Check for updates

ACM DIGITAL LIBRARY | Association for Computing Machinery | acm open

Latest updates: https://dl.acm.org/doi/10.1145/3626772.3657769

RESEARCH-ARTICLE

Efficient Inverted Indexes for Approximate Retrieval over Learned Sparse Representations

SEBASTIAN BRUCH

FRANCO MARIA NARDINI, Institute of Information Science and Technologies "Alessandro Faedo", Pisa, PI, Italy

COSIMO RULLI, Institute of Information Science and Technologies "Alessandro Faedo", Pisa, PI, Italy

ROSSANO VENTURINI, University of Pisa, Pisa, PI, Italy

Open Access Support provided by:

Institute of Information Science and Technologies "Alessandro Faedo"

University of Pisa

PDF Download
3626772.3657769.pdf
30 March 2026
Total Citations: 25
Total Downloads: 2701

Published: 11 July 2024

Citation in BibTeX format

SIGIR 2024: The 47th International ACM SIGIR Conference on Research and Development in Information Retrieval
July 14 - 18, 2024
Washington DC, USA

Conference Sponsors:
SIGIR

https://doi.org/10.1145/3626772.3657769
ISBN: 9798400704314

Efficient Inverted Indexes for Approximate Retrieval over Learned Sparse Representations

Sebastian Bruch
Pinecone
New York, USA
sbruch@acm.org

Franco Maria Nardini
ISTI-CNR
Pisa, Italy
francomaria.nardini@isti.cnr.it

Cosimo Rulli
ISTI-CNR
Pisa, Italy
cosimo.rulli@isti.cnr.it

Rossano Venturini
University of Pisa
Pisa, Italy
rossano.venturini@unipi.it

ABSTRACT

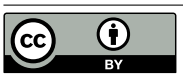
Learned sparse representations form an attractive class of contextual embeddings for text retrieval. That is so because they are effective models of relevance and are interpretable by design. Despite their apparent compatibility with inverted indexes, however, retrieval over sparse embeddings remains challenging. That is due to the distributional differences between learned embeddings and term frequency-based lexical models of relevance such as BM25. Recognizing this challenge, a great deal of research has gone into, among other things, designing retrieval algorithms tailored to the properties of learned sparse representations, including *approximate* retrieval systems. In fact, this task featured prominently in the latest BigANN Challenge at NeurIPS 2023, where approximate algorithms were evaluated on a large benchmark dataset by throughput and recall. In this work, we propose a novel organization of the inverted index that enables fast yet effective approximate retrieval over learned sparse embeddings. Our approach organizes inverted lists into geometrically-cohesive blocks, each equipped with a summary vector. During query processing, we quickly determine if a block must be evaluated using the summaries. As we show experimentally, single-threaded query processing using our method, Seismic, reaches sub-millisecond per-query latency on various sparse embeddings of the Ms Marco dataset while maintaining high recall. Our results indicate that Seismic is one to two orders of magnitude faster than state-of-the-art inverted index-based solutions and further outperforms the winning (graph-based) submissions to the BigANN Challenge by a significant margin.

CCS CONCEPTS

• **Information systems → Retrieval models and ranking**.

KEYWORDS

Learned sparse representations, maximum inner product search, inverted index.



This work is licensed under a Creative Commons Attribution International 4.0 License.

SIGIR '24, July 14–18, 2024, Washington, DC, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0431-4/24/07
https://doi.org/10.1145/3626772.3657769

ACM Reference Format:
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 (SIGIR '24), July 14–18, 2024, Washington, DC, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3626772.3657769

1 INTRODUCTION

Neural Information Retrieval (NIR) has gained increasing popularity since the introduction of pre-trained Large Language Models (LLMs) [28]. NIR models learn a vector representation of short pieces of text, known as an *embedding*, that captures the contextual semantics of the input, thereby enabling more effective matching of queries to documents and, thus, first-stage retrieval [5].

One major focus in NIR is what we call *learned sparse retrieval* (LSR) [17, 18, 20, 25, 31]. LSR repurposes an LLM to encode an input into *sparse* embeddings, a vector in an inner product space where each dimension corresponds with a term in the model's vocabulary. When a coordinate is nonzero in an embedding, that indicates that the corresponding term is semantically relevant to the input. Similarity between embeddings is typically determined by inner product, so that retrieval given a query becomes the problem known as Maximum Inner Product Search (MIPS): Finding the top-k vectors that maximize inner product with a query vector.

LSR is attractive for three reasons. First, LSR models are competitive with *dense retrieval* models that encode text into dense vectors [23, 24, 28, 42, 48, 51, 58]. Importantly, evidence suggests that some LSR models generalize better to out-of-domain datasets [4, 25].

Second, because of the one-to-one mapping between dimensions and vocabulary terms, sparse embeddings are *interpretable* by design. A user can easily understand the embedding space, explain retrieval results, and debug relevance issues. Such properties may be of interest in medical and security applications, for example.

The final reason for their popularity is that sparse embeddings retain many of the benefits of classical lexical models such as BM25 [49] while addressing one of their major weaknesses. That is because, sparse embeddings can, at least in theory, be indexed and retrieved using the all-too-familiar inverted index-based machinery [53], while at the same time, remedying the *vocabulary mismatch* problem due to the incorporation of contextual signals.

Their performance, interpretability, and similarity to lexical models make LSR an important area of research. Efforts in this space include improving the effectiveness of sparse embeddings [17, 18] and the efficiency of sparse retrieval algorithms [6, 7, 19, 35, 37].

The latter category is justified because, despite the apparent compatibility of sparse embeddings with inverted indexes, efficient retrieval remains a challenge. That is so because the weights learned by LSR models exhibit statistical properties that do not conform to the assumptions under which popular inverted index-based retrieval algorithms operate [6, 9, 34]. For example, algorithms such as WAND [2] and MaxScore [54], that are designed for term frequency-based lexical models, function far better than their worst-case complexity would suggest, *if* queries are short and term frequencies follow a Zipfian distribution. In LSR, queries are often longer and, crucially, frequencies are no longer Zipfian [6]. That deviation from assumptions often translates to increased per-query latency.

Overcoming these limitations requires either forcing LSR models to produce the "right" distribution, or designing retrieval algorithms that have fewer restrictive assumptions. As an example of the first direction, Efficient Splade [25] applies L_1 regularization and uses dedicated query and document encoders to make queries shorter. As another, [26] statically prunes documents (or inverted lists) to produce embeddings that approximately maintain semantics but with statistics that are more friendly to dynamic pruning algorithms.

Works in the second direction [6, 7] take a leaf out of the Approximate Nearest Neighbor (ANN) literature [3]: Algorithms that produce *approximate*, as opposed to *exact*, top-k sets. This relaxation makes it easier to trade off accuracy for large gains in efficiency.

Approximate retrieval offers great potential and serves as a bridge between dense and sparse retrieval [7]. So appealing is this paradigm that the 2023 BigANN Challenge[1] at NeurIPS dedicated a track to learned sparse embeddings. Submissions were evaluated on the Splade [19] embeddings of the Ms Marco [43] Passage dataset, and were ranked by the highest throughput past 90% accuracy (i.e., recall with respect to exact search). The results were intriguing: the top two submissions were graph-based ANN methods designed for dense vectors, while other approaches, including an optimized approximate inverted index-based design struggled.

Inspired by BigANN, we present a novel ANN algorithm that we call Seismic (**S**pilled Clust**e**ring of **I**nverted Lists with **S**ummaries for **M**aximum **I**nner Produ**c**t Search) and that admits effective and efficient retrieval over learned sparse embeddings. Pleasantly, our design uses in a new way two familiar data structures: the inverted and the forward index. In particular, we extend the inverted index by introducing a novel organization of inverted lists into geometrically-cohesive blocks. Each block is equipped with a "sketch," serving as a *summary* of the vectors contained in it. The summaries allow us to skip over a large number of blocks during retrieval and save substantial compute. When a summary indicates that a block must be examined, we use the forward index to retrieve exact embeddings of its documents and compute inner products.

