Measure	Rank of the primary draw	z	Other draws as high or higher
Standardised deficit in the three lower bins	21st of 21 (largest deficit)	−2.3	—
Intercept of the pair-level logistic regression	21st of 21 (smallest)	−1.8	—
Slope of the pair-level logistic regression	1st of 21 (largest)	2.5	—
Recall, J ≥ 0.8	7th of 221 linear	1.3	2.73% of linear; 4.50% of random
Recall, J from 0.70 to 0.80	2nd of 221 linear	2.0	0.45% of linear; none of random

Note. Ranks count from the largest value; z: distance from the mean of the other draws, in their standard deviations. The calibration measures are compared with the twenty further draws of Amendment 6, the recall on the natural pairs with the 220 further draws of the linear family and the 200 draws of fully random hashing of Amendment 8.
Equation 13 gives the cluster-robust variance [41] of the mean residual $\bar{e}$ of a bin (Equation 8), clustered by base document, without a small-sample correction and with residuals $e_j = X_j - P_j$ taken about the theoretical probabilities; the outer sum runs over the base documents g, and the inner sum over the pairs built from g. The clustered standard errors are close to those of the independence formula in every bin (Table 21), and H1 remains supported.

$$\widehat{\mathrm{Var}}(\bar{e}) = \frac{1}{m^2}\sum_{g}\left(\sum_{j\in g} e_j\right)^2 \qquad 13)$$

Finer checks (Amendments 7 and 8) compare the primary draw with the average over draws. The logistic regression of Equation 14, fitted to the 15,000 bin pairs, where the theory predicts $\beta_0 = 0$ and $\beta_1 = 1$, gives $\beta_0 = -0.041$ and $\beta_1 = 1.035$ for the primary draw, with standard errors robust to clustering by base document of 0.026 and 0.019, almost those of the model, which assume independent pairs; the joint Wald test of β^* with that cluster-robust covariance Ω and two degrees of freedom gives p = 0.032, and with the slope fixed at 1, logit P_j entering as an offset, the intercept is −0.048

(0.025). Over the twenty other draws the slope averages 0.999 (95% interval of the mean 0.992 to 1.006; standard deviation 0.014) and the intercept 0.005 (−0.007 to 0.017); on both, the primary draw is the most extreme of the 21 draws (Table 22).

$$\operatorname{logit}\Pr(X_j = 1) = \beta_0 + \beta_1 \operatorname{logit} P_j,$$
$$W = (\hat{\beta} - \beta^*)^\top \hat{\Omega}^{-1} (\hat{\beta} - \beta^*), \qquad (14)$$
$$\beta^* = (0,\ 1)^\top$$

The share of the 128 coordinates on which a pair agrees differs from the mean J of each bin by at most 0.0015 in the primary draw and 0.0004 averaged over the draws; with keys of full entropy it checks the implementation rather than the family. The share of the 16 bands on which a pair agrees is, unlike that check, sensitive to how far the collision probabilities of individual pairs depart from J. It differs from the mean of J^r of each bin by at most 0.0023 in the primary draw, in the top bin, and by at most 0.0005 averaged over the draws; in the lowest bin, where J^r is smallest, it lies 0.0004 below that mean in the primary draw and 0.00002 below it averaged over the draws. Below the bins, among the generated pairs with J from 0.20 to 0.45, the candidates follow Equation 3 in each stratum, in the primary draw and over the twenty draws (Table 23; totals in Table 20).

Table 23. Candidates among the generated pairs below the calibration bins, by stratum of J (Amendment 7; exploratory).

J	Pairs	Candidates	Expected	Over the twenty draws
0.20 to 0.30	13,755	4	4.2	4.0 (0 to 8)
0.30 to 0.40	9,596	38	38.3	36.1 (24 to 49)
0.40 to 0.45	3,910	69	67.3	64.7 (53 to 77)

Note. Candidates: with the primary draw; expected: sum of P(J) of Equation 3 over the pairs of the stratum; over the twenty further draws: mean, with the range.

A second implementation of the calibration was written from the plan documents alone (the formulation, the analysis plan and its configuration file, Amendment 1 and the data sheet), without access to our code; it was produced under the direction of the same author, so it gives independence of implementation, not of analyst. It read several points of the plan differently from our code (Table 24), above all the limit on variants, and obtained smaller bins. Its own generator and shingle encoding make it, in effect, another hash draw on another sample. The equivalence test passed in four bins; in the failing bin, 0.65–0.75, it estimated −0.012 (standard error 0.012), and in the lowest bin +0.008 (0.005), where the primary run has −0.009. Its validity check gave 0 candidates among 3,000 pairs, and its H2 bound was 0.0084. With its smaller bins, an exactly calibrated detector would pass its test of H1 with probability 0.522 (Amendment 8), and its failing bin is the one that fails most often over the hash draws, so that failure points to power rather than to a difference between the implementations. In the lowest bin the two estimates differ by 0.017, 2.6 combined standard errors; two hash draws on two samples can produce such a gap, the more so as the primary draw has the largest deficit of the 21 draws in the lower bins, but this compatibility is no evidence that the implementations agree there.

Table 24. Points on which the frozen plan was silent or ambiguous, as the second implementation of the calibration read them and as our code had settled them before the confirmatory run (exploratory).

Point	Second implementation	Our code
Limit of 60,000 variants	For the five bins together: 1,100 to 2,477 pairs per bin	For each bin, pooled into 300,000: 3,000 pairs per bin after 153,162 variants
Variants per base document	A base drawn for each variant, about 30 per base	Bases in turn, 76 or 77 per base
Donor of the replacement tokens	A document with at least five tokens, not the base	A document with at least 200 tokens, possibly the base
Pairs of the validity check	Random pairs of documents with at least five tokens, rejected if J ≥ 0.05	The same, among documents with at least 200 tokens

Point	Second implementation	Our code
Encoding of a 5-gram	Tokens joined by a space	Tokens joined by the unit separator U+001F
Wilson bounds	Two-sided; one-sided as an alternative	Two-sided; the one-sided reading adopted after the result (Amendment 2)
Level of the two one-sided tests	$\alpha = 0.05$ for each	The same
Replaced tokens	Scattered positions	The same
Pool of H2	Every generated pair with $J \geq 0.8$	The same
Hash family	That of Amendment 1; the configuration file still named a prime modulus	The same

Note. The last four rows coincide and are kept because they bear on a decision; the replica's readings are those it recorded with its result.

