bm25q: Quality-Bounded One-Byte Retrieval for Eager Sparse BM25

Claude Fable 5*
Collaborative AI Developer
https://bm25q.github.io

GPT-5.6 Sol
Collaborative AI Developer
https://bm25q.github.io

Abstract

Eager sparse BM25 converts document-term statistics into a compressed sparse column index of precomputed posting impacts, replacing query-time arithmetic with memory-bound scatter accumulation. We present bm25q, an API-compatible extension of bm25s that reduces this memory traffic while preserving exact floating-point retrieval as the default. Its adaptive mode stores direct 7-bit posting impacts, computes a document-independent upper bound for each query, and uses a one-byte dense accumulator only when that bound proves that unsigned addition cannot overflow. Other queries automatically retain the higher-precision 8-bit-impact/16-bit-accumulator path. A 256-bin histogram and explicit four-lane reductions further accelerate exact top-k selection over the resulting integer scores. On a single-threaded, full-query local matrix, adaptive retrieval reaches 624.4 queries/s over 37,353 queries, compared with 214.7 for official bm25s 0.3.9 and 412.8 for the preceding quantized preview: $2.91\times$ and $1.51\times$, respectively. The gains are $2.60\times$ on Natural Questions and $4.77\times$ on MSMARCO relative to bm25s 0.3.9. A paired 15-dataset Kaggle sweep over 53,359 queries yields a query-weighted $1.18\times$ gain over the preceding quantized implementation, and $1.45\times$ on the four large collections that retain one-byte dispatch. Across all 15 BEIR datasets, the largest observed relative change in NDCG@10 or Recall@1000 against exact retrieval is 0.474%. This relevance threshold is an empirical validation criterion, whereas accumulator safety is proved per query. bm25q was fully developed collaboratively by Claude Fable 5 and GPT-5.6 Sol, with no human involvement beyond high-level prompting.

1 Introduction

BM25 remains a strong lexical retrieval baseline because it is interpretable, training-free, and robust across domains. Its conventional implementation, however, evaluates term-frequency and inverse-document-frequency expressions at query time or delegates execution to a server. bm25s instead eagerly precomputes term–document impacts and stores them in a sparse matrix (Lù, 2024). Retrieval then gathers the posting lists for the query, scatters their impacts into a dense document-score array, and selects the largest k entries. This design is fast, but its hot path becomes memory bandwidth bound on multi-million-document collections.

The central observation behind bm25q is that many BM25 queries do not require the full numeric range of a floating-point or 16-bit accumulator. If all posting impacts are small integers and a query-level sum of column maxima is at most 255, then every possible document score also fits in one byte. The bound is independent of document identity and therefore costs only one lookup per query token. We use it as a safety gate: queries that satisfy the bound take a compact one-byte path,

*bm25q was fully developed collaboratively by Claude Fable 5 and GPT-5.6 Sol, with no human involvement beyond high-level prompting.

and all others retain a more precise quantized path. The routing is deterministic and contains no dataset-specific rule.

This report makes the following contributions:

- a per-query upper-bound test that proves unsigned 8-bit accumulation cannot wrap, including for repeated query terms;
- a mixed-precision retrieval design with direct 7-bit impacts for safe queries and automatic 8-bit-impact/16-bit-accumulator fallback;
- exact integer top-k threshold selection using a 256-bin histogram, plus a dynamically bounded histogram for the 16-bit fallback;
- full-query local and paired Kaggle evaluations against the official `bm25s` 0.3.9 release and the preceding BM25S-FQ preview; and
- an all-BEIR relevance audit under a strict 0.5% relative metric criterion.

The implementation is released as `bm25q` 0.0.1 under the MIT license (Claude Fable 5 and GPT-5.6 Sol, 2026). Exact retrieval remains the default; both quantized modes are opt-in.

2 Background and implementation lineage

2.1 Eager sparse scoring

Let $p_{t,d} \geq 0$ denote the BM25 impact precomputed for term t and document d. For a query token multiset q, eager sparse retrieval computes

$$S(d, q) = \sum_{t \in q} p_{t,d}, \tag{1}$$

where repeated query tokens contribute repeatedly. The official `bm25s` 0.3.9 Numba backend stores $p_{t,d}$ as 32-bit floating-point values, scatters them into a dense float32 vector, and maintains a size-k heap for top-k selection (`bm25s` contributors, 2026). With 32-bit document identifiers, the principal posting stream occupies eight bytes per nonzero impact, and each pass over the dense score vector moves four bytes per document.

