Scalable K-Means Guided Partitioning for Block-based Sparse Document Retrieval

Parker Carlson*
University of California, Santa Barbara
Santa Barbara, CA, USA
parker_carlson@ucsb.edu

Sammy Lesner*
University of California, Santa Barbara
Santa Barbara, CA, USA
samanthalesner@ucsb.edu

Antonio Mallia
Seltz
San Francisco, CA, USA
antonio@seltz.ai

Tao Yang
University of California, Santa Barbara
Santa Barbara, CA, USA
tyang@cs.ucsb.edu

Abstract

Document clustering is a common technique used in sparse retrieval to group similar documents together. The K-means method has been widely adopted to group similar sparse vectors together, but it is not scalable when dealing with a large number of clusters, and the bipartite graph partitioning (BP) method is a preferred choice for block-based document retrieval to partition a large document set efficiently. This paper revisits such a partitioning approach and proposes a balanced K-means-guided bisection method with log-linear complexity while maintaining a good similarity-based clustering quality. Our evaluation with several IR datasets for block-based sparse retrieval algorithms show that the proposed method outperforms the baseline variants of K-means in scalability with 29-414x faster partitioning even for small datasets and reduces retrieval latency by up to 32% compared to BP when clustering 8.8M MS MARCO passages using SPLADE++ embeddings.

CCS Concepts

• **Information systems → Clustering and classification**; **Search engine architectures and scalability**.

Keywords

Document clustering, K-means, Learned Sparse Retrieval, Efficiency

ACM Reference Format:

Parker Carlson, Sammy Lesner, Antonio Mallia, and Tao Yang. 2026. Scalable K-Means Guided Partitioning for Block-based Sparse Document Retrieval . In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '26), July 20–24, 2026, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3805712.3809971

1 Introduction

Document clustering is extensively used in sparse document retrieval in which documents are divided into a set of blocks of similar documents for fast index skipping [6, 8, 30, 35, 36, 39]. The bipartite graph partitioning (BP) [14] with bisection is used by live-block filtering, BMP, and superblock pruning [8, 9, 35, 36] to produce a larger number of clusters, while a variant of K-means [10] is used in the Seismic retriever [6] for clustering. Anytime Ranking and ASC [30, 39] adopt a combination of K-means and BP methods for document reordering and clustering. In this paper we use "block" and "cluster" interchangeably to describe a group of documents.

The K-means algorithm [10] is a popular method to divide a dataset of n vectors into K clusters by minimizing the mean distance from documents to the cluster centroids. There are various studies to optimize K-means for sparse datasets [13, 17, 18, 23, 27, 38] and their time complexity is at least $O(n \cdot K)$. The main advantage of BP over K-means and these variants is BP's log-linear complexity, scalable in producing up to $O(n)$ clusters. That is desired by the BMP [36] and LSP [9] retrieval engines, as witnessed by their configuration setting for MS MARCO's 8.8M passages with around one million blocks. The motivation of having such a large number of blocks is to avoid the loose estimation by block-level rank maximum scores and increase the chance of block-level index pruning.

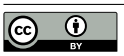
This paper revisits the problem of scalable clustering with K-means and proposes Guided K-means Partitioning (GuideKP) to inject the centroid distance cost of K-means into the bisection paradigm of BP while retaining its low complexity. During bisecting steps, GuideKP incorporates low-cost gain prediction to swap documents between two partitions and produces uniform partition sizes suitable for retrieval with BMP and LSP.

Our evaluation with small BEIR datasets shows that GuideKP produces clusters of the best quality compared to the four tested variants of K-means on multiple metrics while being two to three orders of magnitude faster (or more) in partitioning time. For MS MARCO with around 1M clusters, which is too large for these K-means variants to handle, GuideKP yields up to 32% shorter retrieval latency than BP with more successful runtime index pruning with manageable extra offline partitioning time.

2 Background and Related Work

Document clustering. K-means [28] minimizes the sum of squared error (SSE) in the targeted K clusters, where c_j is the centroid of cluster B_K: $SSE = \sum_{j=1}^{K} \sum_{x \in B_j} \|x - c_j\|^2$. Bisecting K-means [43] is shown to be competitive with a complexity $O(m \cdot \log K \cdot I \cdot r)$ for balanced splitting. But in practice binary splitting is often

*Equal Contribution



SIGIR '26, Melbourne, VIC, Australia

ACM ISBN 979-8-4007-2599-9/2026/07
https://doi.org/10.1145/3805712.3809971

unbalanced, which yields actual complexity $O(m \cdot K \cdot I \cdot r)$, similar to that of the standard K-means algorithm. Here, m is the number of non-zero elements in all data vectors, I is the maximum number of iterations per K-means split to converge, and r is the number of trials for varying different initial splits.

Previous studies for improving K-means on sparse data typically consider a number of clusters in an order of thousands, and their algorithmic complexity remains at least $O(n \cdot K)$, not scalable to a large number of clusters [1, 13, 17, 18, 23, 27, 38]. Another category of K-means studies target dense vectors with different storage structure and dimensionality considerations [2, 11, 37, 42].

Sparse document retrieval. Given a query q for searching a collection of n documents, sparse retrieval computes the dot product similarity of a given query q and a document d as: $RankScore(d) = L(q) \cdot L(d)$ where $L(.)$ is a sparse vector of weighted terms for a document or a query based on the BM25 formula [40] or a learned neural representation [20, 26, 33, 41, 46]. $L(.)$ may contain original and/or expanded terms. Online sparse retrieval historically uses an inverted index to assist fast query processing.