The replica listed 16 points on which it found the plan silent or ambiguous; Table 24 gives those on which the two implementations differ or that bear on a decision. Our code had settled each of these before the confirmatory run; only the code's two-sided reading of the validity bound was replaced after it (Amendment 2). We report our reading of the plan, with its target of 3,000 pairs per bin, as the primary result.

The plan's approximation ratio is a recall in subsamples: the first 1,000, 2,000, 4,000 and 8,000 documents with at least five tokens, exact copies included. The LSH stage found all 7, 29, 131 and 603 pairs with $J \geq 0.8$ (95% Wilson lower bounds $m/(m + z^2)$ of Equation 9: 0.645, 0.883, 0.971 and 0.993), but most of them are identical sets from the exact copies; in a split added after the first run, at $n = 8{,}000$ only 18 are not, with a lower bound of 0.824. The full-file comparison of Table 17 is the informative one.

8 Cost

Table 25 gives the media time and peak memory of each stage on the first n documents with at least five tokens, in file order and with exact copies included. Each time is the median of five repetitions, or three when the first takes more than a second, after a discarded warm-up; peak memory is the maximum traced by Python's tracemalloc module, in a separate pass, not the resident memory. The candidate stage includes tokenisation, shingling, signatures and band tables, but neither the enumeration of the pairs within each bucket nor the exact check. The repetitions of a point can differ by about a quarter: at the last point they took 105.1 to 131.1 s, and at 128,000 documents 63.3 to 80.7 s, so the bump at that point is within timing noise; the t interval of the slope does not allow for noise that differs between points, nor for the misfit of a single power law. The interquartile ranges and CPU times are in the result file. The fitted log-log slope of the candidate stage is 1.06, with 95% interval 1.03 to 1.09. The exact baseline has slope 1.08, with interval 0.94 to 1.23. The candidate stage lies inside its range, but the exact baseline does not, so H3 is not supported (Figure 11).

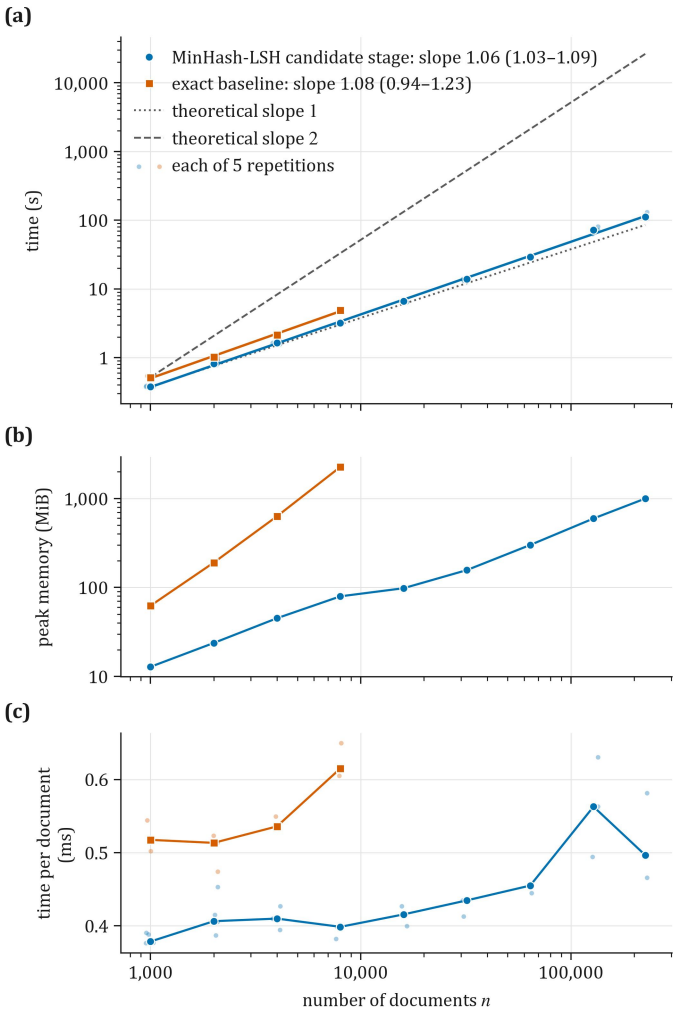


Figure 11.

Figure 11. (a) Time against number of documents on log-log axes, for the candidate stage and the exact baseline: each of the five repetitions of every point (faint dots), their medians, the log-log line fitted to the medians, with its slope and 95% interval (legend), and lines of slope 1 and 2 from the first median of the candidate stage and of the baseline, respectively. (b) Peak memory of each point, traced in a separate pass. (c) Time per document: each repetition and each median divided by the number of documents.

The design could not have tested the slope of the exact baseline well. Its time was measured only up to 8,000 documents, where tokenisation and hashing, which grow linearly, may still dominate; we did not profile the baseline to confirm it, and a single power law fitted to four points cannot separate the two terms. Its peak memory grows with a log-log slope, a reading added after the run, of 1.73, and by a factor of 3.6 between 4,000 and 8,000 documents, where a dense n × n array alone would give four. The candidate stage grows almost linearly; its median time for the 225,557 documents with at least five tokens is 112.0 seconds on one thread.

