Balancing the Blend: An Experimental Analysis of Trade-offs in Hybrid Search

Mengzhao Wang
Zhejiang University
wmzssy@zju.edu.cn

Boyu Tan
Zhejiang University
jacktby@gmail.com

Yunjun Gao
Zhejiang University
gaoyj@zju.edu.cn

Hai Jin
Infiniflow
hai.jin@infiniflow.ai

Yingfeng Zhang
Infiniflow
yingfeng.zhang@infiniflow.ai

Xiangyu Ke
Zhejiang University
xiangyu.ke@zju.edu.cn

Xiaoliang Xu
Hangzhou Dianzi University
xxl@hdu.edu.cn

Yifan Zhu
Zhejiang University
xtf_z@zju.edu.cn

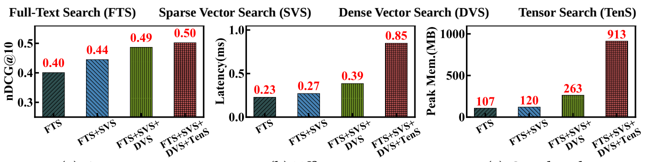
ABSTRACT

Hybrid search, the integration of lexical and semantic retrieval, has become a cornerstone of modern information retrieval systems, driven by demanding applications like Retrieval-Augmented Generation (RAG). The architectural design space for these systems is vast and complex, yet a systematic, empirical understanding of the trade-offs among their core components—retrieval paradigms, combination schemes, and re-ranking methods—is critically lacking. To address this, and informed by our experience building the `Infinity` open-source database, we present the first systematic benchmark of advanced hybrid search architectures. Our framework evaluates four retrieval paradigms—Full-Text Search (FTS), Sparse Vector Search (SVS), Dense Vector Search (DVS), and Tensor Search (TenS)—benchmarking their combinations and re-ranking strategies across 11 real-world datasets. Our results reveal three key findings for practitioners and researchers: (1) A "weakest link" phenomenon, where a single underperforming retrieval path can disproportionately degrade overall accuracy, highlighting the need for path-wise quality assessment before fusion. (2) A data-driven map of the performance trade-offs, demonstrating that optimal configurations depend heavily on resource constraints and data characteristics, moving beyond a one-size-fits-all approach. (3) The identification of Tensor-based Re-ranking Fusion (TRF) as a high-efficacy alternative to mainstream fusion methods, offering the semantic power of tensor search at a fraction of the computational and memory cost. Our findings offer concrete guidelines for designing the next generation of adaptive, scalable hybrid search systems while also identifying key directions for future research.

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/whenever5225/infinity.

1 INTRODUCTION

The increasing complexity of modern information retrieval tasks requires database systems to move beyond single-paradigm search [102, 135]. Hybrid search architectures, which blend multiple retrieval techniques, have emerged as a powerful solution, driven by the demands of applications like Retrieval-Augmented Generation (RAG) [32, 42, 133]. However, designing these systems for optimal performance is a critical challenge, as the trade-offs between retrieval quality, efficiency, and cost are complex [61, 116, 138]. This challenge is particularly significant for advanced architectures combining three or more distinct paradigms, as their synergistic benefits



(a) Accuracy (b) Efficiency (c) Overhead

Figure 1: The multi-faceted performance of hybrid search, with all metrics evaluated on the CQAD(en) dataset.

and interference effects have not been systematically studied. This leaves system designers without a clear guide, raising an important question for the database community: **how can we systematically navigate this complex design space of hybrid search to make evidence-based architectural decisions?**

To answer this question, we begin by reviewing the development of modern retrieval. The study of retrieval has a long and rich history [110], originating in the 1960s and driven by decades of work in the Database (DB) [27, 113, 117, 124] and Information Retrieval (IR) [20, 53, 125, 129] communities. This body of work has produced two complementary approaches: lexical and semantic search. Lexical methods, such as Full-Text Search (FTS) [41, 114] and Sparse Vector Search (SVS) [22, 45], excel at exact keyword matching and produce interpretable results but often fail to capture contextual meaning [54]. In contrast, semantic approaches like Dense Vector Search (DVS) [67, 118] and Tensor Search (TenS) [63, 100] leverage neural models to understand context and nuance [115, 121], though they may lack precision for keyword-specific queries. Hybrid search strategically combines these paradigms to leverage their respective strengths [33, 74, 126]. While this blended approach can significantly improve retrieval accuracy, it introduces a fundamental tension between effectiveness and system cost. As our preliminary results in Figure 1 demonstrate, each additional retrieval path, while boosting accuracy, can dramatically increase query latency and memory consumption. This creates a complex, multi-dimensional optimization problem, making it crucial for system designers to carefully balance these competing factors.

This challenge, however, is exacerbated by several critical research gaps. First, existing studies [31, 73, 135] of hybrid search are often narrow in scope, typically focusing on pairwise combinations like dense and sparse vectors. A systematic understanding of how three or more distinct paradigms perform and interfere with one another in a single system is consequently lacking. Second, the re-ranking strategies employed in practice often rely on simple fusion methods like Reciprocal Rank Fusion (RRF) [21, 38]. While lightweight, the effectiveness of these methods is unverified in complex hybrid search scenarios where candidate lists from up to four paradigms must be consolidated. Finally, and most critically, the community lacks a standardized public benchmark for advanced hybrid search. There is no unified framework available to systematically evaluate and directly compare the trade-offs of all four major paradigms (full-text [113], sparse [137], dense [49], and tensor [84]), which hinders reproducible research and consistent progress. Given the proven, yet distinct, value of each paradigm [49, 84, 95, 102, 137], such a study is essential for understanding their intrinsic trade-offs and guiding future system design.

To bridge these gaps, this paper presents the first comprehensive benchmark of advanced hybrid search architectures. We design and build a novel, modular evaluation framework—informed by our work on the `Infinity`[1] open-source database—that allows for the systematic evaluation of arbitrary combinations of the four retrieval paradigms. Using this framework, we conduct a rigorous experimental study across 11 real-world datasets, creating a data-driven map of the hybrid search performance landscape. Our investigation is guided by three central research questions: (**RQ1**) Does integrating additional retrieval paths inherently improve search accuracy? (**RQ2**) What criteria should be used to select the appropriate combination scheme for different scenarios? (**RQ3**) How can candidates from different retrieval paths be effectively re-ranked in databases?

Key Findings and Implications. Our experimental analysis yields three key findings that provide both practical guidelines for system builders and new directions for researchers: **First**, we identify a "weakest link" phenomenon in multi-path architectures. While combining more retrieval paths can improve accuracy, our results provide the first systematic evidence that a single, underperforming path can disproportionately degrade overall performance. This highlights the critical need for path-wise quality assessment before fusion. **Second**, we provide a data-driven map for navigating performance trade-offs. Our benchmark reveals that no single hybrid architecture is universally optimal. Instead, the ideal configuration is highly dependent on specific constraints (e.g., latency, memory) and data characteristics, and our analysis provides the quantitative evidence needed to guide these crucial design choices. **Third**, we demonstrate that Tensor-based Re-ranking Fusion (TRF) is a high-efficacy and practical re-ranking strategy. TRF consistently outperforms mainstream fusion methods like Reciprocal Rank Fusion (RRF) and offers the semantic power of a full tensor search at a fraction of the computational and memory cost, positioning it as a powerful re-ranking tool for modern database systems.

Our major contributions are as follows:

- **A Systematic Survey of Retrieval Paradigms**. We present the first systematic survey that comparatively analyzes four major retrieval paradigms (full-text, sparse, dense, and tensor) from a database perspective. We analyze their historical evolution, technical underpinnings, and respective trade-offs, establishing a clear foundation for the necessity of advanced hybrid search.
- **A Novel Open-Source Evaluation Framework**. We design, build, and open-source a modular framework for evaluating advanced hybrid search, informed by our work on the `Infinity`[1] database. It is the first publicly available tool that supports the flexible combination and benchmarking of all four major retrieval paradigms, along with multiple re-ranking strategies.
- **A Comprehensive Benchmark and Data-Driven Analysis**. Using our framework, we conduct a rigorous benchmark of various retrieval combinations and re-ranking strategies across 11 real-world datasets. Our analysis produces a quantitative evaluation of accuracy, efficiency, and operational overhead, offering concrete, evidence-based guidelines for practitioners.
- **Identification of Key Challenges and Future Directions**. Based on our empirical findings, we provide a set of novel observations to guide scenario-specific architectural choices and articulate current key challenges and promising research directions for future work in hybrid search.

The remainder of this paper is organized as follows. We review the background on retrieval paradigms in Section 2 and detail our evaluation framework in Section 3. Section 4 describes the experimental setup, and Section 5 presents our comprehensive results. We discuss the key findings, their practical implications, and future research directions in Section 6. Finally, Section 7 concludes the paper. To ensure full reproducibility and facilitate future research, all source code and experimental data from this study are made publicly available[1].

2 BACKGROUND

To inform the design and implementation of effective hybrid search systems, a clear understanding of the foundational retrieval methods is essential. This section reviews the technical underpinnings of the four major paradigms that constitute modern hybrid search: two lexical—Full-Text Search (FTS) and Sparse Vector Search (SVS)—and two semantic—Dense Vector Search (DVS) and Tensor Search (TenS). To make their distinct behaviors more concrete, we will refer to the illustrative example in Figure 2 throughout our discussion. By analyzing the strengths and limitations of each approach, we establish the necessary context for understanding the complexities of their combination and the motivation for our benchmark study.