Block or cluster based sparse index pruning. The state-of-the-art fast retrieval algorithms for learned sparse retrieval BMP, Seismic, and Superblock Pruning all use block- or cluster-based pruning strategies [6, 8, 9, 36]. These methods follow or extend classical index skipping techniques [5, 15, 16, 35, 45]. Block-based index skipping divides documents into blocks to estimate the block-wise maximum rank score for pruning [15, 35, 36]. Cluster-based skipping uses a similar strategy [7, 22, 30, 39]. Assume a document collection is divided into a set of K blocks $\{B_1, \cdots, B_K\}$. The key to block-level pruning is computing the BoundSum for each block: $BoundSum(B_i) = \sum_{t \in Q} q_t \cdot BlockMax(B_i, t)$ where each $BlockMax(B_i, t) = \max_{d \in B_i} w_{t,d}$ and is precomputed offline for every block and term in an index. The visitation to block B_i can be safely pruned if $BoundSum(B_i) \leq \theta$, where θ is the current top-k threshold. While $BoundSum(B_i)$ is the upper bound of possible rank scores within this block, the tightness of this bound depends on the ratio of $BlockMax(B_i, t)$ and average term weights of t in each block and can greatly affect the chance of pruning such a block during retrieval [39]. Thus, it is critical to group documents with similar term structures together.

BMP optimizes *BoundSum* tightness by using a massive number of fine-grain blocks and efficiently scoring them using SIMD operations [36]. SP and LSP add a two-level hierarchy that first conducts coarse-grain pruning which enables the use of smaller blocks (with tighter *BoundSums*) [8, 9]. Seismic relies on an approximate query-centroid distance rather than *BoundSum* but improves its cluster selection accuracy by using a small set of unique clusters per posting list [6]. In terms of clustering or block assignment methods, all of the above studies employ a similarity-based method to group similar documents together. BMP, SP, and LSP use BP [14, 29], whereas Seismic uses an approximate K-means algorithm [10].

Bipartite graph partitioning. BP [14, 29] recursively divides a document set D into two partitions to minimize the following cost (called LogGap) when swapping documents between partitions: $\sum_t |Deg(t)| \log_2 \frac{|D|}{Deg(t)+1}$. Here $|D|$ is the document set size and $|D|/(Deg(t) + 1)$ approximates the average gap between two consecutive document IDs in the posting list of term t. As such a loss exploits the term structure during bisection, BP is effective for index compression by arranging similar documents in contiguous blocks [31, 34].

The purpose of employing BP in BMP and Superblock Pruning is solely for grouping similar documents, but not for compressing document IDs [8, 36]. This is because the small block structure in BMP and SP allows compact ID coding within each block instead of delta encoding. Since the LogGap cost in BP was designed for the ID-gap based compression with delta encoding, we revisit the traditional K-means approach which can align better with BMP and SP's goal to group similar documents in each block.

3 K-Means Guided Partitioning

3.1 Design considerations

Clusters produced by K-means have nonuniform sizes, while the BMP and SP retrieval algorithms use a uniform block size. This can simplify index compression and reduce the overhead of accessing blocks during online retrieval. There have been various balanced K-means studies [4, 13, 25, 32] with hard or soft constraints so that each cluster has n/K data points. These techniques target various K-means applications (e.g. workload balancing) that demand relatively equal cluster sizes, and their complexity varies from $O(n^3)$ to $O(n^2K)$ and $O(n \cdot K)$, which is too high when $K = O(n)$.

Our strategy is to adopt the cost function of K-means but impose a balanced partition size constraint while retaining the near linear time complexity advantage of BP, which scales to large sparse datasets when both n and K grow large. For the datasets tested, our evaluation shows imposing the size balancing restriction does not have a visible impact on the goal of minimizing the overall SSE metric in GuideKP.

3.2 Proposed GuideKP method

Given a document collection with n documents, we aim to divide the document set into K blocks with uniform size b: $B_1, B_2, \cdots, B_K$ where $K = \lceil \frac{n}{b} \rceil$. GuideKP conducts a sequence of bisection steps to recursively divide a data partition D into D_1 and D_2 (near) evenly until the partition does not exceed size b. Algorithm 1 shows the substeps involved in partitioning D, which minimizes the following cost:

$$\mathcal{J}(D_1, D_2) = \sum_{\mathbf{x} \in D_1} \|\mathbf{x} - \mathbf{c}_1\|^2 + \sum_{\mathbf{x} \in D_2} \|\mathbf{x} - \mathbf{c}_2\|^2.$$

Algorithm 1 Partitioning in GuideKP during each bisection step

(1) Split D into D_1 and D_2 with $|D_1| = \lceil \frac{|D|}{2b} \rceil * b$ and $D_2 = D - D_1$.
(2) Compute centroids of D_1 and D_2 as c_1 and c_2.
(3) For each document in D_1, compute the gain to move D_1 to D_2. Sort these documents by their gain. Select a top list L_1.
(4) For each document in D_2, compute the gain to move D_2 to D_1. Sort these documents by their gain. Select a top list L_2.
(5) For selected pairs of L_1 and L_2, perform a swap if there is a gain.
(6) Repeat Substeps (3) to (5) for r times or until no more swapping.

We use two heuristics to guide our algorithm:

- The gain for moving a single document u from D_1 to D_2 is estimated as $SingleMove(u, D_1, D_2) = \mathcal{J}(D_1, D_2) - \mathcal{J}(D_1 - \{u\}, D_2 +$

$\{u\}$). A positive gain of $SingleMove(u, D_1, D_2)$ means that there is a potential cost decrease when moving u from D_1 to D_2. This indicates a document could belong better in the other cluster, but does not take into account the documents in the other cluster.

