Efficient Inverted Index-based Approximate Retrieval over High-dimensional Learned Sparse Representations

Sebastian Bruch[1], Franco Maria Nardini[2], Cosimo Rulli[2], Rossano Venturini[3,2]

[1] Northeastern University, Boston, USA
s.bruch@northeastern.edu

[2] ISTI-CNR, Pisa, Italy
{name.surname}@isti.cnr.it

[3] University of Pisa, Italy
rossano.venturini@unipi.it

Abstract

Learned sparse representations form a very interesting class of contextual embeddings for text retrieval. In the last few years, they have proven to be effective models of relevance. Moreover, they are interpretable by design. Despite their compatibility with inverted indexes, 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 summarize a recent—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.

1 Introduction

Neural Information Retrieval (NIR) has gained increasing popularity since the introduction of pre-trained Large Language Models (LLMs) [41]. 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].

There are three main families of embeddings in Information Retrieval. The first is dense embeddings, where queries and documents are encoded into single, high-dimensional vectors [29, 33, 43, 76, 79]. These methods use the BERT [17] `[CLS]` token as a latent representation of the input text and fine-tune it to optimize a ranking loss. Relevance is estimated using the dot product between the query and document embeddings. Consequently, document collections are encoded *offline*, and the retrieval task is performed by extracting the top-k results through Maximum Inner Product Search (MIPS).

The second family, Multi-Vector Retrieval (MVR), encodes each term as a vector, producing a collection of vectors as the embedding for a given piece of text [34, 39, 57, 67, 68]. The similarity between queries and documents is computed using the *late interaction* mechanism, which involves a max-sum reduction across the rows and columns of the resulting matrix multiplication. While MVR offers a more fine-grained representation of input text and achieves higher retrieval effectiveness than dense embeddings, it is significantly more resource-intensive in terms of retrieval time and memory usage.

The third family and the main focus of this paper is represented by *Learned Sparse Retrieval* (LSR) [23, 25, 26, 35, 46, 58], where text is encoded into a *sparse* embeddings, where each dimension corresponds with a term (or token) 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. Similar to Dense Retrieval, the similarity between a query and a document is estimated using the dot product, thus raising the problem of MISP in the sparse domain.

LSR is attractive for three reasons. First, LSR models are competitive with *dense retrieval* models that encode text into dense vectors. Importantly, evidence suggests that some LSR models generalize better to out-of-domain datasets [4, 35]. Out-of-domain evaluation refers to the scenario where the encoder is trained on one dataset (commonly MSMARCO [59]) and then evaluated on a different collection to assess the model's resilience to distribution shifts in the data. A widely used framework for out-of-domain evaluation is the BEIR dataset collection [69]. Second, because of the one-to-one mapping between dimensions and vocabulary terms, sparse embeddings are *interpretable* by design. Third, because sparse embeddings retain many of the benefits of classical lexical models such as BM25 [65] 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 [71], 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 [23, 25] and the efficiency of sparse retrieval algorithms [6, 7, 24, 50, 52].

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, 12, 49]. 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 [35] applies L_1 regularization and uses dedicated query and document encoders to make queries shorter. 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 [24] embeddings of the Ms Marco [59] 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.

This paper summarizes recent work [8, 9] focusing on defining an 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. Our design uses in a new way two

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

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 [59]. 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 inverted index-based algorithms, are consistently one to two orders of magnitude slower than Seismic.

2 Related Work

We summarize the thread of work on learned sparse embeddings, then discuss methods that approach the problem of retrieval over such vector collections.

Learned Sparse Representations. Learned sparse representations were investigated [77] 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, 13–15, 24, 25, 40, 46, 82]. First attempts at this include DeepCT and HDCT by Dai and Callan [13–15].

DeepCT used the Transformer [73] encoder of BERT [18] 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. 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 [61] or docT5query [60], 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 [40, 45] expands its input with DocT5Query [60] before learning and producing a sparse representation.