2.1 Lexical Search Paradigms

Lexical search paradigms operate on the textual representation of documents and queries, primarily relying on keyword frequency and statistical signals to determine relevance. These methods are valued for their efficiency, interpretability, and precision in exact keyword matching. However, their effectiveness can be constrained by the "vocabulary mismatch" problem [47]; they often struggle to understand contextual nuance, treating text as an unordered collection of words. This section reviews the two most prominent lexical paradigms: traditional Full-Text Search (FTS) and modern learned Sparse Vector Search (SVS), whose distinct characteristics are illustrated in Figure 2 (left).

2.1.1 Full-Text Search (FTS). FTS is a foundational lexical retrieval technology that has evolved through decades of optimization. Its development progressed from early Boolean models, which offered efficient filtering but lacked relevance scoring [57, 93, 96], to statistical weighting schemes like TF-IDF that introduced the concept of term importance [16, 87, 90]. This formed the basis for modern

[1]https://github.com/infiniflow/infinity

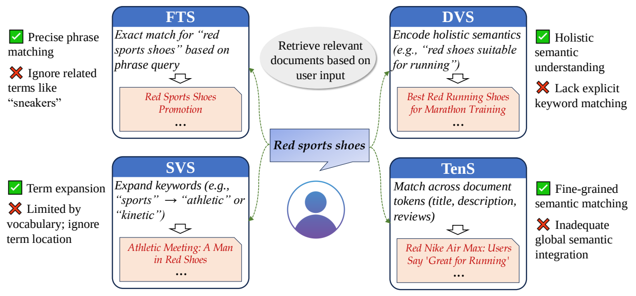

FTS
Exact match for "red sports shoes" based on phrase query
Red Sports Shoes Promotion
...
✓ Precise phrase matching
✗ Ignore related terms like "sneakers"
Retrieve relevant documents based on user input
Red sports shoes
DVS
Encode holistic semantics (e.g., "red shoes suitable for running")
Best Red Running Shoes for Marathon Training
...
✓ Holistic semantic understanding
✗ Lack explicit keyword matching
SVS
Expand keywords (e.g., "sports" → "athletic" or "kinetic")
Athletic Meeting: A Man in Red Shoes
...
✓ Term expansion
✗ Limited by vocabulary; ignore term location
TenS
Match across document tokens (title, description, reviews)
Red Nike Air Max: Users Say 'Great for Running'
...
✓ Fine-grained semantic matching
✗ Inadequate global semantic integration

Figure 2: A toy example of different retrieval paths.

probabilistic models, with the Okapi BM25 (Best Matching 25) algorithm now serving as the de-facto standard for relevance ranking in most production systems [55, 99, 108, 109]. BM25 enhances earlier frameworks by incorporating two key heuristics: non-linear term frequency saturation, which prevents overly frequent terms from disproportionately dominating the relevance score, and document length normalization, which adjusts scores to account for varying document sizes. Given a query Q and a document D, the BM25 score is calculated as:

$$sim(Q, D) = \sum_{i=1}^{n} IDF(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot (1 - b + b \cdot \frac{len(D)}{avgdl})}, \quad (1)$$

where $f(q_i, D)$ is the term frequency of query term q_i in document D, $len(D)$ is the document length, and $avgdl$ is the average document length. The parameters k_1 and b control term frequency saturation and length normalization, respectively. A higher score indicates greater estimated relevance. The behavior of FTS is rooted in its reliance on these exact term statistics. As illustrated in Figure 2 (top left), for a query like "red sports shoes", FTS excels at precise phrase matching, retrieving documents that contain those exact keywords. This precision is a key strength. However, this reliance on exact terms also reveals its primary limitation: a lack of semantic understanding. It will consequently fail to retrieve a relevant document that uses the synonym "sneakers" instead of "shoes".

For large-scale retrieval, the efficiency and effectiveness of FTS engines relies heavily on advanced dynamic pruning [39, 53, 79, 107, 110] and flexible query capabilities [1, 98]. Pruning algorithms such as WAND (Weak AND) [20] and its modern successor, Block-Max WAND [41], facilitate early termination by calculating real-time score bounds for documents, effectively filtering low-value candidates without needing to fully score them. Beyond ranking, a key feature of FTS is its support for rule-based query operations, including phrase searches, wildcard queries, and flexible term weighting, which allow for highly controllable information retrieval. These capabilities, combined with complementary optimizations like inverted index compression [80, 92], form the efficient technical stack that enables modern FTS engines to handle massive corpora.

2.1.2 Sparse Vector Search (SVS). SVS represents a modern evolution of lexical retrieval, enhancing traditional term-based methods with learned term weights derived from deep neural models [31, 69]. Unlike FTS, which relies on raw corpus statistics, SVS utilizes specialized encoder models like SPLADE [44, 45, 64] to transform text into high-dimensional sparse vectors. In these vectors, each dimension corresponds to a term in an expanded vocabulary, and the non-zero values represent the learned contextual importance of that term. The relevance between a query Q and a document D is then computed as the inner product of their respective sparse vectors, $\boldsymbol{q}$ and $\boldsymbol{d}$:

$$sim(Q, D) = \boldsymbol{q}^\top \boldsymbol{d} = \sum_{i=1}^{n} q_i d_i, \quad (2)$$

A higher inner product indicates greater relevance. This learning-based approach grants SVS a key advantage over FTS: term expansion. As shown in Figure 2 (bottom left), SVS can map a query term like "sports" to semantically related vocabulary terms such as "athletic" or "kinetic", thereby mitigating the vocabulary mismatch problem to a degree. However, SVS is still fundamentally a lexical method. It is constrained by its fixed, albeit expanded, vocabulary and typically disregards the position and ordering of terms, which limits its ability to capture more complex phrasal or high-level semantics.

The high dimensionality of sparse vectors makes efficient search a critical challenge, which is addressed using techniques analogous to those in FTS. SVS systems leverage inverted indexes built from the non-zero dimensions of the vectors, enabling the use of dynamic pruning algorithms to accelerate query processing [23–25, 76, 77, 85, 128]. Advanced methods such as Block-Max Pruning (BMP) [82] precompute maximum term impact scores for blocks of documents, allowing the system to aggressively prune large portions of the search space without full evaluation. By combining learned relevance with fast pruning, SVS offers a powerful lexical search paradigm that bridges the gap between traditional keyword statistics and deeper semantic understanding.

2.1.3 Comparison and Synthesis: FTS vs. SVS. While both FTS and SVS are lexical paradigms that typically rely on inverted indexes and pruning for efficiency, their foundational differences in relevance modeling lead to distinct trade-offs. The core distinction lies in the origin of term weights: FTS derives relevance from unsupervised and corpus-level statistics via formulas like BM25, making it robust to any text corpus without prior training. In contrast, SVS relies on pre-trained neural models to learn the contextual importance of terms, allowing it to capture nuances that statistical methods miss. This leads to a key practical trade-off between query control and term expansion. FTS offers highly controllable, rule-based operations (e.g., precise phrase and wildcard searches), whereas SVS excels at learned term expansion, mitigating vocabulary mismatch but offering less direct user control. From a systems perspective, this also introduces a significant difference in operational overhead: FTS indexes raw text directly, while SVS requires a computationally intensive inference step to generate vectors before indexing. Ultimately, the choice between them involves balancing the statistical robustness and query controllability of FTS against the powerful contextual matching of SVS, highlighting their complementary nature within the lexical search landscape.

2.2 Semantic Search Paradigms

In contrast to lexical methods, semantic search paradigms leverage deep neural networks to transform text into rich numerical representations, aiming to capture meaning and context rather than relying on exact keyword statistics. The primary strength of these approaches lies in their ability to understand synonyms, related concepts, and high-level semantic intent. However, this semantic depth can come at the cost of keyword precision, and a reliance on pre-trained models may introduce significant computational overhead and challenges in domain adaptation. This section examines the two dominant semantic paradigms: Dense Vector Search (DVS), which models text with a single global vector, and Tensor Search (TenS), which uses a multi-vector representation, as illustrated in Figure 2 (right).

2.2.1 Dense Vector Search (DVS). DVS operates on the principle of representing the entire semantic meaning of a text—be it a sentence or a document—as a single, fixed-dimensional dense vector, often called an embedding. These embeddings are typically generated using bi-encoder architectures [62, 75, 106], where models like Sentence-BERT [97] are trained to aggregate token-level representations into a holistic summary, for instance by using the output of a special token (e.g., `[CLS]` [31, 37]) or mean-pooling all token hidden states [94]. The semantic similarity between a query Q and a document D is then measured by the inner product of their dense vectors. A higher inner product indicates greater semantic similarity. This global representation allows DVS to achieve a holistic semantic understanding, as illustrated in Figure 2 (top right). For the query "red sports shoes", DVS can retrieve documents about "running shoes" by capturing the query's high-level intent, even if the exact keywords are absent. However, this aggregation process can obscure fine-grained details, leading to its primary weakness: a lack of explicit keyword matching and reduced precision for queries that depend on specific, exact terms.

Efficiently searching through millions or billions of dense vectors is computationally prohibitive for exact methods. Consequently, DVS systems are fundamentally reliant on Approximate Nearest Neighbor Search (ANNS) algorithms [18, 26, 88, 91, 123, 139] to achieve low-latency, high-throughput retrieval. The development of ANNS has progressed through several algorithmic families, including tree-based [103, 132], hashing-based [48, 52, 58, 70], and quantization-based approaches [15, 17, 50, 59]. In recent years, graph-based algorithms [46, 105, 118], most notably HNSW (Hierarchical Navigable Small World graph) [78], have emerged as the state-of-the-art, offering a superior balance of accuracy and efficiency. These graph-based ANNS techniques now form the core engine of nearly all modern vector databases [28, 115, 130], enabling practical semantic search at a massive scale.