- The true gain of switching documents u and v is $SwapGain(u, v) = \mathcal{J}(D_1, D_2) - \mathcal{J}(D_1^*, D_2^*)$, where $D_1^* = D_1 - \{u\} + \{v\}$ and $D_2^* = D_2 - \{v\} + \{u\}$. This confirms that a swap improves the clustering.

Details on the swapping in bisection Substep (5) is shown in Algorithm 2. The SingleMove computed earlier is used to estimate the possible gain of exchanging u and v, and reject unpromising swapping in advance. Additional details on *SingleMove* and *SwapGain* are described in Section 3.3. Note the partitioning choice in Substep (1) of Algorithm 1 aligns the clusters with the blocks used during indexing with BMP or SP, not splitting evenly like BP.

Algorithm 2 Swapping in Substep (5) of Algorithm 1 for GuideKP

Remove u from L_1 with the highest SingleMove;
Remove v from L_2 with the highest SingleMove;
If $SingleMove(u, D_1, D_2) + SingleMove(v, D_2, D_1) \geq 0$
and if $SwapGain(u, v) > 0$, swap u and v.
Repeat the above steps until L_1 or L_2 becomes empty or $SingleMove(u, D_1, D_2) + SingleMove(v, D_2, D_1) < 0$.

3.3 Gain computing and GuideKP complexity

We present low cost methods for our two gain estimators. Let $n_1 = |D_1|$ and $n_2 = |D_2|$.

SingleMove$(\mathbf{u}, \mathbf{D_1}, \mathbf{D_2})$. The new partitions after moving u are $D_1' = D_1 - \{u\}$ and $D_2' = D_2 + \{u\}$. Their centroids are:

$$c_1' = \frac{c_1 n_1 - u}{n_1 - 1} = c_1 - \frac{u - c_1}{n_1 - 1};\ c_2' = \frac{c_2 n_2 + u}{n_2 + 1} = c_2 + \frac{u - c_2}{n_2 + 1}.$$

We will use the following property for a centroid c of set D:

$$\sum_{x \in D} \|x - c\|^2 = (\sum_{x \in D} \|x\|^2) - |D|\|c\|^2. \tag{1}$$

Then, *SingleMove* can be calculated as:

$$SingleMove(u, D_1, D_2) = \mathcal{J}(D_1, D_2) - \mathcal{J}(D_1', D_2') \tag{2}$$

$$= (\sum_{x \in D_1} \|x\|^2) - n_1\|c_1\|^2 + (\sum_{x \in D_2} \|x\|^2) - n_2\|c_2\|^2 \tag{3}$$
$$- (\sum_{x \in D_1'} \|x\|^2) - (n_1 - 1)\|c_1'\|^2$$
$$- (\sum_{x \in D_2'} \|x\|^2) - (n_2 + 1)\|c_2'\|^2$$

$$= (n_1 - 1)\|c_1'\|^2 - n_1\|c_1\|^2 + (n_2 + 1)\|c_2'\|^2 - n_2\|c_2\|^2 \tag{4}$$

$$= \frac{n_1}{n_1 - 1}(\|c_1\|^2 - 2c_1 \cdot u + \|u\|^2) \tag{5}$$
$$- \frac{n_2}{n_2 + 1}(\|c_2\|^2 - 2c_2 \cdot u + \|u\|^2).$$

$\|c_1\|^2$ and $\|c_2\|^2$ can be precomputed in Substep (2) of Algorithm 1. The cost of $u \cdot c_1$ and $u \cdot c_2$ is proportional to the number of nonzeros in u. Thus, for all documents in D_1 and D_2 in all partitions at the same bisection depth, the total cost including sorting for Substeps (3) and (4) is $O(m + n \log n)$ where m is the number of nonzeros in all documents.

SwapGain$(\mathbf{u}, \mathbf{D_1}, \mathbf{v}, \mathbf{D_2})$. The centroids for the new partitions D_1^* and D_2^* are:

$$c_1^* = c_1 + \frac{v - u}{|D_1|};\ c_2^* = c_2 + \frac{u - v}{|D_2|}$$

Then,

$$SwapGain(u, v) = \mathcal{J}(D_1, D_2) - \mathcal{J}(D_1^*, D_2^*) \tag{6}$$

$$= (\|u\|^2 - \|v\|^2 - n_1\|c_1\|^2 + n_1\|c_1^*\|^2) \tag{7}$$
$$+ (\|v\|^2 - \|u\|^2 - n_2\|c_2\|^2 + n_2\|c_2^*\|^2)$$

$$= n_1(-\|c_1\|^2 + \|c_1^*\|^2) + n_2(-\|c_2\|^2 + \|c_2^*\|^2) \tag{8}$$

$$= n_1(-\|c_1\|^2 + \|c_1 + \frac{v - u}{n_1}\|^2) \tag{9}$$
$$+ n_2(-\|c_2\|^2 + \|c_2 + \frac{u - v}{n_2}\|^2)$$

$$= 2(c_2 - c_1) \cdot (u - v) + \|u - v\|^2 \left(\frac{1}{n_1} + \frac{1}{n_2}\right). \tag{10}$$

Following the same argument for computing *SingleMove*, the above expression can be computed at the same complexity level. Notice that $SwapGain(u, v)$ is called in Substep (5) once for every ranked-pair of documents in the candidate list, then the time complexity of Substep (5) is $O(m)$ for all documents at the same bisection depth. Substep (2) costs $O(m)$. The total complexity of all substeps of GuideKP for all bisection steps is $O(r(m + n \log n) \log K))$, which is the same as the BP algorithm.