Formal *et al.* build on SparTerm [1] and propose Splade [26]. 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 [26]. 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 [59] passage ranking task as well as the Beir [70] zero-shot evaluation benchmark [25]. The Splade model has undergone many other rounds of improvements which have been documented in the latest work by the same authors [24]. 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.

Retrieval Algorithms. The Information Retrieval literature offers a wide array of algorithms tailored to retrieval on text collections [71]. They are often *exact* and scale easily to massive datasets. MaxScore [72] and WAND [2], and subsequent improvements [19, 20, 53, 54], are examples that, essentially, solve the MIPS problem over "bag-of-words" representations of text, such as BM25 [65] or TF-IDF [66]. 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, 49]. Mackenzie *et al.* [50] 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 [51], 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 [56] 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 [56] 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 $1 \leq i \leq d$ 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:

$$\mathcal{S} = \underset{x \in \mathcal{X}}{\overset{(k)}{\operatorname{argmax}}}\ \langle q, x \rangle. \tag{14}$$

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

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

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

4 Concentration of Importance

Recently, Daliri *et al.* [16] 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 [16], we examined two state-of-the-art LSR techniques: Splade [23] and Efficient Splade [35]. 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 [59] 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 13a.

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 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 13b.

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%.

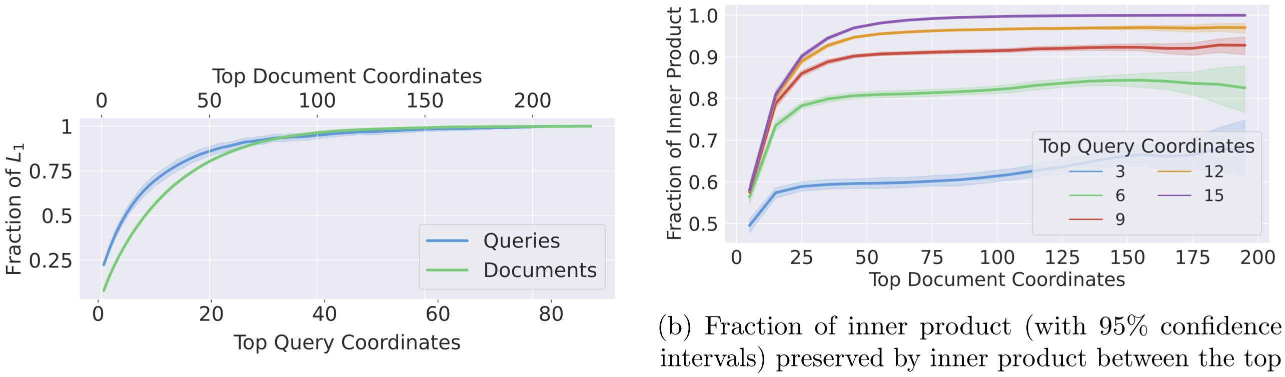
Our results confirm that LSR tends to concentrate importance on a few coordinates. Furthermore, a partial inner product between the largest entries (by absolute value) approximates the full inner product with arbitrary accuracy. As we will see shortly, this property, which is in agreement with [16], can help speed up query processing and reduce space consumption rather substantially.

5 Seismic

We introduce Seismic [8, 9], 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.

First, Seismic 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.

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



(a) Fraction of L_1 mass preserved by keeping only the top non-zero entries with the largest absolute value.

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

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 [11] 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 $j^* = \text{argmax}_j \langle x, \mu^{(j)} \rangle$, 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 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 [20].

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:

$$\phi(B)_i = \max_{x \in B} x_i. \tag{15}$$

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 (15), 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.

Seismic adopts a coordinate-at-a-time traversal (Line 3) of the inverted index. For each coordinate $i \in q_{\mathsf{cut}}$, 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

Algorithm 1: Indexing with Seismic.

```
Input: Collection $\mathcal{X}$ of sparse vectors in $\mathbb{R}^d$; $\lambda$: Maximum length of each inverted list; $\beta$: Maximum number of blocks per inverted list; $\alpha$: Fraction of the overall importance preserved by each summary.
Result: Seismic index.
for $i \in \{1, \ldots, d\}$ 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 $x \in \mathcal{S}$
    $\mathcal{I}_i \leftarrow \{\mathcal{S}_{i,1}, \mathcal{S}_{i,2}, \ldots, \mathcal{S}_{i,\lambda}\}$
    Cluster $\mathcal{I}_i$ into $\beta$ 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 (15)}
    end for
end for
return $\mathcal{I}_i$, $\{S_{i,j}\}$ $\forall i, j$
```

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 $\in (0, 1)$.