2.2.2 Comparison and Synthesis: SVS vs. DVS. While both SVS and DVS leverage vector representations, they embody fundamentally different philosophies of semantic modeling and are supported by distinct architectures. The primary distinction lies in the granularity of representation. SVS focuses on term-level semantic expansion, creating high-dimensional vectors where each dimension corresponds to a vocabulary term with a learned weight. This retains a strong lexical grounding, making it an evolution of traditional "bag-of-words" models. In contrast, DVS pursues holistic semantic summarization, compressing the entire meaning of a text into a single, compact dense vector where individual dimensions lack explicit meaning. This conceptual abstraction allows DVS to capture high-level intent. This representational difference dictates their underlying architectures: SVS typically utilizes the classic inverted index and dynamic pruning stack inherited from traditional retrieval, while DVS relies on an entirely different class of algorithms for ANNS. Ultimately, these differences make them highly complementary: SVS offers a powerful, learned extension of lexical search, while DVS provides a purely semantic pathway, explaining their frequent pairing in modern hybrid search systems.

2.2.3 Tensor Search (TenS). TenS represents a paradigm shift from single-vector representations, instead modeling a text as a set of contextualized embeddings for each of its constituent tokens. Pioneered by models like ColBERT [2, 35, 60, 63, 71, 101], TenS employs a "late-interaction" architecture. Rather than aggregating a text's meaning into one vector before comparison, it retains a tensor of token embeddings for both the query and the document, performing fine-grained similarity computations at query time. The relevance score is typically calculated using a max-similarity (MaxSim) operation, which aggregates the maximum similarity score for each query token embedding against all token embeddings in the document:

$$sim(Q, D) = \sum_{i=1}^{N} \max_{j=1}^{M} \boldsymbol{q}_i^\top \cdot \boldsymbol{d}_j, \tag{3}$$

where $\boldsymbol{q}_i$ and $\boldsymbol{d}_j$ are the token embeddings of the query and document, respectively. This token-level interaction process effectively aligns each query token with its most relevant semantic counterpart in the document. As depicted in Figure 2 (bottom right), TenS can match query tokens across different parts of a document; for instance, for the query "red sports shoes", it could match the token "red" in a product's title, a related token like "Nike" in its description, and "running" in a user review. The final relevance score, an aggregation of these best-token matches, can capture fine-grained semantic relationships. However, this focus on local token alignment is also the source of its weakness: inadequate global semantic integration. By summing individual token scores, TenS may fail to capture the holistic context of a sentence or paragraph.

The primary drawback of the TenS paradigm is its significant computational and memory overhead [51, 66, 89, 101], stemming from the need to store and process an embedding for every token in the corpus. A naive implementation can result in index sizes that are orders of magnitude larger than those for DVS. Consequently, a substantial body of research has focused on mitigating the high cost of TenS. Techniques range from representation compression, such as the residual representations in ColBERTv2 [101] that dramatically reduce index size, to multi-stage filtering pipelines like PLAID [100] and quantization-based methods like EMVB [84] that leverage hardware acceleration (e.g., SIMD) to improve storage and retrieval efficiency. Despite these advancements, the high cost of TenS remains a significant barrier to its widespread adoption, positioning it as a powerful but resource-intensive "heavyweight" semantic search paradigm.

2.2.4 Comparison and Synthesis: DVS vs. TenS. While both DVS and TenS are semantic paradigms that model text using neural embeddings, they differ fundamentally in their semantic granularity and resulting retrieval architectures. DVS operates on a principle of holistic summarization, compressing an entire text into a single, fixed-dimensional vector. This global representation is powerful for capturing high-level topics and intent but inherently sacrifices fine-grained, local details in the process [62, 94]. In contrast, TenS adopts a fine-grained, multi-vector approach, retaining a contextualized embedding for each token. This allows for a "late-interaction" mechanism that can capture nuanced, token-level relationships, but it may struggle to form a coherent understanding of the text's overall global meaning [63]. This core difference in representation dictates their architectural trade-offs. DVS, with its single-vector-per-document model, is perfectly suited for the mature and highly optimized ecosystem of ANNS algorithms, enabling efficient search over massive datasets. TenS, however, presents a far greater retrieval challenge due to the explosion in the number of embeddings to be indexed and queried. This necessitates more complex, multi-stage retrieval pipelines and specialized optimizations to remain computationally tractable. Ultimately, DVS can be seen as the efficient, general-purpose semantic workhorse, while TenS represents a resource-intensive, "heavyweight" paradigm that trades scalability for fine-grained matching precision. Their distinct strengths and costs make them complementary.

2.3 Hybrid Search Architectures.

As previously discussed, lexical and semantic paradigms possess complementary strengths and weaknesses, making neither consistently sufficient on its own. Hybrid search aims to synthesize these capabilities, combining multiple retrieval paths to achieve a level of accuracy and robustness that a single paradigm often cannot match alone. By integrating these diverse methods, hybrid systems can satisfy complex queries that require both the precision of keyword matching and a deep understanding of semantic intent. However, the central challenges of hybrid search lie in how to strategically combine the retrieval paths and how to effectively merge the disparate candidate lists. This section examines the two core components of this process: the architectural combination schemes used to integrate retrieval paths, and the candidate re-ranking strategies employed to produce a single, unified result set.

2.3.1 Retrieval-Path Combination Schemes. Architecturally, current hybrid search implementations [22, 73, 131] predominantly focus on dual-path combination schemes, which can be broadly categorized by the primary system they extend. One common approach is vector-centric [5, 10, 11], where SVS is integrated into vector databases to augment their native DVS capabilities, thereby addressing the weakness of DVS in handling precise keyword queries. Conversely, a text-centric approach integrates DVS into established FTS engines like Elasticsearch [6], retrofitting them with semantic search capabilities [7, 29, 127]. The underlying integration strategies for these dual-path systems vary: some create a unified index over both sparse and dense vectors to enable joint pruning [135], while others advocate for separate, independent indexes managed by a multi-index engine that coordinates queries across them [34]. However, these approaches are typically limited to two paradigms. As noted in our previous discussion, comprehensive architectural studies and systems that effectively integrate three or more paradigms—especially the TenS—remain scarce.

2.3.2 Candidate Re-ranking Strategies. Once multiple retrieval paths produce their individual candidate lists, a re-ranking strategy is required to merge them into a single, unified result set. In modern database and search systems, one of the most prevalent lightweight techniques is Reciprocal Rank Fusion (RRF) [12, 38]. RRF is a score-agnostic method that synthesizes ranked lists by prioritizing candidates that consistently appear near the top of multiple lists, making it particularly effective when the raw scores from different paradigms (e.g., a BM25 score and a vector similarity score) are not directly comparable [33]. The RRF score for a candidate c is formally defined as:

$$\mathrm{RRF}(c) = \sum_{i=1}^{n} \frac{1}{\kappa + \mathrm{rank}_i(c)}, \tag{4}$$

where $\mathrm{rank}_i(c)$ is the rank of candidate c in the i-th list, n is the number of lists, and κ is a smoothing parameter.

An alternative approach is the Weighted Sum (WS) [13, 21], which linearly combines the normalized scores from each retrieval path. The WS score for a candidate c is computed as:

$$\mathrm{WS}(c) = \sum_{i=1}^{n} w_i \cdot \mathrm{score}_i(c), \tag{5}$$

where w_i is the pre-defined weight for the i-th list and $\mathrm{score}_i(c)$ is the candidate's score in that list. Unlike the rank-based RRF, WS is score-aware and is most suitable when the scores from each path are normalized and their relative importance can be reasonably estimated [119].

Remarks. It is important to distinguish these lightweight fusion methods from more powerful, but heavyweight, learning-based re-rankers, such as those based on cross-encoder architectures [63, 136]. This framework performs a full, computationally intensive interaction between the query and each candidate document's tokens at query time. This deep interaction allows them to achieve superior accuracy but makes them impractical for in-database deployment on large initial candidate sets due to high latency and the typical need for specialized hardware like GPUs [97]. Consequently, these models are generally used in a second, cascaded stage: a fast, in-database fusion method like RRF or WS first retrieves an initial set of candidates, which are then re-ranked externally by a cross-encoder. Our study focuses on the critical first stage of this process—the efficient, in-database fusion of multiple retrieval paths—and evaluates its effectiveness, which remains largely unverified in complex hybrid search architectures.

3 EVALUATION FRAMEWORK

To systematically answer the research questions posed in Section 1, a specialized evaluation framework is required. Such a framework must be (1) **comprehensive**, supporting all four major retrieval paradigms; (2) **modular**, allowing for their flexible and arbitrary combination; and (3) **performant**, incorporating state-of-the-art algorithms for a fair and rigorous comparison. To meet these requirements, we designed and implemented a novel evaluation framework, whose architecture is informed by our development of the `Infinity`[1] open-source database.

3.1 Architectural Principles and Overview

Our framework is designed around three core architectural principles derived from the requirements of our study: modularity to

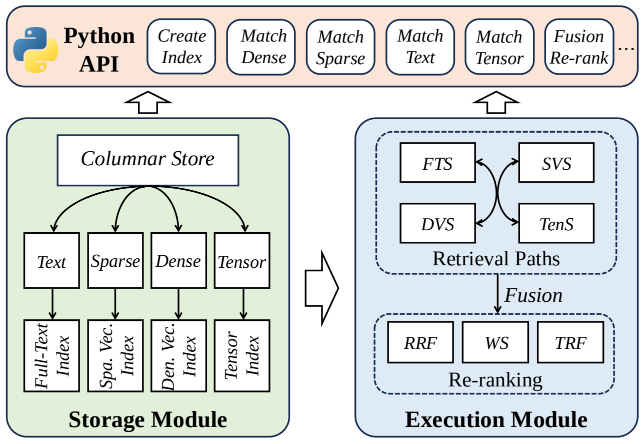


Figure 3: Overview of the evaluation framework.

support flexible paradigm combination, specialization to ensure high performance for each individual retrieval path, and pipelined execution to minimize query latency. As illustrated in Figure 3, these principles are realized through a two-module architecture: a Storage Module responsible for data management and specialized indexing, and an Execution Module that handles pipelined hybrid query processing.