4 Evaluation

We extend the Faster BP code using Rust [31], compiled with -O3 optimization. Our code is publicly available.[1] We compare GuideKP with BP [14, 29, 31] and also the following four variants of K-means: 1) Bisect-KM: Bisecting K-means [43] which is competitive with regular K-means and can outperform other hierarchical methods. It has high complexity as its splitting is often not balanced. 2) SparseKM [38]: A state-of-the-art efficient implementation of K-means for large sparse datasets. Its complexity is $O(m \cdot K)$. 3) Bi-SparseKM: We implement a variant of Bisecting-KM [43] using SparseKmeans [38] to perform each bisecting step. 4) BKM++ [13]: A balanced K-means algorithm with complexity $O(n \cdot K)$.

We examine our partitioning quality on the four smallest publicly-available BEIR datasets (NFCorpus, SciFact, Arguana, SciDocs) [44], ranging in size from 3,633 to 25 thousand documents. We choose these small datasets because the above K-means variants are too slow to handle larger datasets with a large number of clusters. We also evaluate the scalability of our partitioning using the MS MARCO passage dataset [12] with 8.8 million English passages, with standard metrics (Recall and mean query latency) on the development (Dev) query set of 6980 queries. All online query latencies are collected after multiple runs with a single thread on a Linux server using Intel i7-1260P with 12 cores, AVX2 SIMD support, and 64GB memory. Before timing queries, all posting lists and metadata for tested queries are pre-loaded into memory, following the common practice. All partitioning times are run using 12 threads. We select SPLADE++ as a representative learned sparse retrieval model and use its embeddings for all datasets and methods [19, 20]. We

[1] https://github.com/pisa-engine/guidekp

found similar results with SPLADEv3 embeddings [24], but omit these results from most tables for space.

Partitioning time and quality. Table 1 compares the document partitioning time and cluster quality of all baselines over four BEIR datasets when b=8, and targeted $K = \frac{n}{b}$. Each dataset reports the size and number of clusters as N and K respectively. This table reports the following metrics. (1) Partitioning time. (2) The average sum of squared errors (SSE). (3) The standard deviation of cluster sizes (σ) which reveals if partitioning is balanced or not [3, 13]. (4) The estimated BoundSum tightness (Bound Tight.), which is the average, over all terms, of the ratio of the maximum weight divided by the average weight for each term within a block. This is a query-independent tightness estimation, and prior work has linked tighter BoundSum estimation with improved query latency [8, 39]. (5) LogGap, which is the cost metric used in BP.

Table 1: Partitioning time and quality metrics on BEIR-4.

Method	Partition Time (s) ↓	SSE ↓ (10^6)	σ ↓	Bound Tight. ↑	LogGap ↓
NFCorpus (N = 3633, K = 454)					
Random	–	161	**0.329**	0.112	3.48
Bisect-KM	12	4.13	8.96	0.377	2.95
SparseKM	2.9	7.67	19.9	0.350	3.41
Bi-SparseKM	39	6.65	8.41	0.319	3.45
BKM++	17200	4.61	**0.329**	0.296	3.41
BP	**0.04**	4.16	**0.329**	0.372	**2.85**
GuideKP	0.10	**3.98**	**0.329**	**0.416**	2.91
SciFact (N = 5183, K = 647)					
Random	–	189	**0.039**	0.120	3.61
Bisect-KM	30	4.80	9.29	0.427	3.05
SparseKM	4.3	11.9	30.2	0.400	3.61
Bi-SparseKM	56	7.69	8.17	0.341	3.62
BP	**0.05**	4.83	**0.039**	0.408	**2.93**
GuideKP	0.14	**4.65**	**0.039**	**0.462**	3.00
Arguana (N = 8674, K = 1084)					
Random	–	153	**0.182**	0.080	3.80
Bisect-KM	58	3.51	9.05	0.403	2.99
SparseKM	9.0	13.7	42.9	0.295	3.34
Bi-SparseKM	101	5.62	8.27	0.276	3.48
BP	**0.11**	3.58	**0.182**	0.390	**2.91**
GuideKP	0.27	**3.37**	**0.182**	**0.451**	2.97
SciDocs (N = 25657, K = 3207)					
Random	–	145	**0.124**	0.094	3.90
Bisect-KM	455	3.53	10.8	0.441	3.20
SparseKM	43	19.2	41.4	0.304	3.89
Bi-SparseKM	359	6.51	9.79	0.308	3.90
BP	**0.32**	3.58	**0.124**	0.410	**3.03**
GuideKP	1.1	**3.41**	**0.124**	**0.487**	3.16

In terms of partitioning time, BP and GuideKP are significantly faster than traditional K-means algorithms. The released code of BKM++ was too slow to converge on any dataset except NFCorpus, where it took 4.8 hours to complete, and GuideKP is 172,000x faster. This was caused by the high dimensionality of our sparse vectors and large number of clusters. GuideKP is 29-414x faster than other K-means variants, while it is 62% slower than BP.

For partitioning quality measured by SSE and BoundSum tightness, GuideKP performs the best, followed by Bisect-KM which is the best variant of traditional K-means. SparseKM is the fastest of the traditional K-means, but is still 29x slower than GuideKP. Bi-SparseKM is 15-295% better than SparseKM in terms of SSE, but tends to be slightly worse on estimated BoundSum tightness. As the number of clusters increases, Bi-SparseKM begins to perform much better than SparseKM. All of the unbalanced K-means methods have high variance in their cluster sizes (σ), while GuideKP, BP, and BKM++ produce balanced clusters with a low σ value. BP performs best on LogGap, which it directly minimizes, while GuideKP and Bisect-KM are also competitive. Later we show that SSE and BoundSum tightness are better clustering metrics than LogGap for predicting retrieval latency.

Table 2: Effect of `max_iter` (r) on MS MARCO.