The relevant exact method for this file is not the baseline but the prefix-filtering join of Section 7. Amendment 8 timed that join, with its prefixes for $\theta = 0.8$, and the LSH stage from the same input, the shingle sets of the first n distinct documents, with the protocol of Table 25 but three repetitions, timing each step of the LSH stage and, apart from the join, the global order of the exact join (Table 26 gives the totals and the signatures; the result file, every step). Both loop in Python over documents, buckets or posting lists and candidates, with NumPy within each iteration, so the table compares implementations as much as methods. Reading and shingling all 215,536 distinct documents, with 61,225,840 shingles, took 87.6 s once. From those sets, the LSH stage took 23.1 s, 20.0 s of it for the signatures, and the exact join 145.1 s, more time than the LSH stage; their peak memory, traced as in Table 25 on top of the shingle sets they share, was 452 and 2,201 MiB.

Figure 12. Documents and tokens left after each cleaning step, applied in order (Amendment 7, addendum), as a double waterfall on a common scale, the share of the whole file: documents with fewer than five tokens; copies of whole documents beyond the first; near-duplicates beyond the first of each group at $J \geq 0.8$; and, in the documents left, occurrences of repeated paragraphs beyond the first, which remove tokens but no documents. Each floating bar is what a step removes (upper bars: documents; lower bars: tokens), labelled with its share of the file and its count.

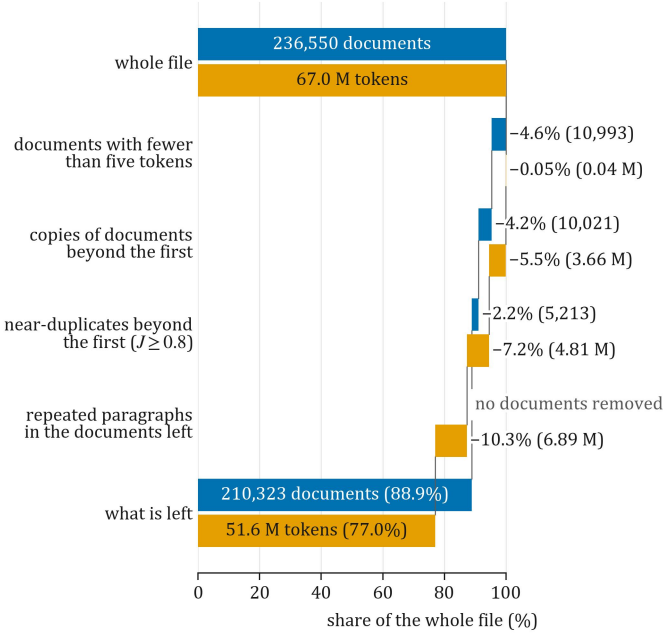


Figure 12.

Figure 12. Documents and tokens left after each cleaning step, applied in order (Amendment 7, addendum), as a double waterfall on a common scale, the share of the whole file: documents with fewer than five tokens; copies of whole documents beyond the first; near-duplicates beyond the first of each group at J ≥ 0.8; and, in the documents left, occurrences of repeated paragraphs beyond the first, which remove tokens but no documents. Each floating bar is what a step removes (upper bars: documents; lower bars: tokens), labelled with its share of the file and its count.

The exact join of the second implementation is another algorithm: it drops shingles that occur in a single document, processes documents by increasing size, indexes shorter prefixes than it probes with, and used prefixes for θ = 0.7, which yields more candidates, to count the three thresholds at once. Timed once by itself, on the same machine with one thread and without its 104.8 s of reading and shingling, it took 18.6 s to count document frequencies and drop those shingles and then 14.8 s for the join, 33.4 s in all, more than our LSH stage and less than our exact join; on the reduced sets it also compared the 128,877,820 pairs that share a shingle, without prefix or size filters, in 31.8 s. At this size, then, even an exact computation without those filters is affordable; the LSH stage was the fastest, by a margin that depends on the filtering refinements and on the implementation.

Table 28. Other bands from the first b·r of the same 128 coordinates of each draw (120 to 128 of them; Amendment 8; exploratory): recall on the exact pairs with J ≥ 0.8 and with J ≥ 0.7, expected (mean of $1 - (1 - J^r)^b$ over the pairs) and averaged over the 221 draws of the linear family, the primary included (Table 17 and Section 7 leave it out); Pairs: the distinct candidate pairs among the distinct documents with the primary draw.

b × r	J ≥ 0.8, expected	J ≥ 0.8, mean	J ≥ 0.7, expected	J ≥ 0.7, mean	Pairs
8 × 16	0.7241	0.7217	0.5136	0.5119	6,645
12 × 10	0.9472	0.9453	0.7980	0.7950	13,954
16 × 8	0.9922	0.9917	0.9279	0.9264	25,426
18 × 7	0.9982	0.9981	0.9688	0.9679	41,511
21 × 6	0.9998	0.9998	0.9922	0.9920	69,974
25 × 5	1.0000	1.0000	0.9992	0.9992	143,056
32 × 4	1.0000	1.0000	1.0000	1.0000	488,358

Re-banding the first b·r of the same 128 coordinates of each draw (Amendment 8, Table 28) shows the price of a higher recall at the threshold. With 18 bands of 7 rows the mean recall at J ≥ 0.8 rises to 0.9981, and the candidate pairs of the primary draw from 25,426 to 41,511; with 21 bands of 6 rows the recall at J ≥ 0.7 reaches 0.9920 for 69,974 candidates. In the LSH stage of Table 26, building the keys and buckets of a band took $\tau_1 \approx 0.13$ s, enumerating a pair within a bucket $\tau_2 \approx 7.5$ µs and verifying a candidate $\tau_3 \approx 17$ µs. If each step's time is proportional to its count (an estimate, not a measurement), Equation 15 gives the extra time of b' bands of r' rows, where Q counts the pairs enumerated within buckets, with repetition (92,672 with 16 bands of 8 rows, 127,875 with 18 of 7 and 196,701 with 21 of 6, for the primary draw), and |C| the distinct candidate pairs: 18 bands of 7 rows would add about 0.8 s and 21 of 6 about 2.2 s to the 23.1 s of the stage, while the expected misses among the 8,119 pairs with J ≥ 0.8 (Equation 4) fall from about 63 to 14 and 1. For every banding, the mean recall over the draws lies within 0.0031 of the value that Equation 3 gives for its b and r; with six or more rows per

band it lies below that value at both thresholds, by 1.2 to 2.7 standard errors of the mean, but the bandings reuse the same draws and nested coordinates, so these are not independent findings; with five or four rows the two agree within 0.0001. The standard deviation of the recall at $J \geq 0.8$ over the draws falls with the number of bands, from 0.0143 with 8 bands of 16 rows to 0.0030 with 16 of 8 and 0.0002 with 21 of 6.

