Bridging Dense and Sparse Maximum Inner Product Search

SEBASTIAN BRUCH, Pinecone, New York, NY, USA
FRANCO MARIA NARDINI, ISTI-CNR, Pisa, Italy
AMIR INGBER, Pinecone, Tel Aviv, Israel
EDO LIBERTY, Pinecone, New York, NY, USA

Maximum inner product search (MIPS) over dense and sparse vectors have progressed independently in a bifurcated literature for decades; the latter is better known as top-k retrieval in Information Retrieval. This duality exists because sparse and dense vectors serve different end goals. That is despite the fact that they are manifestations of the same mathematical problem. In this work, we ask if algorithms for dense vectors could be applied effectively to sparse vectors, particularly those that violate the assumptions underlying top-k retrieval methods. We study clustering-based approximate MIPS where vectors are partitioned into clusters and only a fraction of clusters are searched during retrieval. We conduct a comprehensive analysis of dimensionality reduction for sparse vectors, and examine standard and spherical k-means for partitioning. Our experiments demonstrate that clustering-based retrieval serves as an efficient solution for sparse MIPS. As byproducts, we identify two research opportunities and explore their potential. First, we cast the clustering-based paradigm as dynamic pruning and turn that insight into a novel organization of the inverted index for approximate MIPS over general sparse vectors. Second, we offer a unified regime for MIPS over vectors that have dense and sparse subspaces, that is robust to query distributions.

CCS Concepts: • **Information systems → Retrieval models and ranking**;

Additional Key Words and Phrases: Maximum inner product search, top-k retrieval, sparse vectors, dense vectors, hybrid vectors, sketching, clustering-based approximate nearest neighbor search

ACM Reference format:
Sebastian Bruch, Franco Maria Nardini, Amir Ingber, and Edo Liberty. 2024. Bridging Dense and Sparse Maximum Inner Product Search. *ACM Trans. Inf. Syst.* 42, 6, Article 151 (August 2024), 38 pages.
https://doi.org/10.1145/3665324

1 Introduction

Retrieval is one of the most fundamental questions in **Information Retrieval (IR)**, as the name of the discipline itself reflects. Simply put, given a large number of objects, we wish to find, in an efficient manner, the closest subset of those objects to a query according to some notion of

This work was partially supported by the Horizon Europe RIA "EFRA - Extreme Food Risk Analytics" (Grant agreement no. 101093026) and the PNRR—M4C2—Investimento 1.3, Partenariato Esteso PE00000013—"FAIR—Future Artificial Intelligence Research"—Spoke 1 "Human-centered AI." Horizon Europe and the PNRR programs are funded by the European Commission under the NextGeneration EU program. The views and opinions expressed are solely those of the authors and do not necessarily reflect those of the European Union, nor can the European Union be held responsible for them.
Authors' Contact Information: Sebastian Bruch (Corresponding author), Pinecone, New York, NY, USA; e-mail: sbruch@acm.org; Franco Maria Nardini, ISTI-CNR, Pisa, Italy, e-mail: francomaria.nardini@isti.cnr.it; Amir Ingber, Pinecone, Tel Aviv, Israel; e-mail: ingber@pinecone.io; Edo Liberty, Pinecone, New York, NY, USA; e-mail: edo@pinecone.io.

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

© 2024 Copyright held by the owner/author(s).
ACM 1558-2868/2024/8-ART151
https://doi.org/10.1145/3665324

closeness. The data structure and algorithmic inventions [70, 84] that have emerged from the IR literature to address this deceptively simple question have had enormous impact on the field and birthed major research directions. They provide the machinery to scale ranking to massive datasets within multi-stage ranking systems [6, 7, 15, 42], for instance, or power large-scale applications, of which search is a notable and ubiquitous example.

Much of the IR research on retrieval targets textual data, where documents and queries are texts in natural languages. Unsurprisingly, then, the retrieval machinery that exists today is highly optimized for data that is governed by the laws of natural languages (such as Zipf's law) and the way users interact with retrieval and search systems (e.g., by means of short, keyword queries). The inverted index [84], e.g., is inspired by how we historically organized and found information in a book or at a library. Our measures of closeness, such as TF-IDF and BM25 [64], rely on statistics that reflect our understanding of the relevance between two pieces of text. The dynamic pruning algorithms that help us traverse inverted indexes efficiently [11, 19, 26, 43, 49, 55, 61, 70] to find the top-k most relevant documents to a query, too, rely on the statistical properties of language and relevance measures.

While the form of retrieval above is the bedrock of flurry of other research and applications in IR, the rise of deep learning in recent years brought a different form of retrieval into the IR spotlight: **Approximate Nearest Neighbor (ANN)** search [12] in dense vector spaces.

ANN search has for decades played an outsize role in research problems that are adjacent to text retrieval such as image and multimedia retrieval [60, 81]. Its machinery is optimized for objects and queries that are real vectors in some high-dimensional space, and where closeness is determined by inner product or proper metrics such as Euclidean distance. Today, efficient and effective data structures and algorithms for this problem are often critical components in, among other applications, *semantic* search, where, using deep learning, we learn a vector representation of documents and queries in a space where closeness of vectors implies semantic similarity of their corresponding texts [42].

1.1 Maximum Inner Product Search (MIPS) as the Unifying Problem

The fact that these two branches of retrieval have historically progressed independently makes a great deal of sense: they have targeted quite different applications. Today's reality driven by the burgeoning role of deep learning in IR and the effectiveness of learnt representations in many related domains, however, begins to challenge the status quo. Let us illustrate our point by considering joint lexical-semantic search [13, 18, 37, 39, 46, 47, 73, 76] as an example. In that setup, documents and queries are represented as learnt vectors *and* as bags of words. Retrieval is then performed over both representations to find the documents that are both lexically and semantically close to a query. This application is at the confluence of (inverted index-based) top-k retrieval and ANN search. The challenge presented by the historical dichotomy is that researchers and practitioners alike must study and develop two disparate systems that are characteristically different.

At the same time, we are witnessing the success of methods that *learn* term importance weights from texts [9, 22, 27, 29, 41, 53, 80, 83], rather than compute it based on term frequency and propensity. It has been shown that the weights learnt this way exhibit distributional properties that do not conform to the expectations of inverted index-based retrieval algorithms [17, 51]. This challenges some of the assumptions underlying dynamic pruning algorithms and thus the efficacy of inverted index-based retrieval in the face of arbitrarily-distributed term weights [17, 50].

The existing literature gives effective solutions of various degrees of complexity to each and every one of the shortcomings above [48, 51, 54, 76, 79]. In this work, we wish to investigate a more general question that arises if we returned to the principles and re-examined the most glaring fact: It should come as no surprise that both branches of retrieval operate on vectors and, often, attempt

to solve MIPS. It just so happens that in one branch the vectors are *dense* (i.e., all coordinates are almost surely non-zero) and in the other *sparse* (i.e., where, relative to the dimensionality of the space, very few coordinates are non-zero). We call the former "dense MIPS" and the latter "sparse MIPS" for brevity.

1.2 Sparse MIPS as a Subclass of Dense MIPS

It is clear that solutions devised for sparse MIPS are not immediately applicable to dense MIPS. That is because sparse MIPS algorithms operate under stricter distributional assumptions than dense MIPS algorithms do; in other words, the class of sparse vectors for which MIPS solutions exist is a subset of the class of dense vectors. For example, inverted index-based solutions are only efficient if the vectors are sparse[1] and non-negative, and if their sparsity pattern takes on a Zipfian shape. Dense MIPS algorithms, on the other hand, have fewer inherent limitations. A natural question that arises given the observation above is whether dense MIPS algorithms remain effective and efficient when applied to sparse vectors. That is the primary motivation behind this study.

While conceptually simple and admittedly pedestrian, applying dense MIPS solutions to sparse vectors faces many challenges. And therein lies our technical contribution: We present, as a proof of concept, the machinery that enables such a formulation.

We start by foregoing *exactness* and instead developing ideas on the principle of **Probably Approximately Correctness (PAC)**. In other words, instead of insisting on finding the exact set of top-k documents, we settle with an approximate set that may erroneously contain some farther-afield documents and mistakenly miss other close-by documents. In the IR literature, this is the familiar notion of *rank-unsafe* retrieval [70].

Having accepted some (quantifiable) error in the retrieval outcome, we are faced with the next, rather debilitating challenge of working with often extremely high dimensional sparse vectors. It is here that we appeal to results from related disciplines that study data-oblivious $\ell 2$-subspace embedding [74] and non-linear sketching[2] (itself sparse) of sparse vectors [17]. These dimensionality reduction techniques use the elegant yet simple idea of random projections to preserve Euclidean distance or inner product between vectors. To understand the ramifications of reducing dimensions (and thereby losing information) for sparse MIPS, we study the behavior of two particular random projection techniques when applied to sparse vectors: the linear **Johnson–Lindenstrauss (JL)** [1, 4, 36] transform and the non-linear Sinnamon [17] transform. We study this particular topic in-depth in Section 4.

By projecting sparse high-dimensional vectors into a (possibly dense) low-dimensional subspace, we have removed the main barrier to applying dense MIPS solutions to sparse vectors and are therefore prepared to investigate our main research question above. We are particularly interested in a method commonly known as clustering-based or **Inverted File (IVF)**-based retrieval: It begins by clustering vectors into partitions in an unsupervised manner. When it receives a query vector, it identifies a subset of the more "promising" partitions, and conducts (exact or approximate) retrieval only over the subset of documents assigned to them. The search over the sub-collection can be delegated to another MIPS algorithm, the most naïve of which is an exhaustive, exact search. To understand how (sketches of) sparse vectors behave in an IVF retrieval system, we empirically evaluate standard and spherical k-means [24] on a range of datasets. This analysis is the main topic of Section 5.

[1]In fact, query vectors are often required to be much more sparse than document vectors for a sparse MIPS solution to remain reasonably efficient.

[2]We use "sketch" to describe a compressed representation of a high-dimensional vector, and "to sketch" to describe the act of compressing a vector into a sketch.

Algorithm 1: Indexing

Input: Collection $\mathcal{X}$ of sparse vectors in $\mathbb{R}^N$; Number of clusters, P; Random projector, $\phi : \mathbb{R}^N \to \mathbb{R}^n$ where $n \ll N$; Clustering algorithm Cluster that returns partitions of input data and their representatives.
Result: Cluster assignments $\mathcal{P}_i = \left\{ j \mid x^{(j)} \in \text{Partition} i \right\}$ and cluster representatives C_i's.

$\tilde{\mathcal{X}} \leftarrow \{\phi(x) \mid x \in \mathcal{X}\}$
Partitions, Representatives $\leftarrow$ Cluster$(\tilde{\mathcal{X}}; P)$
$\mathcal{P}_i \leftarrow \left\{ j \mid \tilde{x}^{(j)} \in \text{Partitions}[i] \right\}, \quad \forall 1 \leq i \leq P$
$C_i \leftarrow \text{Representatives}[i], \quad \forall 1 \leq i \leq P$
return $\mathcal{P}$ and C

Algorithm 2: Retrieval

Input: Sparse query vector, $q \in \mathbb{R}^N$; Clusters and representatives, $\mathcal{P}$, C obtained from Algorithm 1; Random projector $\phi : \mathbb{R}^N \to \mathbb{R}^n$ where $n \ll N$; Minimum number of data points to examine, $\ell \leq |\mathcal{X}|$, where $|\mathcal{X}|$ denotes the collection size; MIPS sub-algorithm $\mathcal{R}$.
Result: Approximate set of top-k vectors that maximize inner product with q.

$\tilde{q} \leftarrow \phi(q)$
SortedClusters $\leftarrow$ **SortDescending**$(\mathcal{P}$ by $\langle \tilde{q}, C_i \rangle)$
TotalSize $\leftarrow 0$
$\mathcal{I} \leftarrow \emptyset$; ▹ `Index of partitions` $\mathcal{R}$ `should probe.`
for $\mathcal{P}_{\pi_i} \in$ SortedClusters **do**
  $\mathcal{I} \leftarrow \pi_i$
  TotalSize $\leftarrow$ TotalSize $+ |\mathcal{P}_{\pi_i}|$
  break if TotalSize $\geq \ell$
end for
return Top-k vectors from partitions $\mathcal{P}_{\mathcal{I}} \triangleq \{\mathcal{P}_i \mid i \in \mathcal{I}\}$ w.r.t $\langle q, \cdot \rangle$ using $\mathcal{R}$

Together, dimensionality reduction via random projections and clustering, enable the IVF paradigm for sparse vectors. Algorithm 1 describes the end-to-end indexing procedure, and Algorithm 2 gives details of the retrieval logic. We encourage the reader to refer to Section 3 for an overview of our adopted notation.

1.3 Research Byproducts

As we demonstrate, it is certainly feasible and—given an appropriate tolerance for error—often effective, to apply Algorithms 1 and 2 to sparse vectors. That possibility immediately leads to two important observations that we explore later in this work.

First, we remark that, in effect, clustering a document collection and performing search over only a fraction of the resulting clusters, constitutes a dynamic pruning method—albeit a rank-unsafe one. We use this insight to propose an organization of the inverted index where inverted lists comprise of blocks, with each block containing documents that fall into the same partition, and sorted by partition identifier. We show that, appropriately using skip pointers over inverted lists facilitates fast approximate top-k retrieval for *general* sparse vectors—vectors that need not conform to any distributional requirements. Experiments confirm the efficiency and effectiveness of our proposal.

Second, we offer a fresh but natural perspective to unify the two worlds of dense and sparse MIPS into a single, elegant framework at the systems level. In particular, we consider *hybrid* vectors

(i.e., vectors that may contain dense *and* sparse subspaces) in an IVF retrieval system. We demonstrate empirically that the clusters formed by our proposal are effective, and, regardless of how the $\ell 2$ mass is distributed between the dense and sparse subspaces, retrieval can be arbitrarily accurate.

1.4 Contributions

We summarize our contributions as follows:

- We analyze the effect of linear and non-linear random projection algorithms on the inner product approximation of sparse vectors;
- We extend the clustering-based IVF method of dense MIPS to (sketches of) sparse vectors, and, in that context, empirically evaluate standard and spherical k-means clustering algorithms;
- We use our findings to propose a novel organization of the inverted index that facilitates approximate MIPS over general sparse vectors, thereby freeing sparse MIPS from strict distributional requirements of traditional top-k retrieval algorithms in IR; and,
- We propose a unification of dense and sparse MIPS using IVF, and present a preliminary empirical evaluation of the proposal.

Throughout our presentation, we hope to convey the simplicity that our proposals provide in working with vectors, regardless of their density or sparsity, for both researchers and practitioners. But we are more excited by what this new perspective enables and the major research questions it inspires. To start, we believe our framework and the retrieval machinery it offers provide substantial flexibility to researchers who wish to study learnt term weights without the constraints imposed by traditional inverted index-based retrieval algorithms. We are equally encouraged by our initial findings on hybrid vector retrieval and hope our framework enables further research on lexical-semantic search, multi-modal retrieval, multimedia retrieval, and other domains.

We additionally claim, as we argue later, that our proposed view opens the door to new and exciting research directions in IR, while, as a meta-algorithm, still allowing the incorporation of decades of research. From principled distributed system design, to the mathematics of alternative sparse vector sketching, to improved clustering or partitioning algorithms, our conceptual framework motivates a number of research questions to pursue. Moreover, our proposal gives a new flavor to the important research on efficient and effective systems in IR [14, 16]: the PAC nature of the framework offers intrinsic levers to tradeoff efficiency for effectiveness that deserve a thorough theoretical and empirical examination.

1.5 Outline

The remainder of this article is organized as follows. We review the relevant parts of the literature in Section 2. We then describe our notation and setup in Section 3. That will let us put in context our analysis and discussion of the behavior of linear and non-linear random projections for sparse vectors in Section 4, and subsequently clustering in Section 5. In Section 6, we show that clustering for IVF and dynamic pruning for inverted indexes are intimately connected, and describe a natural organization of the inverted index through clustering. We philosophize on a unified, density-agnostic framework for MIPS in Section 7. We conclude this article in Section 8.

2 Related Work

This section sets the stage by briefly reviewing the literature on sparse and dense MIPS.

2.1 Sparse MIPS

Numerous sparse MIPS algorithms exist in the IR literature that are specifically tailored to text data and that are behind the success of the field in scaling to massive text collections. We refrain from reviewing this vast literature here and, instead, refer the reader to excellent existing surveys [70, 84] on the topic. But to give context to our work, we quickly make note of key algorithms and explain what makes them less than ideal for the setup we consider in this work.

