C-MinHash as a Replacement for MinHash in Jaccard LSH

Noah Kasdorf

Carleton University

November 2025

Big picture

- Standard Jaccard LSH pipeline
  - text $\rightarrow$ shingles $\rightarrow$ MinHash signatures
  - banding $\rightarrow$ candidate pairs $\rightarrow$ exact Jaccard
- The expensive part in practice is building MinHash signatures.
- Goal: swap MinHash for a cheaper variant without losing much recall.
- C-MinHash (Li & Li, 2022) reuses one permutation with cyclic shifts.
- I will focus on how C-MinHash works and what happens on a lyrics Jaccard LSH pipeline.

Quick recap: Jaccard, MinHash, LSH

- Jaccard similarity for sets A, B:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}.$$

- MinHash:
  - random permutation π of the universe,
  - $h_\pi(S)$ is the minimum element of S under π,
  - $\mathbb{P}[h_\pi(A) = h_\pi(B)] = J(A, B)$.
- With K independent permutations we stack K rows to get a signature of length $H = K$.
- LSH banding:
  - split H rows into b bands of r rows ($H = br$),
  - two signatures are candidates when any band matches exactly.
- Under independence the candidate probability is

$$f(s) = 1 - (1 - s^r)^b,$$

Why C-MinHash and what it does

- Standard MinHash:
  - K independent permutations,
  - each element is hashed K times.
- C-MinHash idea:
  - use 1 (or 2) permutations,
  - reuse them with many cyclic shifts,
  - all rows share the same base ordering, just with different offsets.
- Motivation:
  - fewer permutations to store,
  - less heavy hashing work per element,
  - hope to keep similar retrieval quality in LSH.

C-MinHash details (two-permutation version)

- Fix permutations σ and π on $\{0, \ldots, D-1\}$ and shifts $s_1, \ldots, s_K$.
- For a binary vector v (or set S) and row k:

$$h_k(v) = \min\{(\pi(\sigma(i)) + s_k) \bmod D : v_i = 1\}.$$

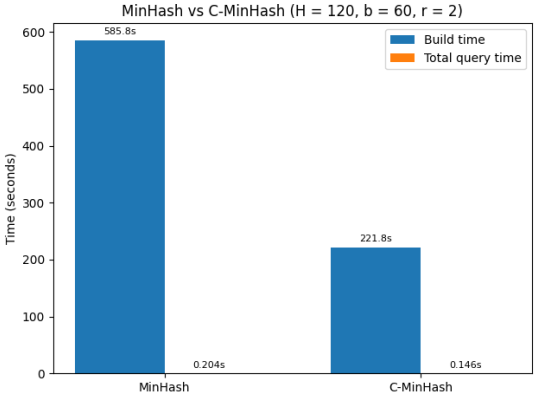
- Implementation view:
  - for each element x:
    - compute base rank $r = \pi(\sigma(x))$ once,
    - for each k, update h_k with $(r + s_k) \bmod D$ if smaller.
- Estimator:
  - same Jaccard estimator

$$\widehat{J} = \frac{1}{K} \sum_{k=1}^{K} \mathbf{1}[h_k(A) = h_k(B)],$$

  - still unbiased: $\mathbb{E}[\widehat{J}] = J(A, B)$,
  - Li & Li show a smaller variance than MinHash for the same K.

Cost and build time

- Permutations:
  - MinHash: K permutations.
  - C-MinHash: 1–2 permutations plus K integer shifts.
- Per-element work:
  - MinHash: K hashes per element.
  - C-MinHash: evaluate σ, π once, then cheap shifts.
- Equal-H run on the lyrics corpus ($H = 120$):
  - MinHash build time $\approx$ 586 s.
  - C-MinHash build time $\approx$ 222 s.
  - About 2.6$\times$ faster for signature construction.