$$\Delta t \approx \begin{array}{l}(b' - b)\,\tau_1 + (Q' - Q)\,\tau_2 \\ + \left(|C'| - |C|\right)\tau_3\end{array} \qquad (15)$$

9 Discussion and limitations

For users of the file, the main result is how much of its text repeats and at which level (Section 6). Paragraph occurrences beyond the first hold 22.5% of the tokens, exact copies of documents 5.5% and near-duplicate documents at $\theta = 0.8$ up to 7.3%, depending on which document of each group is kept. The levels overlap: applied in sequence (short documents, exact copies, near-duplicates, then repeated paragraphs in the documents that remain), they remove 23.0% of the tokens and keep 51,571,718 tokens in 210,323 documents; the document-level steps alone remove 12.7% (Figure 12). Each level calls for a different step. Exact copies are found by hashing whole documents. Near-duplicates at this threshold can be found exactly: from the same shingle sets, our prefix-filtering join of the whole file took 145.1 s on one thread and 2,201 MiB of traced memory, and our LSH stage 23.1 s and 452 MiB, after 87.6 s of reading and shingling (Section 8), while the exact join of the second implementation, with its filtering refinements, took 33.4 s from its shingle sets. At this size an approximate detector is therefore needed neither for speed nor for memory (RQ3), although the LSH stage was the fastest. Repeated paragraphs need deduplication at the level of paragraphs. Under the monthly-snapshot premise (Section 6), the file is not deduplicated across whole snapshots: the occurrences that such deduplication would have removed are about half of the excess occurrences and a quarter of their tokens (Table 13). Anyone building frequency lists, collocation counts or language models from the file should consider removing the occurrences of repeated paragraphs beyond the first, since some words occur almost only in them. Not all repeated paragraphs need be boilerplate: quoted verse, proverbs, formulae and headlines may also recur, and which occurrence is kept is arbitrary; since we did not read them, this guidance does not separate those cases. For the ranking of common words, removing exact copies or near-duplicate documents changes little, while removing repeated paragraphs changes it more and nearly removes words that occur almost only in repeated paragraphs.

Once the draw of hash functions is taken into account, the detector behaves almost as the theory predicts. Averaged over draws, the candidate rate of each bin differs from the curve by at most 0.0022, and the misses near the threshold are close to what Equation 4 predicts. On the natural pairs, where fully random hashing finds as many pairs as predicted, the recall of the linear family is 0.0005 below the prediction at $J \geq 0.8$ and 0.0016 below at 0.7 (2.5 and 2.3 standard errors of the mean), but within 1.5 standard errors of fully random hashing, and the other bandings, which share those draws, do not confirm this possible shortfall independently (Table 28). The linear family also produces 104.7 candidates on average among the generated pairs with J from 0.20 to 0.45, where 109.8 are expected. Any such shortfall is small next to the variation between draws. A single draw is less stable. The confirmatory test passes with the primary draw but fails in 4 of the 20 other draws, each time in one bin, and with these sample sizes a perfectly calibrated detector would fail it about one time in ten, and four or more failures in twenty draws have probability 0.109. A user who relies on one draw should expect the recall at $\theta = 0.8$ to vary by about a percentage point between draws, and a miss probability near 0.053 for pairs at the threshold, not the pooled 0.007 of H2. In Table 26 the exact join grows faster with n than the LSH stage; we did not measure where it would become impractical, at larger sizes or lower thresholds. With fewer rows per band, the recall at the threshold is higher at the price of more candidates (Table 28); when every candidate is verified exactly, as here, that price is small on this file by our estimate (Section 8), so users of an LSH detector who want the pairs at $\theta = 0.8$ would be better served by 18 bands of 7 rows or 21 of 6 than by 16 of 8; with fewer rows the estimated cost keeps growing for little more recall.

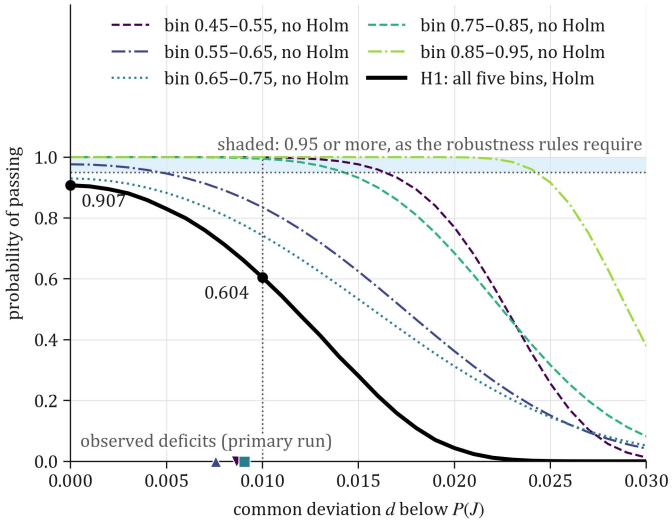


Figure 13.

Figure 13. Probability that the test of H1 passes (all five bins, Holm's adjustment; black) and that each bin passes its two one-sided tests before Holm's adjustment, when the candidate curve lies d below Equation 3 in every bin, with the standard

errors of the primary run (Amendment 10; normal approximation: each bin from Equation 16, H1 from 100,000 simulated replicates with Holm's adjustment for each d). Markers on the horizontal axis: the observed deficit of each of the three lower bins in the primary run (Table 19); dots: H1 at d = 0 and d = 0.01; shaded: 0.95 or more, the share that the robustness rules of Amendments 5 and 6 require.