	Partitioning Time (s) ↓				SSE ↓ (10^6)				Bound Tightness ↑			
`max_iter` =	**1**	**3**	**5**	**20**	**1**	**3**	**5**	**20**	**1**	**3**	**5**	**20**
SPLADE++												
BP	**45.3**	**109**	**163**	**382**	1.75	1.54	1.51	1.48	0.336	0.402	0.407	0.410
GuideKP	104	230	332	804	**1.57**	**1.38**	**1.37**	**1.35**	**0.386**	**0.444**	**0.450**	**0.455**
+ Unaligned	113	214	279	812	1.62	1.45	1.44	1.42	0.350	0.402	0.407	0.412
SPLADE-v3												
BP	**63.0**	**133**	**208**	**577**	3.01	2.59	2.53	2.48	0.375	0.458	0.465	0.470
GuideKP	139	259	380	975	**2.65**	**2.30**	**2.27**	**2.25**	**0.441**	**0.501**	**0.507**	**0.510**
+ Unaligned	166	289	413	970	2.76	2.44	2.41	2.38	0.401	0.457	0.463	0.466

Design choices. Table 2 shows the impact of varying `max_iter` (r) for BP and GuideKP on MS MARCO. The default setting of BP uses $r = 20$, with smart early stopping from [31]. When $r = 3$ for GuideKP, it already outperforms BP in terms of both the SSE and BoundSum tightness. Choosing $r \geq 3$ has diminishing returns and thus we recommend $r = 5$ for fast clustering and $r = 20$ for the best quality. All experiments use $r = 20$ unless otherwise noted. "+ Unaligned" replaces our block-aligned partitioning from Substep (1) of Algorithm 1 with BP's even split. This only decreases SSE by 3-5%, but decreases BoundSum tightness by about 10%. For SPLADE-v3, varying `max_iter` shows a similar pattern, with $r = 3$ still achieving better SSE and Bound Tightness than BP at $r = 20$.

Table 3: Relation between cluster count (K) and quality on BEIR-4. SSE scaled by a factor of 10^6, "BT" is Bound Tightness.

	K = 84		K = 165		K = 337		K = 674		K = 1348	
	SSE ↓	BT ↑	SSE ↓	BT ↑	SSE ↓	BT ↑	SSE ↓	BT ↑	SSE ↓	BT ↑
Bisect-KM	77.4	0.128	38.1	0.170	18.4	0.227	8.74	0.300	3.99	0.410
SparseKM	134	**0.143**	86.0	0.166	42.6	0.215	24.5	0.260	13.1	0.332
BP	77.5	0.131	38.3	0.170	18.5	0.222	**6.78**	0.294	4.04	0.394
GuideKP	**76.4**	0.135	**37.5**	**0.181**	**18.1**	**0.237**	8.51	**0.326**	**3.85**	**0.452**

Impact of varying the number of clusters K on quality. Table 3 studies the cluster quality when varying K from 84 up to 1348 for the 4 BEIR datasets. In terms of SSE, GuideKP is the best for 4 out of 5 cases and outperforms all K-means variants. In terms of the estimated BoundSum tightness, GuideKP outperforms all K-means variants except for small $K = 84$. Thus for a modestly large number of clusters K, GuideKP outperforms these K-means variants in both metrics. GuideKP outperforms BP in all cases except SSE with $K = 674$. Considering results of Table 2 for MS MARCO, GuideKP outperforms BP in both metrics with large K values.

Correlation to retrieval latency. To test if LogGap, SSE and estimated BoundSum tightness are good clustering metrics for retrieval, we compute their correlation separately with retrieval latency in Figure 1 using two block-based retrievers BMP and LSP [9, 36]. Notice that for each retriever, we have applied either BP or GuideKP while the correlation efficient R^2 is computed by aggregating them together. Figure 1 shows that all three metrics are correlated with retrieval time, but SSE and BoundSum tightness have a stronger correlation than LogGap. SSE has the best correlation with query latency for both retrievers, with a correlation coefficient R^2 of over 0.75. BoundSum tightness still has strong correlation with a R^2 of 0.71 to 0.80, while LogGap only has a R^2

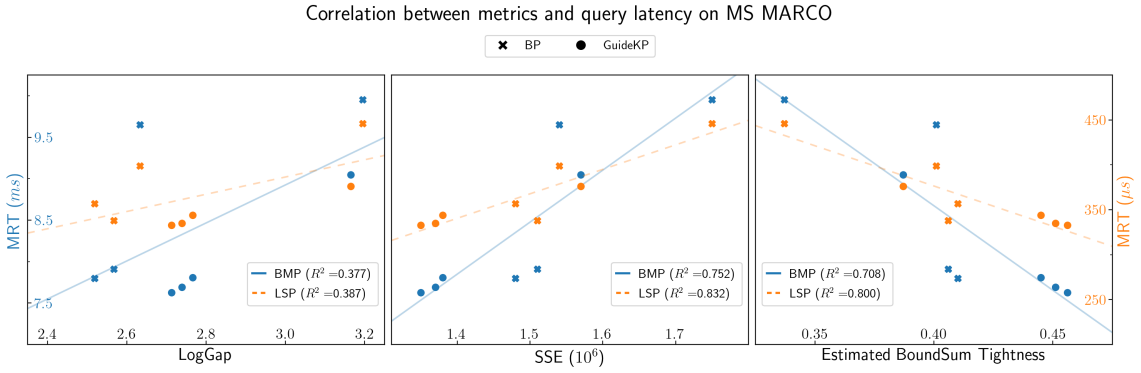


Figure 1: Correlation between LogGap, Est. Bound. Tightness, SSE and retrieval latency with two block-based retrieval engines. The Y axes show the mean query latency (MRT) at retrieval depth 10, with the latency of BMP (using safe search) shown in blue, and the latency of LSP (using their recommended "Config. 1") in orange. Indexes partitioned using BP are shown with an "X" marker, whereas indexes partitioned with GuideKP are shown with an "O". Both BP and GuideKP vary `max_iter={1,3,5,20}`.

of about 0.38. All metrics demonstrate the improvement of partitioning by GuideKP has a stronger effect on BMP than LSP. For example, with a 18% drop in BoundSum tightness resulting in a 19% drop in query latency in BMP, versus a 13% drop in LSP.