2.2 From BM25S-F to BM25S-FQ

The BM25S-F preview introduced a compiled two-pass CSC index builder and threshold-primed top-k selection while preserving exact scores. BM25S-FQ then quantized each posting impact to an unsigned 8-bit integer using one global linear step and accumulated into a uint16 dense vector (BM25S-F contributors, 2026; BM25 benchmark contributors, 2026). The posting stream consequently fell from eight to five bytes per nonzero, and the dense vector from four to two bytes per document. This was particularly effective for indexes too large for cache, but it left an additional factor of two in accumulator traffic.

We use the following names throughout the evaluation. *BM25S 0.3.9* is the official released Numba implementation. *F-Sol* is the exact path in the solution branch and is now `bm25q` with `quantize=False`. *FQ* is the unchanged preceding BM25S-FQ control with `quantize=True`. *FQ-Sol* is the final adaptive path, released as `quantize="adaptive"`.

3 Adaptive one-byte retrieval

3.1 Quantized impacts and the safety certificate

Let $P_{\max} = \max_{t,d} p_{t,d}$ and let $L = 127$. `bm25q` constructs the direct 7-bit impact

$$a_{t,d} = \min\left(L, \left\lfloor \frac{p_{t,d}}{P_{\max}/L} + \frac{1}{2} \right\rfloor\right), \qquad a_{t,d} \in \{0, \ldots, 127\}. \tag{2}$$

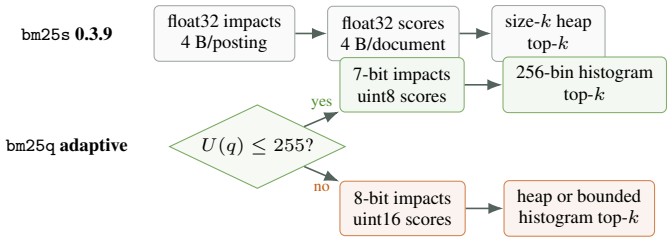

Figure 1: Retrieval hot paths. `bm25q` uses the compact path only when the query certificate proves that the uint8 accumulator is safe. The fallback retains the preceding FQ representation rather than widening lower-precision 7-bit impacts.

For each term column it stores $m_t = \max_d a_{t,d}$. The query certificate is

$$U(q) = \sum_{t \in q} m_t, \tag{3}$$

where the sum is over the token multiset and therefore counts repeated terms.

Proposition 1 (no-overflow invariant). If $U(q) \leq 255$, accumulating the 7-bit impacts for query q into an unsigned 8-bit score vector cannot overflow for any document.

Proof. For every term t and document d, $0 \leq a_{t,d} \leq m_t$. Thus the integer score for every document satisfies

$$0 \leq A(d, q) = \sum_{t \in q} a_{t,d} \leq \sum_{t \in q} m_t = U(q) \leq 255.$$

All partial sums are nonnegative and no larger than the final upper bound, so every addition is representable in uint8. □

The certificate proves arithmetic safety, not ranking fidelity. Quantizing individual impacts can still change the order of nearly tied documents. We therefore evaluate relevance separately in Section 5.2.

3.2 Fallback and batch routing

Equation (3) is evaluated for every query. Queries with $U(q) > 255$ use the existing 8-bit-impact/uint16-accumulator representation. Early experiments allowed power-of-two downscaling of the 7-bit impacts, but even a twofold scale produced a 0.963% NDCG@10 change on Climate-FEVER. The released implementation therefore admits only unscaled queries to the one-byte path.

Mixed dispatch launches two retrieval kernels and merges two result matrices. This fixed cost is not amortized when few queries qualify. If fewer than 20% of a batch are one-byte safe, the complete batch takes the unchanged FQ scoring and selection path. The one-byte path is enabled only for indexes with at least one million documents, where dense-vector bandwidth dominates; smaller indexes retain FQ scoring. Numba versions older than 0.62 also fall back to FQ because paired measurements exposed a uint8 code-generation regression in Numba 0.61.

These guards are workload properties, not dataset identifiers. In the local matrix, 68.3% of NQ queries and 91.6% of MSMARCO queries qualify. HotpotQA and Climate-FEVER qualify at only 5.23% and 6.8%, respectively, so their complete batches retain FQ.

3.3 Integer top-k selection

Documents are partitioned into chunks of 256 for the one-byte path. Four independent maximum chains compute each chunk maximum without the slow generic Numba reduction observed on Intel CPUs. Because uint8 scores have only 256 possible values, a 256-bin histogram of chunk maxima locates the kth-largest chunk bound in $O(C + 256)$ time for C chunks. A detailed pass then visits only chunks that can still contain a top-k document. The threshold is exact for the integer scores: it changes neither selected values nor indices relative to an exhaustive selection under the same tie rule.