We evaluate Seismic against strong baselines, including the top (open-source) submissions to the BigANN Challenge. We additionally include classic inverted index-based retrieval and impact-sorted indexes as reference points for completeness. Experimental results show average per-query latency in **microsecond territory** on various sparse embeddings of Ms Marco [43]. Impressively, Seismic **outperforms the graph-based winning solutions of the BigANN Challenge by a factor of at least 3.4 at 95% accuracy on Splade and 12 on Efficient Splade**, with the margin widening substantially as accuracy increases. Other baselines, including state-of-the-art inverted index-based algorithms, are **consistently one to two orders of magnitude slower than Seismic**.

In summary, we make the following contributions in this work:

- We study an empirical property of learned sparse embeddings that we call the "concentration of importance";
- We present Seismic, a novel ANN algorithm for retrieval over learned sparse vectors that is based on a geometrical organization of the inverted index, and leverages the concentration of importance;
- We report, through extensive experiments, remarkable gains in query latency in exchange for a negligible loss in *retrieval* accuracy, outperforming several state-of-the-art baselines, including the winning submissions to the 2023 BigANN Challenge; and,
- We given an in-depth analysis of Seismic in an ablation study.

2 RELATED WORK

This section reviews notable related research. We summarize the thread of work on learned sparse embeddings, then discuss methods that approach the problem of retrieval over such vector collections.

2.1 Learned Sparse Representations

Learned sparse representations were investigated [59] even before the emergence of pre-trained LLMs. But the rise of LLMs supercharged this research and led to a flurry of activity on the topic [1, 10–12, 17, 19, 27, 31, 60]. First attempts at this include DeepCT and HDCT by Dai and Callan [10–12].

DeepCT used the Transformer [55] encoder of BERT [14] to extract contextual features of a word into an embedding, which can be viewed as a feature vector that characterizes the term's syntactic and semantic role in a given context. DeepCT linearly combines a term's contextualized embedding and summarizes it as a term *weight* for terms that are present in a document. Because the vocabulary associated with a document remains the same, it does not address the vocabulary mismatch problem.

One way to address vocabulary mismatch is to use a generative model, such as doc2query [45] or docT5query [44], to expand documents with relevant terms *and* boost existing terms by repeating them in the document, implicitly performing term re-weighting. In fact, uniCoil-T5 [27, 30] expands its input with DocT5Query [44] before learning and producing a sparse representation.

Formal *et al.* build on SparTerm [1] and propose Splade [20]. Their construction introduces sparsity-inducing regularization and a log-saturation effect on term weights, so that the sparse representations learned by Splade are typically relatively sparser. Interestingly, Splade showed competitive results with respect to state-of-the-art dense and sparse methods [20].

In a later work, Formal *et al.* make adjustments to Splade's pooling and expansion mechanisms, and introduce distillation into its training. This second version, called Splade v2, reached state-of-the-art results on the Ms Marco [43] passage ranking task as well

[1] https://big-ann-benchmarks.com/neurips23.html

as the BEIR [52] zero-shot evaluation benchmark [17]. The SPLADE model has undergone many other rounds of improvements which have been documented in the latest work by the same authors [19]. Among these, one notable extension is the Efficient SPLADE which, as we already noted, attempts to make the learned embeddings more friendly to inverted index-based algorithms.

2.2 Retrieval Algorithms

The Information Retrieval literature offers a wide array of algorithms tailored to retrieval on text collections [53]. They are often *exact* and scale easily to massive datasets. MaxScore [54] and WAND [2], and subsequent improvements [15, 16, 38, 39], are examples that, essentially, solve the MIPS problem over "bag-of-words" representations of text, such as BM25 [49] or TF-IDF [50].

These algorithms operate on an inverted index, augmented with additional data to speed up query processing. One that features prominently is the maximum attainable partial inner product—an upper-bound. This enables the possibility of navigating the inverted lists, one document at a time, and deciding quickly if a document may belong to the result set. Effectively, such algorithms (safely) *prune* the parts of the index that cannot be in the top-k set. That is why they are often referred to as *dynamic pruning* techniques.

Although efficient in practice, dynamic pruning methods are designed specifically for text collections. Importantly, they ground their performance on several pivotal assumptions: non-negativity, higher sparsity rate for queries, and a Zipfian shape of the length of inverted lists. These assumptions are valid for TF-IDF or BM25, which is the reason why dynamic pruning works well and the worst-case time complexity of MIPS is seldom encountered in practice.

These assumptions do not necessarily hold for collections of learned sparse representations, however. Learned vectors may be real-valued, with a sparsity rate that is closer to uniform across dimensions [6, 34]. Mackenzie *et al.* [35] find that learned sparse embeddings reduce the odds of pruning or early-termination in the document-at-a-time (DaaT) and Score-at-a-Time (SaaT) paradigms.

The most similar work to ours is [7]. The authors investigate if *approximate* MIPS algorithms for *dense* vectors port over to *sparse* vectors. They focus on *inverted file* (IVF) where vectors are partitioned into clusters during indexing, with only a fraction of clusters scanned during retrieval. They show that IVF serves as an efficient solution for sparse MIPS. Interestingly, the authors cast IVF as dynamic pruning and turn that insight into a novel organization of the inverted index for approximate MIPS for general sparse vectors. Our index structure can be viewed as an extension of theirs.

Finally, we briefly describe another ANN algorithm over dense vectors: HNSW [36], a graph-based algorithm that constructs a graph where each document is a node and two nodes are connected if they are deemed "similar." Similarity is based on Euclidean distance, but [41] shows inner product results in a structure that is capable of solving MIPS rather quickly and accurately. As we learn in the presentation of our empirical analysis, algorithms that adapt IP-HNSW [41] to sparse vectors work remarkably well.

3 DEFINITIONS AND NOTATION

Suppose we have a collection $\mathcal{X} \subset \mathbb{R}_+^d$ of nonnegative *sparse* vectors. If $x \in \mathcal{X}$, then x is a d-dimensional vector where the vast majority of its coordinates are 0 and the rest are real positive values. We use superscript to enumerate a collection: $x^{(j)}$ is the j-th vector in $\mathcal{X}$.

We use lower-case letters (e.g., x) to denote a vector, call $1 \le i \le d$ its *coordinate*, and write x_i for its i-th *value*. Together, we refer to a coordinate and value pair as an *entry*, and say an entry is non-zero if it has a non-zero value. A sparse vector can be identified as a set of non-zero entries: $\{(i, x_i) \mid x_i \neq 0\}$.

Sparse MIPS aims to solve the following problem to find, from $\mathcal{X}$, the set $\mathcal{S}$ of top k vectors whose inner product with the query vector $q \in \mathbb{R}^d$ is maximal:

$$\mathcal{S} = \arg\max_{x \in \mathcal{X}}^{(k)} \langle q, x \rangle. \tag{1}$$

Let us define a few concepts that we frequently refer to. The L_p norm of a vector denoted by $\|\cdot\|_p$ is defined as $\|x\|_p = (\sum_i |x_i|^p)^{1/p}$. We call the L_p norm of a vector its L_p *mass*. Additionally:

Definition 3.1 (α-mass subvector). Consider a vector x and a permutation π that sorts the entries of x by their absolute value: $|x_{\pi_i}| \ge |x_{\pi_{i+1}}|$. For a constant $\alpha \in [0, 1]$, denote by $1 \le j \le d$ the smallest integer such that:

$$\sum_{i=1}^{j} |x_{\pi_i}| \le \alpha \|x\|_1.$$

We call $\tilde{x}$ made up of $\{(\pi_i, x_{\pi_i})\}_{i=1}^{j}$, the *$\alpha$-mass subvector* of x. Clearly, $\|\tilde{x}\|_1 \le \alpha \|x\|_1$.