Four threats to validity remain. Construct validity: near-duplication is defined by the Jaccard similarity of surface word 5-grams, not by the judgement of readers of Kurmanji. Texts that differ only in orthography, such as spellings without the circumflex of ê, î and û, or that circulate in both Latin and Arabic script, are not matched, and pages built from one template can share most of their shingles while their content differs. The synthetic variants replace scattered single tokens at a fixed length, whereas natural near-duplicates can differ in contiguous blocks, by truncation or in names and dates, and the base documents have at least 200 tokens. Internal validity: the plan is an internal record, not a public registration; the reading that decides the validity check was chosen after its result (Amendment 2); and the primary draw of hash functions, from which the corpus-level figures of the detector come, is favourable on the natural pairs at lower similarities, although on the generated pairs of the calibration it has the largest deficit of the 21 draws in the lower bins (Section 7). External validity: one file of Latin-script Kurmanji web text from one crawl year (2018), whose language label we did not check. Conclusion validity: the margin 0.03 is absolute, so it is lax for small proportions. Equation 16 gives the probability $\pi(d)$ that a bin passes its two one-sided tests, before Holm's adjustment, when the candidate curve lies d below Equation 3 in every bin: $\bar{e}$ then has mean $-d$ and the variance $\sigma^2(d)$ of pairs with probabilities $P_j - d$, while the test keeps the standard error σ of the plan, and the bin passes if $|\bar{e}| < \eta$, with $z = 1.645$. A candidate curve 0.02 below Equation 3 near the threshold would pass the test of the bin 0.75–0.85 with probability 0.685, and would miss 0.073 of the pairs at $J = 0.8$ instead of 0.053; the plan's statement that power is high for deviations below 0.01 does not hold near 0.01: a curve 0.01 below Equation 3 would pass the test of the bin 0.65–0.75 with probability 0.743, and the whole test of H1, simulated with Holm's adjustment, with probability 0.604 (Figure 13; Amendment 10); a margin on the log-odds scale would have been stricter near 0 and 1; in the top bin, where the prediction is close to one, the test reduces to non-inferiority; the normal approximation is rough there, with about three non-candidates expected among 3,000 pairs; Holm's adjustment is conservative for a claim about all bins jointly; and many analyses, which we report as robustness or exploratory analyses, were added after the primary results.

$$\pi(d) = \Phi\left(\frac{\eta + d}{\sigma(d)}\right) - \Phi\left(\frac{d - \eta}{\sigma(d)}\right)$$
$$\eta = \delta - z\,\sigma \qquad (16)$$
$$\sigma^2(d) = \sigma^2 + \frac{d(2\bar{P} - 1) - d^2}{m}$$

The distribution page of CC-100 states no licence; it says that no claims of intellectual property are made on the work of preparation of the corpus [2], and the web text keeps whatever rights its authors hold. We therefore do not redistribute text. The repository holds code and aggregated counts and, for users of the file, the positions of the documents in each group of exact copies and of near-duplicates, the exact pairs with $J \geq 0.7$ with the sizes of the intersection and union of their shingle sets, and the digest and number of occurrences of each of the 192,466 paragraphs that occur more than once. Positions count documents in file order from 0, and a distinct document with exact copies is identified by the position of its first occurrence; a digest is the 16-byte BLAKE2b of the paragraph's UTF-8 bytes, without its line break or a carriage return before it and with any leading byte-order mark, which our exact counts also treat as content. No repository file contains running text from the corpus; one result file lists the twenty word types of Section 6, two of which appear to be a given name and a surname. Web text of this kind can include content about named people; the paper reports only aggregate counts and descriptors, quotes no text and names no person. Several limits follow from the design. We did not read the repeated paragraphs, the near-duplicate pairs or a sample of documents, so the kinds of text that repeat are described only by automatic descriptors, and the language label is unverified. Word frequencies are counted on surface forms, which spread the inflected forms of one Kurmanji lemma over several types. The second implementation of the calibration, which reads the variant budget as a total, gives smaller bins and a failed equivalence test in one bin. The provenance of the primary results records a working tree that was not clean when they were produced; reproductions from clean clones gave identical analysis results (Section 10).

10 Reproducibility and software

The study is organised as a pipeline in Python 3.10 with NumPy 1.22.0 and SciPy 1.7.3; the exact versions, the hardware and the time of each run are recorded with each result file, except the output of the replica of the calibration. One Makefile target runs every stage, from the download of the corpus to the document, including the sensitivity analyses and the tests; the two replications and the clean-clone reproductions are run on their own, as the repository's README describes. The test suite has 50 unit tests for the analysis and the document, all passing: 13 for the detector (tokenisation, the four invariants and the Monte Carlo checks of Section 4), 5 for the exact join (including a comparison with all pairs on random sets), 8 for the statistical functions, 8 for the functions of the later analyses, 8 for the figure guard (clipping, width, legends crossed by a line or outside their axes, and text that cites an element by its number), 3 for the two steps that post-process the exported document and 5 for the quality gates, one of which runs every gate on an input it must accept and on inputs it must reject. Quality gates check that every number written in digits comes from a result file, directly or computed from it by the document builder, from the frozen plan or from a short list of documented constants, or is an integer up to 40 or a year

up to 2021 (numbers written as words are not checked), and they check the references, the structure of the document and that the text layer of the exported document is the text of its source; the release step does not copy a document that fails them. Reporting follows the machine learning reproducibility checklist in the version used at NeurIPS 2019 [42]; the completed checklist is in the repository.

Table 29. Reproductions from clean clones of the repository.