2.1.1 Sparse MIPS for Text Collections. MaxScore [71] and WAND [11], along with their intellectual descendants [25, 26, 55, 56] are the *de facto* sparse MIPS algorithms, applied typically to vectors obtained from a BM25-encoding [64] of text. This family of algorithms augment a document identifier-sorted inverted index with upper-bounds on the partial score contribution of each coordinate to the final inner product. With that additional statistic, it is possible to traverse the inverted lists one document-at-a-time and decide if a document may possibly end up in the top-k set: If the document appears in enough inverted lists whose collective score upper-bound exceeds the current threshold (i.e., minimum of scores in the current top-k set), then that document should be fully evaluated; otherwise, it has no prospect of ever making it to the top-k set and can therefore be safely rejected.

As articulated elsewhere [17], the logic above is effective when vectors have very specific properties: non-negativity, asymmetricly higher sparsity rate in queries, and a Zipfian distribution of the length of inverted lists. It should be noted that these assumptions are true of relevance measures such as BM25 [64]; sparse MIPS algorithms were designed for text distributions after all.

The limitations of existing algorithms render them inefficient for the general case of sparse MIPS, where vectors may be real-valued and whose sparsity rate is closer to uniform across dimensions. That is because, coordinate upper-bounds become more uniform, leading to less effective pruning of the inverted lists. That, among other problems [17, 19], renders the particular dynamic pruning strategy in MaxScore and WAND ineffective, as demonstrated empirically in the past [17, 50].

2.1.2 Signatures for Logical Queries. There are alternatives to the inverted index, however, such as the use of signatures for retrieval and sketches for inner product approximation [30, 63, 72]. In this class of algorithms, Goodwin et al. [30] describe the BitFunnel indexing machinery. BitFunnel stores a bit signature for every document vector in the index using Bloom filters. These signatures are scanned during retrieval to deduce if a document contains the terms of a conjunctive query. While it is encouraging that a signature-based replacement to inverted indexes appears not only viable but very much practical, the query logic BitFunnel supports is limited to logical ANDs and does not generalize to the setup we are considering in this work.

Pratap et al. considered a simple algorithm [63] to sketch sparse *binary* vectors so that the inner product of sketches approximates the inner product of original vectors. They do so by randomly projecting each coordinate in the original space to coordinates in the sketch. When two or more non-zero coordinates collide, the sketch records their logical OR. While a later work extends this idea to categorical-valued vectors [72], it is not obvious how the proposed sketching mechanisms may be extended to real-valued vectors.

2.1.3 General Sparse MIPS. The most relevant work to ours is the recent study of general sparse MIPS by Bruch et al. [17]. Building on random projections, the authors proposed a sketching algorithm, dubbed Sinnamon, that embeds sparse vectors into a low-dimensional sparse subspace. Sinnamon, as with the previous approach, randomly projects coordinates from the original space to the sketch space. But the sketch space is a union of two subspaces: One that records the upper-bound on coordinate values and another that registers the lower-bound instead. It was shown that

reconstructing a sparse vector from the sketch approximates inner product with any arbitrary query with high-accuracy.

Bruch et al. [17] couple the sketches with an inverted index, and empirically evaluate a coordinate-at-a-time algorithm for sparse MIPS. They show considerable compression rate in terms of the size of the index as well as latencies that are sometimes an order of magnitude better than WAND on embedding vectors produced by Splade [27, 28].

2.2 Dense MIPS

Let us note that there exists an extremely vast body of works on ANN search that is in and of itself an interesting area of research [12]. Strictly speaking, however, MIPS is a fundamentally different (and, in fact, a much harder) problem because inner product is not a proper metric; in fact, maximum cosine similarity search and ANN with Euclidean distance are special cases of MIPS. In spite of this, many MIPS solutions for dense vectors adapt ANN solutions to inner product, often without any theoretical justification.

Consider, e.g., the family of MIPS solutions that is based on proximity graphs such as IP-NSW [57] and its many derivatives [44, 67, 82]. These classes of algorithms construct a graph where each data point is a node in the graph and two nodes are connected if they are deemed "similar." Typically, similarity is based on Euclidean distance. But the authors of [57] show that when one uses inner product (albeit improperly) to construct the graph, the resulting structure is nonetheless capable of finding the maximizers of inner product rather quickly and accurately.

Graph-based methods may work well but they come with two serious issues. First, while we can reason about their performance in the Euclidean space, we can say very little about why they do or do not work for inner product, and under what conditions they may fail. It is difficult, e.g., to settle on a configuration of hyperparameters without conducting extensive experiments and evaluation on a validation dataset. The second and even more limiting challenge is the poor scalability and slow index construction of graph methods.

Another family of MIPS algorithms can best be described as different realizations of **Locality Sensitive Hashing (LSH)** [32, 33, 45, 58, 65, 66, 75, 78]. The idea is to project data points such that "similar" points are placed into the same "bucket." Doing so enables sublinear search because, during retrieval, we limit the search to the buckets that collide with the query.

Many LSH methods for MIPS transform the problem to Euclidean or angular similarity search first, in order to then recycle existing hash functions. One of the main challenges with this way of approaching MIPS is that inner product behaves oddly in high dimensions, in a way that is different from, say, Euclidean distance: the maximum inner product between vectors is typically much smaller than the average vector norm. Making LSH-based MIPS accurate requires an increasingly larger number of projections, which leads to an unreasonable growth in index size [69].

Another method that is borrowed from the ANN literature is search using an IVF. This method takes advantage of the geometrical structure of vectors to break a large collection into smaller partitions. Points within each partition are expected to result in a similar inner product with an arbitrary query point—though there are no theoretical guarantees that that phenomenon actually materializes. Despite that, clustering-based IVF is a simple and widely-adopted technique [34, 35], and has been shown to perform well for MIPS [8]. Its simplicity and well-understood behavior are the reasons we study this particular technique in this work.

Finally, in our review of the dense MIPS literature, we exclusively described space partitioning algorithms that reduce the search space through some form of partitioning or hashing, or by organizing vectors in a graph structure and traversing the edges toward the nearest neighbors of a given query. It should be noted, however, that the other and often critical aspect of MIPS is the actual computation of inner product. There are many works that address that particular challenge

Table 1. Datasets of Interest Along with Select Statistics

Dataset	Document Count	Query Count	Splade	Efficient Splade
MS Marco Passage	8.8 M	6,980	127 (49)	185 (5.9)
NQ	2.68 M	3,452	153 (51)	212 (8)
Quora	523 K	10,000	68 (65)	68 (8.9)
HotpotQA	5.23 M	7,405	131 (59)	125 (13)
Fever	5.42 M	6,666	145 (67)	140 (8.6)
DBPedia	4.63 M	400	134 (49)	131 (5.9)

The rightmost two columns report the average number of non-zero entries in documents and, in parentheses, queries for sparse vector representations of the datasets.

often via quantization (see [31] and references therein) but that are beyond the scope of this article.

3 Notation and Experimental Setup

We begin by laying out our notation and terminology. Furthermore, throughout this work, we often interleave theoretical and empirical analysis. To provide sufficient context for our arguments, this section additionally gives details on our empirical setup and evaluation measures.

3.1 Notation

Suppose we have a collection $\mathcal{X} \subset \mathbb{R}^{m+N}$ of possibly *hybrid* vectors. That means, if $x \in \mathcal{X}$, then x is a vector that is comprised of an m-dimensional *dense*, and an N-dimensional *sparse* array of coordinates, where dense and sparse are as defined in Section 1. We abuse terminology and call the dense part of x its "dense vector" and denote it by $x^d \in \mathbb{R}^m$. Similarly, we call the sparse part, $x^s \in \mathbb{R}^N$, its "sparse vector." We can write $x = x^d \oplus x^s$, where $\oplus$ denotes concatenation.

The delineation above will prove helpful later when we discuss the status quo and our proposal within one mathematical framework. Particularly, we can say that a sparse retrieval algorithm operates on the sparse collection $\mathcal{X}^s = \{x^s \mid x = x^d \oplus x^s \in \mathcal{X}\}$, and similarly dense retrieval algorithms operate on $\mathcal{X}^d$, defined symmetrically. Hybrid vectors collapse to dense vectors when $N = 0$ (or when $x^s = \mathbf{0}$ for all $x \in \mathcal{X}$), and reduce to sparse vectors when $m = 0$ (or $x^d = \mathbf{0}\ \forall x \in \mathcal{X}$).

In our notation, MIPS aims to solve the following problem:

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

to find, from $\mathcal{X}$, the set $\mathcal{S}$ of top-k vectors whose inner product with the query vector $q = q^d \oplus q^s \in \mathbb{R}^{m+N}$ is maximal. Sparse and dense MIPS are then special cases of the formulation above, when query and document vectors are restricted to their sparse or dense subspaces respectively.

We write $nz(u)$ for the set of non-zero coordinates in a vector, $nz(u) = \{i \mid u_i \neq 0\}$, and denote the average number of non-zero coordinates with $\psi = \mathbb{E}[|nz(X)|]$ for a random vector X. We denote coordinate i of a vector u using subscripts: u_i. To refer to the jth vector in a collection of vectors, we use superscripts: $u^{(j)}$. We write $\langle u, v \rangle$ to express the inner product of two vectors u and v. We denote the set of consecutive natural numbers $\{1, 2, \ldots, m\}$ by $[m]$ for brevity. Finally, we reserve capital letters to denote random variables (e.g., X) and calligraphic letters for sets (e.g., $\mathcal{X}$).

3.2 Experimental Configuration

3.2.1 Datasets. We perform our empirical analysis on a number of publicly available datasets, summarized in Table 1. The largest dataset used in this work is the MS Marco[3] Passage Retrieval v1 dataset [59], a retrieval and ranking collection from Microsoft. It consists of about 8.8 million short passages which, along with queries in natural language, originate from Bing. The queries are split into train, dev, and eval subsets. We use the small dev query set (consisting of 6,980 queries) in our analysis.

In addition to the Bing queries, we evaluate all methods with the Trec Deep Learning track queries from Trec 2019 [21] and Trec 2020 [20], each consisting of 200 queries. We denote the former by Trec DL-2019 and the latter by Trec DL-2020. Because the statistics of Trec DL datasets are similar to the MS Marco dataset (except query count), we omit these from Table 1.

We also experiment with 5 datasets from the BeIR [68] collection[4]: Natural Questions (NQ, question answering), Quora (duplicate detection), HotpotQA (question answering), Fever (fact extraction), and DBPedia (entity search). For a more detailed description of each dataset, we refer the reader to [68]. We chose these particular datasets because they represent the largest, publicly available subsets of the BeIR collection; other BeIR datasets are either much smaller in size or are not easily accessible.

3.2.2 Sparse Vectors. We convert the datasets above into sparse vectors by using Splade [27] and Efficient Splade [40]. *Splade*[5] [27] is a deep learning model that produces sparse representations for text. The vectors have roughly 30,000 dimensions, where each dimension corresponds to a term in the BERT [23] WordPiece [77] vocabulary. Non-zero entries in a vector reflect learnt term importance weights.

Splade representations allow us to test the behavior of our algorithm on query vectors with a large number of non-zero entries. However, we also create another set of vectors using a more efficient variant of Splade, called *Efficient Splade*[6] [40]. This model produces queries that have far fewer non-zero entries than the original Splade model, but documents that may have a larger number of non-zero entries.

These two models give us a range of sparsity rates to work with and examine our algorithms on. As a way to compare and contrast the more pertinent properties of the learnt sparse representations, Table 1 shows the differences in the sparsity rate of the two embedding models for all datasets considered in this work.

3.2.3 Evaluation. Our main metric of interest is the *accuracy*[7] of approximate algorithms, measured as follows: For every test query, we obtain the exact solution to MIPS by exhaustively searching over the entire dataset. We then obtain an approximate set of top-k documents using a system of interest. Accuracy is then measured as the ratio of exact documents that are present in the approximate set. This metric helps us study the impact of the different sources of error.

We also report *throughput* as **Queries Per Second (QPS)** in a subset of our experiments where efficiency takes center stage. When computing QPS, we include the time elapsed from the moment query vectors are presented to the algorithm to the moment the algorithm returns the requested

[3]Available at https://microsoft.github.io/msmarco/

[4]Available at https://github.com/beir-cellar/beir

[5]Pre-trained checkpoint from HuggingFace available at https://huggingface.co/naver/splade-cocondenser-ensembledistil

[6]Pre-trained checkpoints for document and query encoders were obtained from https://huggingface.co/naver/efficient-splade-V-large-doc and https://huggingface.co/naver/efficient-splade-V-large-query, respectively.

[7]What we call "accuracy" in this work is also known as "recall" in the ANN literature. However, "recall" is an overloaded term in the IR literature as it also refers to the portion of *relevant* documents returned for a query. We use "accuracy" instead to avoid that confusion.

top-k document vectors for all queries—we emphasize that the algorithms used in this work do not operate in batch mode. We note that, because this work is a study of retrieval of vectors, we do not factor into throughput the time it takes to embed a given piece of text.

3.2.4 Hardware and Code. We conduct experiments on a commercially available platform with an Intel Xeon Platinum 8481C Processor (Sapphire Rapids) with a clock rate of 1.9 GHz, 20 virtual CPUs (2 vCPUs per physical core), and 44 GB of main memory. This setup represents a typical server in a production environment—in fact, we rented this machine from the Google Cloud Platform.

We further note that, we implemented all the methods discussed in this work in the Rust programming language. We rely on the Rust compiler for any platform-specific optimization and do not otherwise optimize the code for the Intel platform (such as by developing SIMD code).

4 Analysis of Random Projections for Sparse Vectors

As noted earlier, the historical bifurcation of the retrieval machinery can, in no small part, be attributed to the differences between sparse and dense vectors—in addition to the application domain. For example, sparse vectors are plagued with a much more serious case of the *curse of dimensionality*. In extremely high-dimensional spaces where one may have thousands to millions of dimensions, the geometrical properties and probabilistic certainty that power clustering start to break down. So does our intuition of the space.

The high dimensionality of sparse vectors poses another challenge: greater computation required to perform basic operations. While optimized implementations (see, e.g., [38] and references therein) of spherical k-means exist for sparse vectors, e.g., their efficiency nonetheless degrades with the number of dimensions. Standard k-means is even more challenged: Cluster centroids are likely to be high-dimensional *dense* vectors, leading to orders of magnitude more computation to perform cluster assignments in each iteration of the algorithm.

These difficulties—computational complexity and geometrical oddities—pose a fundamental challenge to clustering over sparse vectors. That leads naturally to dimensionality reduction, and in particular *sketching* [74]: summarizing a high-dimensional vector into a lower-dimensional space such that certain properties, such as the distance between points or inner products, are preserved with some quantifiable error.

The reason sketching is appealing is that the mathematics behind it offer guarantees in an *oblivious* manner: with no further assumptions on the source and nature of the vectors themselves or their distribution. Additionally, sketching a vector is often fast since it is a requisite for their application in streaming algorithms. Finally, the resulting *sketch* in a (dense and) low-dimensional space facilitates faster subsequent computation in exchange for a controllable error.

In this work, we explore two functions ($\phi(\cdot)$ in the notation of Algorithm 1): One classical result that has powered much of the research on sketching is the linear JL transform [36], which produces dense sketches of its input and enables computing an unbiased estimate of inner product (or Euclidean distance). Another, is the non-linear Sinnamon function [17] that produces sparse sketches of its input that enable deriving upper-bounds on inner product.

In the remainder of this section, we review these two algorithms in-depth and compare and contrast their performance. Importantly, we consider the approximation error in isolation: How does sketching affect MIPS if our MIPS algorithm itself were exact? In other words, if we searched exhaustively for the top-k maximizers of inner product with a query, what accuracy may we expect if that search were performed on sketches of vectors versus the original vectors?

4.1 The JL Transform

4.1.1 Review. Let us repeat the result due to Johnson and Lindenstrauss [36] for convenience:

Lemma 4.1 (JL). *For $0 < \epsilon < 1$ and any set $\mathcal{V}$ of $|\mathcal{V}|$ points in $\mathbb{R}^N$, and an integer $n = \Omega(\epsilon^{-2} \ln |\mathcal{V}|)$, there exists a Lipschitz mapping $f : \mathbb{R}^N \to \mathbb{R}^n$ such that*

$$(1-\epsilon)\|u-v\|_2^2 \leq \|f(u)-f(v)\|_2^2 \leq (1+\epsilon)\|u-v\|_2^2,$$

for all $u, v \in \mathcal{V}$.

This result has been extensively studied and further developed since its introduction. Using simple proofs, e.g., it can be shown that the mapping f may be a linear transformation by an $n \times N$ random matrix Φ drawn from a certain class of distributions. Such a matrix Φ is said to form a JL transform [74].