The storage module employs a decoupled, columnar storage design where each data type (Text, Sparse, Dense, Tensor) is managed by its own specialized indexing fleet. This architectural choice is critical for modularity and performance, as it allows each retrieval path to use its own state-of-the-art indexing algorithms without compromise, a key limitation in some unified index approaches. This decoupled structure also ensures the framework is extensible, enabling the seamless integration of new retrieval paradigms or advanced indexing techniques in the future.

The execution module orchestrates hybrid search by transforming a query into a pipelined Directed Acyclic Graph (DAG) of execution operators. It utilizes a push-based, concurrent execution model, where each operator can run independently to maximize resource utilization and reduce overall latency. The final fusion operator is responsible for applying the chosen re-ranking strategy to the candidate sets retrieved from the various parallel paths to produce a single, unified result. For ease of use and reproducibility, the framework is exposed through a user-friendly Python API[2].

3.2 Storage Module: Decoupled Indexing

The storage module implements the principle of decoupled indexing through a columnar data layout and a suite of specialized index structures. During data ingestion, each source document is processed and partitioned into its respective columns: a raw text column for FTS, and separate columns for the sparse vector, dense vector, and tensor representations generated by the embedding models. Each of these columns is then indexed by a dedicated indexing fleet applying the most effective algorithm for its data type. For example, the dense vector fleet implements a state-of-the-art HNSW index, while the full-text and sparse vector fleets use their own highly optimized inverted indexes with specialized pruning logic. All indexes are persisted to disk and accessed via memory mapping to ensure efficient data management and scalability beyond main memory capacity.

3.3 Execution Module: Pipelined Hybrid Query Processing

The execution module is responsible for orchestrating complex hybrid queries across the multiple, independent indexes of the storage module. To achieve this, it first transforms a hybrid query into a Directed Acyclic Graph (DAG) of physical operators. Each node in the DAG represents a specific task, such as a scan operation on an inverted index or a vector index. This DAG is then compiled into a query pipeline where each retrieval path is executed in parallel to maximize throughput. The results from these parallel execution paths are streamed to a final fusion operator, which applies the specified re-ranking strategy (e.g., RRF) to merge the candidate lists and produce the unified result.

To minimize latency and support high-concurrency workloads, the framework implements a fine-grained, push-based execution model. Unlike traditional pull-based models where operators request data, data is actively pushed from one operator to the next as soon as it becomes available. This approach reduces idle computational time and is highly effective for the streaming, data-intensive nature of search queries. Each operator in the pipeline is designed to execute independently and concurrently. The framework manages a dedicated thread pool, strategically assigning operator tasks to available cores to maximize parallelism while mitigating resource contention between adjacent CPU cores. This combination of DAG-based planning and a concurrent, push-based execution engine allows the framework to efficiently coordinate complex, multi-path queries with minimal overhead.

3.4 Algorithm Implementations

Our framework's modularity allows each component to be implemented using specialized, state-of-the-art techniques. The key implementation details are as follows.

We implemented our FTS engine from scratch to allow for tight integration with our storage and execution models, inspired by the production-grade designs of `Lucene`[3] and `Tantivy`[4]. It utilizes the BM25 ranking function, with query performance accelerated by a Block-Max WAND (BMW) [41] dynamic pruning algorithm. The implementation also supports configurable tokenizers [8, 9] and integrates document ID reassignment to improve data locality and pruning efficiency. For SVS, we implement Block-Max Pruning (BMP) [82], a highly-efficient dynamic pruning algorithm. Our implementation includes its core components of block filtering [81] and candidate block evaluation, which are heavily optimized with SIMD instructions [40] and a hybrid inverted-forward index structure to accelerate the top-k selection process. The DVS component is built upon a state-of-the-art HNSW graph index [78]. To enhance performance beyond standard implementations, our version integrates Locally-adaptive Vector Quantization (LVQ) [15]. This is a two-stage quantization process that significantly reduces

[2]https://infiniflow.org/docs/pysdk_api_reference

[3]https://github.com/apache/lucene
[4]https://github.com/quickwit-oss/tantivy

the in-memory index footprint and accelerates distance computations during the graph traversal phase of the search. To manage the high overhead of TenS, we implement the EMVB technique [84]. This includes its key optimizations for efficiency: a bit-vector pre-filtering mechanism to quickly discard irrelevant documents, SIMD-accelerated centroid computations, and Product Quantization (PQ) to compress the large number of residual token vectors. Finally, the fusion operator supports three distinct strategies: the lightweight and widely-used Reciprocal Rank Fusion (RRF) and Weighted Sum (WS), alongside our Tensor-based Re-ranking Fusion (TRF), which uses the fine-grained MaxSim score (Equation (3)) for more accurate semantic re-ranking.

4 EVALUATION SETUP

This section details the experimental setup for our comprehensive evaluation of advanced hybrid search architectures. Our methodology is designed to be rigorous and reproducible, providing the empirical evidence needed to answer the three research questions posed in Section 1. We first describe the core components of our experiment, including the datasets and methods under evaluation. We then define our evaluation metrics, followed by the specific implementation details and the hardware environment.

4.1 Experimental Components

This section details the core components—datasets, models, and methods—that constitute our experimental evaluation.

4.1.1 Datasets. To ensure the generalizability of our findings, we evaluate all methods on a diverse collection of 11 real-world datasets, chosen for their variety in domain, task, and scale. As summarized in Table 1, most datasets are from the widely-used BEIR benchmark [106]. To enhance diversity, we also include two multilingual datasets (Chinese and English) from MTEB [83] and a Chinese news question-answering dataset from AIR-Bench [30]. The collection covers 9 distinct tasks, such as question answering and fact-checking, across 7 domains, with corpus sizes ranging from 5.2K to 8.8M documents. With the exception of MSMA(en), all datasets are out-of-domain. This means the embedding models were not fine-tuned on them, which allows for a robust test of zero-shot retrieval performance.

4.1.2 Embedding Models. We utilize mainstream embedding models to generate embeddings of dense vector search (DVS), sparse vector search (SVS), and tensor search (TenS). For dense and sparse vectors, we use the `BGE-M3` model [31], which supports a wide range of input lengths (from short sentences to documents up to 8,192 tokens) and is designed to concurrently produce high-quality dense and sparse representations. While `BGE-M3` can also generate token-level tensors, its 1,024-dimensional token embeddings render it impractical for large-scale TenS. For instance, we observed that generating tensors for fewer than 50% of the documents in the MLDR(en) dataset would result in a prohibitive index size of up to 1.1TB. To address this, for TenS on English datasets, we adopt the highly efficient `answerai-colbert-small-v1` model [2]. With only 33 million parameters (compared to `BGE-M3`'s 560 million), it produces much more manageable 96-dimensional token representations. For Chinese datasets, we use the `Jina-ColBERT-v2` model [60], which employs Matryoshka Representation Learning; we select its most compact 64-dimensional variant to ensure optimal space efficiency.

4.1.3 Evaluated Methods. Our evaluation covers four single-path retrieval methods: FTS, DVS, SVS, and TenS, each implemented with a state-of-the-art algorithm as detailed in Section 3.4. From these, we construct and evaluate 11 different hybrid search combinations, including all six two-path, four three-path, and the single four-path configuration. For all hybrid methods, we evaluate three re-ranking strategies: the widely-used Reciprocal Rank Fusion (RRF) and Weighted Sum (WS), alongside our Tensor-based Rank Fusion (TRF), which uses tensor representations for a fine-grained re-ranking of candidates.

4.2 Evaluation Metrics

To provide a holistic assessment of each method, we evaluate three key dimensions: effectiveness, efficiency, and resource overhead.

4.2.1 Effectiveness. We measure retrieval effectiveness using Normalized Discounted Cumulative Gain at rank k (nDCG@k), with k set to 10 by default. Following the standard for mainstream information retrieval benchmarks like BEIR, we selected nDCG@k for its ability to handle both binary and graded relevance judgments and to properly account for the rank of retrieved documents [68, 106, 120]. Furthermore, theoretical analyses [122] demonstrate its ability to ensure comparability across different methods and tasks. These properties make it a more robust and informative metric than rank-unaware measures like Precision/Recall or those limited to binary relevance, such as Mean Reciprocal Rank (MRR) and Mean Average Precision (MAP) [106].

4.2.2 Efficiency and Overhead. We evaluate system performance using metrics that cover both online query processing and offline index construction. **Query Performance**: We measure online efficiency using Query Latency (the average response time per query) and Throughput (measured in Queries Per Second, or QPS, under a multi-threaded batch workload). **Indexing Performance**: For offline evaluation, we measure the total Indexing Time (including data import and construction) to assess efficiency, and the final on-disk Index Size to assess storage overhead. **Resource Overhead**: We quantify resource usage by measuring the Peak Memory Consumption during both online query processing and offline index construction. Our framework utilizes memory mapping (mmap) for index access, so the observed online memory footprint may be lower than the total index size.

4.3 Implementation and Environment

This section details the specific implementation parameters and the hardware environment used to conduct our experiments.

4.3.1 Implementation Details. All components are configured using parameters consistent with established best practices to ensure a fair comparison. **FTS**: For the BM25 scoring function (refer to Equation (1)), we use the standard default parameters of $k_1 = 1.2$ and $b = 0.75$. We use the standard tokenizer [9] for English datasets and the IK tokenizer [8] for Chinese. **SVS**: The Block-Max Pruning algorithm is configured with a block size of 8, in line with

Table 1: Statistics of datasets. The last four columns denote the corpus size for each data type.