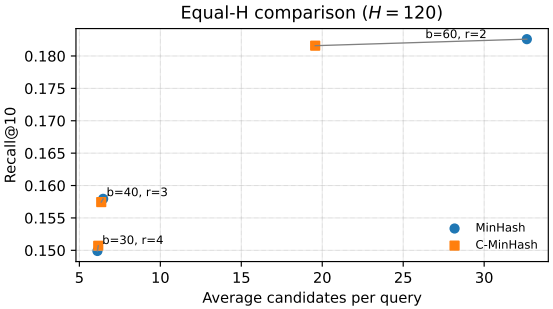


Experimental setup (lyrics + equal-H)

- Dataset:
  - *150K Lyrics Labeled with Spotify Valence* (Kaggle, via HF).
  - Use $\approx$ 157,000 songs as corpus, 1,000 as queries.
- Preprocessing:
  - lowercase, strip punctuation, tokenize on whitespace,
  - build word-3 shingles (stride 1),
  - hash shingles to 64-bit IDs, treat each song as a shingle set.
- Metrics:
  - recall@3, @5, @10,
  - candidates per query,
  - signature build time and total query time.
- Equal-H setup:
  - fix $H = K = 120$ for both methods,
  - banding choices

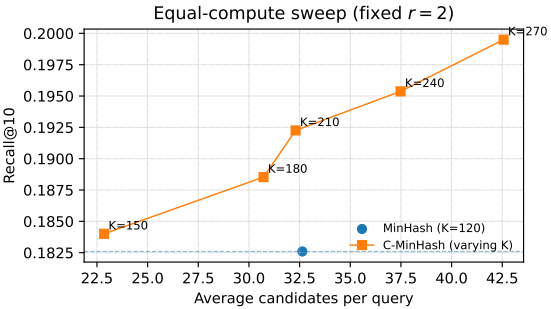
$$(b, r) \in \{(40, 3), (60, 2), (30, 4)\}, \quad H = br.$$

Equal-H results: recall vs candidates



- For all (b, r) choices, MinHash and C-MinHash have very similar recall@10.
- C-MinHash often has similar or slightly fewer candidates per query.
- For $(b, r) = (60, 2)$:
  - recall@10 is almost identical,
  - C-MinHash gives fewer candidates.
- So at fixed H we get comparable quality with cheaper signatures.

Equal-compute experiment (fixed time budget)



- Take the MinHash baseline:
  - $K = 120$, $(b, r) = (60, 2)$,
  - build time ≈ 586 s.
- For C-MinHash:
  - keep $r = 2$,
  - increase K and set $b = K/2$ so $H = K$,
  - only keep settings whose build time stays under the ≈ 586 s budget.
- Results:
  - for moderate K (around 150–210), recall@10 is higher than MinHash with similar or fewer candidates,
  - around $K = 270$ we see the best recall@10

Design note: changing r instead of b

- I also tried using the extra compute to increase r instead of increasing b.
- This pushed candidates per query down to about 6, but recall@10 stayed around 0.15–0.16 and then started to drop with very strict banding.
- Trade-off:
  - extremely fast verification,
  - but limited or worse recall improvements.
- Using C-MinHash to lengthen the signatures by increasing the bands gave a better recall vs. cost trade-off in this setting.

Takeaways

- C-MinHash is a practical drop-in replacement for MinHash in this Jaccard LSH lyrics pipeline.
- At fixed H:
  - recall and candidate counts are very similar,
  - signature construction is about 2–3$\times$ faster.
- Under a fixed build-time budget:
  - C-MinHash can support longer signatures,
  - which gives higher recall@k for similar total work.
- Caveat:
  - rows are correlated, so the textbook LSH curve is only approximate,
  - candidate rates, especially for borderline pairs, can shift a bit compared to fully independent rows.

References

- Y. Li and P. Li. *C-MinHash: Improving Minwise Hashing with Circulant Permutation*. ICML, PMLR, 2022.
- A. Z. Broder. *On the resemblance and containment of documents*. Compression and Complexity of Sequences, 1997.
- J. Leskovec, A. Rajaraman, and J. D. Ullman. *Mining of Massive Datasets*, 3rd Ed., 2020.
- P. Li and A. König. *Theory and applications of b-bit minwise hashing*. WWW, 2010.
- edenbd. *150K Lyrics Labeled with Spotify Valence*. Kaggle dataset, 2020.