4 CONCENTRATION OF IMPORTANCE

Recently, Daliri *et al.* [13] presented a sketching algorithm for sparse vectors that rest on the following simple principle: Coordinates that contribute more heavily to the L_2 norm of a vector, weigh more significantly on the inner product between vectors. Using that intuition, they report that if we were to drop the non-zero coordinates of a sparse vector with a probability proportional to its contribution to the L_2 mass, we can reduce the size of a collection while approximately maintaining inner products between vectors.

Inspired by [13], we examined two state-of-the-art LSR techniques: SPLADE [18] and Efficient SPLADE [25]. Our analysis reveals a parallel property, which we call the "concentration of importance." In particular, we observe that the LSR techniques place a disproportionate amount of the total L_1 mass of a vector on just a small subset of the coordinates.

Let us demonstrate this phenomenon on the MS MARCO Passage dataset [43] with the SPLADE embeddings.[2] We take every vector, sort its entries by value, and measure the fraction of the L_1 mass preserved by considering a given number of top entries. For queries, the top 10 entries yield 0.75-mass subvectors. For documents, the top 50 (about 30% of non-zero entries), give 0.75-mass subvectors. We illustrated our measurements in Figure 1.

These results bring us naturally to our next question: What are the ramifications of the concentration of importance for inner product between queries and documents? One way to study that is as follows: We take the top-10 document vectors for each query, prune each document vector by keeping a fraction of its non-zero

[2]The `cocondenser-ensembledistill` checkpoint was obtained from https://huggingface.co/naver/splade-cocondenser-ensembledistil.

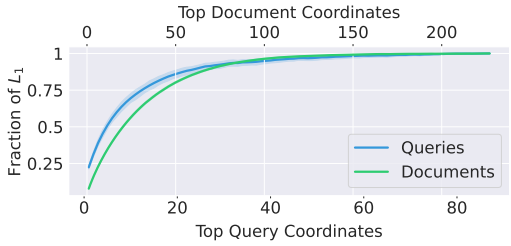


Figure 1: Fraction of L_1 mass preserved by keeping only the top non-zero entries with the largest absolute value.

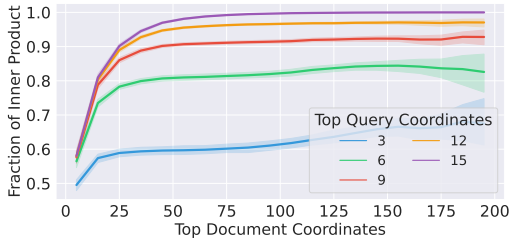


Figure 2: Fraction of inner product (with 95% confidence intervals) preserved by inner product between the top query and document coordinates with the largest absolute value.

entries with the largest value. We do the same for query vectors. We then compute the inner product between the trimmed-down queries and documents and report the results in Figure 2.

The figure shows that, if we took the top 10% of the most "important" coordinates from queries (9) and documents (20), we preserve, on average, 85% of the full inner product. Keeping 12 query and 25 document coordinates bumps that up to 90%.

Our results confirm that LSR tends to concentrate importance on a few coordinates. Furthermore, a partial inner product between the largest entries (by absolute value) approximates the full inner product with arbitrary accuracy. As we will see shortly, this property, which is in agreement with [13], can help speed up query processing and reduce space consumption rather substantially.

5 PROPOSED ALGORITHM

We now introduce Seismic, a novel ANN algorithm that allows effective and efficient approximate retrieval over learned sparse representations. The design of Seismic uses two important and familiar data structures: the inverted index and the forward index. In an nutshell, we use a forward index for inner product computation, and an inverted index to pinpoint the subset of documents that must be evaluated. Figure 3 gives an overview of the overall design.

Seismic is novel in the following ways. First, it uses an organization of the inverted index that blends together *static* and *dynamic* pruning to significantly reduce the number of documents that must be evaluated during retrieval. Second, it partitions inverted lists into geometrically-cohesive blocks to facilitate efficient skipping of blocks. Finally, we attach a *summary* to each block, whose inner product with a query approximates—albeit not necessarily in an unbiased manner—the inner product of the query with documents contained in the block.

5.1 Static Pruning

Seismic heavily relies on the concentration of importance property discussed in Section 4. The property shows that a small subset of the most important coordinates of the sparse embedding of a query and document vector can be used to effectively approximate their inner product. We incorporate this result in Seismic during the construction of the inverted index through *static pruning*.

Concretely, for coordinate i, we build its inverted list by gathering all $x \in \mathcal{X}$ whose $x_i \neq 0$. We then sort the inverted list by x_i's value in decreasing order (breaking ties arbitrarily), so that the document whose i-th coordinate has the largest value appears at the beginning of the list. We then prune the inverted list by keeping at most the first λ entries for a fixed λ—our first hyper-parameter. We denote the resulting inverted list for coordinate i by $\mathcal{I}_i$.

5.2 Blocking of Inverted Lists

Seismic also introduces a novel blocking strategy on inverted lists. It partitions each inverted list into β small blocks—our second hyper-parameter. The rationale behind a blocked organization of an inverted list is to group together documents that are *similar* in terms of their local representations, so as to facilitate a *dynamic pruning* strategy, to be described shortly.

We defer the determination of similarity to a clustering algorithm. In other words, the documents whose ids are present in an inverted list are given as input to a clustering algorithm, which subsequently partitions them into β clusters. Each cluster is then turned into one block, consisting of the id of documents whose vectors belong to the same cluster. Conceptually, each block is "atomic" in the following sense: if the dynamic pruning algorithm decides we must visit a block, *all* the documents in that block are fully evaluated.

We note that any geometrical (supervised or unsupervised) clustering algorithm may be readily used. We use a shallow variant [8] of K-Means as follows. Given a set of vectors $\mathcal{S}$, we uniformly-randomly sample β vectors, $\{\mu^{(j)}\}_{j=1}^{\beta}$, from $\mathcal{S}$, and use them as cluster representatives. For each $x \in \mathcal{S}$, we find $j^* = \arg\max_j \langle x, \mu^{(j)} \rangle$, and assign x to the j^*-th cluster.

5.3 Per-block Summary Vectors

So far we have described how we statically prune inverted lists to the top λ entries and then partition them into β blocks using a clustering algorithm. We now describe how this structure can be used as a basis for a novel dynamic pruning method.

We need an efficient way to determine if a block should be evaluated. To that end, Seismic leverages the concept of a *summary* vector: a d-dimensional vector that "represents" the documents in a block. The summary vectors are stored in the inverted index, one per block, and are meant to serve as an efficient way to compute a good-enough approximation of the inner product between a query and the documents within the block.

One realization of this idea is to upper-bound the full inner product attainable by documents in a block. In other words, the i-th coordinate of the summary vector of a block would contain the

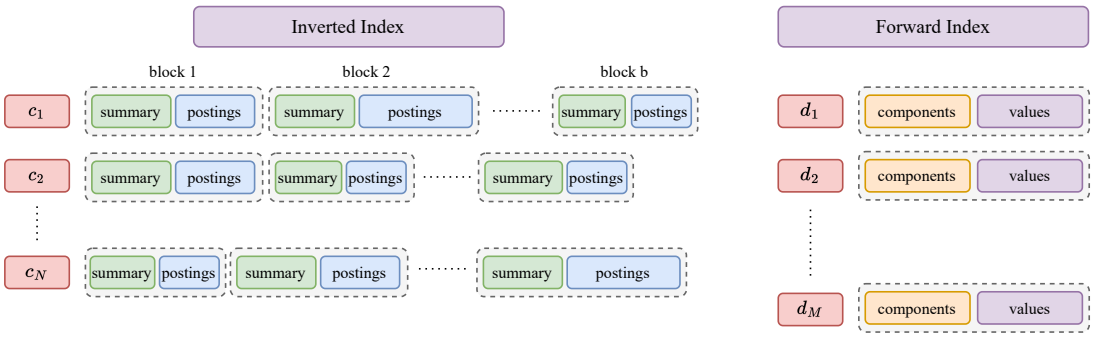


Figure 3: The design of Seismic. Inverted lists are independently partitioned into geometrically-cohesive blocks. Each block is a set of document identifiers with a summary vector. The inner product of a query with the summary approximates the inner product attainable with the documents in that block. The forward index stores the complete vectors (including values).

maximum x_i among the documents in that block. This construction can be best described as a vectorization of the upper-bound *scalars* in blocked variants of WAND [16].

