Seismic: Efficient and Effective Retrieval over Learned Sparse Representation

Sebastian Bruch[1], Franco Maria Nardini[2], Cosimo Rulli[2] and Rossano Venturini[4]

[1]Pinecone, USA
[2]ISTI-CNR, Italy
[3]University of Pisa, Italy

Abstract
Learned sparse representations form an attractive class of contextual embeddings for text retrieval thanks to their effectiveness and interpretability. Retrieval over sparse embeddings remains challenging due to the distributional differences between learned embeddings and term frequency-based lexical models of relevance, such as BM25. Recognizing this challenge, recent research trades off exactness for efficiency, moving to *approximate* retrieval systems. In this work[1], 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. Experiments on the Splade and E-Splade embeddings on the Ms Marco and NQ datasets show that our approach is up to 21× time faster than the winning (graph-based) submissions to the BigANN Challenge.

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

1. Introduction

Learned Sparse Retrieval (LSR) [2, 3, 4, 5, 6] repurposes Large Language Models 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. LSR models are of pivotal interest as they i) compete with *dense retrieval* models that encode text into dense vectors in terms of effectiveness [7, 8, 9, 10, 11, 12, 13], ii) tend to generalize better to out-of-domain datasets [14, 6], iii) are *interpretable* by design [6, 1]. The straightforward usage of standard inverted index for sparse embeddings is hindered by the statistical properties of the weights learned by LSR, which do not conform to the assumptions under which popular inverted index-based retrieval algorithms operate [15, 16, 17]. Hence, many recent solutions give up on exact search to boost the efficiency of the search algorithm [15, 18], taking a leaf out of the Approximate Nearest Neighbor (ANN) literature [19]. As a clear example, the 2023 BigANN Challenge[1] at NeurIPS dedicated a track to learned sparse embeddings. 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. Our solution (Section 2) uses in a new way two familiar data structures:

[1]This contribution is an extended abstract of Bruch *et al.* [1]

IIR 2024: The 14th Italian Information Retrieval Workshop. September 5-6, Udine, Italy.
sbruch@acm.org (S. Bruch); francomaria.nardini@isti.cnr.it (F. M. Nardini); cosimo.rulli@isti.cnr.it (C. Rulli); rossano.venturini@unipi.it (R. Venturini)

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

the inverted and the forward index. 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. Our experimental evaluation (Section 3) shows that Seismic outperforms the state-of-the-art competitors up to 21× on the Splade and E-Splade embeddings on the Ms Marco and NQ datasets.

2. Methodology

Algorithm 1: Indexing.

Input: $\mathcal{X}$: sparse vectors in $\mathbb{R}^d$;
λ: Maximum inverted list length;
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 $\phi(B_{i,j})$
return $\mathcal{I}_i$, $\{S_{i,j}\}\ \forall i, j$

Algorithm 2: Query processing

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

$q_{\text{cut}} \leftarrow$ the top cut entries of q
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 $r < \frac{\text{Heap.min()}}{\text{heap\_factor}}$ **then**
      continue {Skip the block}
    for $d \in B_j$ **do**
      $p = \langle q, \text{ForwardIndex}[d] \rangle$
      UpdateHeap(Heap, p, d)
return Heap

The design of Seismic relied both on an inverted index and a forward index. Seismic uses an organization of the inverted index that blends together *static* and *dynamic* pruning. The documents pinpointed by the inverted index are then evaluated using the forward index. The data structure and the indexing / query processing algorithm are described in detail below.

Static Pruning. Seismic heavily relies on the concentration of importance property discussed by Bruch *et al.* [1]. 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. Concretely, *static pruning* means that 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 $\mathcal{I}_i$.

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* so as to facilitate a *dynamic pruning* strategy.

A clustering algorithm is used to partition the documents whose ids are present in an inverted list 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 [20] of K-Means; see the original paper for more details [1].

Per-block summary Vectors. Seismic leverages the concept of a *summary* vector to determine whether a block should be evaluated. A summary is d-dimensional vector built with the idea 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. More precisely, our summary function $\phi : 2^X \rightarrow \mathbb{R}^d$ takes a block B from the universe of all blocks 2^X, and produces a vector whose i-th coordinate is simply $\phi(B)_i = \max_{x \in B} x_i$. 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 number of non-zero entries in summary vectors grows quickly with the block size, increasing the memory footprint and the search time of Seismic. To this end, we prune $\phi(B)$ by keeping only its α-mass subvector. See the original work for the definition of α-mass subvector [1]. That, α, is our third and last indexing hyper-parameter. We further reduce the size of summaries by applying scalar quantization after min-max scaling, employing only a single byte for each value.

Indexing. 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).

Query Processing. Algorithm 2 shows the query processing logic in Seismic. We select a subset of the query coordinates q_{cut} (Line 1), sorted by magnitude, 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, using the Forward Index. A document whose inner product is greater than the minimum score in the Min-Heap is inserted into the heap (UpdateHeap).