There are many constructions of Φ that form a JL transform. It is trivial to show that when the entries of Φ are independently drawn from $\mathcal{N}\left(0, \frac{1}{n}\right)$, then Φ is a JL transform with parameters $(\epsilon, \delta, \theta)$ if $n = \Omega(\epsilon^{-2} \ln(\theta/\delta))$. $\Phi = \frac{1}{\sqrt{n}}R$, where $R_{n\times N}$ is a matrix whose entries are independent Rademacher random variables, is another simple-to-prove example of a JL transform. The literature offers a large number of other, more efficient constructions such as the Fast JL Transform [1], as well as specific theoretical results for sparse vectors (e.g., [10]). We refer the interested reader to [74] for an excellent survey of these results.

4.1.2 Theoretical Analysis. In this work, we are interested in the transformation in the context of inner product rather than the $\ell 2$ norm and Euclidean distance. Let us take $\phi(u) = Ru$, with $R \in \{-1/\sqrt{n}, 1/\sqrt{n}\}^{n\times N}$, as one candidate sketching function in Algorithm 1 and state the following results for our particular construction:

Theorem 4.2. *Fix two vectors u and $v \in \mathbb{R}^N$. Define $Z_{SKETCH} = \langle \phi(u), \phi(v) \rangle$ as the random variable representing the inner product of sketches of size n, prepared using the projection $\phi(u) = Ru$, with $R \in \{-1/\sqrt{n}, 1/\sqrt{n}\}^{n\times N}$ being a random Rademacher matrix. Z_{SKETCH} is an unbiased estimator of $\langle u, v \rangle$. Its distribution tends to a Gaussian with variance:*

$$\frac{1}{n}\left(\|u\|_2^2\|v\|_2^2 + \langle u, v\rangle^2 - 2\sum_i u_i^2 v_i^2\right). \tag{2}$$

We give our proof of the claim above in Appendix A. We next make the following claim for a fixed query vector q and a *random* document vector, thereby taking it a step closer to the MIPS setup. We present a proof in Appendix B.

Theorem 4.3. *Fix a query vector $q \in \mathbb{R}^N$ and let X be a random vector drawn according to the following probabilistic model. Coordinate i, X_i, is non-zero with probability $p_i > 0$ and, if it is non-zero, draws its value from a distribution with mean μ and variance σ^2. $Z_{SKETCH} = \langle \phi(q), \phi(X) \rangle$, with $\phi(u) = Ru$ and $R \in \{-1/\sqrt{n}, 1/\sqrt{n}\}^{n\times N}$, has expected value $\mu \sum_i p_i q_i$ and variance:*

$$\frac{1}{n}\left[(\mu^2+\sigma^2)\left(\|q\|_2^2 \sum_i p_i - \sum_i p_i q_i^2\right) + \mu^2\left(\left(\sum_i q_i p_i\right)^2 - \sum_i (q_i p_i)^2\right)\right]. \tag{3}$$

Consider the special case where $p_i = \psi/N$ for some constant ψ for all dimensions i. Further assume, without loss of generality, that the (fixed) query vector has unit norm: $\|q\|_2 = 1$. It can be observed that the variance of Z_{SKETCH} decomposes into a term that is $(\mu^2+\sigma^2)(1-1/N)\psi/n$, and a second term that is a function of $1/N^2$. The mean is a linear function of the non-zero coordinates in the query: $(\mu \sum_i q_i)\psi/N$. As N grows, the mean of Z_{SKETCH} tends to 0 at a rate proportional to the sparsity rate (ψ/N), while its variance tends to $(\mu^2+\sigma^2)\psi/n$.

The analysis above suggests that the ability of $\phi(\cdot)$, as defined in this section, to preserve the inner product of a query vector with a randomly drawn document vector deteriorates as a function

of the number of non-zero coordinates. For example, when the number of non-zero coordinates becomes larger, $\langle \phi(q), \phi(X) \rangle$ for a fixed query q and a random vector X becomes less reliable because the variance of the approximation increases. Nonetheless, as we see later in this work, the degree of noise is often manageable in practice as evidenced by the accuracy of Algorithm 2.

4.2 The Sinnamon Transform

4.2.1 Review. Like JL transform, Sinnamon [17] aims to reduce the dimensionality of (sparse) vectors. Unlike JL transform, it does so through a non-linear mapping.

Sinnamon uses half the sketch to record upper-bounds on the values of non-zero coordinates in a vector, and the other half to register lower-bounds. For notational convenience, let us assume that the sketch size is $n = 2m$. Given a vector $u \in \mathbb{R}^N$ and h independent random mappings $\pi_o : [N] \rightarrow [m]$ $(1 \leq o \leq h)$, Sinnamon constructs the upper-bound sketch $\overline{u} \in \mathbb{R}^m$ where its kth coordinate is assigned the following value:

$$\overline{u}_k \leftarrow \max_{\{i \in nz(u) \mid \exists\, o \text{ s.t. } \pi_o(i)=k\}} u_i. \tag{4}$$

The lower-bound sketch, $\underline{u}$, is filled in a symmetric manner, in the sense that the algorithmic procedure is the same but the operator changes from $\max(\cdot)$ to $\min(\cdot)$.

Computing the inner product between a query vector $q \in \mathbb{R}^N$ and a vector u given its sketch $(\phi(u) = \underline{u} \oplus \overline{u})$ uses the following procedure: Positive query values are multiplied by the least upper-bound from $\overline{u}$, and negative query values by the greatest lower-bound from $\underline{u}$:

$$\sum_i \mathbb{1}_{i \in nz(u)} q_i \left(\mathbb{1}_{q_i > 0} \min_{k \in \{\pi_o(i)\ 1 \leq o \leq h\}} \overline{u}_k + \mathbb{1}_{q_i < 0} \max_{k \in \{\pi_o(i)\ 1 \leq o \leq h\}} \underline{u}_k \right). \tag{5}$$

The indicator $\mathbb{1}_{i \in nz(u)}$, which is kept in conjunction with the sketch, guarantees that the partial inner product between a query coordinate q_i and the sketch of a document vector (i.e., individual summands in Equation (5)) is 0 if $i \notin nz(u)$. That pairing of the sketch with the indicator function improves the bound on error dramatically while maintaining a large compression rate. For formal results on the probability of the inner product error, we refer the reader to the original work [17].

4.2.2 Theoretical Analysis. In this work, we use a simplified instance of Sinnamon, which we call Weak Sinnamon, by (a) setting the number of random mappings to 1, which we denote by π; and (b) removing $\mathbb{1}_{i \in nz(u)}$ from the inner product computation. These two reductions have important side effects that ultimately enable us to apply existing clustering algorithms and compute inner product between vectors.

Let us focus on the upper-bound sketch to illustrate these differences; similar arguments can be made for the lower-bound sketch. First, notice that the upper-bound sketch of a document vector simplifies to $\overline{u}$ where:

$$\overline{u}_k \leftarrow \max_{\{i \in nz(u) \mid \pi(i)=k\}} u_i, \tag{6}$$

and that the upper-bound sketch of a query vector, $\overline{q}$, becomes:

$$\overline{q}_k \leftarrow \sum_{\{i \in nz(q) \mid \pi(i)=k \,\wedge\, q_i > 0\}} q_i. \tag{7}$$

We denote the former by $\phi_d(\cdot)$ (for document) and the latter by $\phi_q(\cdot)$ (for query).

Second, the inner product computation between the sketches of query and document vectors reduces to:

$$\langle \phi_q(q), \phi_d(u) \rangle = \langle \overline{q}, \overline{u} \rangle + \langle \underline{q}, \underline{u} \rangle = \sum_{i:\, q_i > 0} q_i \overline{u}_{\pi(i)} + \sum_{i:\, q_i < 0} q_i \underline{u}_{\pi(i)}. \tag{8}$$

We now extend the analysis in [17] to the setup above. We begin by stating the following claim that is trivially true:

Theorem 4.4. *For a query vector q and document vector u, $\langle q, u\rangle \leq \langle \phi_q(q), \phi_d(u)\rangle$.*

Importantly, the inner product between query and document sketches is not an unbiased estimator of the inner product between the original vectors. Let us now model the probability of the approximation error.

Consider the upper-bound sketch first. Using a similar argument to Theorem 5.4 of [17], we state the following result and provide a proof in Appendix C:

Theorem 4.5. *Let X be a random vector drawn according to the following probabilistic model. Coordinate i, X_i, is non-zero with probability $p_i > 0$ and, if it is non-zero, draws its value from a distribution with PDF ϕ and CDF Φ. Then:*

$$\mathbb{P}\left[\overline{X}_{\pi(i)} - X_i \leq \delta\right] \approx (1-p_i)\left(e^{-\frac{1}{m}(1-\Phi(\delta))\sum_{j\neq i} p_j}\right) + p_i \int e^{-\frac{1}{m}(1-\Phi(\alpha+\delta))\sum_{j\neq i} p_j}\phi(\alpha)d\alpha. \quad (9)$$

A symmetric argument can be made for the error of the lower-bound sketch. Crucially, given the result above, which formalizes the CDF of the sketching approximation error, we can obtain the expected value and variance of the random variables $\overline{X}_{\pi(i)} - X_i$ and $\underline{X}_{\pi(i)} - X_i$ for all dimensions i. From there, and following similar arguments as the proof of Theorem 5.8 of [17], it is easy to show that the approximation error takes on a Gaussian distribution with mean:

$$\sum_{i:\, q_i>0} q_i \mathbb{E}\left[\overline{X}_{\pi(i)} - X_i\right] + \sum_{i:\, q_i<0} q_i \mathbb{E}\left[\underline{X}_{\pi(i)} - X_i\right]$$

and variance that is:

$$\sum_{i:\, q_i>0} q_i^2 Var\left[\overline{X}_{\pi(i)} - X_i\right] + \sum_{i:\, q_i<0} q_i^2 Var\left[\underline{X}_{\pi(i)} - X_i\right].$$

Let us illustrate the implications of Theorem 4.5 by considering the special case where $p_i = \psi/N$ for all dimensions i. As the sparsity rate increases and N grows, the second term in Equation (9) tends to 0 at a rate proportional to ψ/N, while the first term dominates, tending approximately to $\exp\left(-(1-\Phi(\delta))\psi/m\right)$. By making ψ/m smaller, we can control the approximation error and have it concentrate on smaller magnitudes. That subsequently translates to a more accurate inner product between a fixed query and a randomly drawn document vector.

As a final remark on Weak Sinnamon, we note that when n is larger than the number of non-zero coordinates in a document vector, the resulting sketch itself is *sparse*. Furthermore, sketching using Weak Sinnamon only requires $O(\psi)$ operations, with ψ denoting the number of non-zero coordinates, while the JL transform has a sketching complexity of $O(n\psi)$. As we explain later, these properties will play a key role in the efficiency of sparse MIPS.

4.3 Empirical Comparison

Our results from the preceding sections shed light on how JL and Weak Sinnamon transformations are expected to behave when applied to sparse vectors. Our main conclusion is that the sparsity rate heavily affects the approximation error. In this section, we design experiments that help us observe the expected behavior in practice and compare the two algorithms on real data.

Given a sparse dataset and a set of queries, we first obtain the exact top-1 document for each query by performing an exhaustive search over the entire collection. We then create a second dataset wherein each vector is a sketch of a vector in the original dataset. We now perform exact

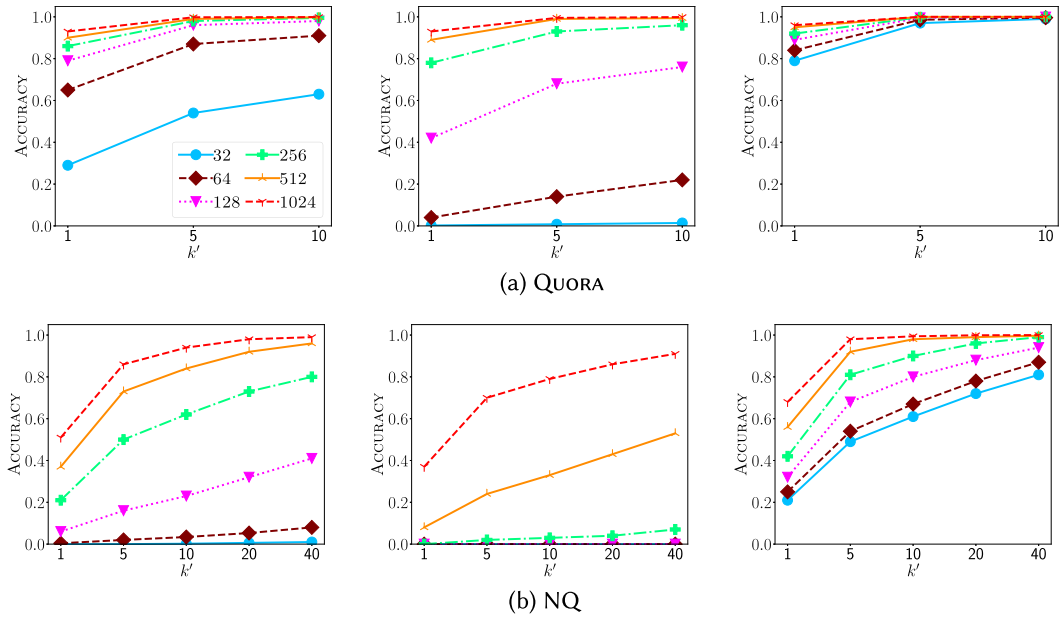


Fig. 1. Top-1 accuracy of retrieval for test queries over sketches produced by JL transform (left column), Weak Sinnamon (middle column), and, as a point of reference, the original Sinnamon algorithm (right column). We retrieve the top-k' documents by performing an exhaustive search over the sketch collection and re-ranking the candidates by exact inner product to obtain the top-1 document and compute accuracy. Each line in the figures represents a different sketch size n. We note that Weak Sinnamon and Sinnamon only use half the sketch to record upper-bounds but leave the lower-bound sketch unused because Splade vectors are non-negative. That implies that their effective sketch size is half that of the JL transform's.

search over the sketch dataset to obtain top-k' ($k' \geq 1$) documents, and report the accuracy of the approximate retrieval.

There are two parameters in the setup above that are of interest to us. First is the sketch size, n. By fixing the dataset (thus its sparsity rate) but increasing the sketch size, we wish to empirically quantify the effect of using larger sketches on the ability of each algorithm to preserve inner product. Note that, because the vectors are non-negative, Weak Sinnamon only uses half the sketch capacity to form the upper-bound sketch—reducing its effective sketch size to $n/2$.

The second factor is k' which controls how "hard" a retrieval algorithm must work to compensate for the approximation error. Changing k' helps us understand if the error introduced by a particular sketch size can be attenuated by simply retrieving more candidates and later re-ranking them according to their exact score.

The results of our experiments are presented in Figure 1 for select datasets embedded with the Splade model. We chose these datasets because they have very different sizes and sparsity rates, as shown in Table 1, with Quora having the largest sparsity rate and fewest documents, and NQ the smallest sparsity rate and a medium collection size.

Naturally, our observations are consistent with what the theoretical results predict. The sketch quality improves as its size increases. That shows the effect of the parameter n on the approximation variance of the JL transform and the concentration of error in Weak Sinnamon sketches.