More precisely, our summary function $\phi : 2^{\mathcal{X}} \rightarrow \mathbb{R}^d$ takes a block B from the universe of all blocks $2^{\mathcal{X}}$, and produces a vector whose i-th coordinate is simply:

$$\phi(B)_i = \max_{x \in B} x_i. \tag{2}$$

This summary is *conservative*: its inner product with the query is no less than the inner product between the query and any of its document: $\langle q, \phi(B) \rangle \geq \langle q, x \rangle$ for all $x \in B$ and an arbitrary query q.

The caveat, however, is that the number of non-zero entries in summary vectors grows quickly with the block size. That is the source of two potential issues: 1) the space required to store summaries increases; and 2) as inner product computation takes time proportional to the number of non-zero entries, the time required to evaluate a block could become unacceptably high.

We may address that caveat by applying pruning and quantization, with the understanding that any such method may take away the conservatism of the summary. As we will empirically show, there are many pruning or quantization candidates to choose from.

In particular, we use the following technique that builds on the concentration of importance property: We prune $\phi(B)$, obtained from Equation (2), by keeping only its α-mass subvector. That, α, is our third and last indexing hyper-parameter.

We further reduce the size of summaries by applying scalar quantization. With the goal of reserving a single byte for each value, we subtract the minimum value m from each summary entry, and divide the resulting range into 256 sub-intervals of equal size. A value in the summary is replaced with the index of the sub-interval it maps to. To reconstruct a value approximately, we multiply the id of its sub-interval by the size of the sub-intervals, then add m.

5.4 Forward Index

Seismic blends together two data structures. The first is an inverted index that tells us which documents to examine. To make it practical, we apply approximations that allow us to gain efficiency with a possible loss in accuracy. A forward index, which is simply a look-up table that stores the exact document vectors, helps correct those errors and offers a way to compute the exact inner products between a query and the documents within a block, whenever that block is deemed a good candidate for evaluation.

We must note that, documents belonging to the same block are not necessarily stored consecutively in the forward index. This is simply infeasible because the same document may belong to different inverted lists and, thus, to different blocks. Because of this layout, computing the inner products may incur many cache misses, which are detrimental to query latency. In our implementation, we extensively use prefetching instructions to mitigate this effect.

5.5 Recap

We summarize the discussion above in Algorithm 1. When indexing a collection $\mathcal{X} \subset \mathbb{R}^d$, for every coordinate $i \in \{1, \ldots, d\}$, we form its inverted list, recording only the document identifiers (Line 2). We then sort the list in decreasing order of values (Line 3), and apply static pruning by keeping, for each inverted list, the λ elements with the largest value (Line 4). We then apply clustering to the inverted list to derive at most β blocks (Line 5). Once documents are assigned to the blocks, we then build the block summary using the procedure described earlier (Line 7).

Algorithm 2 shows the query processing logic in Seismic. We use the concentration of importance property to (a) select a subset of the query coordinates q_{cut} (Line 1), and (b) define a novel dynamic pruning strategy (Lines 5–7) that allows to skip blocks in the inverted lists of the coordinates in q_{cut}.

Seismic adopts a coordinate-at-a-time traversal (Line 3) of the inverted index. For each coordinate $i \in q_{\text{cut}}$, it evaluates the blocks using their summary. The documents within a block are evaluated further if the approximation with the summary is greater than a fraction of the minimum inner product in the Min-Heap. That means that, the forward index retrieves the complete document vector in the selected block and computes inner products. A document whose inner product is greater than the minimum score in the Min-Heap is inserted into the heap. Note that, Algorithm 2 takes two hyper-parameters: an integer cut, and heap_factor $\in (0, 1)$.

Algorithm 1: Indexing with Seismic.

Input: Collection $\mathcal{X}$ of sparse vectors in $\mathbb{R}^d$; λ: Maximum length of each inverted list; β: Maximum number of blocks per inverted list; α: Fraction of the overall importance preserved by each summary.
Result: Seismic index.

for $i \in \{1, \dots, d\}$ **do**
  $\mathcal{S} \leftarrow \{j \mid x_i^{(j)} \neq 0,\ x^{(j)} \in \mathcal{X}\}$
  Sort $\mathcal{S}$ in decreasing order by x_i for all $x \in \mathcal{S}$
  $\mathcal{I}_i \leftarrow \{\mathcal{S}_{i,1}, \mathcal{S}_{i,2}, \dots, \mathcal{S}_{i,\lambda}\}$
  Cluster $\mathcal{I}_i$ into β partitions, $\{B_{i,j}\}_{j=1}^{\beta}$
  for $1 \leq j \leq \beta$ **do**
    $S_{i,j} \leftarrow \alpha$-mass subvector of $\phi(B_{i,j})$ {Equation (2)}
return $\mathcal{I}_i$, $\{S_{i,j}\}$ $\forall i, j$

Algorithm 2: Query processing with Seismic.

Input: q: query; k: number of results; cut: number of largest query entries considered; heap_factor: a correction factor to rescale the summary inner product; $\mathcal{I}_i$'s and $S_{i,j}$'s: inverted lists and summaries obtained from Algorithm 1.
Result: A Heap with the top-k documents.

$q_{\text{cut}} \leftarrow$ the top cut entries of q with the largest value
Heap$\leftarrow \emptyset$
for $i \in q_{\text{cut}}$ **do**
  for $B_j \in \mathcal{I}_i$ **do**
    $r \leftarrow \langle q, S_{i,j} \rangle$
    if Heap.len() $= k$ and $r < \frac{\text{Heap.min()}}{\text{heap\_factor}}$ **then**
      continue {Skip the block}
    for $d \in B_j$ **do**
      $p = \langle q, \text{ForwardIndex}[d] \rangle$
      if Heap.len() $< k$ or $p >$ Heap.min() **then**
        Heap.insert(p, d)
      if Heap.len() $= k + 1$ **then**
        Heap.pop_min()
return Heap

6 GENERALIZED ARCHITECTURE

What we presented in Section 5 is an instance of a more general algorithm. Conceptually, Seismic can be viewed as the application of the following logical functions to a collection of sparse vectors.

Clustering with Spillage. We group together documents that share a non-zero coordinate (as inverted lists), then partition them into blocks. This is an instance of *clustering with spillage*, where an item may belong to multiple clusters. The inverted index as *coarse* clustering is efficient for sparse vectors, though other algorithms that allow spillage may very well suit other distributions.

Sketching. We summarize clusters by taking the maximum of each coordinate. While we use the upper-bound vector to obtain a conservative estimate, a more general design admits other types of summaries such as centroids, medoids or any other sketch [56].

Compression. We used pruning and quantization to reduce the total size of summaries by paying particular attention to the L_1 mass. In theory, however, any number of other compression schemes may be utilized, such as [6, 13].

Routing. We identify the subset of clusters that must be fully evaluated by sequentially scanning summaries and comparing their inner product with the minimum score so far. Routing a query to the right cluster, however, need not follow that paradigm strictly. We may consider all summaries at once and decide which clusters to probe in one go—a process akin to the "IVF" approach to ANN [22].

7 EXPERIMENTS

We now evaluate Seismic experimentally. Specifically, we are interested in investigating the performance of Seismic in the following ways: (a) its accuracy, latency, space usage, and indexing time against existing solutions, and (b) an ablation study of the impact of the different components of Seismic on performance.