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

3. Experiments

Experimental Setup. We experiment on two publicly-available datasets: Ms Marco v1 Passage [21] and Natural Questions (NQ) from Beir [22]. We evaluate Seismic on embeddings generated using Splade [5] and E-Splade. [6].

We compare Seismic with five state-of-the-art retrieval solutions. In this manuscript, we only report the comparison against the best competitors, namely the winning solutions of the "Sparse Track" at the 2023 BigANN Challenge at NeurIPS, GrassRMA and PyAnn. See the original work for the complete comparison [1]. We compare the methods using mean query latency (μsec.) and accuracy, i.e., the percentage of true nearest neighbors recalled in the returned set. We implemented Seismic in Rust.[2] We conduct experiments on a server equipped with one Intel i9-9900K CPU, clock rate 3.60 GHz and 64 GiB of RAM, with single-threaded execution.

Results Table 1 details retrieval performance in terms of average per-query latency at various accuracy cut. 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.

4. Conclusions and Future Work

This paper presents Seismic, a novel approach for efficient and effective retrieval over sparse learned representations. Our solution outperforms the state-of-art graph-based solutions for efficient sparse retrieval up to a factor of 21× on the Splade and E-Splade embeddings on the Ms Marco dataset. As future work, we intend to explore the application of compression techniques for inverted lists [23] to further reduce the size of inverted and forward indexes.

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

References

[1] S. Bruch, F. M. Nardini, C. Rulli, R. 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, Association for Computing Machinery, New York, NY, USA, 2024, p. 152–162. doi:10.1145/3626772.3657769.

[2] S. MacAvaney, F. M. Nardini, R. Perego, N. Tonellotto, N. Goharian, O. Frieder, Expansion via prediction of importance with contextualization, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 1573–1576.

[3] T. Formal, B. Piwowarski, S. 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, 2021, pp. 2288–2292.

[4] T. Formal, C. Lassance, B. Piwowarski, S. Clinchant, Splade v2: Sparse lexical and expansion model for information retrieval, 2021. arXiv:2109.10086.

[5] T. Formal, C. Lassance, B. Piwowarski, S. 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, 2022, pp. 2353–2359.

[6] C. Lassance, S. Clinchant, An efficiency study for splade models, in: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2022, pp. 2220–2226.

[7] J. Lin, R. F. Nogueira, A. Yates, Pretrained Transformers for Text Ranking: BERT and Beyond, Synthesis Lectures on Human Language Technologies, Morgan & Claypool Publishers, 2021.

[8] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, W.-t. Yih, Dense passage retrieval for open-domain question answering, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, 2020, pp. 6769–6781.

[9] L. Xiong, C. Xiong, Y. Li, K.-F. Tang, J. Liu, P. Bennett, J. Ahmed, A. Overwijk, Approximate nearest neighbor negative contrastive learning for dense text retrieval, in: International Conference on Learning Representations, 2021.

[10] N. Reimers, I. Gurevych, 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, 2019.

[11] K. Santhanam, O. Khattab, J. Saad-Falcon, C. Potts, M. 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, 2022, pp. 3715–3734.

[12] O. Khattab, M. 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, 2020, pp. 39–48.

[13] F. M. Nardini, C. Rulli, R. Venturini, Efficient multi-vector dense retrieval with bit vectors, in: Advances in Information Retrieval, 2024, pp. 3–17.

[14] S. Bruch, S. Gai, A. Ingber, An analysis of fusion functions for hybrid retrieval, ACM Transactions on Information Systems 42 (2023).

[15] S. Bruch, F. M. Nardini, A. Ingber, E. Liberty, An approximate algorithm for maximum inner product search over streaming sparse vectors, ACM Transactions on Information Systems 42 (2023).

[16] J. Mackenzie, A. Trotman, J. Lin, Wacky weights in learned sparse representations and the revenge of score-at-a-time query evaluation, 2021. arXiv:2110.11540.

[17] M. Crane, J. S. Culpepper, J. Lin, J. Mackenzie, A. 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, 2017, pp. 201–210.

[18] S. Bruch, F. M. Nardini, A. Ingber, E. Liberty, Bridging dense and sparse maximum inner product search, 2023. arXiv:2309.09013.

[19] S. Bruch, Foundations of Vector Retrieval, Springer Nature Switzerland, 2024.

[20] F. Chierichetti, A. Panconesi, P. Raghavan, M. Sozio, A. Tiberi, E. Upfal, Finding near neighbors through cluster pruning, in: Proceedings of the Twenty-Sixth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, 2007, pp. 103–112.

[21] T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, L. Deng, Ms marco: A human generated machine reading comprehension dataset (2016).

[22] N. Thakur, N. Reimers, A. Rücklé, A. Srivastava, I. 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.

[23] G. E. Pibiri, R. Venturini, Techniques for inverted index compression, ACM Computing Surveys 53 (2021) 125:1–125:36.