Another unsurprising finding is that Weak Sinnamon's sensitivity to the ψ/n factor becomes evident in NQ: When the ratio between the number of non-zero coordinates and the sketch size

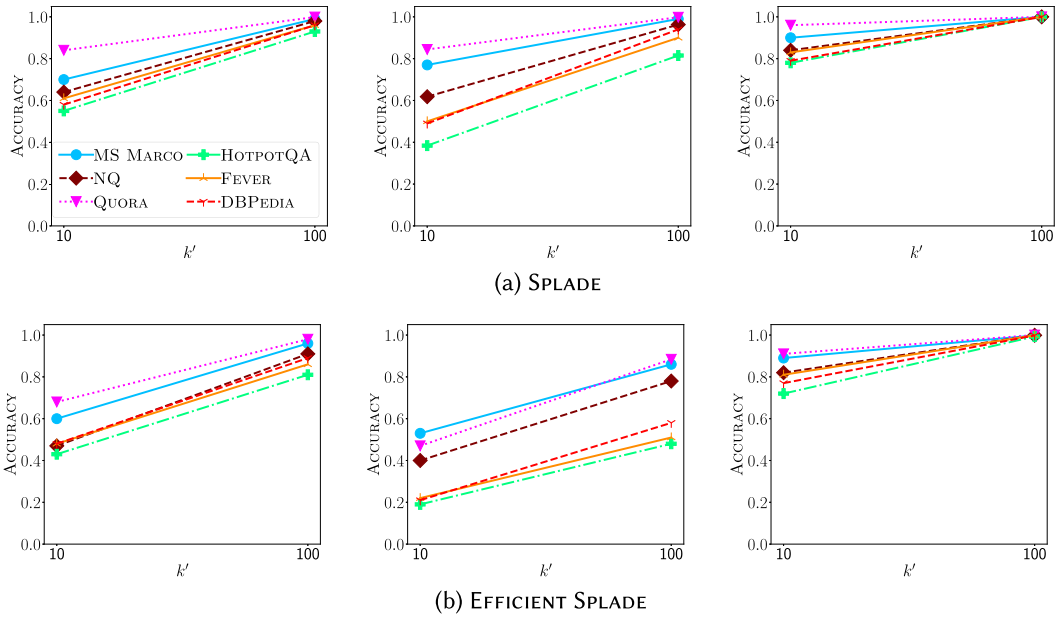


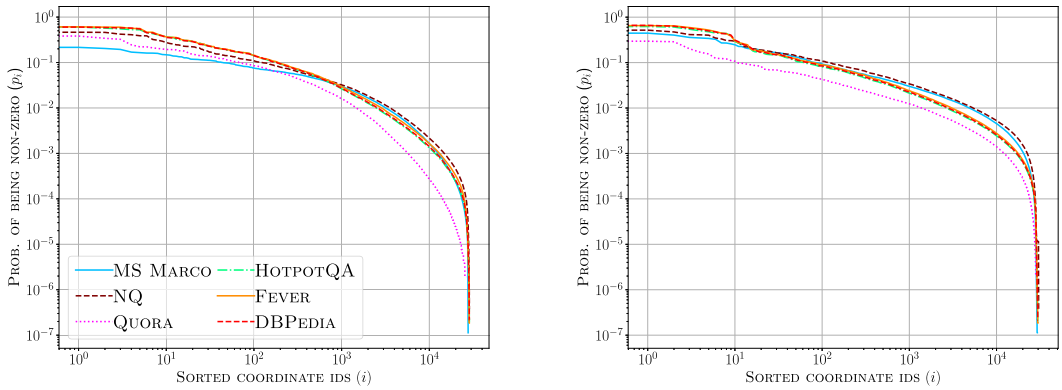
Fig. 2. Top-10 accuracy of retrieval for test queries over sketches of size $n = 1,024$ produced by JL transform (left column), Weak Sinnamon (middle column), and, for reference, the original Sinnamon algorithm (right column). As in Figure 1, we retrieve the top-k' documents by performing an exhaustive search over the sketch collection and re-ranking the candidates by exact inner product to obtain the top-10 documents and compute accuracy. Each line in the figures represents a different dataset. In these experiments, we adjust the effective sketch size of Weak Sinnamon and Sinnamon to match that of the JL transform's.

(ψ/n) is large, the variance of the approximation error becomes larger. The reason is twofold: more non-zero coordinates are likely to collide as vectors become more dense; and, additionally, sketches themselves become more dense, thereby increasing the likelihood of error for *inactive* coordinates. To contextualize Weak Sinnamon and the effects of our modifications to the original algorithm on the approximation error, we also plot in Figure 1 the performance of Sinnamon.

While increasing the sketch size is one way to lower the probability of error, casting a wider net (i.e., $k' > k$) followed by re-ranking appears to also improve retrieval quality.

Now that we have a better understanding of the effect of the parameters on the quality of the sketching algorithms, let us choose one configuration and repeat the experiments above on all our datasets. One noteworthy adjustment is that we set Weak Sinnamon's effective sketch size to match that of the JL transform's: As we noted, because Weak Sinnamon leaves the lower-bound sketch unused for non-negative vectors, we re-allocate it for the upper-bound sketch, in effect giving Weak Sinnamon's upper-bound sketch n dimensions to work with. Another change is that we use a more challenging configuration and perform top-10 retrieval. Finally, we also include Efficient Splade for completeness.

Figure 2 shows the results of these experiments. The general trends observed in these figures are consistent with the findings of Figure 1: Obtaining a larger pool of candidates from sketches and re-ranking them according to their exact inner product is a reliable way of countering the approximation error; and, Weak Sinnamon generally underperforms the JL transform in preserving inner product between vectors. Additionally, as vectors become more dense, the sketching quality degrades, leading to a higher approximation error.



(a) Splade (b) Efficient Splade

Fig. 3. Probability of each coordinate being non-zero (p_i for coordinate i) for Splade and Efficient Splade vectors of several datasets. To aid visualization, we sort the coordinates by p_i's in descending order. A Zipfian distribution would manifest as a line in the log-log plot. Notice that, this distribution is closer to uniform for MS Marco than others.

Another interesting but expected phenomenon is that sketching performs comparatively poorly on Efficient Splade. That is because, query vectors generated by the Efficient Splade model are more sparse than those made by Splade. When a query has few non-zero coordinates, the expected inner product becomes small while the variance of JL transform sketches concentrates around a constant, as predicted by Theorem 4.3. As for Weak Sinnamon, when queries have a large number of non-zero coordinates, the shape of the distribution of error becomes less sensitive to the approximation error of individual coordinates; with fewer non-zero coordinates in the query vector, the opposite happens.

As a final observation, we notice that retrieval accuracy is generally higher for Quora, MS Marco, and NQ datasets. That is easy to explain for Quora as it is a more sparse dataset with a much smaller ψ/n. On the other hand, the observed trend is rather intriguing for a larger and more dense dataset such as MS Marco. On closer inspection, however, it appears that the stronger performance can be attributed to the probabilities of coordinates being non-zero (i.e., p_i's). In Figure 3, we plot the distribution of p_i's but, to make the illustration cleaner, sort the coordinates by their p_i in descending order. Interestingly, the distribution of p_i's is closer to uniform for MS Marco and NQ, while it is more heavily skewed for Fever, DBPedia, and HotpotQA.

5 Evaluation of Clustering Over Sketches of Sparse Vectors

In the preceding section, we were squarely concerned with the ability of the two sketching algorithms in approximately preserving inner product between query and document vectors. That analysis is relevant if one were to directly operate on sketches as opposed to the original vectors when, say, building a graph-based nearest neighbor search index such as HNSW [52] or IP-NSW [57]. In this work, our primary use for sketches is to form partitions in the context of Algorithms 1 and 2: Whether $\mathcal{R}$ searches over sketches or the original vectors is left as a choice.

In that framework, Section 4 has already studied the first line of the two algorithms: sketching the sparse vectors. In this section, we turn to the clustering procedure and empirically evaluate two alternatives: Standard and spherical k-means. Note that, the clustering choice is the last piece required to complete the two algorithms and apply IVF-style search to sparse vectors.

Standard k-means is an iterative protocol that partitions the input data into a predefined number of clusters, K. It first samples K arbitrary points, called "centroids," from the data distribution at random—though there are other initialization protocols available, such as k-means++ [5]. It then repeats until convergence two steps: It assigns each data point to the nearest centroid by their Euclidean distance to form partitions in the first step; and, in the second step, recomputes the centroids to be the mean of the mass of all data points assigned to each partition. While this Expectation-Maximization procedure may fall into local optima, it generally produces partitions that approximate Voronoi regions in a dataset.

Spherical k-means works similarly, with the notable exception that at the end of each iteration, it normalizes the centroids so that they are projected onto the unit sphere. This form of clustering has been used in the past for a topical analysis of text documents [24] among other applications.

Both of these clustering algorithms are popular choices in the IVF-based ANN search as evidenced by their integration into commonly used software packages such as FAISS [35]. As such, we plug the two methods into Algorithms 1 and 2 and apply them to our datasets. Our objective is to understand the differences between the two clustering choices in terms of overall retrieval quality as well as their sensitivity to the choice of sketching algorithm.

5.1 Empirical Comparison

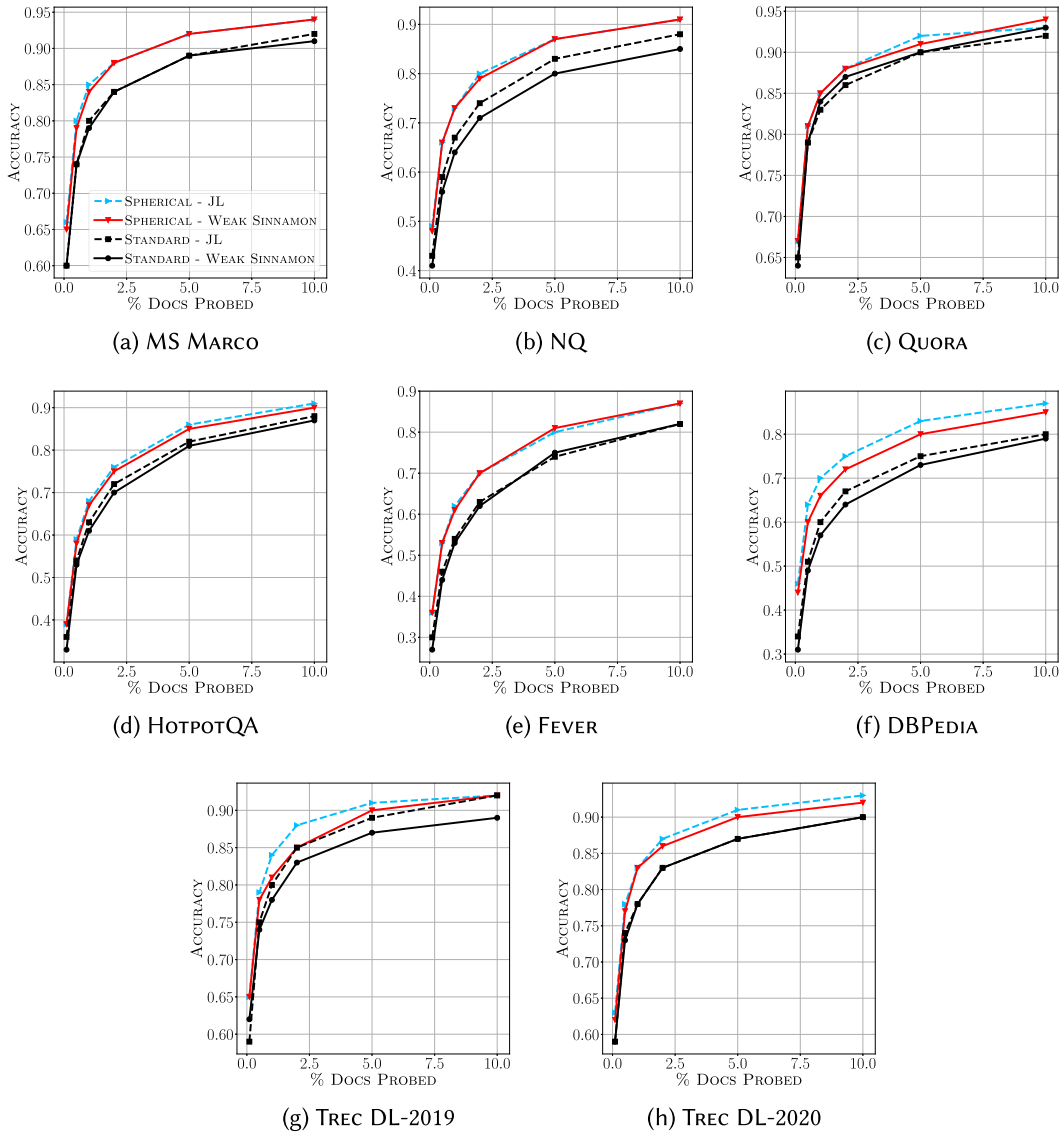
We begin by emphasizing that, in this particular section, we do not pay attention to speed and only report accuracy as a function of the total number of documents examined, ℓ, in Algorithm 2. Additionally, we use an *exact*, exhaustive search algorithm as $\mathcal{R}$ over the original vectors to find the final top-k candidates once the ℓ-subset of a dataset has been identified.

Before we state our findings, a note on our choice of "the number of documents examined" (ℓ) versus the more familiar notion of "the number of clusters searched" (known commonly as nProbe): The standard k-means algorithm is highly sensitive to vector norms. That is natural as the algorithm cares solely about the Euclidean distance between points within a partition. When it operates on a collection of vectors with varying norms, then, it is intuitive that it tends to isolate high-normed points in their own, small partitions, while lumping together the low-normed vectors into massive clusters. As a result of this phenomenon, partitions produced by standard k-means are often imbalanced. Probing a fixed number of partitions at search time puts k-means at an unfair disadvantage compared to its spherical variant. By choosing to work with ℓ rather than fixating on the number of top clusters we remove that variable from the equation.

Figure 4 summarizes our results for the Splade-generated vectors. We plot one figure per dataset, where each figure depicts the relationship between top-10 accuracy and ℓ (expressed as percentage of the collection size). When applying Algorithm 1 to the datasets, we set the sketch size to 1, 024 as per findings of Section 4. Additionally, we fix the number of partitions P to $4\sqrt{|\mathcal{X}|}$ where $|\mathcal{X}|$ is the number of documents in a dataset $\mathcal{X}$. Plots for Efficient Splade are shown separately in Figure 5.

One of the most striking observations is that spherical k-means appears to be a better choice universally on the vector datasets we examine in this work. By partitioning the data with spherical k-means in Algorithm 1 and examining at most 10% of the collection, we often reach a top-10 accuracy well above 0.8 and often 0.9. This is in contrast to the performance of standard k-means which often lags behind.

We are also surprised by how little the choice of JL transform versus Weak Sinnamon appears to matter, in the high-accuracy region, for the purposes of partitioning with spherical k-means and retrieval over the resulting partitions. When the clustering method is the standard k-means, on the other hand, the difference between the two sketching algorithms is sometimes more noticeable.



(a) MS Marco (b) NQ (c) Quora

(d) HotpotQA (e) Fever (f) DBPedia

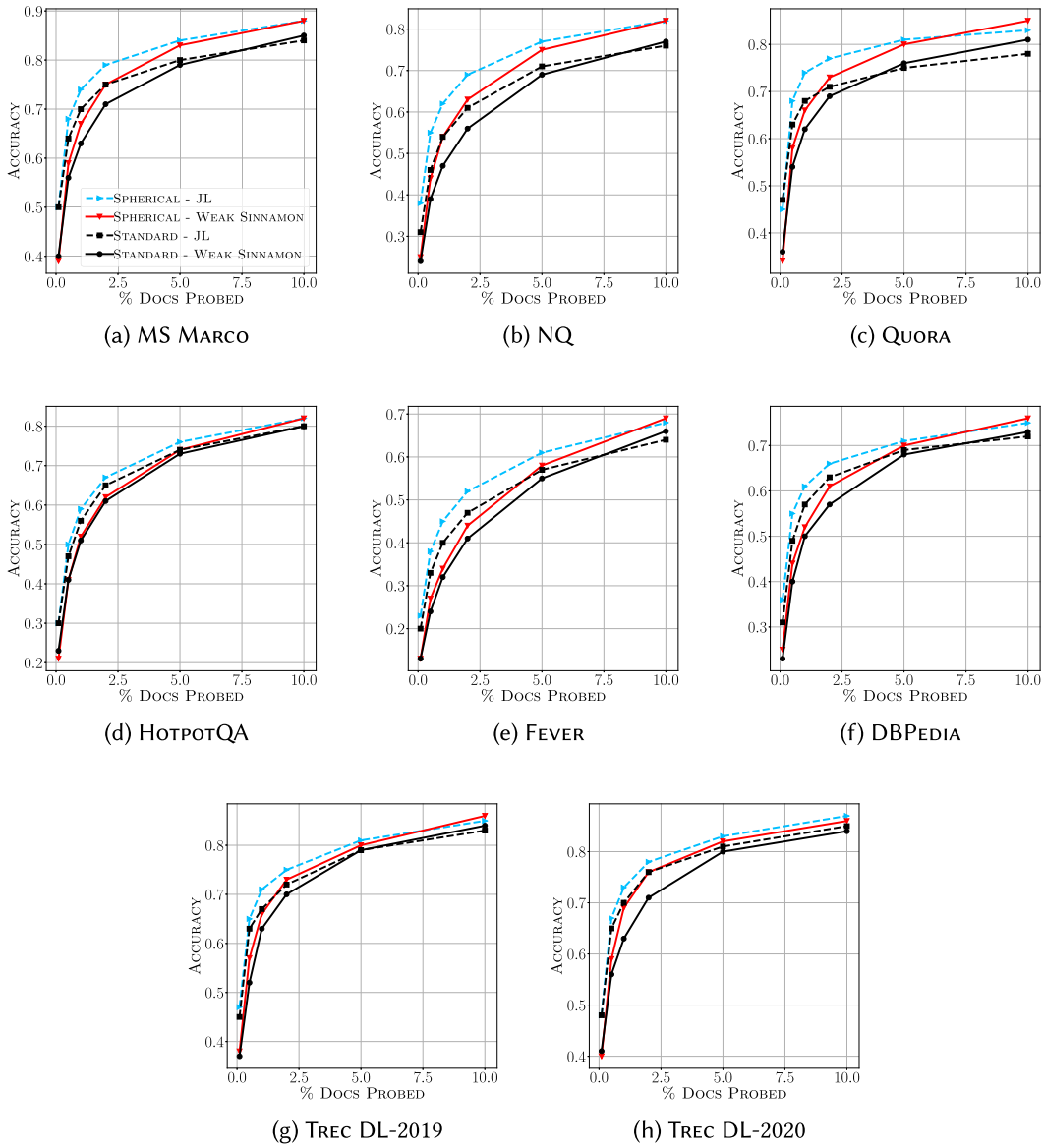
(g) Trec DL-2019 (h) Trec DL-2020

Fig. 4. Top-10 accuracy of Algorithm 2 for Splade vectors versus the number of documents examined (ℓ)—expressed as percentage of the size of the collection—for different clustering algorithms (standard and spherical k-means) and different sketching mechanisms (JL transform and Weak Sinnamon, with sketching size of 1,024). Note that the vertical axis is not consistent across figures.