When we compute the correlation efficient R^2 separately for BP and GuideKP in each metric, the correlation between BMP's latency and LogGap, SSE, and BoundSum tightness of indexes built using GuideKP is 0.999, 0.998, and 0.999 respectively. LSP also achieves correlations above 0.97 to the retrieval time on GuideKP indexes. In comparison, BP indexes have correlations ranging from a decent 0.54 to strong 0.81. Thus, clustering with GuideKP yields a higher correlation coefficient to the retrieval time than with BP, and an improvement of any of these three metrics with GuideKP will likely reduce retrieval latencies.

Table 4: Mean response time (μs) at a fixed recall budget or on a fixed configuration on MS MARCO Dev with BMP & LSP

Recall Preserved	95% MRT	97% MRT	99% MRT	Fixed Config.
		k=10		
BMP+BP	1724 (+16%)	2082 (+16%)	2696 (+17%)	7800 (+2.2%)
BMP+GuideKP	**1482**	**1788**	**2305**	**7630**
LSP+BP	145 (+11%)	184 (+22%)	267 (+31%)	303 (+13%)
LSP+GuideKP	**131**	**151**	**204**	**269**
		k=1000		
BMP+BP	1474 (+6.0%)	1820 (+7.2%)	2652 (8.0%)	17600 (+32%)
BMP+GuideKP	**1390**	**1697**	**2456**	**13300**
LSP+BP	573 (+11%)	583 (+13%)	878 (+8.5%)	1220 (+9.9%)
LSP+GuideKP	**516**	**516**	**809**	**1110**

Impact of GuideKP partitioning on retrieval latency. Table 4 compares retrieval latency of indexes clustered with GuideKP and BP across fixed configurations and targeted recall budgets (95% to 99%) for both BMP and LSP. For fixed configurations, BMP uses safe search (b=8) and LSP follows the recommended "Config. 1." To ensure a fair comparison, we use a grid search[2] to identify the fastest BMP/LSP configuration for each index that meets the targeted recall. This is because GuideKP typically yields higher recall than BP under identical retrieval parameters, and Columns 2 to 4 represent the best speedup achievable under each budget.

[2]Details omitted for space; they are included with our code release.

Across all configurations, GuideKP yields a latency shorter than BP by 16–31% at k=10 and 7–11% at k=1000. Even with fixed configurations, GuideKP produces faster indexes with a 2.2-32% speedup for BMP and a 10-13% speedup for LSP. These gains are driven by GuideKP's strong clustering quality, which enables more aggressive pruning without sacrificing recall. For instance, using LSP at a 99% recall budget, GuideKP achieves an 8.5% speedup at k=1000 by reducing the number of blocks scored by 23% while keeping BoundSum computation stable. The impact is even more pronounced at k=10, where a 31% speedup is realized; here, improved clustering allows for a 66% reduction in blocks scored and a 17% decrease in BoundSum computation time.

5 Conclusion

We proposed a simple balanced K-means guided bisection algorithm with a log-linear complexity to both n and K. Our evaluation shows that GuideKP is 29-414x faster than the three variants of K-means while having similar or better clustering quality metrics. GuideKP outperforms the fourth variant (BKM++) in all metrics on one dataset while BKM++ is too slow to cluster on the remaining small datasets. On MS MARCO passages with SPLADE++, the four K-means variants cannot handle it due to its size, and GuideKP can outperform BP in both quality metrics SSE and BoundSum tightness and result in up to 32% query latency improvement. GuideKP has the same time complexity as BP, while being up-to 2.1x slower for clustering MS MARCO passages with 1 million clusters, which is still reasonable and scalable for offline processing.

We assessed the impact of GuideKP on retrieval latency using LSP and BMP because they require a very large number of blocks. The future work is to study the effects with different learned sparse representations [21, 26, 33, 41, 46] and other retrievers [6].

Acknowledgments. We thank anonymous referees for their valuable comments. This work is supported in part by U.S. NSF IIS-2225942 and has used the computing resource of the NSF's ACCESS program. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the U.S. NSF.

References

[1] Kazuo Aoyama and Kazumi Saito. 2025. Accelerating Spherical K-Means Clustering for Large-Scale Sparse Document Data. *IEEE Transactions on Knowledge and Data Engineering* 37, 12 (2025), 6833–6846. doi:10.1109/TKDE.2025.3608264

[2] David Arthur and Sergei Vassilvitskii. 2007. k-means++: The Advantages of Careful Seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '07)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1027–1035.

[3] Arindam Banerjee and Joydeep Ghosh. 2006. Scalable Clustering Algorithms with Balancing Constraints. *Data Mining and Knowledge Discovery* 13 (2006), 365–395. doi:10.1007/s10618-006-0040-z

[4] P. S. Bradley, K. P. Bennett, and A. Demiriz. 2000. *Constrained K-Means Clustering*. Technical Report MSR-TR-2000-65. Microsoft Research.