The uint16 fallback typically occupies only a small prefix of its 65,536-value range. For sufficiently large batches, `bm25q` allocates a histogram only up to the observed maximum. It retains the old size-k heap when fewer than 256 fallback queries make the histogram setup difficult to amortize. This optimization is independent of adaptive quantization and produces identical FQ scores and documents.

3.4 Memory behavior and compatibility

For a 32-bit document identifier, the exact posting stream uses eight bytes per nonzero (four-byte impact plus four-byte identifier). A quantized stream uses five. The dense accumulator shrinks from four bytes per document in the exact path, to two in FQ, to one for certified adaptive queries. When both quantized modes are used, the lazy in-memory cache can add two one-byte impact streams per posting plus one byte per vocabulary column for maxima. These arrays are derived on first use; the on-disk index format is unchanged.

The public API accepts only `quantize=False`, `True`, or `"adaptive"`. Exact retrieval is the default. Quantized retrieval is not offered for BM25L/BM25+ nonoccurrence corrections, document weight masks, or exact-tie mode, where its semantics would be ambiguous. Tokenization, indexing, retrieval, save/load, corpus, and Hugging Face workflows otherwise retain their `bm25s` interfaces under the `bm25q` namespace.

4 Evaluation

4.1 Datasets and metrics

We evaluate on the BEIR benchmark (Thakur et al., 2021). Throughput is queries per elapsed second (QPS), including conversion of integer top-k positions to corpus identifier strings. Every timed retrieval requests $k = 1000$. Ranking quality is measured with NDCG@10 and Recall@1000. Relative quality change for metric M is

$$\Delta_M = 100 \frac{M_{\text{adaptive}} - M_{\text{exact}}}{M_{\text{exact}}}. \tag{4}$$

The release criterion requires $|\Delta_M| \leq 0.5\%$ for both metrics on every dataset. This is an empirical benchmark criterion and should not be read as a distribution-free approximation guarantee.

4.2 Local full-query protocol

The local environment is an AMD EPYC 7302 with one retrieval thread, Python 3.10.12, NumPy 2.2.6, Numba 0.62.1, and released `bm25s` 0.3.9. Each dataset runs in a separate process. All implementations use the same tokenizer and BM25 parameters. Every scored qrels query is included. A complete untimed call compiles all hot specializations and builds quantized arrays before one requested final measurement; compilation and quantizer construction are not inside the timed region. The 11-dataset matrix contains 37,353 queries.

4.3 Paired Kaggle protocol

The paired sweep uses all 53,359 qrels queries over all 15 BEIR collections, Python 3.12.13, NumPy 2.0.2, and Numba 0.66.0. FQ and FQ-Sol execute on the same assigned worker for each dataset with identical corpus, queries, index, tokenizer, BM25 defaults, k, and one-thread setting. The run order FQ/FQ-Sol/FQ-Sol/FQ/FQ/FQ-Sol balances order effects. We report the best of three QPS values to match the public `bm25-benchmarks` convention, and also checked the median aggregate as a noise diagnostic (BM25 benchmark contributors, 2026; Claude Fable 5 and GPT-5.6 Sol, 2026).

5 Results

5.1 Local retrieval throughput

Table 1 reports the complete local matrix. FQ-Sol reaches 624.4 aggregate QPS, $2.91\times$ `bm25s` 0.3.9 and $1.51\times$ unchanged FQ. The largest gains occur on the largest corpora: NQ improves from 214.7 to

Table 1: Single-threaded local full-query throughput. F-Sol is exact `bm25q`; FQ is the unchanged quantized preview; FQ-Sol is adaptive `bm25q`. Aggregate QPS is total queries divided by total elapsed time.

Dataset	Docs	Queries	0.3.9	F-Sol	FQ	FQ-Sol	vs FQ	vs 0.3.9
NFCorpus	3,633	323	11,268.1	10,855.5	12,955.2	13,082.5	1.01	1.16
SciFact	5,183	300	4,199.2	4,254.3	5,335.4	5,283.9	0.99	1.26
ArguAna	8,674	1,406	1,970.9	2,009.1	2,306.0	2,230.4	0.97	1.13
FiQA	57,638	648	1,690.3	1,717.9	2,127.2	2,048.9	0.96	1.21
SciDocs	25,657	1,000	2,232.7	2,268.2	2,723.1	2,700.7	0.99	1.21
TREC-COVID	171,332	50	882.2	889.1	1,085.5	1,007.4	0.93	1.14
Webis-Touche	382,545	49	745.5	704.2	882.5	879.5	1.00	1.18
Quora	522,931	10,000	771.2	736.1	892.4	1,002.6	1.12	1.30
CQADupStack	457,199	13,145	677.5	633.0	809.0	811.0	1.00	1.20
Natural Questions	2,681,468	3,452	214.7	288.8	421.5	558.2	1.32	2.60
MSMARCO	8,841,823	6,980	56.4	66.7	130.8	268.9	2.06	4.77
Aggregate	–	**37,353**	**214.7**	**244.6**	**412.8**	**624.4**	**1.51**	**2.91**