Additionally, and perhaps unsurprisingly, the difference between the two sketching methods is more pronounced in experiments on the Efficient Splade vector datasets.

6 Clustering as Dynamic Pruning for the Inverted Index

Throughout the previous sections, we simply assumed that once Algorithm 2 has identified the top partitions and accumulated the ℓ-subset of documents to examine, the task of actually finding the



(a) MS Marco (b) NQ (c) Quora

(d) HotpotQA (e) Fever (f) DBPedia

(g) Trec DL-2019 (h) Trec DL-2020

Fig. 5. Top-10 accuracy of Algorithm 2 for Efficient Splade versus the number of documents examined (ℓ).

top-k vectors from that restricted subset would be delegated to a secondary MIPS algorithm, $\mathcal{R}$, which we have thus far ignored. We now wish to revisit $\mathcal{R}$.

There are many ways one could design and implement $\mathcal{R}$ and apply it to the set of partitions $\mathcal{P}_I$ on Line 10 of Algorithm 2. For example, $\mathcal{R}$ may be an exhaustive search—an option we used previously because we argued we were assessing retrieval quality alone and did not concern ourselves with efficiency. As another example, if partitions are stored on separate physical (or logical) retrieval nodes in a distributed system, each node could use an inverted index-based algorithm to find the top-k candidates from their partition of the index. This section proposes a novel alternative for $\mathcal{R}$

Algorithm 3: Constructing a partitioned inverted index

Input: Collection of sparse vectors, $\mathcal{X} \subset \mathbb{R}^N$; Clusters $\mathcal{P}$ obtained from Algorithm 1.
Result: Inverted index, $\mathcal{I}$; Skip list, $\mathcal{S}$.

```
I ←; ∅                                          ▹ Initialize the inverted index
S ←; ∅                                          ▹ Initialize the skip list
for P_i ∈ P do
    SortAscending(P_i)                          ▹ Sort partition by document identifier
    for j ∈ P_i do
        for t ∈ nz(x^(j)) do
            S[t].Append(i, |I[t]|) if it is the first time a document from P_i is recorded in I[t]
            I[t].Append(j, x_t^(j))             ▹ Append document identifier and value to list
        end for
    end for
end for
return I, S
```

that is based on the insight that clustering documents for IVF-based search and dynamic pruning algorithms in the inverted index-based top-k retrieval literature are intimately connected.

6.1 Partitioning Inverted Lists

Consider an *optimal* partitioning $\mathcal{P}^*$ of a collection $\mathcal{X}$ of sparse vectors into P clusters with a set of representative points C^*. In the context of MIPS, optimality implies that for any given sparse query q, we have that the solution to $C_i = \arg\max_{c \in C^*} \langle q, c_i \rangle$ represents the partition $\mathcal{P}_i$ in which we can find the maximizer of $\arg\max_{x \in \mathcal{X}} \langle q, x \rangle$. That implies that, when performing MIPS for a given query, we *dynamically prune* the set of documents in $\mathcal{X} \setminus \mathcal{P}_i$; the procedure is dynamic because i depends on the query vector.

Consider now an inverted index that represents $\mathcal{X}$. Typically, its inverted lists are sorted either by document identifiers or by the "impact" of each document on the final inner product score [70]. The former is consequential for compression [62] and document-at-a-time dynamic pruning algorithms [70], while the latter provides an opportunity for early-termination of score computation—we reiterate that, all of these techniques work only on non-negative vectors or that their extension to negative vectors is non-trivial. But, as we explain, $\mathcal{P}^*$ induces another organization of inverted lists that will enable fast, approximate retrieval in the context of Algorithm 2 for general sparse vectors.

Our construction, detailed in Algorithm 3, is straightforward. At a high-level, when forming an inverted list for a coordinate t, we simply iterate through partitions and add vectors from that partition whose coordinate t is non-zero to the inverted list. As we do so, for each inverted list, we record the *offsets* within the list of each partition in a separate *skip* list. Together the two structures enable us to traverse the inverted lists by only evaluating documents in a given set of partitions.

An alternative way of viewing the joint inverted and skip lists is to think of each inverted list as a set of variable-length segments or blocks, where documents within each block are grouped according to a clustering algorithm.

We must remark on the space complexity of the resulting structure. There are two factors to comment on. First, sorting the inverted lists by partition identifier rather than document identifier may lead to suboptimality for compression algorithms. That is because, the new arrangement of documents may distort the d-gaps (i.e., the difference between two consecutive document identifiers in an inverted list); compression algorithms perform better when d-gaps are smaller and when there

Algorithm 4: Query processing over partitioned inverted lists

Input: Inverted index, $\mathcal{I}$; Skip list, $\mathcal{S}$ obtained from Algorithm 3; Sparse query vector, q; Set of partitions to probe, $\mathcal{P}_\mathcal{I}$ from Algorithm 2.
Result: Top-k vectors.

```
scores ←; ∅                                   ▷ A mapping from documents to scores
for t ∈ nz(q) do
    SLPosition ←; 0                           ▷ Pointer into the skip list S[t]
    for P_i ∈ P_I do
        Advance SLPosition until partition of S[t][SLPosition] matches P_i
        begin ← S[t][SLPosition].Offset
        end ← S[t][SLPosition + 1].Offset
        for (docid, value) ∈ I[t][begin ... end] do
            scores[docid] ← scores[docid] + q_t × value
        end for
    end for
end for
return Top-k documents given scores
```

is a *run* of the same d-gap in the list. But we can address that concern trivially through document identifier reassignment: After partitioning is done by Algorithm 1, we assign new identifiers to documents such that documents within a partition have consecutive identifiers.

The second factor is the additional data stored in $\mathcal{S}$. In the worst case, each inverted list will have documents from every partition. That entails that each $\mathcal{S}[t]$ records P additional pairs of integers consisting of partition identifier and the offset within the inverted list where that partition begins. As such, in the worst case, the inverted index is inflated by the size of storing $2NP$ integers. However, given that P is orders of magnitude smaller than the total number of non-zero coordinates in the collection, and as such $2NP \ll \psi|\mathcal{X}|$, the increase to the total size of the inverted index is mild at worst. Moreover, skip lists can be further compressed using an integer or integer-list codec.

6.2 Query Processing over Partitioned Inverted Lists

When Algorithm 2 gives us a set of partitions $\mathcal{P}_\mathcal{I}$ to probe, we use a simple coordinate-at-a-time scheme to compute the scores of documents in $\bigcup \mathcal{P}_\mathcal{I}$ and return the top-k vectors.

When processing coordinate t and accumulating partial inner product scores, we have two operations to perform. First, we must take the intersection of the skip list and the list of whitelisted partitions: $\mathcal{P}_\mathcal{I} \cap \mathcal{S}[t]$.PartitionId (where the operator PartitionId returns the partition identifier of every element in the skip list). Only then do we traverse the inverted list $\mathcal{I}[t]$ by looking at the offsets of partitions in the intersection set. One possible instance of this procedure is described in Algorithm 4.

6.3 Empirical Evaluation

There are four key properties that we wish to evaluate. Naturally, we care about the efficiency of Algorithms 3 and 4 when we use them as $\mathcal{R}$ in Algorithm 2. But, seeing as the partitioning performed by Algorithm 1 is not guaranteed to be the optimal partitioning $\mathcal{P}^*$, we understand there is a risk of losing retrieval accuracy by probing a fraction of partitions, as demonstrated in Section 5. As such, the second important property is the effectiveness of the methods presented here. We thus report throughput versus accuracy as one tradeoff space of interest.

We also presented Algorithms 3 and 4 as a new dynamic pruning method for the inverted index. To show that for different levels of accuracy, we indeed prune the inverted lists, we additionally report the size of the pruned space as we process queries.

A third factor is the size of the inverted index and the inflation due to (a) the additional data structure that holds skip pointers and (b) the partition centroids produced by Algorithm 1. We also evaluate this aspect, but we do not apply compression anywhere in our evaluation: We consider compression to be orthogonal to this work and only report the overhead.

Finally, we implemented Algorithms 1–4 by enabling parallelism within and across queries. We believe, therefore, it is important to measure the effect of the number of CPU cores on throughput. As such, we present throughput measurements by changing the number of cores we make available to the algorithms.

6.3.1 Baseline Retrieval Algorithm. As argued earlier, we are interested in general sparse vectors, such as those produced by Splade, which exhibit distributional properties that differ from traditional sparse vectors based on lexical models of relevance. It has been noted by others [17, 50] that an exhaustive disjunctive query processing over the inverted index—a method Bruch et al. referred to as LinScan—outpeforms all dynamic pruning-based optimization methods and represents a strong baseline. We therefore use LinScan as our baseline system.

LinScan is a safe algorithm as it evaluates every qualified document (i.e., documents that contain at least one non-zero coordinate of the query vector). But as Bruch et al. show in [17], there is a simple strategy to turn LinScan into an approximate algorithm: By giving the algorithm-a-time budget, we can ask it to process as many coordinates as possible until the budget has been exhausted. At that point, LinScan returns the approximate top-k set according to the accumulated partial inner product scores. We use this variant to obtain approximate top-k sets for comparison with our own approximate algorithms.

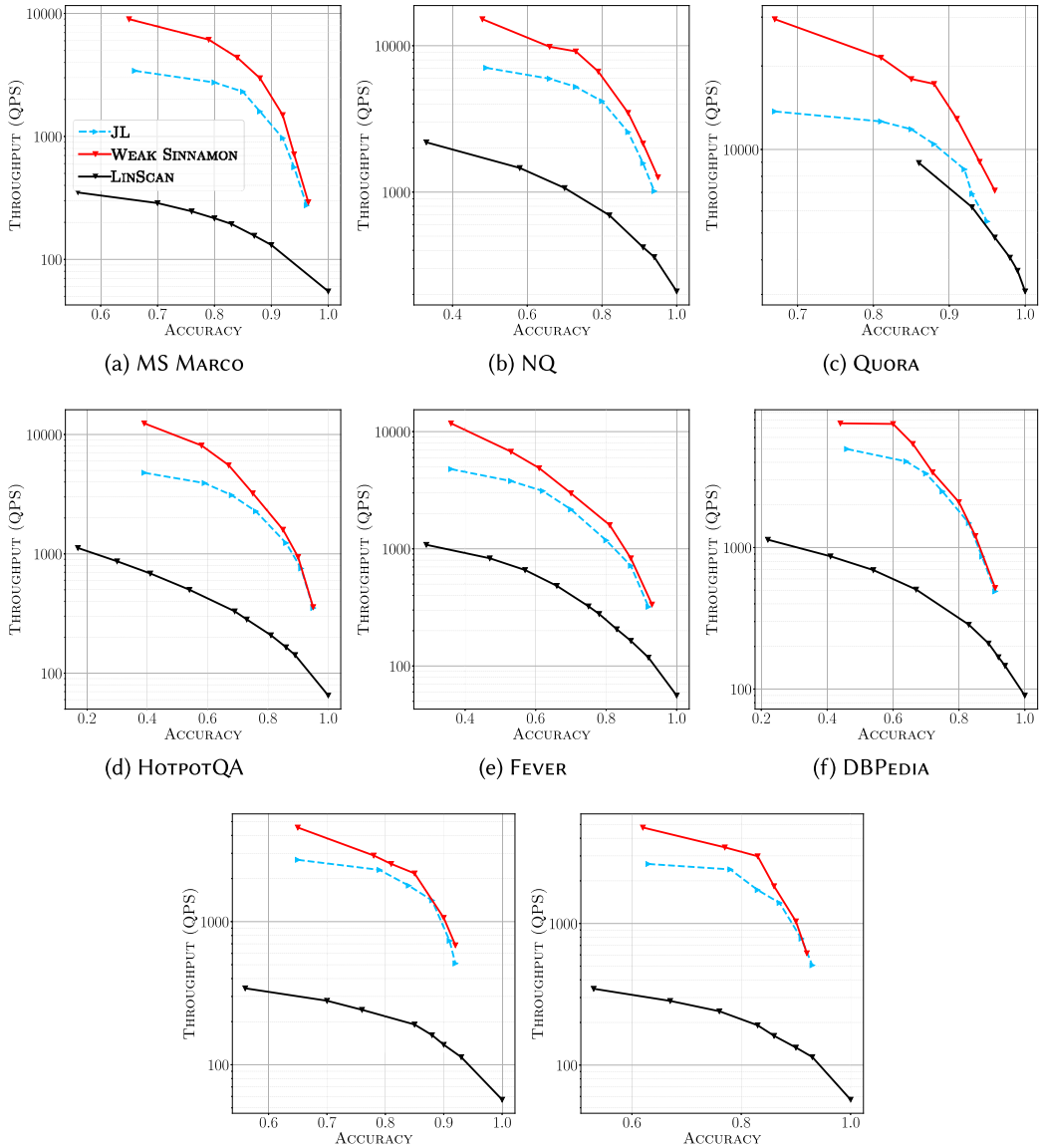
6.3.2 Throughput versus Accuracy. The first topic of evaluation is the tradeoff between throughput and accuracy. We can trade one factor off for the other by adjusting the parameter ℓ in Algorithm 2: A smaller ℓ will result in probing fewer partitions, which in turn leads to faster retrieval but lower quality. Letting ℓ approach the size of the collection, on the other hand, results in the algorithm probing every partition, leading to a slower but higher-quality retrieval.

We tune this knob as we perform top-10 retrieval over our datasets. We use Splade and Efficient Splade vectors as input to the algorithms, sketch them using the JL and Weak Sinnamon transforms, but partition the data only using spherical k-means. The results of our experiments are shown in Figures 6 and 7.

In order to digest the trends, we must recall that the throughput of our retrieval method is affected by two factors: the time it takes to perform inner product of a query vector with cluster centroids, and the time it takes to execute algorithm $\mathcal{R}$ on the subset of partitions identified from the previous step. In the low-recall region, we expect the first factor to make up the bulk of the processing time, while in the high-recall region the cost of executing $\mathcal{R}$ starts to dominate the overall processing time.

That phenomenon is evident in the figures for both Splade and Efficient Splade experiments. That also explains why when sketching is done with Weak Sinnamon, throughput is much better than the JL transform: Weak Sinnamon creates sparse query sketches which lead to faster inner product computation with partition centroids.