Dataset	Domain	Task	#Corpus	#Query	Avg. Length of Docs	Full-Text	Sparse	Dense	Tensor
MSMA(en) [86]	Miscellaneous	Passage Retrieval	8,841,823	43	56	2.9GB	13GB	36GB	254GB
DBPE(en) [56]	Wikipedia	Entity Retrieval	4,635,922	400	50	1.4GB	6.9GB	18.2GB	58GB
MCCN(zh) [30]	News	Question Answering	935,162	339	1,263	2.3GB	6.9GB	3.8GB	148GB
TOUC(en) [19]	Miscellaneous	Argument Retrieval	382,545	49	292	184MB	1.6GB	1.5GB	56GB
MLDR(zh) [31]	Wiki., Wudao	Long-Document Retrieval	200,000	800	4,249	3.1GB	5.2GB	791MB	186GB
MLDR(en) [31]	Wikipedia	Long-Document Retrieval	200,000	800	3,308	3.1GB	4.4GB	791MB	94GB
TREC(en) [111]	Bio-Medical	Bio-Medical Information Retrieval	171,332	50	161	184MB	554MB	688MB	16GB
FIQA(en) [4]	Finance	Question Answering	57,638	648	132	43MB	137MB	232MB	4GB
CQAD(en) [3]	StackExchange	Duplicate-Question Retrieval	40,221	1,570	129	19MB	63MB	164MB	1GB
SCID(en) [36]	Scientific	Citation Prediction	25,657	1,000	176	30MB	89MB	106MB	2.4GB
SCIF(en) [112]	Scientific	Fact Checking	5,183	809	214	7.5MB	23MB	24MB	676MB

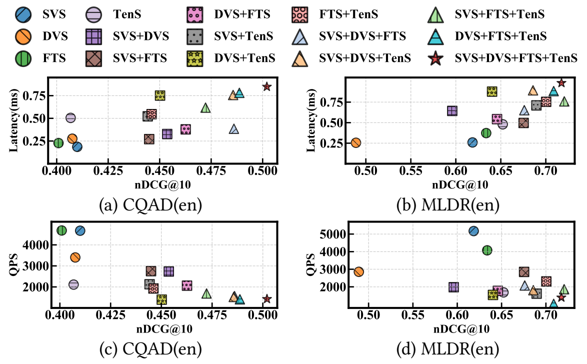


Figure 4: Accuracy-efficiency trade-offs across different combination schemes.

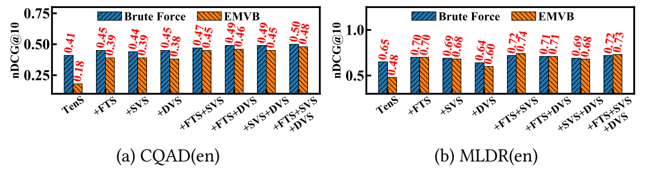


Figure 5: Accuracy comparison of brute-force and EMVB.

recommended settings for the method. **DVS**: Our HNSW index is constructed with a maximum neighbor size (`M`) of 16 and a candidate neighbor size (`efconstruction`) of 200, which are common settings in practice [65, 72]. The LVQ enhancement is applied as detailed in its original work [15]. **TenS**: The EMVB algorithm is configured with 8192 centroids and 32 subspaces for its Product Quantization (PQ) stage, conforming to typical configurations for this technique [17, 59, 134]. **Re-ranking**: For RRF, the smoothing parameter κ is set to 60, a widely-used value [33, 38]. For WS, the weight of each path is set to its nDCG@10 score from its single-path evaluation on the respective dataset. Unless otherwise specified, RRF is the default re-ranking strategy in all hybrid search evaluations.

4.3.2 Experimental Environment. All embedding generation was performed on two NVIDIA RTX 5000 Ada Generation GPUs. The core retrieval framework is implemented in C++ for performance, with a user-friendly Python API for accessibility. All experiments were conducted on a Linux server equipped with two Intel Xeon E5-2650 v4 CPUs (2.20 GHz). Each CPU comprises 12 cores and supports hyper-threading, providing 48 logical processors. The system is configured with 125GB of main memory (RAM). To ensure reliability, each experiment was executed at least three times, and the results were averaged to minimize environmental interference.

5 EXPERIMENTAL RESULTS

This section presents the results of our experimental evaluation. We analyze the data from our benchmark across 11 real-world datasets to answer the three research questions posed in Section 1. Due to space constraints, we present the primary results in this section; a more exhaustive set of results is available in [14].

5.1 Multi-Path Retrieval Effectiveness (RQ1)

Finding 1: While hybrid search accuracy generally improves with more retrieval paths, the overall performance is disproportionately constrained by the effectiveness of its weakest component.

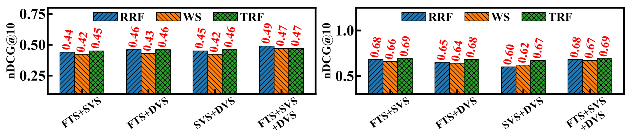
The primary motivation for hybrid search is complementarity, where different paradigms identify different but overlapping sets of relevant documents, and their combination produces a richer candidate pool for re-ranking. This additive effect is evident on datasets like DBPE(en) (Table 2), where combining FTS (nDCG@10 of 0.565) and SVS (0.479) yields a significantly higher hybrid score of 0.608. In this case, FTS contributes candidates via exact keyword matches while SVS adds candidates with expanded, semantically related terms. The resulting pool is more comprehensive than what either path could produce alone, increasing the likelihood of capturing the user's full intent.

However, our benchmark reveals that this synergy can fail, or even reverse, under specific conditions. A path with low accuracy can pollute the candidate pool with many irrelevant documents. These "noisy" candidates may displace relevant results from stronger paths during the final re-ranking stage. This problem is particularly acute for fusion methods like RRF, which are sensitive to a document's mere presence across multiple lists. The "weakest link" phenomenon is clear on our results (Table 2). On the TOUC(en) dataset, for instance, adding a weak DVS path (nDCG@10 of 0.390) to a strong FTS path (0.650) degrades the final score to 0.604. A similar dilutive effect is visible on MSMA(en), where even a state-of-the-art DVS path (nDCG@10 of 0.830) is hindered when combined with weaker paths like FTS (resulting in 0.816) or SVS (0.814).

This finding has critical implications for system design. A hybrid architecture is not a simple summation of strengths but a sensitive system where component quality is paramount. Naively adding a retrieval path is therefore a high-risk strategy, as a weak path can

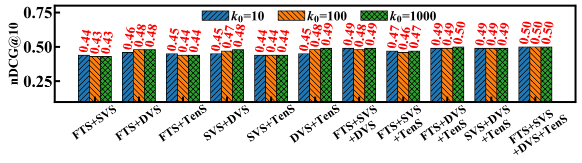
Table 2: Retrieval accuracy (nDCG@10) of different methods with FTS, SVS, and DVS across 11 datasets. In hybrid search, for each dataset column, the left values use RRF re-ranking, and the right values use TRF re-ranking. The best score on a given dataset is bolded, and the second-best is underlined. TenS is omitted due to high storage and computation costs.

Method(↓)	Dataset(→)	MSMA(en)		DBPE(en)		MCCN(zh)		TOUC(en)		MLDR(zh)		MLDR(en)		TREC(en)		FIQA(en)		CQAD(en)		SCID(en)		SCIF(en)	
Single-Path	FTS	.744		.565		.222		.650		.411		.634		.839		.328		.401		.310		.704	
	SVS	.645		.479		.354		.626		.405		.619		.781		.375		.410		.263		.647	
	DVS	.830		.692		.543		.390		.262		.489		.751		**.532**		.408		.326		.715	
Two-Path	FTS+SVS	.734	.867	.608	.678	.328	.438	**.690**	<u>.677</u>	.455	<u>.495</u>	.675	<u>.690</u>	.835	.911	.393	.457	.445	.449	.312	.343	.700	.747
	FTS+DVS	.816	**.894**	.668	**.722**	.440	**.624**	.604	.618	.411	.471	.646	.680	.832	<u>.920</u>	.471	.508	.463	<u>.465</u>	.347	**.354**	.748	<u>.761</u>
	DVS+SVS	.814	.887	.674	.711	.518	<u>.622</u>	.566	.596	.365	.448	.596	.666	.803	.899	.506	<u>.510</u>	.454	.463	.320	<u>.353</u>	.716	.746
Three-Path	FTS+SVS+DVS	<u>.819</u>	<u>.888</u>	.692	<u>.713</u>	.474	**.624**	.654	.627	.451	**.496**	.676	**.693**	.865	**.924**	.490	.496	**.486**	<u>.465</u>	.345	<u>.353</u>	.739	**.762**

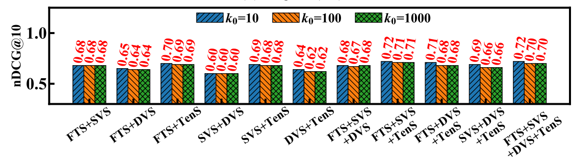


(a) CQAD(en) (b) MLDR(en)

Figure 6: Accuracy comparison of RRF, WS, and TRF.



(a) CQAD(en)



(b) MLDR(en)

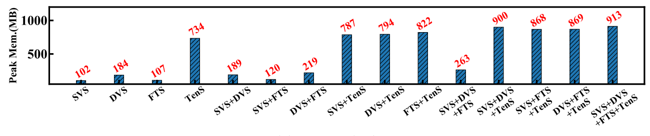
Figure 7: Accuracy under different numbers of candidates.

pollute the candidate pool with irrelevant documents. This introduces performance instability and forces the re-ranker to arbitrate between a few good candidates and a large volume of noise, increasing the likelihood of ranking errors. This underscores the need for robust design principles, such as path-wise quality assessment or dynamic pruning, to mitigate the impact of underperforming components. Our benchmark confirms this "weakest link" principle is a consistent phenomenon. The phenomenon persists across various configurations. It is independent of the re-ranking strategy chosen (Figure 6). It also holds true for different candidate list sizes (Figure 7) and specific algorithmic implementations (Figure 5).