6 Experiments

We now evaluate Seismic in terms of its accuracy, latency, space usage, and indexing time against existing solutions.

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.

6.1 Setup

Datasets. We experiment on two publicly-available datasets: Ms Marco v1 Passage [59] and Natural Questions (NQ) from Beir [70]. 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 [23]. Each non-zero entry is the importance weight of a term in the BERT [18] WordPiece [75] 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 [35]. 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

[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`.

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.
$q_{\mathsf{cut}} \leftarrow$ the top cut entries of $q$ with the largest value
Heap$\leftarrow \emptyset$
for $i \in q_{\mathsf{cut}}$ 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}
        end if
        for $d \in B_j$ do
            $p = \langle q, \mathsf{ForwardIndex}[d] \rangle$
            if Heap.len() $< k$ or $p >$ Heap.min() then
                Heap.insert($p, d$)
            end if
            if Heap.len() $= k + 1$ then
                Heap.pop_min()
            end if
        end for
    end for
end for
return Heap
```

38.8 on the Ms Marco dev set. We refer to this model as E-Splade.

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] After generating 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[6] at NeurIPS. These include:

- GrassRMA: A graph-based method for dense vectors adapted to sparse vectors that appears in the BigANN challenge as "sHnsw."[7]

- PyAnn: Another graph-based ANN solution.[8]

The other three baselines are inverted index-based solutions:

[5] https://github.com/naver/splade

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

[7] C++ code is publicly available at https://github.com/Leslie-Chung/GrassRMA.

[8] C++ code is publicly available at https://github.com/veaaaab/pyanns.

Splade on Ms Marco								
Accuracy (%)	90	91	92	93	94	95	96	97
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 –
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.

- Ioqp [47]: Impact-sorted query processor written in Rust. We choose Ioqp because it is known to outperform JASS [42], 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 [55]: An inverted index-based C++ library based on `ds2i` [62] 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.* [37]. 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, [37] 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[9] and apply recursive graph bisection [48]. 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 $\text{Sinnamon}_{\text{Weak}}$ 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.

[9] `https://github.com/castorini/anserini`

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.[10] 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.

6.2 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. its average latency is 7,947 microsecond. We note that the 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 [30] 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

[10] Our code is publicly available at `https://github.com/TusKANNy/seismic`.

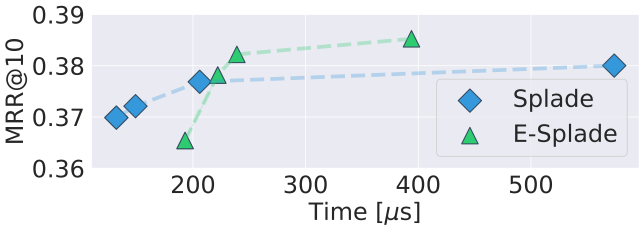


Figure 14: MRR@10 on Ms Marco.

in Figure 14. 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.

6.3 Comparison against Learned Dense Representations

The large speedup provided by Seismic allows sparse representation to be competitive with Dense Retrieval methods, where documents and queries are encoded into *dense* high-dimensional dense vectors. Here, researchers rely on the vast Approximate Nearest Neighbors literature for the dense domain, yielding high-recall results on a million-scale dataset in a few milliseconds. We compare Seismic with a state-of-the-art dense encoder, i.e., DRAGON [43], paired with the well-known HNSW [51] graph-based index. Dragon is a highly effective dense retrieval method that reaches 39.0 of MRR@10 thanks to the multi-teacher distillation from which it benefits during training. Encoding documents and queries with DRAGON yields dense vectors with 768 dimensions each. We rely on the well-known `faiss` [21] library to build a HNSW graph with $M = 200$ and $ef_c = 400$.