[5] Andrei Z. Broder, David Carmel, Michael Herscovici, Aya Soffer, and Jason Zien. 2003. Efficient query evaluation using a two-level retrieval process. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management* (New Orleans, LA, USA) *(CIKM '03)*. Association for Computing Machinery, New York, NY, USA, 426–434. doi:10.1145/956863.956944

[6] Sebastian Bruch, Franco Maria Nardini, Cosimo Rulli, and Rossano Venturini. 2024. Efficient Inverted Indexes for Approximate Retrieval over Learned Sparse Representations. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Washington DC, USA) *(SIGIR '24)*. Association for Computing Machinery, New York, NY, USA, 152–162. doi:10.1145/3626772.3657769

[7] Fazli Can, Ismail Sengör Altingövde, and Engin Demir. 2004. Efficiency and effectiveness of query processing in cluster-based retrieval. *Information Systems* 29, 8 (2004), 697–717. doi:10.1016/S0306-4379(03)00062-0

[8] Parker Carlson, Wentai Xie, Shanxiu He, and Tao Yang. 2025. Dynamic Superblock Pruning for Fast Learned Sparse Retrieval. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Padua, Italy) *(SIGIR '25)*. Association for Computing Machinery, New York, NY, USA, 3004–3009. doi:10.1145/3726302.3730183

[9] Parker Carlson, Wentai Xie, Rohil Shah, and Tao Yang. 2026. Efficient Sparse Retrieval with Lightweight Superblock Pruning. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '26)*. Association for Computing Machinery, New York, NY, USA. doi:10.1145/3805712.3809739

[10] Flavio Chierichetti, Alessandro Panconesi, Prabhakar Raghavan, Mauro Sozio, Alessandro Tiberi, and Eli Upfal. 2007. Finding near neighbors through cluster pruning. In *Proceedings of the Twenty-Sixth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems* (Beijing, China) *(PODS '07)*. Association for Computing Machinery, New York, NY, USA, 103–112. doi:10.1145/1265530.1265545

[11] Vincent Cohen-Addad, Silvio Lattanzi, Ashkan Norouzi-Fard, Christian Sohler, and Ola Svensson. 2020. Fast and Accurate k-means++ via Rejection Sampling. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 16235–16245. https://proceedings.neurips.cc/paper_files/paper/2020/file/babcff88f8be8c4795bd6f0f8cccca61-Paper.pdf

[12] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Fernando Campos, and Ellen M. Voorhees. 2020. Overview of the TREC 2020 Deep Learning Track. *ArXiv* abs/2102.07662 (2020).

[13] Rieke de Maeyer, Sami Sieranoja, and Pasi Fränti. 2023. Balanced k-means revisited. *Applied Computing and Intelligence* 3, 2 (2023), 145–179. doi:10.3934/aci.2023008

[14] Laxman Dhulipala, Igor Kabiljo, Brian Karrer, Giuseppe Ottaviano, Sergey Pupyrev, and Alon Shalita. 2016. Compressing Graphs and Indexes with Recursive Graph Bisection. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) *(KDD '16)*. Association for Computing Machinery, New York, NY, USA, 1535–1544. doi:10.1145/2939672.2939862

[15] Constantinos Dimopoulos, Sergey Nepomnyachiy, and Torsten Suel. 2013. A Candidate Filtering Mechanism for Fast Top-k Query Processing on Modern CPUs. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Dublin, Ireland) *(SIGIR '13)*. Association for Computing Machinery, New York, NY, USA, 723–732. doi:10.1145/2484028.2484087

[16] Shuai Ding and Torsten Suel. 2011. Faster Top-k Document Retrieval Using Block-Max Indexes. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Beijing, China) *(SIGIR '11)*. Association for Computing Machinery, New York, NY, USA, 993–1002. doi:10.1145/2009916.2010048

[17] Yufei Ding, Yue Zhao, Xipeng Shen, Madanlal Musuvathi, and Todd Mytkowicz. 2015. Yinyang K-means: a drop-in replacement of the classic K-means with consistent speedup. In *Proceedings of the 32nd International Conference on Machine Learning - Volume 37* (Lille, France) *(ICML'15)*. JMLR.org, 579–587.

[18] Charles Elkan. 2003. Using the triangle inequality to accelerate k-means. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning* (Washington, DC, USA) *(ICML'03)*. AAAI Press, 147–153.

[19] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2022. From Distillation to Hard Negative Sampling: Making Sparse Neural IR Models More Effective. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Madrid, Spain) *(SIGIR '22)*. Association for Computing Machinery, New York, NY, USA, 2353–2359. doi:10.1145/3477495.3531857

[20] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, Canada) *(SIGIR '21)*. Association for Computing Machinery, New York, NY, USA, 2288–2292. doi:10.1145/3404835.3463098

[21] Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021. COIL: Revisit Exact Lexical Match in Information Retrieval with Contextualized Inverted List. *NAACL* (2021).

[22] Fatih Hafizoglu, Emre Can Kucukoglu, and Ismail Sengor Altingovde. 2017. On the Efficiency of Selective Search. In *Advances in Information Retrieval - 39th European Conference on IR Research, ECIR (Lecture Notes in Computer Science, Vol. 10193)*. Aberdeen, Scotland, UK, 705–712. doi:10.1007/978-3-319-56608-5_69

[23] Johannes Knittel, Steffen Koch, and Thomas Ertl. 2021. Efficient sparse spherical k-means for document clustering. In *Proceedings of the 21st ACM Symposium on Document Engineering* (Limerick, Ireland) *(DocEng '21)*. Association for Computing Machinery, New York, NY, USA, Article 6, 4 pages. doi:10.1145/3469096.3474937

[24] Carlos Lassance, Hervé Déjean, Thibault Formal, and Stéphane Clinchant. 2024. SPLADE-v3: New baselines for SPLADE. arXiv:2403.06789 [cs.IR]

[25] Zhiyong Li, Jing Liu, and Tommy WS Chow. 2018. Size-regularized cut for cluster analysis. *IEEE Transactions on Knowledge and Data Engineering* 31, 11 (2018), 2095–2108.