What is also clear from our experiments is that our approximate method always compares favorably to the approximate baseline. In fact, for the same desired accuracy, our method often reaches a throughput that is orders of magnitude larger than that of the baseline's. For instance, on MS Marco encoded with Splade, an instance of our algorithm that operates on Weak Sinnamon



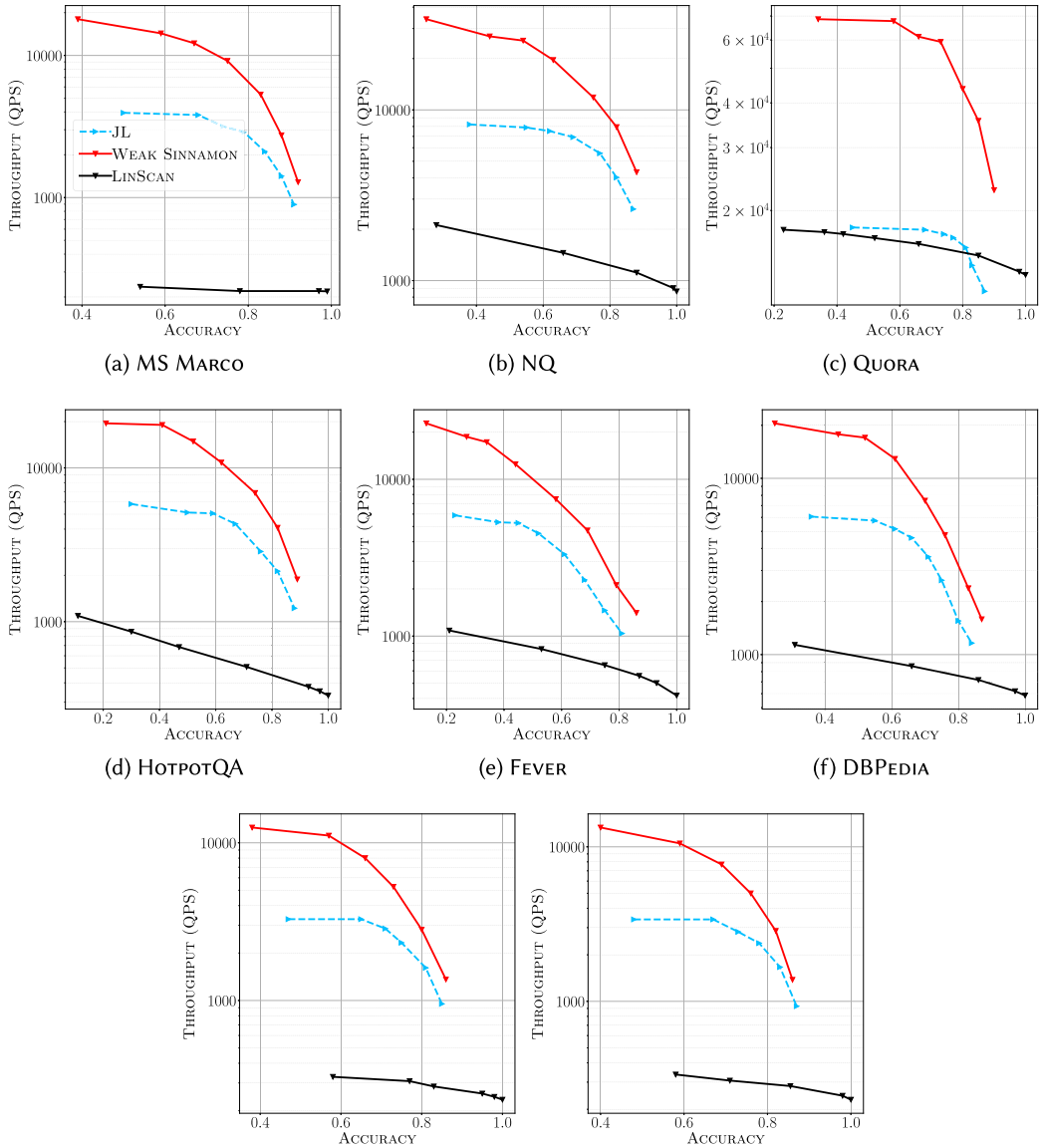
(a) MS Marco (b) NQ (c) Quora

(d) HotpotQA (e) Fever (f) DBPedia

(g) Trec DL-2019 (h) Trec DL-2020

Fig. 6. Throughput (queries/second) versus top-10 retrieval accuracy on Splade-encoded datasets. We limit the experiments to an instance of Algorithm 1 that uses spherical k-means. We set ℓ to 0.1%, 0.5%, 1%, 2%, 5%, and 10% of the size of the collection. Included here is an approximate variant of an exhaustive disjunctive query processor (LinScan). We use 20 CPU cores and repeat each experiment 10 times for a more reliable throughput measurement. Axes are not consistent across figures.

sketches processes queries at an extrapolated rate of approximately 2,000 QPS and delivers 90% accuracy, while the baseline method yields a throughput of roughly 150 QPS. At lower recalls, the gap is substantially wider.



(a) MS Marco (b) NQ (c) Quora

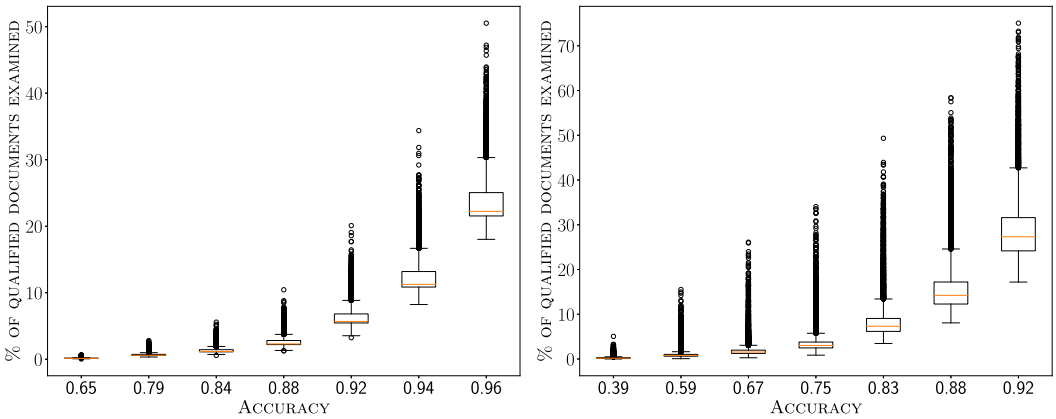
(d) HotpotQA (e) Fever (f) DBPedia

(g) Trec DL-2019 (h) Trec DL-2020

Fig. 7. Throughput versus top-10 retrieval accuracy on Efficient Splade-encoded datasets. Setup is as in Figure 6.

As we require a higher accuracy, all methods become slower. Ultimately, of course, if we set ℓ too high, our algorithms become slower than the exact baseline. That is because, our approximate algorithms have to pay the price of computing inner product with centroids *and* must execute the additional step of intersecting $\mathcal{P}_I$ with the skip lists. We do not show this empirically, however.

6.3.3 Effect of Dynamic Pruning. As we already explained, when we adjust the parameter ℓ in Algorithm 2, we control the number of documents the sub-algorithm $\mathcal{R}$ is allowed to evaluate. While



(a) Splade (b) Efficient Splade

Fig. 8. Percentage of qualified documents (i.e., documents that contain at least one non-zero coordinate of the query) probed versus top-10 accuracy for the MS Marco dataset. In this setup, Algorithm 1 uses Weak Sinnamon along with spherical k-means for partitioning. Note the irregular spacing of the horizontal axes.

we studied the impact of ℓ on efficiency as measured by throughput, here we wish to understand its effect in terms of the amount of pruning it induces. While throughput measurements depend on our specific implementation of Algorithm 4, measuring the portion of documents pruned is implementation-agnostic and, as such, serves as a more definitive measure of efficiency.

To that end, we count, for each query, the actual number of documents evaluated by Algorithm 4 as we gradually increase ℓ. We plot this quantity in Figure 8 for MS Marco from a configuration of our algorithms that uses Weak Sinnamon and spherical k-means. To improve visualization, we show not raw counts, but the percentage of qualified documents—defined, once again, as the number of documents that contain at least one non-zero coordinate of the query—that Algorithm 4 evaluates. That is indicative of how much of the inverted lists the algorithm manages to skip.

As one observes, in the low-recall region, the algorithm probes only a fraction of the inverted lists. On Splade dataset, the algorithm reaches a top-10 accuracy of 0.94 by merely evaluating, on average, about 10% of the total number of documents in the inverted lists. On Efficient Splade, as expected, the algorithm is relatively less effective.

These results are encouraging. It shows the potential that a clustering-based organization of the inverted index has for dynamic pruning in approximate MIPS. Importantly, this method does not require the vectors to follow certain distributions or be non-negative.

6.3.4 Index Size Overhead. As we mentioned earlier, our algorithms add overhead to the index structure required for query processing. If our reference point is the LinScan algorithm with a basic (uncompressed) inverted index, our methods introduce two additional structures: (a) the skip list, $\mathcal{S}$, in Algorithm 3; and, (b) the array of $4\sqrt{|\mathcal{X}|}$ centroids produced by Algorithm 1. We next measure this overhead.

We report our findings in Table 2 for Splade and Efficient Splade vector datasets, measured in GB of space after serialization to disk. We reiterate that, we do not apply compression to the index. That is because there is an array of compression techniques that can be applied to the different parts of the data structure (such as quantization, approximation, and d-gap compression). Choosing any of those would arbitrarily conflate the inflation due to the overhead and the compression rate.

Table 2. Index Sizes in GB

	Method	MS Marco	NQ	Quora	HotpotQA	Fever	DBPedia
Splade	LinScan	8.4	3.1	0.27	5.1	5.9	4.7
	Ours	9.0(+7%)	3.43(+10%)	0.32(+18%)	5.5(+8%)	6.3(+7%)	5.0(+6%)
E. Splade	LinScan	12	4.2	0.27	4.9	5.7	4.6
	Ours	13(+8%)	4.7(+12%)	0.37(+37%)	5.4(+10%)	6.2(+9%)	5.0(+9%)

The index in LinScan is made up of an inverted index with document identifiers and floating point values (uncompressed). The index in our method stores $4\sqrt{|\chi|}$ centroids from the application of spherical k-means to Weak Sinnamon for dataset χ, an inverted index with the same size as LinScan, and the skip list structure $\mathcal{S}$.

We observe that the overhead of our method on larger datasets is relatively mild. The increase in size ranges from 6% to 10% (Quora excluded) for the Splade-encoded datasets and a slightly wider and large range for Efficient Splade-encoded datasets.

6.3.5 Effect of Parallelism. We conclude the empirical evaluation of our approximate algorithm by repeating the throughput-accuracy experiments with a different number of CPUs. In our implementation, we take advantage of access to multiple processors by parallelizing the computation of inner product between queries and centroids (in Algorithm 2) for each query, in addition to distributing the queries themselves to the available CPUs. As a result of this concurrent paradigm, we expect that, by reducing the number of CPUs available to the algorithm, throughput will be more heavily affected in low-recall regions (when ℓ is small).

Figure 9 shows the results of these experiments on the Splade- and Efficient Splade-encoded MS Marco dataset. The figures only include a configuration of our algorithms with spherical k-means and Weak Sinnamon. It is easy to confirm that our hypothesis from above holds: In low-recall regions where computation is heavily dominated by the cost of computing inner product with centroids, throughput decreases considerably as we reduce the number of CPUs.

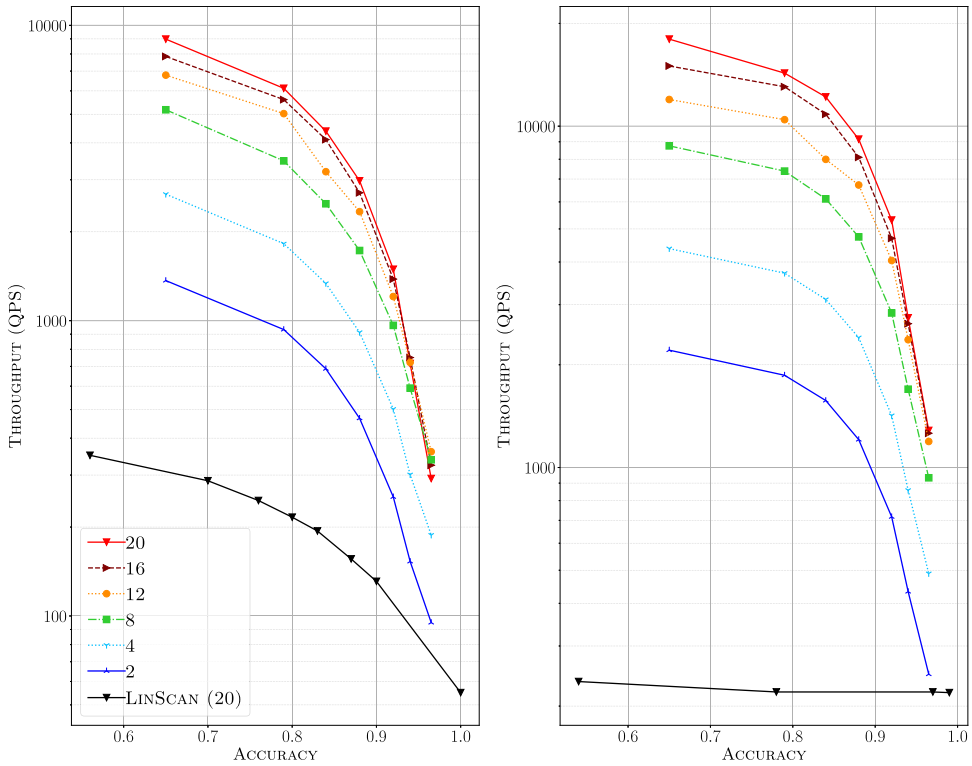
7 Toward A Unified Framework for MIPS

Sections 4–6 presented a complete instance of Algorithm 2 for IVF-based MIPS over *sparse* vectors. But, recall that, we borrowed the idea of IVF-based search from the dense MIPS literature. So it is only natural to pose the following question: Now that we have an arbitrarily-accurate IVF algorithm for sparse vectors, can we extend it to *hybrid* vectors in $\mathbb{R}^{m+N}$? In this section, we unpack that question superficially and investigate possible directions at a high-level to explore the feasibility and benefits of such an approach. First, however, let us motivate this question.

7.1 Motivation

We described the changing landscape of retrieval in Section 1. From lexical-semantic search to multi-modal retrieval, for many emerging applications the ability to conduct MIPS over hybrid vectors efficiently and effectively is a requisite. One viable approach to searching over a collection of hybrid vectors $\mathcal{X}$ is to simply decompose the process into separate MIPS questions, one over the dense subspace $\mathcal{X}^d$ and the other over the sparse one $\mathcal{X}^s$, followed by an aggregation of the retrieved sets. Indeed this approach has become the *de facto* solution for hybrid vector retrieval [13, 18].

The two-stage retrieval system works as follows: When a hybrid query vector $q \in \mathbb{R}^{m+N}$ arrives and the retrieval system is expected to return the top-k documents, commonly, q^d is sent to the dense MIPS system with a request for the top $k' \geq k$ vectors, and q^s to the sparse retrieval component



(a) Splade (b) Efficient Splade

Fig. 9. Effect of changing the number of CPUs on throughput. The figures illustrate these measurements for MS Marco, and a particular configuration of our algorithm that uses spherical k-means over Weak Sinnamon sketches. We include LinScan executed on 20 CPUs from Figures 6 and 7 as a point of reference.

with a similar request. Documents in the union of the two sets are subsequently scored and reranked to produce an approximate set of top-k vectors, $\tilde{\mathcal{S}}$:

$$\tilde{\mathcal{S}} = \underset{x \in \mathcal{S}^d \cup \mathcal{S}^s}{\overset{(k)}{\arg\max}} \langle q, x \rangle, \tag{10}$$

$$\mathcal{S}^d = \underset{x \in \mathcal{X}}{\overset{(k')}{\arg\max}} \langle q^d, x^d \rangle \text{ and, } \mathcal{S}^s = \underset{x \in \mathcal{X}}{\overset{(k')}{\arg\max}} \langle q^s, x^s \rangle. \tag{11}$$

Let us set aside the effectiveness of the setup above for a moment and consider its complexity from a systems standpoint. It is clear that, both for researchers and practitioners, studying and creating two disconnected, incompatible systems adds unwanted costs. For example, systems developers must take care to keep all documents in sync between the two indexes at all times. Reasoning about the (mis)behavior of the retrieval system, as another example, requires investigating one layer of indirection and understanding the processes leading to two separate retrieved sets. These collectively pose a challenge to systems researchers, and add difficulty to operations in production. Furthermore, it is easy to see that the least scalable of the two systems dictates or shapes the overall latency and throughput capacity.

Even if we accepted the cost of studying two separate systems or deemed it negligible, and further decided scalability is not a concern, it is not difficult to show that such a heterogeneous

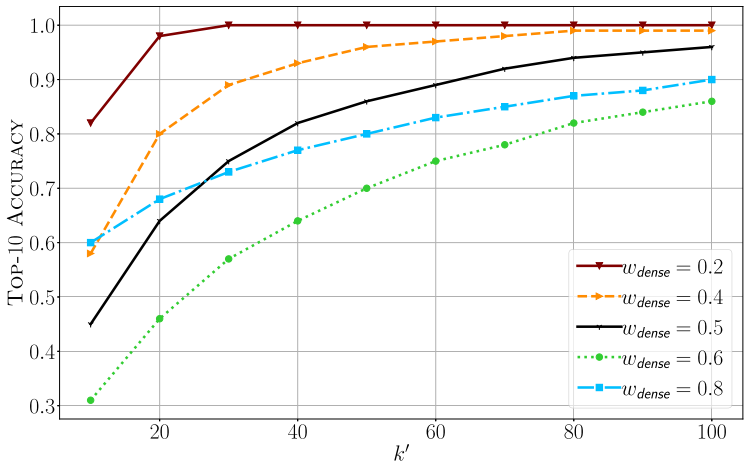


Fig. 10. Top-10 accuracy of the two-stage retrieval system for hybrid vectors. We retrieve k' candidates from each sub-system and rerank them to find the top-10 set. We prepare the hybrid vectors by first normalizing the dense and sparse parts separately, then constructing query vectors as follows: $q = w_{dense}q^d + (1 - w_{dense})q^s$, where q^d and q^s are sampled from the data distribution. In effect, w_{dense} shifts the $\ell 2$ mass from the sparse to the dense subspace, giving more importance to one subspace over the other during retrieval.

design may prove wasteful or outright ineffective in the general case. More concretely, depending on how the $\ell 2$ mass of the query and document vectors is split between the dense subspace and the sparse subspace, the two sub-systems involved may have to resort to a large k' in order to ensure an accurate final retrieved set at rank k.