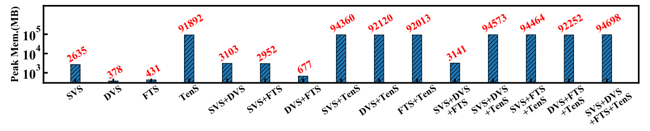
5.2 Combination Scheme Selection (RQ2)

Finding 2: There is no universally optimal hybrid search architecture; the ideal combination of retrieval paths is governed by a multi-dimensional trade-off between accuracy, query performance, and resource overhead.

Our benchmark data reveals a clear trade-off between retrieval accuracy and online query performance. As shown across multiple datasets (Figures 4 and 11), adding more retrieval paths to a combination, while generally boosting nDCG@10, consistently increases query latency and reduces throughput (QPS). This performance cost is significant even within our optimized execution

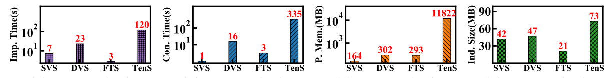


(a) CQAD(en)

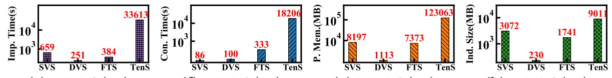


(b) MLDR(en)

Figure 8: Memory consumption across retrieval methods.



(a) CQAD(en) (b) CQAD(en) (c) CQAD(en) (d) CQAD(en)



(e) MLDR(en) (f) MLDR(en) (g) MLDR(en) (h) MLDR(en)

Figure 9: Indexing overhead of different retrieval paths.

framework, which uses DAG-based scheduling and parallel execution to minimize latency. For instance, on the CQAD(en) dataset, while the four-path FTS+DVS+SVS+TenS combination achieves a high nDCG@10 of 0.50, its query latency is 3.7 times higher than that of the single-path FTS, which has a lower accuracy of 0.40. This presents a critical decision point for system designers: applications requiring the highest possible accuracy must be provisioned to tolerate higher latency, while latency-critical applications may need to select a less accurate, but more efficient, configuration.

This trade-off extends beyond query performance to both online memory consumption and offline indexing costs. Online memory overhead, as illustrated in Figure 8, directly scales with the number of combined paths. This cost is particularly acute for TenS, whose peak memory consumption can be orders of magnitude higher than other paradigms. This is a direct consequence of its per-token embedding representation; unlike DVS where storage is constant per document, TenS's storage footprint scales linearly with document length. The offline costs of indexing (Figure 9) reflect this same challenge. Here, TenS again imposes an extremely high resource burden, requiring substantially more time and peak memory to build its massive index of token embeddings. These findings have significant practical implications for system deployment, as the substantial resource costs of TenS may render it infeasible as a primary retrieval path in many large-scale scenarios.

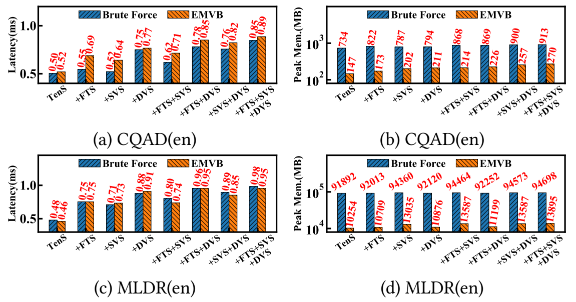


Figure 10: Online overhead of brute-fore and EMVB.

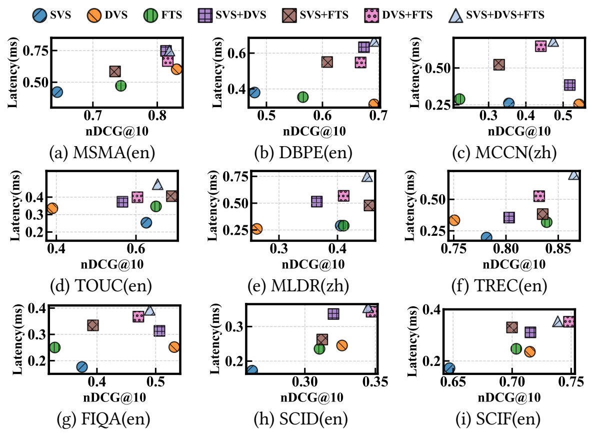


Figure 11: Cross-dataset latency-accuracy trade-offs.

Furthermore, the optimal trade-off point is not static; it is influenced by the maturity of algorithmic implementations and the specific characteristics of the dataset. The choice of a specific algorithmic implementation often embodies a deliberate trade-off between performance and resource consumption. Our TenS evaluation (Figure 10) shows that an optimized EMVB algorithm can reduce peak memory usage by up to 88% compared to a brute-force approach. This gain, however, comes at the direct cost of lower retrieval accuracy. Dataset characteristics also play a crucial role. On corpora with long documents like MLDR(en), the fixed-size representation of DVS offers a clear advantage in indexing efficiency. However, compressing a long document's semantics into a single vector can lead to significant information loss and lower accuracy. In contrast, the representations for FTS and SVS scale with document length, allowing them to retain more term-level detail. Ultimately, these factors reinforce our central finding: selecting a hybrid search architecture is a complex balancing act that requires a deep understanding of the application's specific performance priorities and data context.

5.3 Cross-Path Re-ranking (RQ3)

Finding 3: While lightweight fusion methods like RRF are efficient, Tensor-based Re-ranking Fusion (TRF) offers a path to significantly higher accuracy by leveraging fine-grained semantics at a fraction of the cost of a full FTS path.

The choice of re-ranking strategy substantially impacts a hybrid system's final accuracy. Our results consistently show that TRF holds a clear advantage in effectiveness over the widely-used RRF

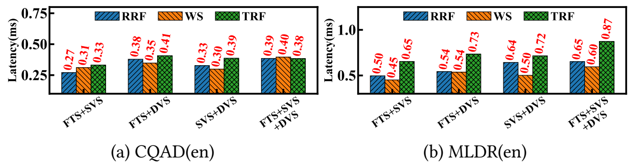


Figure 12: Retrieval latency under RRF, WS, and TRF.

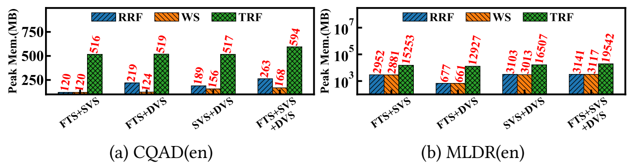


Figure 13: Memory overhead across re-ranking methods.

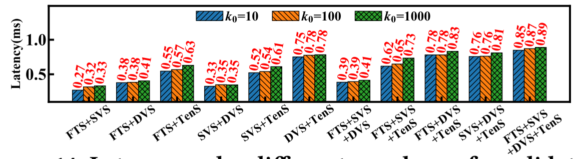


Figure 14: Latency under different numbers of candidates.

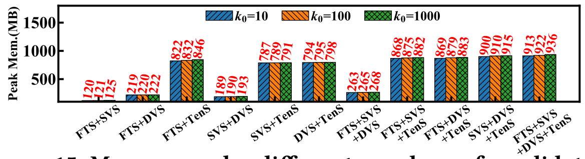


Figure 15: Memory under different numbers of candidates.

and WS methods (Figure 6, Table 2). While RRF and WS perform fusion based on coarse, document-level signals (ranks or scores), TRF performs a fine-grained re-scoring using the underlying token embeddings. It leverages the MaxSim computation to explicitly model the token-to-token interactions between the query and each candidate, verifying their contextual alignment. This acts as a powerful semantic verification step, allowing TRF to identify true positives that these coarser methods might miss. This effect is pronounced on the DBPE(en) dataset for the FTS+DVS combination, where switching from RRF (nDCG@10 of 0.668) to TRF (0.722) yields an 8.1% accuracy improvement. This highlights a key architectural insight: the power of tensor-based late interaction can be harnessed more efficiently as a re-ranking mechanism on a small set of promising candidates, rather than as a primary retrieval path.

This superior accuracy, however, comes with increased computational and memory overhead compared to the lightweight fusion methods. As shown in Figure 12, TRF consistently exhibits higher query latency than both RRF and WS. This is because RRF and WS operate only on pre-computed ranks and scores, while TRF must load and process tensor data from disk to compute the MaxSim score for each candidate. Similarly, TRF's peak memory consumption is noticeably higher as it requires loading candidate tensor data into memory (Figure 13). Crucially, however, this overhead is substantially lower than that of including TenS as a primary retrieval path. The reason for this efficiency gain lies in the difference in operational scope. A full TenS path must search through the massive, corpus-wide index of all token embeddings to find its top-k candidates. In contrast, TRF operates only on the small, pre-filtered set of candidates (e.g., the top 100) provided by the cheaper initial retrieval paths. It completely avoids the cost of a large-scale search, only loading the tensor data for the most promising documents.

This architectural difference leads to dramatic resource savings; for example, when used with the FTS+DVS combination on MLDR(en), TRF reduces peak memory usage by 86% compared to including TenS as a primary retrieval path.

The performance overhead is also a direct function of the number of candidates (k_0) being re-ranked, offering a clear tuning parameter. As shown in Figures 14 and 15, both query latency and peak memory consumption for TRF scale predictably with an increasing k_0. This allows system designers to directly balance accuracy against latency and memory budgets by adjusting the candidate set size. These results position TRF as a promising and practical "middle-ground" in the re-ranking landscape, even in its current, unoptimized implementation. It provides a significant accuracy boost over traditional fusion methods without incurring the prohibitive resource costs of a full TenS path. This highlights a promising direction for future research focused on optimizing TRF's resource footprint. Our follow-up work already indicates that with techniques like scalar or binary quantization [15, 50], TRF's high effectiveness can be largely maintained even with tensor compression rates of up to 32×, dramatically enhancing its practicality.