We build a Seismic index on the Splade-V3 [36] embeddings. Splade-V3 is the current state-of-the-art method for learned sparse representation, which is an improved version of Splade, relying on self-distillation with hard negatives, that reaches 40.3 of MRR@10 on Ms Marco. We use $\lambda = 5000$, $\beta = 1000$, and $\alpha = 0.5$. We report the result of our comparison in Figure 15. Seismic proves to be a highly efficient solution compared to dense retrieval. The combination of Splade-V3 and Seismic largely dominates over DRAGON + HNSW on the entire Pareto-frontier, being both more efficient and more effective. Moreover, observe that the dense forward index takes about 12.5 GBytes to be stored (assuming to use 16 bit floating point values), while the forward index of Splade-V3 takes about as half (5.5Gbytes), considering to use 16 bits integers to store the ids of the components and 16 bit floating point for the values.

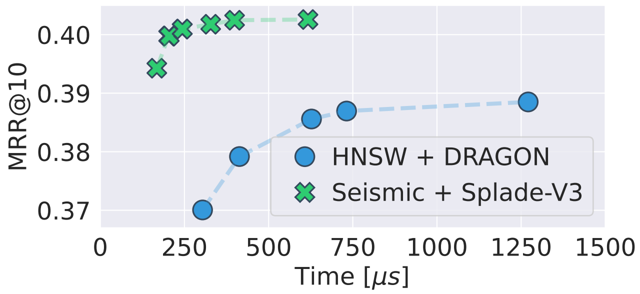


Figure 15: DRAGON indexed with HSNW against Splade-V3 indexed with Seismic on Ms Marco.

Splade on Ms Marco		
Model	Index size (MiB)	Index build time (min.)
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.

6.4 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 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 by speeding up retrieval over Splade so dramatically that switching to E-Splade becomes unnecessary and, in fact, detrimental to both efficiency and effectiveness.

8 Future Work

Several future work can be foreseen to extend this result, which can be summarized in different lines of research:

- Current approaches in approximate nearest neighbors retrieval over learned representations often employ: 1) graph-based, and 2) inverted-index based solutions for indexing data. A first line of research we intend to investigate regards mixing the two "worlds" to fully exploit specific properties of each of them. Specifically, we intend to improve inverted-index based solutions with the "locality" intrinsically modeled by graph-based methods. This can be achieved via the exploitation of the trade-off enabled by the reduction of the average query latency achieved via the storage of the neighbor of the documents in the collection. We expect this to have an impact as pre-computed neighbors can allow inverted indexed to work at lower recall cuts, thus significantly reducing their query processing time.

- We also intend to investigate novel compression techniques for inverted lists [63] to further reduce the size of inverted and forward indexes.

- A third line of research asks for a comprehensive evaluation of the performance of Seismic and its competitors on big datasets. In fact, the evaluation of these approaches in information retrieval has always been conducted on datasets of limited sizes— typically Ms Marco version 1.0— which consists of less than ten million documents. A detailed analysis of what are the challenges arising from the application of these methods to bigger datasets, which are in the order of hundreds of millions of documents, is missing. We believe that this would spark novel, interesting research questions contributing to the definition of new—fast and effective—approximate nearest neighbors search methods.

- Recent work have been done to develop indexing data structures that jointly exploit both dense and sparse embeddings [64, 81]. Linear combinations of the relevance estimated by the two representations separately have been shown to improve the retrieval accuracy [10, 27, 38, 44, 74]. We plan to explore the application of Seismic to hybrid representations to exploit the compact representation provided by dense embeddings together with the per-term space partitioning induced by sparse ones.

- In Section 6.3, we highlighted how sparse embeddings are generally more memory-compact than dense ones. Yet, dense embeddings can rely on a vast amount of literature on vector compression, including the well-known Product Quantization approach [22, 28, 31, 32]. PQ has shown to be highly effective in reducing the memory burden of dense representations by one order of magnitude without sacrificing accuracy [78, 80]. At the moment, sparse methods lack a comparable method for vector compression. A compelling research direction is to develop novel compression methods tailored for learned sparse representations.

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. Sparterm: Learning term-based sparse representation for fast text retrieval, 2020.