In what follows, we unpack these questions through an empirical evaluation on two public datasets. We note that, due to space constraints, we excluded many combinations of datasets and LSR models (e.g., uniCoil-T5 embeddings of NQ) from the presentation of our results. However, the reported trends hold consistently.

7.1 Setup

Datasets. We experiment on two publicly-available datasets: Ms Marco v1 Passage [43] and Natural Questions (NQ) from Beir [52]. Ms Marco is a collection of 8.8M passages in English. In our evaluation, we use the smaller "dev" set of queries for retrieval, which includes 6,980 questions. NQ is a collection of 2.68M questions in English. We use it in combination with its "test" set of 7,842 queries.

Learned Sparse Representations. We evaluate Seismic with embeddings generated by three LSR models:

- Splade [18]. Each non-zero entry is the importance weight of a term in the BERT [14] WordPiece [57] vocabulary consisting of 30,000 terms. We use the `cocondenser-ensembledistil`[3] version of Splade that yields MRR@10 of 38.3 on the Ms Marco dev set. The number of non-zero entries in documents (queries) is, on average, 119 (43) for Ms Marco and 153 (51) for NQ.
- Efficient Splade [25]. Similar to Splade, but there are 181 (5.9) non-zero entries in Ms Marco documents (queries). We use the `efficient-splade-V-large`[4] version, yielding MRR@10 of 38.8 on the Ms Marco dev set. We refer to this model as E-Splade.
- uniCoil-T5 [27, 30]. Expands passages with relevant terms generated by DocT5Query [44]. uniCoil-T5 achieves MRR@10 of 35.2 on the Ms Marco dev set. There are, on average, 68 (6) non-zero entries in Ms Marco documents (queries).

It is worth highlighting that these embedding models belong to different families. Splade and E-Splade perform expansion for both queries and documents. On the other hand, uniCoil-T5 only performs document expansion and does so using a generative model.

We generate the Splade and E-Splade embeddings using the original code published on GitHub.[5] uniCoil-T5 embeddings are based on the original implementation on GitHub.[6] After generating

[3] Checkpoint at https://huggingface.co/naver/splade-cocondenser-ensembledistil
[4] Checkpoints at https://huggingface.co/naver/efficient-splade-V-large-doc and https://huggingface.co/naver/efficient-splade-V-large-query.
[5] https://github.com/naver/splade
[6] https://github.com/castorini/pyserini/blob/master/docs/experiments-unicoil.md

the embeddings, we replicate the performance in terms of MRR@10 on the Ms Marco dev set to confirm that our replication achieves the same performance presented in the original papers.

Baselines. We compare Seismic with five state-of-the-art retrieval solutions. Two of these are the winning solutions of the "Sparse Track" at the 2023 BigANN Challenge[7] at NeurIPS. These include:

- GrassRMA: A graph-based method for dense vectors adapted to sparse vectors that appears in the BigANN challenge as "sHnsw."[8]
- PyAnn: Another graph-based ANN solution.[9]

The other three baselines are inverted index-based solutions:

- Ioqp [32]: Impact-sorted query processor written in Rust. We choose Ioqp because it is known to outperform JASS [29], a widely-adopted open-source impact-sorted query processor.
- SparseIvf [7]: An inverted index where lists are partitioned into blocks through clustering. At query time, after finding the N closest clusters to the query, a coordinate-at-a-time algorithm traverses the inverted lists. The solution is approximate because only documents that belong to top N clusters are considered.
- Pisa [40]: An inverted index-based C++ library based on `ds2i` [46] that uses highly-optimized blocked variants of WAND. Pisa is *exact* as it traverses inverted lists in a rank-safe manner.

We also considered the method by Lassance *et al.* [26]. Their approach statically prunes either inverted lists (by keeping p-quantile of elements), documents (by keeping a fixed number of top entries), or all coordinates whose value is below a threshold. While simple, [26] is only able to speed up query processing by 2–4× over Pisa on E-Splade embeddings of Ms Marco. We found it to be ineffective on Splade and generally far slower than GrassRMA and PyAnn. As such we do not include it in our discussions.

We build Ioqp and Pisa indexes using Anserini[10] and apply recursive graph bisection [33]. For Ioqp, we vary the *fraction* (of the total collection) hyper-parameter in $[0.1, 1]$ with step size of 0.05. For SparseIvf, we sketch documents using Sinnamon$_{\text{Weak}}$ and a sketch size of 1,024, and build $4\sqrt{N}$ clusters, where N is the number of documents in the collection. For GrassRMA and PyAnn, we build different indexes by running all possible combinations of $ef_c \in \{1000, 2000\}$ and $M \in \{16, 32, 64, 128, 256\}$. During search we test $ef_s \in [5, 100]$ with step size 5, then $[100, 400]$ with step 10, $[100, 1000]$ with step 100, and finally $[1000, 5000]$ with step 500. We apply early stopping when accuracy saturates.

Our grid search for Seismic on Ms Marco is over: $\lambda \in [1500, 7500]$ with step size of 500, $\beta \in [150, 750]$ with step 50, and $\alpha \in [0.1, 0.5]$ with 0.1. Best results are achieved with $\lambda = 6{,}000$, $\beta = 400$, and $\alpha = 0.4$. The grid search for Seismic on NQ is over: $\lambda \in \{4500, 5250, 6000\}$, $\beta \in \{300, 350, 400, 450, 525, 600, 700, 800\}$, and $\alpha \in \{0.3, 0.4, 0.5\}$. Best results are achieved with $\lambda = 5{,}250$, $\beta = 525$, and $\alpha = 0.5$. Seismic employs 8-bit scalar quantization for summaries. Moreover, Seismic uses matrix multiplication to efficiently multiply the query vector with all quantized summaries of an inverted list.

Evaluation Metrics. We evaluate all methods using three metrics:

- Latency (μsec.). The time elapsed, in *microseconds*, from the moment a query vector is presented to the index to the moment it returns the requested top k document vectors running in single thread mode. Latency does not include embedding time.
- Accuracy. The percentage of true nearest neighbors recalled in the returned set. By measuring the recall of an approximate set given the exact top-k set, we study the impact of the different levers in an algorithm on its overall accuracy as a retrieval engine.
- Index size (MiB). The space the index occupies in memory.

Reproducibility and Hardware Details. We implemented Seismic in Rust.[11] We compile Seismic by using the version 1.77 of Rust and use the highest level of optimization made available by the compiler. We conduct experiments on a server equipped with one Intel i9-9900K CPU with a clock rate of 3.60 GHz and 64 GiB of RAM. The CPU has 8 physical cores and 8 hyper-threaded ones. We query the index using a single thread.

7.2 Results

We now present our experimental results. We begin by comparing the performance of Seismic with baselines. We then ablate Seismic to understand the impact of our design choices on performance.

7.2.1 Accuracy-Latency Trade-off. Table 1 details retrieval performance in terms of average per-query latency for Splade, E-Splade, and uniCoil-T5 on Ms Marco, and Splade on NQ. We frame the results as the trade-off between effectiveness and efficiency. In other words, we report mean per-query latency at a given accuracy level.

The results on these datasets show Seismic's remarkable relative efficiency, reaching a latency that is often one to two orders of magnitude smaller. Overall, Seismic consistently outperforms all baselines at all accuracy levels, including GrassRMA and PyAnn, which in turn perform better than other inverted index-based baselines—confirming the findings of the BigANN Challenge.

We make a few additional observations. Ioqp appears to be the slowest method across datasets. This is not surprising considering the distributional abnormalities of learned sparse vectors, as discussed earlier. SparseIvf generally improves over Ioqp, but Seismic speeds up query processing further. In fact, the minimum speedup over Ioqp (SparseIvf) on Ms Marco is 84.6× (22.3×) on Splade, 24.9× (20.9×) on E-Splade, and 143.3× (53.6×) on uniCoil-T5.