While the phenomenon above is provable, we demonstrate its effect by a simple (though contrived) experiment. We generate a collection of 100,000 documents and 1,000 queries. Each vector is a hybrid of a dense and a sparse vector. The dense vectors are in $\mathbb{R}^{64}$, with each coordinate drawing its value from the exponential distribution (with scale 0.5). The sparse vectors are in $\mathbb{R}^{1,000}$ with an average of $\psi = 16$ non-zero coordinates, where non-zero values are drawn from the exponential distribution (scale 0.5). We use different seeds for the pseudo-random generator when creating document and query vectors.

In order to study how the ratio of $\ell 2$ mass between dense and sparse subspaces affects retrieval quality, we first normalize the generated dense and sparse vectors separately. During retrieval, we amplify the dense part of the query vector by a weight between 0 and 1 and multiply the sparse part by one minus that weight. In the end, we are performing retrieval for a query vector q that can be written as $w_{dense}q^d \oplus (1 - w_{dense})q^s$. By letting w_{dense} sweep the unit interval, we simulate a shift of the $\ell 2$ mass of the hybrid vector from the sparse to the dense subspace.

Over the generated collection, we conduct *exact* retrieval using exhaustive search and obtain the top $k = 10$ vectors for each query by maximizing the inner product. We then use the two-stage design by asking each sub-system to return the (exact) top k' vectors for $k' \in [100]$, and reranking the union set to obtain the final top $k = 10$ documents. We then measure the top-k accuracy of the two-stage architecture.

Figure 10 plots accuracy versus k' for different values of w_{dense}. It is easy to see that, as one subspace becomes more important than the other, the retrieval quality too changes. Importantly, a larger k' is often required to attain a high-accuracy.

The factors identified in this section—systems complexity, scalability bottleneck, and the sub-optimality of retrieval quality—nudge us in the direction of a unified framework for MIPS.

Algorithm 5: Indexing of hybrid vectors

Input: Collection $\mathcal{X}$ of hybrid vectors in $\mathbb{R}^{m+N}$; Number of clusters, P; Random projector, $\phi : \mathbb{R}^N \to \mathbb{R}^n$ where $n \ll N$; Clustering algorithm Cluster that returns partitions of input data and their representatives.
Result: Cluster assignments $\mathcal{P}_i = \{j \mid x^{(j)} \in \text{ Partition } i\}$ and cluster representatives C_i's.

$\tilde{\mathcal{X}} \leftarrow \{x^d \oplus \phi(x^s) \mid x^d \oplus x^s \in \mathcal{X}\}$
Partitions, Representatives $\leftarrow$ Cluster$(\tilde{\mathcal{X}}; P)$
$\mathcal{P}_i \leftarrow \{j \mid \tilde{x}^{(j)} \in \text{Partitions}[i]\}, \quad \forall 1 \le i \le P$
$C_i \leftarrow \text{Representatives}[i], \quad \forall 1 \le i \le P$
return $\mathcal{P}$ and C

Algorithm 6: Retrieval of hybrid vectors

Input: Hybrid query vector, $q \in \mathbb{R}^{m+N}$; Clusters and representatives, $\mathcal{P}, \mathcal{C}$ obtained from Algorithm 5; random projector $\phi : \mathbb{R}^N \to \mathbb{R}^n$; Number of data points to examine, $\ell \le |\mathcal{X}|$ where $|\mathcal{X}|$ denotes the size of the collection; hybrid MIPS sub-algorithm $\mathcal{R}$.
Result: Approximate set of top-k vectors that maximize inner product with q.

$\tilde{q} \leftarrow q^d \oplus \phi(q^s)$
SortedClusters $\leftarrow$ **SortDescending**$(\mathcal{P}$ by $\langle \tilde{q}, C_i \rangle)$
TotalSize $\leftarrow 0$
$\mathcal{I} \leftarrow; \emptyset$ ▹ `Records the index of the partitions` $\mathcal{R}$ `should probe.`
for $\mathcal{P}_{\pi_i} \in$ SortedClusters **do**
  $\mathcal{I} \leftarrow \pi_i$
  TotalSize $\leftarrow$ TotalSize $+ |\mathcal{P}_{\pi_i}|$
  break if TotalSize $\ge \ell$
end for
return Top-k vectors from partitions $\mathcal{P}_{\mathcal{I}} \triangleq \{\mathcal{P}_i \mid i \in \mathcal{I}\}$ w.r.t $\langle q, \cdot \rangle$ using $\mathcal{R}$

7.2 IVF MIPS for Hybrid Vectors

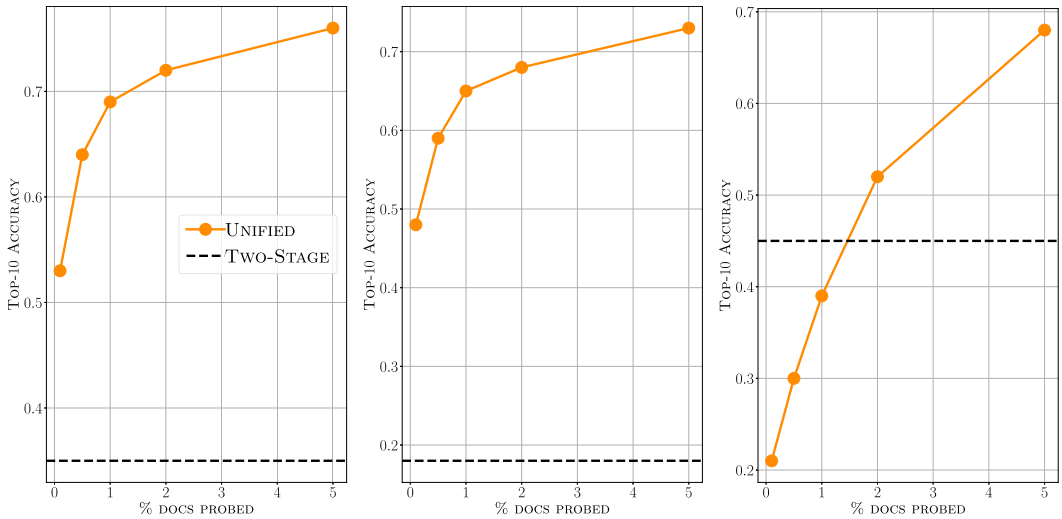
We present a simple extension of the IVF indexing and retrieval duo of Algorithms 1 and 2 to generalize the logic to hybrid vectors. This is shown in Algorithms 5 and 6, where the only two differences with the original algorithms are that (a) sketching is applied only to the sparse portion of vectors to form new vectors in $\mathbb{R}^{m+n}$ instead of $\mathbb{R}^{m+N}$, and (b) that the sub-algorithm $\mathcal{R}$ is assumed to carry out top-k retrieval over hybrid vectors from a given set of partitions.

In this section, we only verify the viability of the extended algorithms and leave an in-depth investigation of the proposal to future work. As such, we use exhaustive search as the sub-algorithm $\mathcal{R}$ and acknowledge that any observations made using such an algorithm only speaks to the effectiveness of the method and not its efficiency.

7.3 Empirical Evaluation

Let us repeat the experiment from Section 7.1 on synthetic vectors and compare the two-stage retrieval process with the unified framework in terms of retrieval accuracy. To that end, we design the following protocol.

First, we perform exact MIPS using exhaustive search over the hybrid collection of vectors. The set of top-k documents obtained in this way make up the ground-truth for each query.



(a) $w_{dense} = 0.2$ (b) $w_{dense} = 0.5$ (c) $w_{dense} = 0.8$

Fig. 11. Top-10 accuracy over hybrid vectors as a function of the percentage of documents probed. w_{dense} controls how much of the $\ell 2$ mass of a hybrid vector is concentrated in its dense subspace. We also plot the performance of the two-stage system where each system returns the set of top-k' documents according to sparse or dense inner product scores, such that the size of the union of the two sets is roughly k.

Next, we consider the two-stage system. We retrieve through exhaustive search the exact set of top-k' (for a large k') documents according to their sparse inner product, and another (possibly overlapping) set by their dense inner product. From the two ranked lists, we accumulate enough documents from the top such that the size of the resulting set is roughly equal to k. In this way, we can measure the top-k accuracy of the two-stage system against the ground-truth.

Finally, we turn to the unified framework. We use the JL transform to reduce the dimensionality of sparse vectors, and spherical k-means to partition the vectors. We then proceed as usual and measure top-k accuracy for different values of ℓ.

From these experiments, we wish to understand whether and when the accuracy of the unified framework exceeds the accuracy of the two-stage setup. If the unified system is able to surpass the accuracy of the two-stage system by examining a relatively small portion of the collection—a quantity controlled through ℓ—then that is indicative of the viability of the proposal. Indeed, as Figure 11 shows, the unified system almost always reaches a top-10 accuracy that is higher than the two-stage system's by evaluating less than 2% of the collection.

8 Discussion and Conclusion

We began this research with a simple question: Can we apply dense MIPS algorithms to sparse vectors? That led us to investigate different dimensionality reduction techniques for sparse vectors as a way to contain the curse of dimensionality. We showed, e.g., that the JL transform and Sinnamon behave differently on sparse vectors and can preserve inner product to different degrees. We also thoroughly evaluated the effect of clustering on sparse MIPS in the context of an IVF-based retrieval system. Coupling dimensionality reduction with clustering realized an effective IVF system for sparse vectors, summarized in Algorithms 1 and 2.

The protocol is easy to describe and is as follows. We sketch sparse vectors into a lower-dimensional (dense or sparse) subspace in a first step. We then apply clustering to the sketches and

partition the data into a predetermined number of clusters, each identified by a representative (e.g., a centroid). When the system is presented with a query, we sketch the query (asymmetrically) and identify the top partitions by taking inner product between the query and cluster representatives. We then execute a secondary sub-algorithm to perform MIPS on the restricted subset of document vectors.

In our presentation of the material above, we observed a strong, natural connection between clustering for IVF and dynamic pruning methods for inverted indexes. We developed that insight into an inverted index-based algorithm that could serve as the sub-algorithm in the above search procedure. Importantly, the algorithm organizes documents within an inverted list by partition identifier—rather than the conventional arrangement by document identifier or impact score. Such an organization, coupled with skip pointers, enables the algorithm to only search over the subset of documents that belong to the top partitions determined by the IVF method. Crucially, the algorithm is agnostic to the vector distribution and admits real-valued vectors.

Finally, we discussed how our proposal leads to a unified retrieval framework for hybrid vectors. By sketching the sparse sub-vectors and constructing an IVF index for the transformed hybrid vectors, we showed that it is possible to achieve better recall than a two-stage system, where dense and sparse sub-vectors are handled separately. The added advantage of the unified approach is that its accuracy remains robust under different vector distributions, where the mass shifts from the dense to the sparse subspace.

We limited our discussion of hybrid MIPS to synthetic vectors as we were only interested in the viability of this byproduct of our primary research question. We acknowledge that we have only scratched the surface of retrieval over hybrid vectors. There are a multitude of open questions within the unified regime that warrant further investigation, including many minor but practical aspects of the framework that we conveniently ignored in our high-level description. We leave those as future work.

We believe our investigation of MIPS for sparse (and hybrid vectors) provides many opportunities for IR researchers. One line of research most immediately affected by our proposal is sparse representation learning. Models such as Splade are not only competitive on in- and out-of-domain tasks, they also produce inherently-interpretable representations of text—a desirable behavior in many production systems. However, sparse embeddings have, by and large, been tailored to existing retrieval regimes. For example, Efficient Splade learns sparser queries for better latency. uniCoil [41] collapses term representations of Coil [29] to a scalar for compatibility with inverted indexes. We claim that our proposed regime is a step toward removing such constraints, enabling researchers to explore sparse representations without much restraint, leading to a potentially different behavior. As we observe in Figures 4 and 5, e.g., Splade vectors are more amenable to clustering than Efficient Splade, and may even prove more efficient within the new framework. That is good news as there is evidence suggesting that Splade is more effective than its other variant on out-of-domain data [40].

Another related area of research that can benefit from our proposed regime is multi-modal and multimedia retrieval. Because our framework is agnostic to the distribution of the hybrid vectors, it is entirely plausible to formulate the multi-modal problem as MIPS over hybrid vectors, especially when one of the modes involves textual data, is data that is partially sparse, or where one may need to engineer (sparse) features to augment dense embeddings.

Appendices

A Proof of Theorem 4.2

Fix two vectors u and $v \in \mathbb{R}^N$. Define $Z_{\text{SKETCH}} = \langle \phi(u), \phi(v) \rangle$ as the random variable representing the inner product of sketches of size n, prepared using the projection $\phi(u) = Ru$, with

$R \in \{-1/\sqrt{n}, 1/\sqrt{n}\}^{n \times N}$. Z_{SKETCH} is an unbiased estimator of $\langle u, v \rangle$. Its distribution tends to a Gaussian with variance:

$$\frac{1}{n}\left(\|u\|_2^2\|v\|_2^2 + \langle u, v\rangle^2 - 2\sum_i u_i^2 v_i^2\right).$$

Proof. Consider the random variable $Z = \left(\sum_j R_j u_j\right)\left(\sum_k R_k v_k\right)$, where R_i's are Rademacher random variables. It is clear that nZ is the product of the sketch coordinate i (for any i): $\phi(u)_i \phi(v)_i$.

We can expand the expected value of Z as follows:

$$\begin{aligned}
\mathbb{E}[Z] &= \mathbb{E}\left[\left(\sum_j R_j u_j\right)\left(\sum_k R_k v_k\right)\right] \\
&= \mathbb{E}\left[\sum_i R_i^2 u_i v_i\right] + \mathbb{E}\left[\sum_{j \neq k} R_j R_k u_j v_k\right] \\
&= \sum_i u_i v_i \underbrace{\mathbb{E}[R_i^2]}_{1} + \sum_{j \neq k} u_j v_k \underbrace{\mathbb{E}[R_j R_k]}_{0} \\
&= \langle u, v\rangle.
\end{aligned}$$

The variance of Z can be expressed as follows:

$$Var(Z) = \mathbb{E}[Z^2] - \mathbb{E}[Z]^2 = \mathbb{E}\left[\left(\sum_j R_j u_j\right)^2 \left(\sum_k R_k v_k\right)^2\right] - \langle u, v\rangle^2.$$

We have the following:

$$\mathbb{E}\left[\left(\sum_j R_j u_j\right)^2 \left(\sum_k R_k v_k\right)^2\right] = \mathbb{E}\left[\left(\sum_i u_i^2 + \sum_{i \neq j} R_i R_j u_i u_j\right)\left(\sum_k v_k^2 + \sum_{k \neq l} R_k R_l v_k v_l\right)\right] \tag{12}$$

$$= \|u\|_2^2\|v\|_2^2 + \underbrace{\mathbb{E}\left[\sum_i u_i^2 \sum_{k \neq l} R_k R_l v_k v_l\right]}_{0} + \underbrace{\mathbb{E}\left[\sum_k v_k^2 \sum_{i \neq j} R_i R_j u_i u_j\right]}_{0} + \mathbb{E}\left[\sum_{i \neq j} R_i R_j u_i u_j \sum_{k \neq l} R_k R_l v_k v_l\right]. \tag{13}$$

The last term can be decomposed as follows:

$$\begin{aligned}
&\mathbb{E}\left[\sum_{i \neq j \neq k \neq l} R_i R_j R_k R_l u_i u_j v_k v_l\right] \\
&+ \mathbb{E}\left[\sum_{i=k, j \neq l \vee i \neq k, j=l} R_i R_j R_k R_l u_i u_j v_k v_l\right] \\
&+ \mathbb{E}\left[\sum_{i \neq j, i=k, j=l \vee i \neq j, i=l, j=k} R_i R_j R_k R_l u_i u_j v_k v_l\right].
\end{aligned}$$

The first two terms are 0 and the last term can be rewritten as follows:

$$2\mathbb{E}\left[\sum_i u_i v_i \left(\sum_j u_j v_j - u_i v_i\right)\right] = 2\langle u, v\rangle^2 - 2\sum_i u_i^2 v_i^2. \tag{14}$$

We now substitute the last term in Equation (13) with Equation (14) to obtain

$$Var(Z) = \|u\|_2^2 \|v\|_2^2 + \langle u, v\rangle^2 - 2\sum_i u_i^2 v_i^2. \tag{15}$$

Observe that $Z_{\text{SKETCH}} = 1/n \sum_i \phi(u)_i \phi(v)_i$ is the sum of independent, identically distributed random variables. Furthermore, for bounded vectors u and v, the variance is finite. By the application of the Central Limit Theorem, we can deduce that the distribution of Z_{SKETCH} tends to a normal distribution with the stated expected value. Noting that $Var(Z_{\text{SKETCH}}) = 1/n^2 \sum_i Var(Z)$ gives the desired variance. □