[2] Andrei Z. Broder, David Carmel, Michael Herscovici, Aya Soffer, and Jason Zien. Efficient query evaluation using a two-level retrieval process. In *Proceedings of the 12th International Conference on Information and Knowledge Management*, page 426–434, 2003.

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

[4] Sebastian Bruch, Siyu Gai, and Amir Ingber. An analysis of fusion functions for hybrid retrieval. *ACM Transactions on Information Systems*, 42(1), August 2023.

[5] Sebastian Bruch, Claudio Lucchese, and Franco Maria Nardini. Efficient and effective tree-based and neural learning to rank. *Foundations and Trends® in Information Retrieval*, 17(1):1–123, 2023.

[6] Sebastian Bruch, Franco Maria Nardini, Amir Ingber, and Edo Liberty. An approximate algorithm for maximum inner product search over streaming sparse vectors. *ACM Transactions on Information Systems*, 42(2), November 2023.

[7] Sebastian Bruch, Franco Maria Nardini, Amir Ingber, and Edo Liberty. Bridging dense and sparse maximum inner product search, 2023.

[8] Sebastian Bruch, Franco Maria Nardini, Cosimo Rulli, and Rossano Venturini. 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, page 152–162, New York, NY, USA, 2024. Association for Computing Machinery.

[9] Sebastian Bruch, Franco Maria Nardini, Cosimo Rulli, and Rossano Venturini. Pairing clustered inverted indexes with k-nn graphs for fast approximate retrieval over learned sparse representations. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, CIKM '24, page 3642–3646, New York, NY, USA, 2024. Association for Computing Machinery.

[10] Tao Chen, Mingyang Zhang, Jing Lu, Michael Bendersky, and Marc Najork. Out-of-domain semantics to the rescue! zero-shot hybrid retrieval models. In *European Conference on Information Retrieval*, pages 95–110. Springer, 2022.

[11] Flavio Chierichetti, Alessandro Panconesi, Prabhakar Raghavan, Mauro Sozio, Alessandro Tiberi, and Eli Upfal. Finding near neighbors through cluster pruning. In *Proceedings of the Twenty-Sixth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 103–112, 2007.

[12] Matt Crane, J. Shane Culpepper, Jimmy Lin, Joel Mackenzie, and Andrew Trotman. 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*, pages 201–210, 2017.

[13] Zhuyun Dai and Jamie Callan. Context-aware sentence/passage term importance estimation for first stage retrieval, 2019.

[14] Zhuyun Dai and Jamie Callan. Context-aware document term weighting for ad-hoc search. In *Proceedings of The Web Conference*, pages 1897–1907, 2020.

[15] Zhuyun Dai and Jamie Callan. 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*, pages 1533–1536, 2020.

[16] Majid Daliri, Juliana Freire, Christopher Musco, Aécio Santos, and Haoxiang Zhang. Sampling methods for inner product sketching, 2023.

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *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)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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)*, pages 4171–4186, June 2019.

[19] Constantinos Dimopoulos, Sergey Nepomnyachiy, and Torsten Suel. Optimizing top-k document retrieval strategies for block-max indexes. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, pages 113–122, 2013.

[20] Shuai Ding and Torsten Suel. 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*, pages 993–1002, 2011.

[21] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. *arXiv preprint arXiv:2401.08281*, 2024.

[22] Matthijs Douze, Hervé Jégou, and Florent Perronnin. Polysemous codes. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pages 785–801. Springer, 2016.

[23] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 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*, pages 2353–2359, 2022.

[24] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. Towards effective and efficient sparse neural information retrieval. *ACM Transactions on Information Systems*, December 2023.

[25] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. Splade v2: Sparse lexical and expansion model for information retrieval, 2021.

[26] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 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*, pages 2288–2292, 2021.

[27] Luyu Gao, Zhuyun Dai, Tongfei Chen, Zhen Fan, Benjamin Van Durme, and Jamie Callan. Complement lexical retrieval model with semantic residual embeddings. In *Advances in Information Retrieval: 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28–April 1, 2021, Proceedings, Part I 43*, pages 146–160. Springer, 2021.