Clone	What was run	Result files identical	Cost: counts and memory	Tests passed
6c5fb5d	The primary analyses, the checks later recorded in Amendment 4 and the second implementation of the calibration	6 of 6, and the replica's output	Identical	36 of 36
04f3b57	The whole Makefile pipeline, from the hash-verified corpus file to the document, with the analyses of Amendment 5	11 of 11	Identical	41 of 41
3238b93	The later analyses of Amendments 4 to 8, except the cost on the same input, from the committed primary results	16 of 16	Not compared	Not run

Note. Result files are compared without provenance, timing and fields derived from timing; the test record of the second clone is left out, since the suite had fewer tests at that commit, the tests of the project clock included.

Table 30. Result files used by the document and the tables and figures that draw on each.

Result file (.json)	Tables and figures
estructura_corpus	Tables 2, 3, 5, 9, 10, 13, 14, 15 and 16; Figures 1, 5, 12 and 14
enmienda8_corpus	Tables 4, 10, 11 and 13; Figures 1 and 14
enmienda8_sorteos	Tables 10, 17, 22 and 28; Figures 3, 7 and 14
calidad_datos	Tables 2, 3, 4, 5 and 8; Figure 14
enmienda9_codificacion	Tables 4 and 5; Figure 14
enmienda10_figuras	Figures 2, 4, 6, 13 and 14
calibracion	Tables 18, 19 and 24; Figures 8, 9, 10, 13 and 14
coste	Tables 7, 8, 18 and 25; Figures 11 and 14
coste_comparable	Tables 7, 26 and 27; Figure 14
replica_calibracion	Tables 20 and 24; Figures 10 and 14
calibracion_extra	Tables 18, 19, 20, 23 and 24; Figures 9 and 14
grupos_duplicados	Figures 4 and 14
marcas_de_repeticion	Figures 4, 5 and 7
corpus	Table 8; Figures 4, 12 and 14
repeticiones	Figure 5
eliminacion_combinada	Table 13; Figures 12 and 14
exacto_corpus	Table 17; Figures 7 and 14
variacion_hash	Tables 17, 20 and 21; Figures 7, 10 and 14
control_unilateral	Table 18
contraste_exacto	Table 20
enmienda8_calibracion	Tables 20, 21 and 22; Figures 8, 10 and 14
s3_bootstrap	Tables 20 and 21; Figures 10 and 14
robustez_semilla	Table 20; Figures 10 and 14
sensibilidad	Tables 20 and 21
s1_shingles	Table 20; Figures 10 and 14
aproximacion	Table 20; Figure 14
pruebas	Figure 14
reproduccion	Table 29
reproduccion_final	Table 29
reproduccion_posteriores	Table 29

Note. Each file is written by the script of the same name in analisis/, except calidad_datos, by data/auditoria; replica_calibracion, by verificacion/replica; grupos_duplicados, by analisis/grupos; marcas_de_repeticion, by analisis/enmienda8_corpus; repeticiones, by data/analisis_repetidos; s3_bootstrap, robustez_semilla and s1_shingles, by

analisis/muestreo_plan; pruebas, by doc/pruebas; reproduccion_final and reproduccion_posteriores, by analisis/reproduccion. Derived from the sources of the document builder and from what the figure script reads.

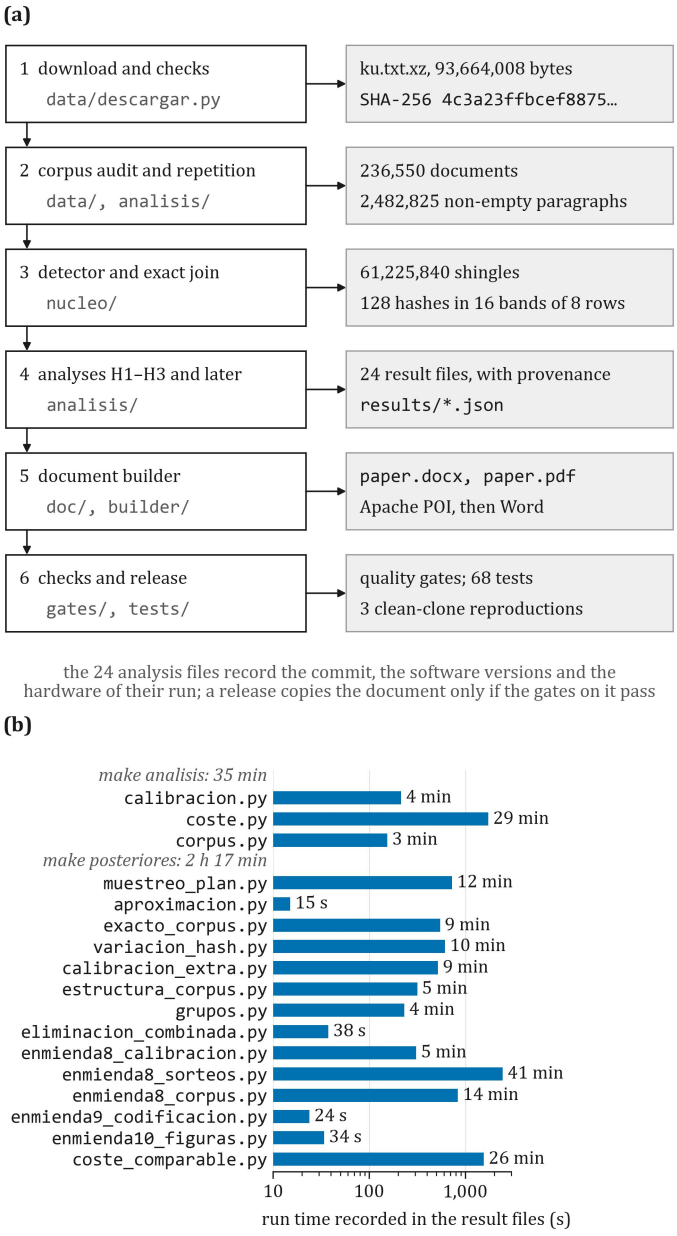


Figure 14.