B Proof of Theorem 4.3

Fix a query vector $q \in \mathbb{R}^N$ and let X be a random vector drawn according to the following probabilistic model. Coordinate i, X_i, is non-zero with probability $p_i > 0$ and, if it is non-zero, draws its value from a distribution with mean μ and variance σ^2. $Z_{\text{SKETCH}} = \langle \phi(q), \phi(X)\rangle$, with $\phi(u) = Ru$ and $R \in \left\{-1/\sqrt{n}, 1/\sqrt{n}\right\}^{n\times N}$, has expected value $\mu \sum_i p_i q_i$ and variance:

$$\frac{1}{n}\left[(\mu^2 + \sigma^2)\left(\|q\|_2^2 \sum_i p_i - \sum_i p_i q_i^2\right) + \mu^2\left(\left(\sum_i q_i p_i\right)^2 - \sum_i (q_i p_i)^2\right)\right].$$

Proof. It is easy to see that:

$$\mathbb{E}[Z_{\text{SKETCH}}] = \sum_i q_i \mathbb{E}[X_i] = \mu \sum_i p_i q_i.$$

As for variance, we start from Theorem 4.2 and arrive at the following expression:

$$\frac{1}{n}\left(\|q\|_2^2 \mathbb{E}[\|X\|_2^2] + \mathbb{E}[\langle q, X\rangle^2] - 2\sum_i q_i^2 \mathbb{E}[X_i^2]\right), \tag{16}$$

where the expectation is with respect to X. Let us consider the terms inside the parentheses one by one. The first term becomes:

$$\begin{aligned}\|q\|_2^2 \mathbb{E}[\|X\|_2^2] &= \|q\|_2^2 \sum_i \mathbb{E}[X_i^2] \\ &= \|q\|_2^2 (\mu^2 + \sigma^2) \sum_i p_i.\end{aligned}$$

The second term reduces to:

$$\begin{aligned}\mathbb{E}[\langle q, X\rangle^2] &= \mathbb{E}\big[\langle q, X\rangle\big]^2 + Var\big[\langle q, X\rangle\big] + \\ &= \mu^2\left(\sum_i q_i p_i\right)^2 + \sum q_i^2\big[(\mu^2 + \sigma^2)p_i - \mu^2 p_i^2\big] \\ &= \mu^2\left(\left(\sum_i q_i p_i\right)^2 - \sum_i q_i^2 p_i^2\right) + \sum_i q_i^2 p_i (\mu^2 + \sigma^2).\end{aligned}$$

Finally, the last term breaks down to:

$$-2\sum_i q_i^2\mathbb{E}[X_i^2] = -2\sum_i q_i^2(\mu^2+\sigma^2)p_i$$
$$= -2(\mu^2+\sigma^2)\sum_i q_i^2 p_i.$$

Putting all these terms back into Equation (16) yields the desired expression for variance. □

C Proof of Theorem 4.5

Let X be a random vector drawn according to the following probabilistic model. Coordinate i, X_i, is non-zero with probability $p_i > 0$ and, if it is non-zero, draws its value from a distribution with PDF ϕ and CDF Φ. Then:

$$\mathbb{P}[\overline{X}_{\pi(i)} - X_i \leq \delta] \approx (1-p_i)\left(e^{-\frac{1}{m}(1-\Phi(\delta))\sum_{j\neq i}p_j}\right) + p_i\int e^{-\frac{1}{m}(1-\Phi(\alpha+\delta))\sum_{j\neq i}p_j}\phi(\alpha)d\alpha.$$

Proof. Decomposing the probability of the event by conditioning on whether X_i is "active" (i.e., its value is drawn from the distribution with PDF ϕ) or "inactive" (i.e., it is 0), we arrive at:

$$\mathbb{P}\left[\overline{X}_{\pi(i)} - X_i \leq \delta\right] = p_i\mathbb{P}\left[\overline{X}_{\pi(i)} - X_i \leq \delta \mid X_i \textit{ is active}\right] + (1-p_i)\mathbb{P}\left[\overline{X}_{\pi(i)} \leq \delta \mid X_i \textit{ is inactive}\right].$$

The term conditioned on X_i being active is given by Theorem 5.4 of [17]. The other event involving an inactive X_i happens when all values that collide with $\overline{X}_{\pi(i)}$ are less than or equal to δ. This event is equivalent to the event that every active coordinate whose value is greater than δ maps to any sketch coordinate except i. Using this alternative event, we can write the conditional probability as follows:

$$\left(1-\frac{1}{m}\right)^{(1-\Phi(\delta))\sum_{j\neq i}p_j} \approx e^{-\frac{1}{m}(1-\Phi(\delta))\sum_{j\neq i}p_j},$$

where we used $e^{-1} \approx (1-1/m)^m$. That completes the proof. □

References

[1] Nir Ailon and Bernard Chazelle. 2006. Approximate Nearest Neighbors and the Fast Johnson-Lindenstrauss Transform. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*. 557–563.

[2] Nir Ailon and Bernard Chazelle. 2009. The Fast Johnson–Lindenstrauss Transform and Approximate Nearest Neighbors. *SIAM Journal on Computing* 39, 1 (2009), 302–322.

[3] Nir Ailon and Edo Liberty. 2011. An Almost Optimal Unrestricted Fast Johnson-Lindenstrauss Transform. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms*. 185–191.

[4] Nir Ailon and Edo Liberty. 2013. An Almost Optimal Unrestricted Fast Johnson-Lindenstrauss Transform. *ACM Transactions on Algorithms* 9, 3, Article 21 (June 2013), 12 pages.

[5] David Arthur and Sergei Vassilvitskii. 2007. K-Means++: The Advantages of Careful Seeding. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*. 1027–1035.

[6] Nima Asadi. 2013. *Multi-Stage Search Architectures for Streaming Documents*. University of Maryland.

[7] Nima Asadi and Jimmy Lin. 2013. Effectiveness/Efficiency Tradeoffs for Candidate Generation in Multi-Stage Retrieval Architectures. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 997–1000.

[8] Alex Auvolat, Sarath Chandar, Pascal Vincent, Hugo Larochelle, and Yoshua Bengio. 2015. Clustering is efficient for approximate maximum inner product search. arXiv:1507.05910 [cs.LG]

[9] 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]

[10] Richard Baraniuk, M. Davenport, Ronald DeVore, and M. Wakin. 2006. The Johnson-Lindenstrauss Lemma Meets Compressed Sensing. *IEEE Transactions on Information Theory* 52 (January 2006), 1289–1306.

[11] 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*. 426–434.

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

[13] 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.

[14] Sebastian Bruch, Claudio Lucchese, and Franco Maria Nardini. 2022. ReNeuIR: Reaching Efficiency in Neural Information Retrieval. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 3462–3465.

[15] 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.

[16] Sebastian Bruch, Joel Mackenzie, Maria Maistro, and Franco Maria Nardini. 2023. ReNeuIR at SIGIR 2023: The Second Workshop on Reaching Efficiency in Neural Information Retrieval. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 3456–3459.

[17] 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.

[18] Tao Chen, Mingyang Zhang, Jing Lu, Michael Bendersky, and Marc Najork. 2022. Out-of-Domain Semantics to the Rescue! Zero-Shot Hybrid Retrieval Models. In *Proceedings of the 44th European Conference on IR Research*. 95–110.

[19] 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*. 201–210.

[20] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2021. Overview of the Trec 2020 deep learning track. arXiv:2102.07662 [cs.IR]

[21] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2020. Overview of the Trec 2019 deep learning track. arXiv:2003.07820 [cs.IR]

[22] 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*. 1533–1536.

[23] 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*. 4171–4186.

[24] Inderjit S. Dhillon and Dharmendra S. Modha. 2001. Concept Decompositions for Large Sparse Text Data Using Clustering. *Machine Learning* 42, 1 (January 2001), 143–175.

[25] Constantinos Dimopoulos, Sergey Nepomnyachiy, and Torsten Suel. 2013. Optimizing Top-k Document Retrieval Strategies for Block-Max Indexes. In *Proceedings of the 6th ACM International Conference on Web Search and Data Mining*. 113–122.

[26] 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*. 993–1002.

[27] 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*. 2353–2359.

[28] 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*. 2288–2292.

[29] Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021. COIL: Revisit Exact Lexical Match in Information Retrieval with Contextualized Inverted List. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics*. 3030–3042.

[30] Bob Goodwin, Michael Hopcroft, Dan Luu, Alex Clemmer, Mihaela Curmei, Sameh Elnikety, and Yuxiong He. 2017. BitFunnel: Revisiting Signatures for Search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 605–614.

[31] Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. 2020. Accelerating Large-Scale Inference with Anisotropic Vector Quantization. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research)*. 3887–3896.

[32] Qiang Huang, Jianlin Feng, Yikai Zhang, Qiong Fang, and Wilfred Ng. 2015. Query-Aware Locality-Sensitive Hashing for Approximate Nearest Neighbor Search. *Proceedings of the VLDB Endowment* 9, 1 (September 2015), 1–12.

[33] Piotr Indyk and Rajeev Motwani. 1998. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*. 604–613.

[34] Hervé 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.

[35] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. Billion-Scale Similarity Search with GPUs. *IEEE Transactions on Big Data* 7 (2021), 535–547.

[36] William B. Johnson and Joram Lindenstrauss. 1984. Extensions of Lipschitz mappings into Hilbert space. *Contemporary Mathematics* 26 (1984), 189–206.

[37] 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*.

[38] Hyunjoong Kim, Han Kyul Kim, and Sungzoon Cho. 2020. Improving Spherical k-Means for Document Clustering: Fast Initialization, Sparse Centroid Projection, and Efficient Cluster Labeling. *Expert Systems with Applications* 150 (2020), 113288.

[39] Saar Kuzi, Mingyang Zhang, Cheng Li, Michael Bendersky, and Marc Najork. 2020. Leveraging semantic and lexical matching to improve the recall of document retrieval systems: A hybrid approach. arXiv:2010.01195 [cs.IR]

[40] 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*. 2220–2226.

[41] 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]

[42] Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. 2021. Pretrained transformers for text ranking: BERT and beyond. arXiv:2010.06467 [cs.IR]

[43] 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*. 301–304.

[44] Jie Liu, Xiao Yan, Xinyan Dai, Zhirong Li, James Cheng, and Ming-Chang Yang. 2019. Understanding and improving proximity graph based maximum inner product search. arXiv:1909.13459 [cs.IR]

[45] Changyi Ma, Fangchen Yu, Yueyao Yu, and Wenye Li. 2021. Learning Sparse Binary Code for Maximum Inner Product Search. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 3308–3312.

[46] Ji Ma, Ivan Korotkov, Keith Hall, and Ryan T. McDonald. 2020. Hybrid First-stage Retrieval Models for Biomedical Literature. In *Working Notes of Conference and Labs of the Evaluation Forum (CLEF '20)*. DOI: https://ceur-ws.org/Vol-2696/paper_92.pdf

[47] Xueguang Ma, Kai Sun, Ronak Pradeep, and Jimmy J. Lin. 2021. A replication study of dense passage retriever. arXiv:2104.05740 [cs.IR]

[48] Joel Mackenzie, Antonio Mallia, Alistair Moffat, and Matthias Petri. 2022. Accelerating Learned Sparse Indexes Via Term Impact Decomposition. In *Proceedings of the Findings of the Association for Computational Linguistics*. 2830–2842.

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

[50] 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]

[51] Joel Mackenzie, Andrew Trotman, and Jimmy Lin. 2022. Efficient Document-at-a-Time and Score-at-a-Time Query Evaluation for Learned Sparse Representations. *ACM Transactions on Information Systems* (December 2022).

[52] Yu. A. Malkov and D. A. Yashunin. 2016. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. arXiv:1603.09320 [cs.DS]

[53] Antonio Mallia, Omar Khattab, Torsten Suel, and Nicola Tonellotto. 2021. Learning Passage Impacts for Inverted Indexes. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1723–1727.

[54] 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*. 1901–1905.

[55] 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*. 625–634.

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

[57] Stanislav Morozov and Artem Babenko. 2018. Non-Metric Similarity Graphs for Maximum Inner Product Search. In *Proceedings of the Advances in Neural Information Processing Systems*.

[58] Behnam Neyshabur and Nathan Srebro. 2015. On Symmetric and Asymmetric LSHs for Inner Product Search. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, Vol. 37. 1926–1934.

[59] 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).

[60] Yuxin Peng, Xin Huang, and Yunzhen Zhao. 2018. An Overview of Cross-Media Retrieval: Concepts, Methodologies, Benchmarks, and Challenges. *IEEE Transactions on Circuits and Systems for Video Technology* 28, 9 (September 2018), 2372–2385.

[61] Matthias Petri, Alistair Moffat, Joel Mackenzie, J. Shane Culpepper, and Daniel Beck. 2019. Accelerated Query Processing Via Similarity Score Prediction. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 485–494.

[62] Giulio Ermanno Pibiri and Rossano Venturini. 2020. Techniques for Inverted Index Compression. *ACM Computing Surveys* 53, 6, Article 125 (December 2020), 36 pages.

[63] Rameshwar Pratap, Debajyoti Bera, and Karthik Revanuru. 2019. Efficient Sketching Algorithm for Sparse Binary Data. In *Proceedings of the 2019 IEEE International Conference on Data Mining*. 508–517.

[64] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at Trec-3. In *Proceedings of the Overview of the Third Text REtrieval Conference (Trec '94)*, Vol. 500-225. National Institute of Standards and Technology, 109–126.

[65] Anshumali Shrivastava and Ping Li. 2014. Asymmetric LSH (ALSH) for Sublinear Time Maximum Inner Product Search (MIPS). In *Proceedings of the 27th International Conference on Neural Information Processing Systems,* Vol. 2. 2321–2329.

[66] Y. Song, Y. Gu, R. Zhang, and G. Yu. 2021. ProMIPS: Efficient High-Dimensional c-Approximate Maximum Inner Product Search with a Lightweight Index. In *Proceedings of the 2021 IEEE 37th International Conference on Data Engineering*. 1619–1630.

[67] Shulong Tan, Zhaozhuo Xu, Weijie Zhao, Hongliang Fei, Zhixin Zhou, and Ping Li. 2021. Norm Adjusted Proximity Graph for Fast Inner Product Retrieval. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 1552–1560.

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

[69] Mo Tiwari, Ryan Kang, Je-Yong Lee, Donghyun Lee, Chris Piech, Sebastian Thrun, Ilan Shomorony, and Martin Jinye Zhang. 2023. Faster maximum inner product search in high dimensions. arXiv:2212.07551 [cs.LG]

[70] 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.

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

[72] Bhisham Dev Verma, Rameshwar Pratap, and Debajyoti Bera. 2022. Efficient Binary Embedding of Categorical Data using BinSketch. *Data Mining and Knowledge Discovery* 36 (2022), 537–565.

[73] Shuai Wang, Shengyao Zhuang, and Guido Zuccon. 2021. 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*. 317–324.

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

[75] Xiang Wu, Ruiqi Guo, Sanjiv Kumar, and David Simcha. 2019. Local orthogonal decomposition for maximum inner product search. arXiv:1903.10391 [cs.LG]

[76] Xiang Wu, Ruiqi Guo, David Simcha, Dave Dopson, and Sanjiv Kumar. 2019. Efficient inner product approximation in hybrid spaces. arXiv:1903.08690 [cs.LG]

[77] 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. arXiv:1609.08144.

[78] Xiao Yan, Jinfeng Li, Xinyan Dai, Hongzhi Chen, and James Cheng. 2018. Norm-Ranging LSH for Maximum Inner Product Search. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 2956–2965.

[79] Jheng-Hong Yang, Xueguang Ma, and Jimmy Lin. 2021. Sparsifying sparse representations for passage retrieval by top-k masking. arXiv:2112.09628 [cs.IR]

[80] 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*. 497–506.

[81] Wengang Zhou, Houqiang Li, and Qi Tian. 2017. Recent advance in content-based image retrieval: A literature survey. arXiv:1706.06064 [cs.MM]

[82] Zhixin Zhou, Shulong Tan, Zhaozhuo Xu, and Ping Li. 2019. Möbius Transformation for Fast Inner Product Search on Graph. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. 8218–8229.

[83] Shengyao Zhuang and Guido Zuccon. 2022. Fast Passage Re-ranking with Contextualized Exact Term Matching and Efficient Passage Expansion. In *Proceedings of the Workshop on Reaching Efficiency in Neural Information Retrieval, the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[84] Justin Zobel and Alistair Moffat. 2006. Inverted Files for Text Search Engines. *ACM Computing Surveys* 38, 2 (July 2006), 56 pages.

Received 15 September 2023; revised 26 February 2024; accepted 29 April 2024