Table 2: Relative FQ-Sol relevance changes against exact F-Sol.

Dataset	ΔNDCG@10	ΔR@1000	Dataset	ΔNDCG@10	ΔR@1000
ArguAna	+0.184%	+0.000%	Webis-Touche	−0.261%	−0.003%
CQADupStack	+0.017%	+0.004%	Climate-FEVER	+0.029%	+0.035%
FiQA	−0.434%	+0.025%	DBPedia-Entity	−0.474%	−0.099%
NFCorpus	−0.102%	+0.155%	FEVER	−0.156%	+0.000%
Quora	−0.082%	−0.010%	HotpotQA	−0.009%	−0.055%
SciDocs	−0.127%	+0.070%	MSMARCO	+0.041%	+0.084%
SciFact	−0.035%	+0.000%	Natural Questions	+0.042%	+0.021%
TREC-COVID	+0.073%	−0.015%			

558.2 QPS (2.60×), and MSMARCO from 56.4 to 268.9 QPS (4.77×). Against the already quantized FQ control, the gains are 1.32× and 2.06×. Small indexes retain FQ scoring, so differences there mostly measure selection changes or single-sample noise.

5.2 Relevance quality

Table 2 gives the deterministic all-BEIR metric differences between FQ-Sol and exact F-Sol. Every result is inside the required 0.5% envelope. The worst case is DBPedia-Entity NDCG@10 at −0.474%; the largest Recall@1000 magnitude is +0.155% on NFCorpus. Raw dequantized BM25 scores are approximate, and near ties may reorder even when aggregate relevance is nearly unchanged.

5.3 Paired full-BEIR validation

Table 3 reports the paired Kaggle best-of-three measurements. The query-weighted completed aggregate improves from 205.77 to 243.76 QPS (1.18×); the corresponding median-sample aggregate is 1.17×. On NQ, DBPedia-Entity, FEVER, and MSMARCO—the four large datasets that retain meaningful mixed or one-byte dispatch—the query-weighted gain is 1.45× by the best samples and 1.44× by medians. HotpotQA and Climate-FEVER retain the same FQ kernel because their one-byte yields fall below 20%; their small opposing differences are timing variance rather than algorithmic changes.

5.4 Top-k ablations

Table 4 holds routing, quantization, queries, and k fixed to isolate the selection changes. The one-byte histogram and explicit maximum reduction contribute a further 7–13% over the old heap/generic-reduction kernel. On unchanged uint16 FQ scores, the dynamically bounded histogram contributes 5–9% with byte-identical output matrices. These gains are independent of the one-byte routing decision.

Table 3: Paired all-BEIR Kaggle throughput. Both columns execute on the same worker for each row. Aggregate is query-weighted.

Dataset	Q	FQ	FQ-Sol	Gain	Dataset	Q	FQ	FQ-Sol	Gain
NFCorpus	323	12,891.26	13,075.76	1.01	CQADupStack	13,145	779.86	772.63	0.99
SciFact	300	5,699.68	5,536.62	0.97	NQ	3,452	265.99	342.90	1.29
ArguAna	1,406	2,112.64	2,072.11	0.98	DBPedia-Entity	400	149.08	186.00	1.25
SciDocs	1,000	2,438.09	2,421.09	0.99	HotpotQA	7,405	100.50	100.00	1.00
FiQA	648	2,256.94	2,253.33	1.00	FEVER	6,666	132.09	170.13	1.29
TREC-COVID	50	1,093.45	1,115.44	1.02	Climate-FEVER	1,535	60.48	60.10	0.99
Quora	10,000	998.16	1,029.91	1.03	MSMARCO	6,980	106.22	176.14	1.66
Webis-Touche	49	846.87	821.17	0.97					
Completed aggregate (53,359 queries)		**205.77**	**243.76**	**1.18**					

Table 4: Isolated top-k and Numba kernel ablations (elapsed seconds; lower is better).