Figure 14. (a) Pipeline architecture, in the order in which the Makefile runs it (left), with what each stage produces (right): the size and the SHA-256 digest of the corpus, the documents and paragraphs, the shingles and the banding, the result files, the document and the checks. The quality gates read the document and the results before the release step copies it; each analysis stage writes a JSON file with its provenance. (b) Run time that each script of the two analysis stages of the Makefile records in its result files, in the order in which they run, and the sum for each stage over the scripts shown; the scripts that record none are left out.

Three clean clones of the repository reproduced the results (Table 29); result files are compared without their provenance and timing fields, and the times of the cost measurements, which vary between runs, are not compared. The first was made before the analyses of Amendment 5 were run; the third used the scripts of the final version for the later analyses. The bootstrap, the clustered variance, the variant generator, the corpus counts and the scripts of Amendments 8 to 10 have no unit tests of their own; the scripts check the counts against each other and stop on a mismatch, and each script of Amendments 8 to 10 stops unless it reproduces the earlier result files it overlaps with. The later analyses that use the calibration sample regenerate it and stop unless it is identical to the primary run; none stopped. The workstation has an eight-core processor with sixteen threads and 16 GB of memory, and the exact model is recorded with each result file but the replica's; the cost measurements used one thread, with the thread counts of the numerical libraries set to one. Figure 14 shows the stages of the pipeline, and Table 30 lists the result files that the document uses, with the tables and figures that draw on each.

Declarations

Data availability. The corpus is publicly available at the address given in the reference list [2], and its SHA-256 digest is given in Section 3. Code, aggregated results, the positions of duplicated and near-duplicate documents, the exact near-duplicate pairs and the digests and counts of repeated paragraphs are kept in a version-controlled repository; the public address will be given with the published article.

Funding. None.

Conflicts of interest. None.

Ethics. The study analyses a publicly available text corpus, reports only aggregate counts and descriptors and quotes no text; no ethics review was sought.

Author contributions. Conceptualisation, methodology, software, validation, formal analysis, data curation, writing and review: L.E.M.G.

References

1. S. Ahmadi, "KLPT – Kurdish Language Processing Toolkit," in Proceedings of the Second Workshop for NLP Open Source Software, 2020, pp. 72–84. doi:10.18653/v1/2020.nlposs-1.11.
2. CC-100: Monolingual Datasets from Web Crawl Data, distribution page, Kurdish (ku) entry, archived 8 April 2021, https://web.archive.org/web/20210408050623/http://data.statmt.org/cc-100/.
3. A. Conneau et al., "Unsupervised Cross-lingual Representation Learning at Scale," in Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 8440–8451. doi:10.18653/v1/2020.acl-main.747.
4. G. Wenzek et al., "CCNet: Extracting High Quality Monolingual Datasets from Web Crawl Data," arXiv:1911.00359, 2019; also in Proceedings of the Twelfth Language Resources and Evaluation Conference, 2020, pp. 4003–4012.
5. I. Caswell, T. Breiner, D. van Esch and A. Bapna, "Language ID in the Wild: Unexpected Challenges on the Path to a Thousand-Language Web Text Corpus," in Proceedings of the 28th International Conference on Computational Linguistics, 2020, pp. 6588–6608. doi:10.18653/v1/2020.coling-main.579.
6. J. Kreutzer et al., "Quality at a Glance: An Audit of Web-Crawled Multilingual Datasets," arXiv:2103.12028v3, 2021.
7. K. Lee et al., "Deduplicating Training Data Makes Language Models Better," arXiv:2107.06499v1, 2021.
8. G. S. Manku, A. Jain and A. Das Sarma, "Detecting Near-Duplicates for Web Crawling," in Proceedings of the 16th International Conference on World Wide Web, 2007, pp. 141–150. doi:10.1145/1242572.1242592.
9. A. Z. Broder, S. C. Glassman, M. S. Manasse and G. Zweig, "Syntactic Clustering of the Web," Computer Networks and ISDN Systems, vol. 29, no. 8–13, pp. 1157–1166, 1997. doi:10.1016/S0169-7552(97)00031-7.
10. M. Charikar, "Similarity Estimation Techniques from Rounding Algorithms," in Proceedings of the Thirty-Fourth Annual ACM Symposium on Theory of Computing, 2002, pp. 380–388. doi:10.1145/509907.509965.
11. M. Henzinger, "Finding Near-Duplicate Web Pages: A Large-Scale Evaluation of Algorithms," in Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2006, pp. 284–291. doi:10.1145/1148170.1148222.
12. P. J. Ortiz Suárez, B. Sagot and L. Romary, "Asynchronous Pipeline for Processing Huge Corpora on Medium to Low Resource Infrastructures," in Proceedings of the Workshop on Challenges in the Management of Large Corpora (CMLC-7), 2019, pp. 9–16. doi:10.14618/ids-pub-9021.
13. Xue et al., "mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer," in Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2021, pp. 483–498. doi:10.18653/v1/2021.naacl-main.41.
14. H. Veisi, M. MohammadAmini and H. Hosseini, "Toward Kurdish Language Processing: Experiments in Collecting and Processing the AsoSoft Text Corpus," Digital Scholarship in the Humanities, vol. 35, no. 1, pp. 176–193, 2020. doi:10.1093/llc/fqy074.
15. D. Ataman, "Bianet: A Parallel News Corpus in Turkish, Kurdish and English," arXiv:1805.05095, 2018.
16. M. Gökırmak and F. M. Tyers, "A Dependency Treebank for Kurmanji Kurdish," in Proceedings of the Fourth International Conference on Dependency Linguistics (Depling 2017), 2017, pp. 64–72. ACL Anthology W17-6509.
17. K. Sheykh Esmaili, S. Salavati and A. Datta, "Towards Kurdish Information Retrieval," ACM Transactions on Asian Language Information Processing, vol. 13, no. 2, article 7, 2014. doi:10.1145/2556948.
18. G. Haig and E. Öpengin, "Introduction to Special Issue – Kurdish: A Critical Research Overview," Kurdish Studies, vol. 2, no. 2, pp. 99–122, 2014. doi:10.33182/ks.v2i2.397.
19. E. Öpengin and G. Haig, "Regional Variation in Kurmanji: A Preliminary Classification of Dialects," Kurdish Studies, vol. 2, no. 2, pp. 143–176, 2014. doi:10.33182/ks.v2i2.399.
20. H. Hassani, "BLARK for Multi-Dialect Languages: Towards the Kurdish BLARK," Language Resources and Evaluation, vol. 52, no. 2, pp. 625–644, 2018. doi:10.1007/s10579-017-9400-0.
21. J. Sheyholislami, Kurdish Identity, Discourse, and New Media. New York: Palgrave Macmillan, 2011. doi:10.1057/9780230119307.
22. M. Baroni, S. Bernardini, A. Ferraresi and E. Zanchetta, "The WaCky Wide Web: A Collection of Very Large Linguistically Processed Web-Crawled Corpora," Language Resources and Evaluation, vol. 43, no. 3, pp. 209–226, 2009. doi:10.1007/s10579-009-9081-4.
23. R. Schäfer and F. Bildhauer, Web Corpus Construction. Synthesis Lectures on Human Language Technologies. Morgan & Claypool, 2013. doi:10.2200/S00508ED1V01Y201305HLT022.