Seismic consistently outperforms GrassRMA and PyAnn by a substantial margin, ranging from 2.6× (Splade on Ms Marco) to 21.6× (E-Splade on Ms Marco) depending on the level of accuracy. In fact, as accuracy increases, the latency gap between Seismic and the two graph-based methods widens. This gap is much larger when query vectors are sparser, such as with E-Splade embeddings. That is because, when queries are highly sparse, inner products between queries and documents become smaller, reducing the efficacy of a greedy graph traversal. As one data point, PyAnn over E-Splade embeddings of Ms Marco visits roughly 40,000 documents to reach 97% accuracy, whereas Seismic evaluates just 2,198 documents.

Finally, we highlight that Pisa is the slowest (albeit, *exact*) solution. On Ms Marco, Pisa processes queries in about 100,325 microseconds on Splade embeddings. On E-Splade and uniCoil-T5, its average latency is 7,947 and 9,214 microseconds, respectively.

[7] https://big-ann-benchmarks.com/neurips23.html
[8] C++ code is publicly available at https://github.com/Leslie-Chung/GrassRMA.
[9] C++ code is publicly available at https://github.com/veaaaab/pyanns.
[10] https://github.com/castorini/anserini
[11] Our code is publicly available at https://github.com/TusKANNy/seismic.

Accuracy (%)	90		91		92		93		94		95		96		97	
SPLADE on MS MARCO																
IOQP	17,423	(93.2×)	17,423	(84.6×)	18,808	(91.2×)	21,910	(98.7×)	24,382	(90.6×)	31,843	(105.1×)	35,735	(102.7×)	51,522	(97.0×)
SPARSEIVF	4,169	(22.3×)	4,984	(24.2×)	6,442	(31.3×)	7,176	(32.3×)	8,516	(31.7×)	10,254	(33.8×)	12,881	(37.0×)	15,840	(29.8×)
GRASSRMA	807	(4.3×)	867	(4.2×)	956	(4.6×)	1,060	(4.8×)	1,168	(4.3×)	1,271	(4.2×)	1,577	(4.5×)	1,984	(3.7×)
PYANN	489	(2.6×)	539	(2.6×)	603	(2.9×)	654	(2.9×)	845	(3.1×)	1,016	(3.4×)	1,257	(3.6×)	1,878	(3.5×)
SEISMIC (ours)	187	–	206	–	206	–	222	–	269	–	303	–	348	–	531	–
E-SPLADE on MS MARCO																
IOQP	7,857	(35.4×)	8,382	(37.8×)	8,892	(37.2×)	9,858	(41.2×)	10,591	(41.4×)	11,536	(30.7×)	11,934	(31.2×)	14,485	(24.9×)
SPARSEIVF	4,643	(20.9×)	5,058	(22.8×)	5,869	(24.6×)	6,599	(27.6×)	7,555	(29.5×)	8,962	(23.8×)	10,414	(27.2×)	13,883	(23.9×)
GRASSRMA	2,074	(9.3×)	2,658	(12.0×)	2,876	(12.0×)	3,490	(14.6×)	4,431	(17.3×)	5,141	(13.7×)	7,181	(18.7×)	12,047	(20.7×)
PYANN	1,685	(7.6×)	1,702	(7.7×)	2,045	(8.6×)	2,409	(10.1×)	3,119	(12.2×)	4,522	(12.0×)	7,317	(19.1×)	12,578	(21.6×)
SEISMIC (ours)	222	–	222	–	239	–	239	–	256	–	376	–	383	–	581	–
UNICOIL-T5 on MS MARCO																
IOQP	22,278	(193.7×)	25,060	(203.7×)	26,541	(199.6×)	30,410	(181.0×)	33,327	(198.4×)	34,061	(189.2×)	38,399	(143.3×)	40,759	(145.6×)
SPARSEIVF	6,375	(55.4×)	7,072	(57.5×)	8,192	(61.6×)	9,207	(54.8×)	10,306	(61.3×)	12,308	(68.4×)	14,359	(53.6×)	17,572	(62.8×)
GRASSRMA	1,318	(11.5×)	1,434	(11.7×)	1,812	(13.6×)	2,004	(11.9×)	2,168	(12.9×)	2,668	(14.8×)	4,140	(15.4×)	5,340	(19.1×)
PYANN	1,133	(9.9×)	1,456	(11.8×)	1,741	(13.1×)	1,755	(10.4×)	2,061	(12.3×)	2,973	(16.5×)	3,883	(14.5×)	6,324	(22.6×)
SEISMIC (ours)	115	–	123	–	133	–	168	–	168	–	180	–	268	–	280	–
SPLADE on NQ																
IOQP	8,313	(42.6×)	8,854	(45.4×)	9,334	(44.2×)	11,049	(46.0×)	11,996	(48.0×)	14,180	(53.3×)	15,254	(53.3×)	18,120	(50.1×)
SPARSEIVF	3,862	(19.8×)	4,309	(22.1×)	4,679	(22.2×)	5,464	(22.8×)	6,113	(24.5×)	6,675	(25.1×)	7,796	(27.3×)	9,109	(25.2×)
GRASSRMA	1,000	(5.1×)	1,138	(5.8×)	1,208	(5.7×)	1,413	(5.9×)	1,549	(6.2×)	2,091	(7.9×)	2,448	(8.6×)	3,038	(8.4×)
PYANN	610	(3.1×)	668	(3.4×)	748	(3.5×)	870	(3.6×)	1,224	(4.9×)	1,245	(4.7×)	1,469	(5.1×)	1,942	(5.4×)
SEISMIC (ours)	195	–	195	–	211	–	240	–	250	–	266	–	286	–	362	–

Table 1: Mean latency (μsec/query) at different accuracy cutoffs with speedup (in parenthesis) gained by SEISMIC over others.

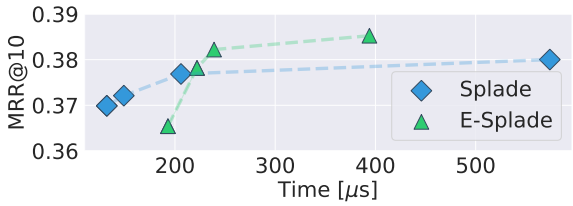


Figure 4: MRR@10 on MS MARCO.

Model	Index size (MiB)	Index build time (min.)
SPLADE on MS MARCO		
IOQP	2,195	-
SPARSEIVF	8,830	44
GRASSRMA	10,489	267
PYANN	5,262	137
SEISMIC (ours)	6,416	5

Table 2: Index size and build time.

We note that its high latency on SPLADE is largely due to the much larger number of non-zero entries in queries.

We conclude with a remark on the relationship between retrieval accuracy (as measured by recall with respect to exact search) and ranking quality (such as MRR and NDCG [21] given relevance judgments). Even though ranking quality is not our primary focus, we measured MRR@10 on MS MARCO for the approximate top-k sets obtained from SEISMIC, and plot that as a function of per-query latency in Figure 4. While MRR@10 is relatively stable, we do notice a drop in the low-latency (and thus low-accuracy) regime. Perhaps more interesting is the fact that SEISMIC can speed up retrieval over SPLADE so much that if the time budget is tight, using SPLADE embeddings gets us to a higher MRR@10 faster.

7.2.2 Space and Build Time. Table 2 records the time it takes to index the entire MS MARCO collection embedded with SPLADE with different methods, and the size of the resulting index. We perform this experiment on a machine with two Intel Xeon Silver 4314 CPUs clocked at 2.40GHz, with 32 physical cores plus 32 hyper-threaded ones and 512 GiB of RAM. We build the indexes by using multi-threading parallelism with 64 cores.

We left out the build time for IOQP because its index construction has many external dependencies (such as Anserini and graph bisection) that makes giving an accurate estimate difficult.

Trends for other datasets are similar to those reported in Table 2. Notably, indexes produced by approximate methods are larger. That makes sense: using more auxiliary statistics helps narrow the search space dynamically and quickly. Among the highly efficient methods, the size of SEISMIC's index is mild, especially compared with GRASSRMA. Importantly, SEISMIC builds its index in a fraction of the time it takes PYANN or GRASSRMA to index the collection.