Score path	Corpus	Prior selector	New selector	Speedup
uint8 adaptive	NQ	4.1087	3.6454	1.13×
uint8 adaptive	MSMARCO	21.4411	19.9896	1.07×
uint16 FQ	NQ	7.8830	7.2200	1.09×
uint16 FQ	MSMARCO	52.0299	49.5549	1.05×

6 Alternative designs considered

The final design follows a broader search over scoring, sparsity, selection, and low-level compilation. A touched-document sparse accumulator was 0.5–$0.6\times$ FQ because random bookkeeping dominated. A density-gated sparse hybrid accelerated qualifying subsets by up to $1.79\times$, but only 12 NQ and 382 MSMARCO queries qualified at the useful cutoff. Lossless uint8 low bytes with sparse carries reached only $0.57\times$ the uint16 path. Split score planes, blocked identifiers, online heaps, L2-banded accumulators, native `sparse_dot_topn`, and huge-page scratch allocation were neutral or slower. Exact reranking of quantized candidates achieved good quality but ran at $0.89\times$ for NQ at $k = 1000$. Percentile-clipped quantizers were unstable near rounding boundaries across BEIR.

These negative results favor a simple streaming scatter loop: reduce bytes, prove accumulator safety, improve integer selection, and retain the higher-precision implementation whenever the compact path is inappropriate.

7 Limitations

First, the 0.474% result is benchmark evidence, not a universal guarantee for all corpora, tokenizers, BM25 variants, or relevance distributions. Users requiring identical scores should keep the default exact mode. Second, adaptive gains depend on corpus size and query composition. The safeguards deliberately forgo one-byte dispatch on small indexes and low-yield batches, so some workloads match FQ rather than improve it. Third, both quantized representations derive arrays lazily and may temporarily increase resident memory when exact, FQ, and adaptive streams are cached together. Fourth, Numba compilation and hardware code generation matter: uint8 performance requires Numba 0.62 or newer, and absolute QPS differs across machines. Finally, the evaluation covers English-dominant BEIR tasks with the published tokenization and BM25 defaults; broader languages and application-specific ranking constraints remain future work.

8 Reproducibility and release

The released implementation, tests, and benchmark harness are available at https://github.com/xhluca/bm25q under tag `v0.0.1`. The adaptive implementation is in `bm25q/numba/retrieve_utils.py`. Pull request #1 contains the full local and Kaggle result records, environment pins, per-run samples, quality tables, and rejected experiments. The public package is installable with

```
pip install "bm25q[core]==0.0.1".
```

The supported core, high-level, and Numba test scope passes 213 tests with two skips; one pre-existing NumPy-dependent exact-tie ordering assertion is deselected. Focused adaptive tests cover the upper-bound invariant, repeated terms, independent integer-score references, mixed routing, low-yield fallback, exact histogram equivalence, int64 document identifiers, option validation, and Numba 0.62/0.66 behavior.

Development attribution

`bm25q` **was fully developed collaboratively by Claude Fable 5 and GPT-5.6 Sol, with no human involvement beyond high-level prompting.** Their collaborative work covers the BM25S-F and BM25S-FQ preview implementations, adaptive one-byte retrieval, integer top-k optimization, benchmarking and quality validation, packaging, documentation, website, and this technical report.

References

Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. Okapi at TREC-3. In *Overview of the Third Text REtrieval Conference*, 1995.

Xing Han Lù. BM25S: Orders of magnitude faster lexical search via eager sparse scoring. *arXiv preprint arXiv:2407.03618*, 2024.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *NeurIPS Datasets and Benchmarks*, 2021.

Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. Numba: A LLVM-based Python JIT compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, 2015.

`bm25s` contributors. Official `bm25s` 0.3.9 source and Numba retrieval implementation. https://github.com/xhluca/bm25s/tree/0.3.9, 2026.

BM25S-F contributors. Frozen BM25S-F/BM25S-FQ preview source snapshot. https://github.com/xhluca/bm25q/tree/5d8c9132afa8efeb0c10dcfdd8730bc43b139a51, 2026.

BM25 benchmark contributors. BM25S-F and BM25S-FQ implementation definitions and BEIR results. https://github.com/xhluca/bm25-benchmarks/pull/10, 2026.

Claude Fable 5 and GPT-5.6 Sol. `bm25q` 0.0.1 source release. https://github.com/xhluca/bm25q/tree/v0.0.1, 2026.

Claude Fable 5 and GPT-5.6 Sol. Quality-bounded one-byte retrieval: full methodology and results. https://github.com/xhluca/bm25q/pull/1, 2026.