[28] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. Optimized product quantization. *IEEE transactions on pattern analysis and machine intelligence*, 36(4):744–755, 2013.

[29] Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin, and Allan Hanbury. Efficiently teaching an effective dense retriever with balanced topic aware sampling. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 113–122, 2021.

[30] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002.

[31] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2010.

[32] Yannis Kalantidis and Yannis Avrithis. Locally optimized product quantization for approximate nearest neighbor search. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2321–2328, 2014.

[33] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen Tau Yih. Dense passage retrieval for open-domain question answering. In *2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020*, pages 6769–6781. Association for Computational Linguistics (ACL), 2020.

[34] Omar Khattab and Matei Zaharia. 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*, pages 39–48, 2020.

[35] Carlos Lassance and Stéphane Clinchant. An efficiency study for splade models. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2220–2226, 2022.

[36] Carlos Lassance, Hervé Déjean, Thibault Formal, and Stéphane Clinchant. Splade-v3: New baselines for splade. *arXiv preprint arXiv:2403.06789*, 2024.

[37] Carlos Lassance, Simon Lupart, Hervé Déjean, Stéphane Clinchant, and Nicola Tonellotto. A static pruning study on sparse neural retrievers. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1771–1775, 2023.

[38] Dohyeon Lee, Seung-won Hwang, Kyungjae Lee, Seungtaek Choi, and Sunghyun Park. On complementarity objectives for hybrid retrieval. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13357–13368, 2023.

[39] Jinhyuk Lee, Zhuyun Dai, Sai Meher Karthik Duddu, Tao Lei, Iftekhar Naim, Ming-Wei Chang, and Vincent Zhao. Rethinking the role of token retrieval in multi-vector retrieval. *Advances in Neural Information Processing Systems*, 36, 2024.

[40] Jimmy Lin and Xueguang Ma. A few brief notes on deepimpact, coil, and a conceptual framework for information retrieval techniques, 2021.

[41] Jimmy Lin, Rodrigo Frassetto Nogueira, and Andrew Yates. *Pretrained Transformers for Text Ranking: BERT and Beyond.* Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2021.

[42] Jimmy Lin and Andrew Trotman. Anytime ranking for impact-ordered indexes. In *Proceedings of the 2015 International Conference on The Theory of Information Retrieval*, pages 301–304, 2015.

[43] Sheng-Chieh Lin, Akari Asai, Minghan Li, Barlas Oguz, Jimmy Lin, Yashar Mehdad, Wen-tau Yih, and Xilun Chen. How to train your dragon: Diverse augmentation towards generalizable dense retrieval. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

[44] Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. Sparse, dense, and attentional representations for text retrieval. *Transactions of the Association for Computational Linguistics*, 9:329–345, 2021.

[45] Xueguang Ma, Ronak Pradeep, Rodrigo Nogueira, and Jimmy Lin. 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*, pages 3187–3197, 2022.

[46] Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, Nicola Tonellotto, Nazli Goharian, and Ophir Frieder. Expansion via prediction of importance with contextualization. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1573–1576, 2020.

[47] J. Mackenzie, M. Petri, and L. Gallagher. Ioqp: A simple impact-ordered query processor written in rust. In *Proc. DESIRES*, pages 22–34, 2022.

[48] J. Mackenzie, M. Petri, and A. Moffat. Faster index reordering with bipartite graph partitioning. In *Proc. SIGIR*, pages 1910–1914, 2021.

[49] Joel Mackenzie, Andrew Trotman, and Jimmy Lin. Wacky weights in learned sparse representations and the revenge of score-at-a-time query evaluation, 2021.

[50] Joel Mackenzie, Andrew Trotman, and Jimmy Lin. Efficient document-at-a-time and score-at-a-time query evaluation for learned sparse representations. *ACM Transactions on Information Systems*, 41(4), 2023.

[51] Yu A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4):824–836, 4 2020.

[52] Antonio Mallia, Joel Mackenzie, Torsten Suel, and Nicola Tonellotto. Faster learned sparse retrieval with guided traversal. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1901–1905, 2022.