7.3 Ablation Study

We now take SEISMIC apart to study the impact of its components. We take the SPLADE embeddings of MS MARCO and analyze the impact of (a) quantization on summaries; (b) two strategies to partition inverted lists; and (c) two methods for building the summary vectors.

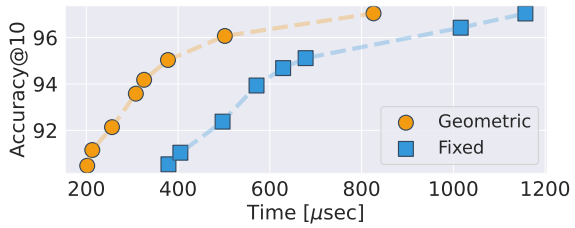


Figure 5: Fixed vs. geometric blocking. Data sampled from parameters: cut $\in \{1, \ldots, 10\}$ and heap_factor $\in \{0.7, 0.8, 0.9, 1.0\}$.

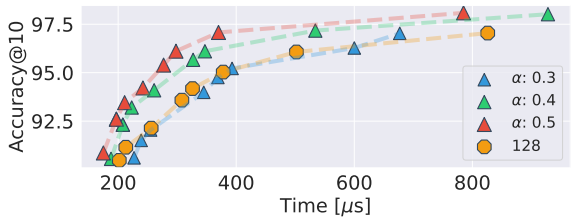


Figure 6: Fixed (128 top entries per summary) vs. importance-based (α-mass subvectors) summaries.

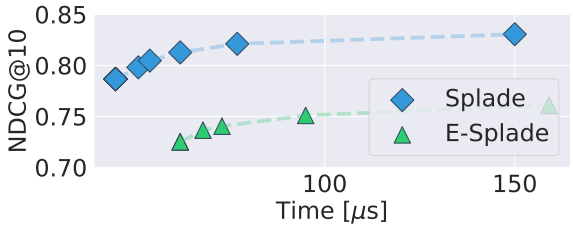


Figure 7: NDCG@10 on the Quora dataset.

Quantization of Summaries. We empirically observe that the scalar quantization applied to summaries does not hinder the effectiveness or the efficiency of Seismic. Indeed, it reduces the memory footprint of the summaries by a factor of 4.

Fixed vs. Geometric Blocking. We delegate inverted list blocking to a clustering algorithm. In this section, we wish to understand the impact of *geometric* clustering on the performance of Seismic. To that end, we compare two realizations of the index. In one, called "geometric" blocking, we use a variant of K-Means as described in Section 5.2. Separately, in what we call "fixed" blocking, we take the impact-sorted inverted lists and chunk them into fixed-size groups. We then compare the performance of these two configurations on the accuracy-latency trade-off space. Figure 5 reports our results, showing that geometric blocking significantly outperforms fixed blocking for all ranges of hyper-parameters considered.

Fixed vs. Importance-based Summaries. Recall that, our summary vectors are α-mass subvectors of the vector produced by Equation (2). In a sense, the summary reflects the distribution of documents within a block. Here, we contrast that "importance-based" summary generation with a simple alternative: Keeping a *fixed* number of top entries of the vector from Equation (2). The drawback of this alternative is that we store the same number of entries for each block regardless of the number of documents in the block or the distribution of their importance, thus weakening the performance of Seismic.

Figure 6 visualizes the latency-accuracy trade-off of these different settings. It is clear that, for a fixed time budget, importance-based summaries lead to better accuracy than fixed-length summaries. Moreover, summaries with 128 top entries take 2,687 MiB of space, while importance-based summaries with $\alpha = 0.5$ consume 2,885 MiB (without quantization). Reducing α to 0.4 and 0.3 lowers the size to 2,303 and 1,801 MiB, respectively.

Forward Index. The forward index could use 32- or 16-bit floating points to store vector values. We use half-precision, leading to 4,113 MiB of space usage at negligible cost to accuracy and no impact on latency. We confirm that PyAnn too uses this representation.

8 CONCLUDING REMARKS

We presented Seismic, a novel approximate algorithm that facilitates effective and efficient retrieval over learned sparse embeddings. We showed empirically its remarkable efficiency on a number of embeddings of publicly-available datasets. Seismic outperforms existing methods, including the winning, graph-based algorithms at the BigANN Challenge in NeurIPS 2023 that use similar-sized (or larger) indexes.

One of the exciting opportunities that our research creates is that it offers a new way of thinking about sparse embedding models. Let us explain how. When Splade proved difficult to scale because state-of-the-art inverted index-based solutions failed to process queries fast enough, the community moved towards E-Splade and other variants that reduce query processing time, but that exhibit degraded performance in zero-shot settings. Evidence suggests, for example, that E-Splade embeddings of Quora—a Beir dataset—yield NDCG@10 of 0.76 while Splade embeddings yield 0.83.

Seismic changes that equation. As we visualize in Figure 7, for any given time budget, Seismic retrieves a better-quality top-k set from the Splade embeddings of Quora. The key take-away message is clear: Seismic speeds up retrieval over Splade so dramatically that switching to E-Splade becomes unnecessary and, in fact, detrimental to both efficiency and effectiveness.

As future work, we intend to explore the application of compression techniques for inverted lists [47] to further reduce the size of inverted and forward indexes.

Acknowledgements. This work was partially supported by the Horizon Europe RIA "Extreme Food Risk Analytics" (EFRA), grant agreement n. 101093026, by the PNRR - M4C2 - Investimento 1.3, Partenariato Esteso PE00000013 - "FAIR - Future Artificial Intelligence Research" - Spoke 1 "Human-centered AI" funded by the European Commission under the NextGeneration EU program, by the PNRR ECS00000017 Tuscany Health Ecosystem Spoke 6 "Precision medicine & personalized healthcare" funded by the European Commission under the NextGeneration EU programme, by the MUR-PRIN 2017 "Algorithms, Data Structures and Combinatorics for Machine Learning", grant agreement n. 2017K7XPAN_003, and by the MUR-PRIN 2022 "Algorithmic Problems and Machine Learning", grant agreement n. 20229BCXNW.

REFERENCES

[1] Yang Bai, Xiaoguang Li, Gang Wang, Chaoliang Zhang, Lifeng Shang, Jun Xu, Zhaowei Wang, Fangshan Wang, and Qun Liu. 2020. SparTerm: Learning Term-based Sparse Representation for Fast Text Retrieval. arXiv:2010.00768 [cs.IR]

[2] 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 12th International Conference on Information and Knowledge Management* (New Orleans, LA, USA). 426–434.

[3] Sebastian Bruch. 2024. *Foundations of Vector Retrieval.* Springer Nature Switzerland.

[4] Sebastian Bruch, Siyu Gai, and Amir Ingber. 2023. An Analysis of Fusion Functions for Hybrid Retrieval. *ACM Transactions on Information Systems* 42, 1, Article 20 (August 2023), 35 pages.

[5] Sebastian Bruch, Claudio Lucchese, and Franco Maria Nardini. 2023. Efficient and Effective Tree-based and Neural Learning to Rank. *Foundations and Trends® in Information Retrieval* 17, 1 (2023), 1–123.

[6] Sebastian Bruch, Franco Maria Nardini, Amir Ingber, and Edo Liberty. 2023. An Approximate Algorithm for Maximum Inner Product Search over Streaming Sparse Vectors. *ACM Transactions on Information Systems* 42, 2, Article 42 (November 2023), 43 pages.

[7] Sebastian Bruch, Franco Maria Nardini, Amir Ingber, and Edo Liberty. 2023. Bridging Dense and Sparse Maximum Inner Product Search. arXiv:2309.09013 [cs.IR]

[8] 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). 103–112.