6 DISCUSSION AND FUTURE WORK

Building upon the extensive experimental results from our benchmark, this section provides a deeper analysis of our findings and their broader implications for the database community. We first discuss our three principal findings, clarifying their novelty and offering practical guidance for system architects. We then use these insights to articulate several key challenges and promising directions for future research in hybrid search.

6.1 Principal Findings and Their Implications

This section provides a detailed discussion of our three principal findings, clarifying their novelty and practical importance. To provide a high-level overview of the complex trade-offs involved, we begin with a qualitative summary of our results in Table 3.

6.1.1 The "Weakest Link": A Principle for Robust Hybrid Search Architectures. The high-level summary in Table 3 suggests a general correlation: more paths often lead to a higher accuracy rating. However, the retrieval accuracy of a hybrid system is often constrained by its least effective component, sometimes to the extent that the combination underperforms its own best constituent path. While the general concept of an ensemble being limited by its inputs is intuitive, our work provides the first systematic, quantitative evidence of this "weakest link" phenomenon in the context of complex hybrid search architectures. This finding challenges the common "more is better" assumption for combining retrieval paths.

The primary implication of this finding is that hybrid architectures must be treated as sensitive systems, not as simple ensembles where strengths are guaranteed to aggregate. This finding underscores the symbiotic relationship between the initial retrieval (candidate generation) and final re-ranking stages. A powerful reranker is crucial for applying deep semantic analysis to precisely order the most promising candidates. However, its effectiveness is fundamentally constrained by the quality of its inputs; it cannot recover relevant documents that were missed during the initial candidate generation. Thus, the quality of the initial retrieval acts as a hard ceiling on the final accuracy of the entire system. Therefore, a critical principle for robust system design is the need for rigorous, component-level performance validation before integrating a new retrieval path, a point we will revisit in Section 6.2.

6.1.2 The Trade-off Map: From Intuition to Evidence-Based Design. Table 3 illustrates a clear pattern: architectures can be optimized to excel in specific dimensions, such as the state-of-the-art accuracy of TenS-inclusive paths or the high query speed of a single DVS path. This reveals a fundamental trade-off: no single architecture is universally optimal, as strengths in one dimension invariably come at the cost of weaknesses in others. Historically, navigating these complex, multi-dimensional trade-offs has been challenging. Architectural choices have often relied on intuition or been informed by ad-hoc studies of simple, pairwise combinations. Our work provides a more principled foundation, allowing the design process to move from conjecture to a quantitative assessment of a configuration's specific performance profile.

This evidence-based foundation has practical implications for system architects. It suggests that a static, one-size-fits-all hybrid architecture is likely suboptimal for diverse workloads. Instead, our findings highlight the need for systems to adaptively select a path combination that best aligns with specific query types, data characteristics, or performance requirements. The trade-off map can also reveal particularly effective architectural patterns. For example, our analysis highlights the FTS+SVS+DVS combination as a highly balanced architecture. This three-path system achieves a high accuracy rating without the severe efficiency trade-offs of TenS-inclusive configurations, identifying a practical "sweet spot" for a wide range of applications that require strong accuracy but cannot tolerate extreme resource demands.

6.1.3 TRF: A Practical Architecture for In-Database Fine-Grained Reranking. Our third principal finding is that Tensor-based Re-ranking Fusion (TRF) provides a practical and highly effective architecture for in-database re-ranking. This positions TRF as a novel architectural pattern that fills a crucial gap in the current re-ranking landscape. As our results in Section 5.3 show, it consistently and substantially outperforms the cheap but coarse-grained fusion methods like RRF and WS. At the same time, it is architecturally distinct from two other high-accuracy approaches. Unlike a full TenS retrieval path, TRF operates only on a small candidate set, making it far more resource-efficient. Furthermore, unlike heavyweight cross-encoder models [63, 136] that typically require a separate, often GPU-based processing stage, TRF is designed as an in-database operation.

The primary implication of this finding is that TRF offers a clear upgrade path for the many systems currently relying on traditional fusion methods. For a moderate increase in latency and memory, practitioners can achieve a significant boost in retrieval accuracy, making TRF an attractive option for applications where accuracy is a key differentiator. It is important to note that these strong results were achieved with our baseline, unoptimized TRF implementation. Its high performance, coupled with the significant potential for future optimization via techniques like tensor compression, marks it as a highly promising direction for the next generation of database retrieval systems. Further optimization of TRF is a key research direction, which we will discuss next.

Table 3: A qualitative summary of hybrid search architecture trade-offs. Ratings are on a 1-to-5 star scale (e.g., ★★★★☆), where more black stars indicate better performance in that dimension (higher accuracy or higher efficiency/lower cost). All ratings are relative to the methods evaluated in this study, using RRF as the baseline re-ranker.

Architecture Type	Retrieval Path	Retrieval Accuracy	Retrieval Efficiency	Memory Overhead	Indexing Cost
Single-Path	FTS	★★☆☆☆	★★★★★	★★★★★	★★★★★
	SVS	★★☆☆☆	★★★★★	★★★★★	★★★★★
	DVS	★★☆☆☆	★★★★★	★★★★★	★★★★★
	TenS	★★★☆☆	★★☆☆☆	★★☆☆☆	★☆☆☆☆
Two-Path	FTS+SVS	★★★☆☆	★★★★☆	★★★★☆	★★★★☆
	FTS+DVS	★★★☆☆	★★★★☆	★★★★☆	★★★★☆
	FTS+TenS	★★★★☆	★★☆☆☆	★★☆☆☆	★☆☆☆☆
	SVS+DVS	★★★☆☆	★★★★☆	★★★★☆	★★★★☆
	SVS+TenS	★★★★☆	★★☆☆☆	★★☆☆☆	★☆☆☆☆
	DVS+TenS	★★★★☆	★★☆☆☆	★★☆☆☆	★☆☆☆☆
Three-Path	FTS+SVS+DVS	★★★★☆	★★★☆☆	★★★☆☆	★★★☆☆
	FTS+SVS+TenS	★★★★★	★★☆☆☆	★★☆☆☆	★☆☆☆☆
	FTS+DVS+TenS	★★★★★	★★☆☆☆	★★☆☆☆	★☆☆☆☆
	SVS+DVS+TenS	★★★★★	★★☆☆☆	★★☆☆☆	★☆☆☆☆
Four-Path	FTS+SVS+DVS+TenS	★★★★★	★☆☆☆☆	★★☆☆☆	★☆☆☆☆

6.2 Challenges and Future Directions

Our findings illuminate several key challenges that open up promising directions for future research. In this section, we articulate three related directions: the development of adaptive architectures, the performance optimization of tensor-based re-ranking, and the expansion of hybrid search to visual documents and into more complex system evaluations like end-to-end RAG.

6.2.1 Towards Adaptive Hybrid Search. Our findings that a "weakest link" can degrade performance (Finding 1) and that no single architecture is universally optimal (Finding 2) both point to a critical limitation in current systems: they are typically static. A fixed combination of retrieval paths is unlikely to be optimal for all queries or across all data subsets. This highlights a major research direction: the development of adaptive hybrid search architectures. Future systems could incorporate a cost-based query optimizer that, for a given query, predicts the likely effectiveness and resource cost of each available retrieval path. Based on these predictions and any user-defined constraints (e.g., a latency budget), the optimizer could then dynamically select the optimal set of paths to execute, potentially pruning or down-weighting a predicted "weak link" at runtime. Moving from static to query-aware execution is a crucial step towards building robust hybrid search systems.

6.2.2 Optimizing the Cost-Performance of Tensor-based Re-ranking. Our third finding highlights TRF as a high-efficacy method that significantly outperforms traditional fusion. However, its broader adoption is currently limited by its higher computational and memory costs compared to methods like RRF. This presents a valuable direction: optimizing the cost-performance of tensor-based re-ranking. A promising avenue is tensor compression. As our follow-up work suggests, techniques like scalar or binary quantization can dramatically reduce the memory footprint of tensor representations while largely preserving re-ranking effectiveness. Complementing this, research into computation acceleration for the MaxSim operation is also critical. This could involve leveraging hardware-specific instructions like SIMD or developing more efficient algorithms for the token-level interaction. Success in these areas could eliminate the primary barriers to TRF's adoption, establishing it as a new standard for high-accuracy re-ranking in modern databases.

6.2.3 Expanding the Hybrid Search Frontier. While our study provides a foundation for textual hybrid search, the frontier of information retrieval is expanding to visual information and more complex system integrations. A significant future direction is extending hybrid search principles to multi-modal documents. This involves investigating the synergy between emerging visual-semantic models like ColPali [43] for visual data and traditional lexical methods operating on text extracted via Optical Character Recognition (OCR) [104]. Another vital research direction is to evaluate these advanced hybrid search architectures within an end-to-end RAG framework. Such a study would need to consider the complex interplay between the retrieval module and other key components, such as document chunking strategies and query intent recognition. Understanding these complex interactions is the next critical step in building optimized and effective RAG systems.

7 CONCLUSION

In this paper, we conducted the first systematic study of advanced hybrid search architectures, a critical but poorly understood area in modern database systems. We designed and implemented a novel evaluation framework to comprehensively benchmark four major retrieval paradigms and their 15 combinations across 11 diverse datasets. Our analysis yielded three principal findings: (1) the "weakest link" phenomenon, where a system's accuracy is constrained by its least effective component; (2) a data-driven map of the performance landscape, which demonstrates that architectural selection is a multi-dimensional trade-off with no universal solution; and (3) the emergence of TRF as a promising re-ranking architecture. We believe that our framework, benchmark results, and the design principles will provide a crucial foundation for both practitioners building the next generation of retrieval systems and for researchers exploring the future of adaptive hybrid search.