[53] Antonio Mallia, Giuseppe Ottaviano, Elia Porciani, Nicola Tonellotto, and Rossano Venturini. Faster blockmax wand with variable-sized blocks. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 625–634, 2017.

[54] Antonio Mallia and Elia Porciani. Faster blockmax wand with longer skipping. In *Advances in Information Retrieval*, pages 771–778, 2019.

[55] Antonio Mallia, Michal Siedlaczek, Joel Mackenzie, and Torsten Suel. 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*, pages 50–56, 2019.

[56] Stanislav Morozov and Artem Babenko. Non-metric similarity graphs for maximum inner product search. In *Advances in Neural Information Processing Systems*, 2018.

[57] Franco Maria Nardini, Cosimo Rulli, and Rossano Venturini. Efficient multi-vector dense retrieval with bit vectors. In *European Conference on Information Retrieval*, pages 3–17. Springer, 2024.

[58] Thong Nguyen, Sean MacAvaney, and Andrew Yates. A unified framework for learned sparse retrieval. In *European Conference on Information Retrieval*, pages 101–116. Springer, 2023.

[59] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. Ms marco: A human generated machine reading comprehension dataset. November 2016.

[60] Rodrigo Nogueira, Jimmy Lin, and AI Epistemic. From doc2query to docttttquery. *Online preprint*, 6:2, 2019.

[61] Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. Document expansion by query prediction, 2019.

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

[63] Giulio Ermanno Pibiri and Rossano Venturini. Techniques for inverted index compression. *ACM Computing Surveys*, 53(6):125:1–125:36, 2021.

[64] Yifan Qiao, Parker Carlson, Shanxiu He, Yingrui Yang, and Tao Yang. 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*, pages 19742–19757, 2024.

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

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

[67] Keshav Santhanam, Omar Khattab, Christopher Potts, and Matei Zaharia. Plaid: an efficient engine for late interaction retrieval. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 1747–1756, 2022.

[68] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 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*, pages 3715–3734, 2022.

[69] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 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)*.

[70] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 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)*, 2021.

[71] Nicola Tonellotto, Craig Macdonald, and Iadh Ounis. Efficient query processing for scalable web search. *Foundations and Trends in Information Retrieval*, 12(4–5):319–500, December 2018.

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

[73] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010, 2017.

[74] Shuai Wang, Shengyao Zhuang, and Guido Zuccon. Bert-based dense retrievers require interpolation with bm25 for effective passage retrieval. In *Proceedings of the 2021 ACM SIGIR international conference on theory of information retrieval*, pages 317–324, 2021.

[75] 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. Google's neural machine translation system: Bridging the gap between human and machine translation, 2016.

[76] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations*.

[77] Hamed Zamani, Mostafa Dehghani, W. Bruce Croft, Erik Learned-Miller, and Jaap Kamps. 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*, pages 497–506, 2018.

[78] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. Jointly optimizing query encoder and product quantization to improve retrieval performance. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 2487–2496, 2021.

[79] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. Optimizing dense retrieval model training with hard negatives. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1503–1512, 2021.

[80] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. Learning discrete representations via constrained clustering for effective and efficient dense retrieval. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pages 1328–1336, 2022.

[81] Haoyu Zhang, Jun Liu, Zhenhua Zhu, Shulin Zeng, Maojia Sheng, Tao Yang, Guohao Dai, and Yu Wang. Efficient and effective retrieval of dense-sparse hybrid vectors using graph-based approximate nearest neighbor search. *arXiv preprint arXiv:2410.20381*, 2024.

[82] Tiancheng Zhao, Xiaopeng Lu, and Kyusong Lee. 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*, pages 565–575, June 2021.