[9] Matt Crane, J. Shane Culpepper, Jimmy Lin, Joel Mackenzie, and Andrew Trotman. 2017. A Comparison of Document-at-a-Time and Score-at-a-Time Query Evaluation. In *Proceedings of the 10th ACM International Conference on Web Search and Data Mining* (Cambridge, United Kingdom). 201–210.

[10] Zhuyun Dai and Jamie Callan. 2019. Context-Aware Sentence/Passage Term Importance Estimation For First Stage Retrieval. arXiv:1910.10687 [cs.IR]

[11] Zhuyun Dai and Jamie Callan. 2020. Context-Aware Document Term Weighting for Ad-Hoc Search. In *Proceedings of The Web Conference* (Taipei, Taiwan). 1897–1907.

[12] Zhuyun Dai and Jamie Callan. 2020. Context-Aware Term Weighting For First Stage Passage Retrieval. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, China). 1533–1536.

[13] Majid Daliri, Juliana Freire, Christopher Musco, Aécio Santos, and Haoxiang Zhang. 2023. Sampling Methods for Inner Product Sketching. arXiv:2309.16157 [cs.DB]

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers).* 4171–4186.

[15] Constantinos Dimopoulos, Sergey Nepomnyachiy, and Torsten Suel. 2013. Optimizing Top-k Document Retrieval Strategies for Block-Max Indexes. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining* (Rome, Italy). 113–122.

[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). 993–1002.

[17] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE v2: Sparse Lexical and Expansion Model for Information Retrieval. arXiv:2109.10086 [cs.IR]

[18] 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). 2353–2359.

[19] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2023. Towards Effective and Efficient Sparse Neural Information Retrieval. *ACM Transactions on Information Systems* (December 2023).

[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). 2288–2292.

[21] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated Gain-based Evaluation of IR Techniques. *ACM Transactions on Information Systems* 20, 4 (2002), 422–446.

[22] Herve Jégou, Matthijs Douze, and Cordelia Schmid. 2011. Product Quantization for Nearest Neighbor Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33, 1 (2011), 117–128.

[23] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing.* 6769–6781.

[24] Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, China). 39–48.

[25] Carlos Lassance and Stéphane Clinchant. 2022. An Efficiency Study for SPLADE Models. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Madrid, Spain). 2220–2226.

[26] Carlos Lassance, Simon Lupart, Hervé Déjean, Stéphane Clinchant, and Nicola Tonellotto. 2023. A Static Pruning Study on Sparse Neural Retrievers. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Taipei, Taiwan). 1771–1775.

[27] Jimmy Lin and Xueguang Ma. 2021. A Few Brief Notes on DeepImpact, COIL, and a Conceptual Framework for Information Retrieval Techniques. arXiv:2106.14807 [cs.IR]

[28] Jimmy Lin, Rodrigo Frassetto Nogueira, and Andrew Yates. 2021. *Pretrained Transformers for Text Ranking: BERT and Beyond.* Morgan & Claypool Publishers.

[29] Jimmy Lin and Andrew Trotman. 2015. Anytime Ranking for Impact-Ordered Indexes. In *Proceedings of the 2015 International Conference on The Theory of Information Retrieval* (Northampton, Massachusetts, USA). 301–304.

[30] Xueguang Ma, Ronak Pradeep, Rodrigo Nogueira, and Jimmy Lin. 2022. Document expansion baselines and learned sparse lexical representations for ms marco v1 and v2. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval.* 3187–3197.

[31] Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, Nicola Tonellotto, Nazli Goharian, and Ophir Frieder. 2020. Expansion via Prediction of Importance with Contextualization. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, China). 1573–1576.

[32] J. Mackenzie, M. Petri, and L. Gallagher. 2022. IOQP: A simple Impact-Ordered Query Processor written in Rust. In *Proc. DESIRES.* 22–34.

[33] J. Mackenzie, M. Petri, and A. Moffat. 2021. Faster Index Reordering with Bipartite Graph Partitioning. In *Proc. SIGIR.* 1910–1914.

[34] Joel Mackenzie, Andrew Trotman, and Jimmy Lin. 2021. Wacky Weights in Learned Sparse Representations and the Revenge of Score-at-a-Time Query Evaluation. arXiv:2110.11540 [cs.IR]

[35] Joel Mackenzie, Andrew Trotman, and Jimmy Lin. 2023. Efficient Document-at-a-Time and Score-at-a-Time Query Evaluation for Learned Sparse Representations. *ACM Transactions on Information Systems* 41, 4 (2023).

[36] Yu A. Malkov and D. A. Yashunin. 2020. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42, 4 (4 2020), 824–836.

[37] Antonio Mallia, Joel Mackenzie, Torsten Suel, and Nicola Tonellotto. 2022. Faster Learned Sparse Retrieval with Guided Traversal. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Madrid, Spain). 1901–1905.

[38] Antonio Mallia, Giuseppe Ottaviano, Elia Porciani, Nicola Tonellotto, and Rossano Venturini. 2017. Faster BlockMax WAND with Variable-Sized Blocks. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Shinjuku, Tokyo, Japan). 625–634.

[39] Antonio Mallia and Elia Porciani. 2019. Faster BlockMax WAND with Longer Skipping. In *Advances in Information Retrieval.* 771–778.

[40] Antonio Mallia, Michal Siedlaczek, Joel Mackenzie, and Torsten Suel. 2019. PISA: Performant Indexes and Search for Academia. In *Proceedings of the Open-Source IR Replicability Challenge co-located with 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, OSIRRC@SIGIR 2019, Paris, France.* 50–56.

[41] Stanislav Morozov and Artem Babenko. 2018. Non-metric Similarity Graphs for Maximum Inner Product Search. In *Advances in Neural Information Processing Systems.*

[42] Franco Maria Nardini, Cosimo Rulli, and Rossano Venturini. 2024. Efficient Multi-vector Dense Retrieval with Bit Vectors. In *Advances in Information Retrieval.* 3–17.

[43] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. (November 2016).

[44] Rodrigo Nogueira, Jimmy Lin, and AI Epistemic. 2019. From doc2query to docTTTTTquery. *Online preprint* 6 (2019), 2.

[45] Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document Expansion by Query Prediction. arXiv:1904.08375 [cs.IR]

[46] Giuseppe Ottaviano and Rossano Venturini. 2014. Partitioned Elias-Fano Indexes. In *Proceedings of the 37th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* 273–282.

[47] Giulio Ermanno Pibiri and Rossano Venturini. 2021. Techniques for Inverted Index Compression. *ACM Computing Surveys* 53, 6 (2021), 125:1–125:36.

[48] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

[49] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at TREC-3.. In *TREC (NIST Special Publication, Vol. 500-225)*, Donna K. Harman (Ed.). National Institute of Standards and Technology (NIST), 109–126.

[50] Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing and Management* 24, 5 (1988), 513–523.

[51] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022. ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 3715–3734.

[52] 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 *35th Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

[53] Nicola Tonellotto, Craig Macdonald, and Iadh Ounis. 2018. Efficient Query Processing for Scalable Web Search. *Foundations and Trends in Information Retrieval* 12, 4–5 (December 2018), 319–500.

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

[55] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA). 6000–6010.

[56] David P. Woodruff. 2014. Sketching as a Tool for Numerical Linear Algebra. *Foundations and Trends in Theoretical Computer Science* 10, 1–2 (Oct 2014), 1–157.

[57] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation.

[58] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. In *International Conference on Learning Representations*.

[59] Hamed Zamani, Mostafa Dehghani, W. Bruce Croft, Erik Learned-Miller, and Jaap Kamps. 2018. From Neural Re-Ranking to Neural Ranking: Learning a Sparse Representation for Inverted Indexing. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management* (Torino, Italy). 497–506.

[60] Tiancheng Zhao, Xiaopeng Lu, and Kyusong Lee. 2021. SPARTA: Efficient Open-Domain Question Answering via Sparse Transformer Matching Retrieval. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 565–575.