24. P. Indyk and R. Motwani, "Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality," in Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, 1998, pp. 604–613. doi:10.1145/276698.276876.
25. A. Z. Broder, M. Charikar, A. M. Frieze and M. Mitzenmacher, "Min-Wise Independent Permutations," Journal of Computer and System Sciences, vol. 60, no. 3, pp. 630–659, 2000. doi:10.1006/jcss.1999.1690.
26. J. Leskovec, A. Rajaraman and J. D. Ullman, Mining of Massive Datasets, 2nd ed. Cambridge: Cambridge University Press, 2014. doi:10.1017/CBO9781139924801.
27. S. Chaudhuri, V. Ganti and R. Kaushik, "A Primitive Operator for Similarity Joins in Data Cleaning," in Proceedings of the 22nd International Conference on Data Engineering, 2006, p. 5. doi:10.1109/ICDE.2006.9.
28. S. Sarawagi and A. Kirpal, "Efficient Set Joins on Similarity Predicates," in Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, 2004, pp. 743–754. doi:10.1145/1007568.1007652.
29. C. Xiao, W. Wang, X. Lin and J. X. Yu, "Efficient Similarity Joins for Near Duplicate Detection," in Proceedings of the 17th International Conference on World Wide Web, 2008, pp. 131–140. doi:10.1145/1367497.1367516.
30. R. J. Bayardo, Y. Ma and R. Srikant, "Scaling Up All Pairs Similarity Search," in Proceedings of the 16th International Conference on World Wide Web, 2007, pp. 131–140. doi:10.1145/1242572.1242591.
31. W. Mann, N. Augsten and P. Bouros, "An Empirical Evaluation of Set Similarity Join Techniques," Proceedings of the VLDB Endowment, vol. 9, no. 9, pp. 636–647, 2016. doi:10.14778/2947618.2947620.
32. D. R. MacIver, Z. Hatfield-Dodds et al., "Hypothesis: A New Approach to Property-Based Testing," Journal of Open Source Software, vol. 4, no. 43, p. 1891, 2019. doi:10.21105/joss.01891.
33. M. Pătraşcu and M. Thorup, "On the k-Independence Required by Linear Probing and Minwise Independence," in Automata, Languages and Programming (ICALP 2010), Lecture Notes in Computer Science, vol. 6198, 2010, pp. 715–726. doi:10.1007/978-3-642-14165-2_60.
34. M. Mitzenmacher and S. Vadhan, "Why Simple Hash Functions Work: Exploiting the Entropy in a Data Stream," in Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2008, pp. 746–755.
35. D. J. Schuirmann, "A Comparison of the Two One-Sided Tests Procedure and the Power Approach for Assessing the Equivalence of Average Bioavailability," Journal of Pharmacokinetics and Biopharmaceutics, vol. 15, no. 6, pp. 657–680, 1987. doi:10.1007/BF01068419.
36. S. Holm, "A Simple Sequentially Rejective Multiple Test Procedure," Scandinavian Journal of Statistics, vol. 6, no. 2, pp. 65–70, 1979. https://www.jstor.org/stable/4615733.
37. E. B. Wilson, "Probable Inference, the Law of Succession, and Statistical Inference," Journal of the American Statistical Association, vol. 22, no. 158, pp. 209–212, 1927. doi:10.1080/01621459.1927.10502953.
38. B. Efron, "Bootstrap Methods: Another Look at the Jackknife," The Annals of Statistics, vol. 7, no. 1, pp. 1–26, 1979. doi:10.1214/aos/1176344552.
39. C. A. Field and A. H. Welsh, "Bootstrapping Clustered Data," Journal of the Royal Statistical Society: Series B (Statistical Methodology), vol. 69, no. 3, pp. 369–390, 2007. doi:10.1111/j.1467-9868.2007.00593.x.
40. J. M. Hoenig and D. M. Heisey, "The Abuse of Power: The Pervasive Fallacy of Power Calculations for Data Analysis," The American Statistician, vol. 55, no. 1, pp. 19–24, 2001. doi:10.1198/000313001300339897.
41. A. C. Cameron and D. L. Miller, "A Practitioner's Guide to Cluster-Robust Inference," Journal of Human Resources, vol. 50, no. 2, pp. 317–372, 2015. doi:10.3368/jhr.50.2.317.
42. J. Pineau et al., "Improving Reproducibility in Machine Learning Research (A Report from the NeurIPS 2019 Reproducibility Program)," Journal of Machine Learning Research, vol. 22, no. 164, pp. 1–20, 2021. https://jmlr.org/papers/v22/20-303.html.