[26] Jimmy Lin and Xueguang Ma. 2021. A Few Brief Notes on DeepImpact, COIL, and a Conceptual Framework for Information Retrieval Techniques. *CoRR* abs/2106.14807 (2021). arXiv:2106.14807 https://arxiv.org/abs/2106.14807

[27] Weiwei Liu, Xiaobo Shen, and Ivor Tsang. 2017. Sparse Embedded k-Means Clustering. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/3214a6d842cc69597f9edf26df552e43-Paper.pdf

[28] Stuart Lloyd. 1982. Least squares quantization in PCM. *IEEE Transactions on Information Theory* 28, 2 (1982), 129–137.

[29] Joel Mackenzie, Antonio Mallia, Matthias Petri, J Shane Culpepper, and Torsten Suel. 2019. Compressing inverted indexes with recursive graph bisection: A reproducibility study. In *Advances in Information Retrieval: 41st European Conference on IR Research, ECIR 2019, Cologne, Germany, April 14–18, 2019, Proceedings, Part I 41*. Springer, 339–352.

[30] Joel Mackenzie, Matthias Petri, and Alistair Moffat. 2021. Anytime Ranking on Document-Ordered Indexes. *ACM Transactions on Information Systems (TOIS)* 40, 1, Article 13 (2021), 32 pages.

[31] Joel Mackenzie, Matthias Petri, and Alistair Moffat. 2021. Faster Index Reordering with Bipartite Graph Partitioning. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, Canada) *(SIGIR '21)*. Association for Computing Machinery, New York, NY, USA, 1910–1914. doi:10.1145/3404835.3462991

[32] Mikko I. Malinen and Pasi Fränti. 2014. Balanced K-Means for Clustering. In *Advances in Intelligent Systems and Computing*. 32–41.

[33] Antonio Mallia, Omar Khattab, Torsten Suel, and Nicola Tonellotto. 2021. Learning Passage Impacts for Inverted Indexes. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, Canada) *(SIGIR '21)*. Association for Computing Machinery, New York, NY, USA, 1723–1727. doi:10.1145/3404835.3463030

[34] Antonio Mallia, Michał Siedlaczek, and Torsten Suel. 2019. An experimental study of index compression and DAAT query processing methods. In *European Conference on Information Retrieval*. Springer, 353–368.

[35] Antonio Mallia, Michał Siedlaczek, and Torsten Suel. 2021. Fast Disjunctive Candidate Generation Using Live Block Filtering. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining* (Virtual Event, Israel) *(WSDM '21)*. Association for Computing Machinery, New York, NY, USA, 671–679. doi:10.1145/3437963.3441813

[36] Antonio Mallia, Torsten Suel, and Nicola Tonellotto. 2024. Faster Learned Sparse Retrieval with Block-Max Pruning. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Washington DC, USA) *(SIGIR '24)*. Association for Computing Machinery, New York, NY, USA, 2411–2415. doi:10.1145/3626772.3657906

[37] Kasper Overgaard Mortensen, Fatemeh Zardbani, Mohammad Ahsanul Haque, Steinn Ymir Agustsson, Davide Mottin, Philip Hofmann, and Panagiotis Karras. 2023. Marigold: Efficient k-Means Clustering in High Dimensions. *Proc. VLDB Endow.* 16, 7 (March 2023), 1740–1748. doi:10.14778/3587136.3587147

[38] Khoi Nguyen Pham Dang, He Zhe Lin, and Chih Jen Lin. 2025. SparseKmeans: Efficient K-means Clustering For Sparse Data. In *Proceedings of the 34th ACM*

International Conference on Information and Knowledge Management (Seoul, Republic of Korea) *(CIKM '25)*. Association for Computing Machinery, New York, NY, USA, 6481–6485. doi:10.1145/3746252.3761646

[39] Yifan Qiao, Parker Carlson, Shanxiu He, Yingrui Yang, and Tao Yang. 2024. Threshold-driven Pruning with Segmented Maximum Term Weights for Approximate Cluster-based Sparse Retrieval. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (Eds.). Association for Computational Linguistics, Miami, Florida, USA, 19742–19757. https://aclanthology.org/2024.emnlp-main.1101

[40] Stephen E. Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Found. Trends Inf. Retr.* 3 (2009), 333–389.

[41] Tao Shen, Xiubo Geng, Chongyang Tao, Can Xu, Xiaolong Huang, Binxing Jiao, Linjun Yang, and Daxin Jiang. 2023. LexMAE: Lexicon-Bottlenecked Pretraining for Large-Scale Retrieval. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. International Conference on Learning Representations (ICLR). https://openreview.net/forum?id=PfpEtB3-csK

[42] Jack Spalding-Jamieson, Eliot Wong Robson, and Da Wei Zheng. 2025. Scalable k-Means Clustering for Large k via Seeded Approximate Nearest-Neighbor Search. arXiv:2502.06163 [cs.LG] https://arxiv.org/abs/2502.06163

[43] Michael Steinbach, George Karypis, and Vipin Kumar. 2000. A comparison of document clustering techniques. *TextMining Workshop at KDD2000* (2000).

[44] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*. https://openreview.net/forum?id=wCu6T5xFjeJ

[45] Howard Turtle and James Flood. 1995. Query Evaluation: Strategies and Optimizations. *Information Processing & Management* 31, 6 (1995), 831–850.

[46] Hansi Zeng, Julian Killingback, and Hamed Zamani. 2025. Scaling Sparse and Dense Retrieval in Decoder-Only LLMs. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Padua, Italy) *(SIGIR '25)*. Association for Computing Machinery, New York, NY, USA, 2679–2684. doi:10.1145/3726302.3730225