ACKNOWLEDGMENTS

The evaluation framework presented in this paper was significantly informed by the architectural principles and practical experience gained from the development of the `Infinity` open-source database[1]. We are deeply grateful to our colleagues on the Infinity team for their foundational work and insightful discussions. Specifically, we would like to extend our sincere thanks to Zhichang Yu, Yushi Shen, Zhiqiang Yang, Ling Qin, and Yi Xiao for their invaluable contributions to the `Infinity` project.

REFERENCES

[1] 2008. *Full-Text Search.* Apress, Berkeley, CA, 217–237.
[2] 2014. Small but Mighty: Introducing answerai-colbert-small. https://www.answer.ai/posts/2024-08-13-small-but-mighty-colbert.html. [Online; accessed 05-November-2024].
[3] 2015. CQADupStack. http://nlp.cis.unimelb.edu.au/resources/cqadupstack/. [Online; accessed 02-November-2024].
[4] 2018. Financial Opinion Mining and Question Answering. https://sites.google.com/view/fiqa/. [Online; accessed 02-November-2024].
[5] 2023. Getting Started with Hybrid Search. https://www.pinecone.io/learn/hybrid-search-intro/. [Online; accessed 06-February-2025].
[6] 2024. Elasticsearch: The heart of the Elastic Stack. https://www.elastic.co/elasticsearch. [Online; accessed 20-October-2024].
[7] 2024. Full-text search vs vector search. https://www.meilisearch.com/blog/full-text-search-vs-vector-search. [Online; accessed 09-October-2024].
[8] 2024. IK Analysis for Elasticsearch and OpenSearch. https://github.com/infinilabs/analysis-ik. [Online; accessed 20-October-2024].
[9] 2024. Standard tokenizer. https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis-standard-tokenizer.html. [Online; accessed 10-December-2024].
[10] 2025. Hybrid Search Explained. https://weaviate.io/blog/hybrid-search-explained. [Online; accessed 06-February-2025].
[11] 2025. Hybrid Search with Milvus. https://milvus.io/docs/hybrid_search_with_milvus.md. [Online; accessed 06-February-2025].
[12] 2025. Relevance scoring in hybrid search using Reciprocal Rank Fusion (RRF). https://learn.microsoft.com/en-us/azure/search/hybrid-search-ranking. [Online; accessed 20-March-2025].
[13] 2025. Reranking. https://milvus.io/docs/reranking.md. [Online; accessed 20-March-2025].
[14] 2025. Supplementary Materials: Detailed Experimental Results. https://github.com/whenever5225/infinity/tree/main/exps/results. [Online; accessed 28-July-2025].
[15] Cecilia Aguerrebere, Ishwar Singh Bhati, Mark Hildebrand, Mariano Tepper, and Theodore L. Willke. 2023. Similarity search in the blink of an eye with compressed indices. *Proc. VLDB Endow.* 16, 11 (2023), 3433–3446.
[16] Gianni Amati and C. J. van Rijsbergen. 2002. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Trans. Inf. Syst.* 20, 4 (2002), 357–389.
[17] Fabien André, Anne-Marie Kermarrec, and Nicolas Le Scouarnec. 2015. Cache locality is not enough: High-Performance Nearest Neighbor Search with Product Quantization Fast Scan. *Proc. VLDB Endow.* 9, 4 (2015), 288–299.
[18] Ilias Azizi, Karima Echihabi, and Themis Palpanas. 2025. Graph-Based Vector Search: An Experimental Evaluation of the State-of-the-Art. *Proc. ACM Manag. Data* 3, 1 (2025), 43:1–43:31.
[19] Alexander Bondarenko, Maik Fröbe, Meriem Beloucif, Lukas Gienapp, Yamen Ajjour, Alexander Panchenko, Chris Biemann, Benno Stein, Henning Wachsmuth, Martin Potthast, and Matthias Hagen. 2020. Overview of Touché 2020: Argument Retrieval. In *Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum, Thessaloniki, Greece, September 22-25, 2020 (CEUR Workshop Proceedings)*, Vol. 2696.
[20] Andrei Z. Broder, David Carmel, Michael Herscovici, Aya Soffer, and Jason Y. Zien. 2003. Efficient query evaluation using a two-level retrieval process. In *Proceedings of the 2003 ACM CIKM International Conference on Information and Knowledge Management, New Orleans, Louisiana, USA, November 2-8, 2003.* 426–434.
[21] Sebastian Bruch, Siyu Gai, and Amir Ingber. 2024. An Analysis of Fusion Functions for Hybrid Retrieval. *ACM Trans. Inf. Syst.* 42, 1 (2024), 20:1–20:35.
[22] 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 (2024), 151:1–151:38.
[23] Sebastian Bruch, Franco Maria Nardini, Cosimo Rulli, and Rossano Venturini. 2024. Efficient Inverted Indexes for Approximate Retrieval over Learned Sparse Representations. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024.* 152–162.
[24] Sebastian Bruch, Franco Maria Nardini, Cosimo Rulli, and Rossano Venturini. 2024. Pairing Clustered Inverted Indexes with κ-NN Graphs for Fast Approximate Retrieval over Learned Sparse Representations. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, CIKM 2024, Boise, ID, USA, October 21-25, 2024.* 3642–3646.
[25] Sebastian Bruch, Franco Maria Nardini, Cosimo Rulli, Rossano Venturini, and Leonardo Venuta. 2025. Investigating the Scalability of Approximate Sparse Retrieval Algorithms to Massive Datasets. *CoRR* abs/2501.11628 (2025).
[26] Yuzheng Cai, Jiayang Shi, Yizhuo Chen, and Weiguo Zheng. 2024. Navigating Labels and Vectors: A Unified Approach to Filtered Approximate Nearest Neighbor Search. *Proc. ACM Manag. Data* 2, 6 (2024), 246:1–246:27.
[27] Kaushik Chakrabarti, Surajit Chaudhuri, and Venkatesh Ganti. 2011. Interval-based pruning for top-k processing over compressed lists. In *Proceedings of the 27th International Conference on Data Engineering, ICDE 2011, April 11-16, 2011, Hannover, Germany.* 709–720.
[28] Cheng Chen, Chenzhe Jin, Yunan Zhang, Sasha Podolsky, Chun Wu, Szu-Po Wang, Eric Hanson, Zhou Sun, Robert Walzer, and Jianguo Wang. 2024. SingleStore-V: An Integrated Vector Database System in SingleStore. *Proc. VLDB Endow.* 17, 12 (2024), 3772–3785.
[29] Haonan Chen, Carlos Lassance, and Jimmy Lin. 2023. End-to-End Retrieval with Learned Dense and Sparse Representations Using Lucene. *CoRR* abs/2311.18503 (2023).
[30] Jianlyu Chen, Nan Wang, Chaofan Li, Bo Wang, Shitao Xiao, Han Xiao, Hao Liao, Defu Lian, and Zheng Liu. 2024. AIR-Bench: Automated Heterogeneous Information Retrieval Benchmark. *CoRR* abs/2412.13102 (2024).
[31] Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. BGE M3-Embedding: Multi-Lingual, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation. *CoRR* abs/2402.03216 (2024).
[32] Sibei Chen, Ju Fan, Bin Wu, Nan Tang, Chao Deng, Pengyi Wang, Ye Li, Jian Tan, Feifei Li, Jingren Zhou, and Xiaoyong Du. 2025. Automatic Database Configuration Debugging using Retrieval-Augmented Language Models. *Proc. ACM Manag. Data* 3, 1 (2025), 13:1–13:27.
[33] 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 *Advances in Information Retrieval - 44th European Conference on IR Research, ECIR 2022, Stavanger, Norway, April 10-14, 2022, Proceedings, Part I (Lecture Notes in Computer Science)*, Vol. 13185. 95–110.
[34] Yaoqi Chen, Ruicheng Zheng, Qi Chen, Shuotao Xu, Qianxi Zhang, Xue Wu, Weihao Han, Hua Yuan, Mingqin Li, Yujing Wang, Jason Li, Fan Yang, Hao Sun, Weiwei Deng, Feng Sun, Qi Zhang, and Mao Yang. 2024. OneSparse: A Unified System for Multi-index Vector Search. In *Companion Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, Singapore, May 13-17, 2024.* 393–402.
[35] Benjamin Clavié. 2024. JaColBERTv2.5: Optimising Multi-Vector Retrievers to Create State-of-the-Art Japanese Retrievers with Constrained Resources. *CoRR* abs/2407.20750 (2024).
[36] Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel S. Weld. 2020. SPECTER: Document-level Representation Learning using Citation-informed Transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020.* 2270–2282.
[37] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised Cross-lingual Representation Learning at Scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020.* 8440–8451.
[38] Gordon V. Cormack, Charles L. A. Clarke, and Stefan Büttcher. 2009. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2009, Boston, MA, USA, July 19-23, 2009.* 758–759.
[39] Matt Crane, J. Shane Culpepper, Jimmy Lin, Joel M. Mackenzie, and Andrew Trotman. 2017. A Comparison of Document-at-a-Time and Score-at-a-Time Query Evaluation. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM 2017, Cambridge, United Kingdom, February 6-10, 2017.* 201–210.
[40] Constantinos Dimopoulos, Sergey Nepomnyachiy, and Torsten Suel. 2013. A candidate filtering mechanism for fast top-k query processing on modern cpus. In *The 36th International ACM SIGIR conference on research and development in Information Retrieval, SIGIR '13, Dublin, Ireland - July 28 - August 01, 2013.* 723–732.
[41] Shuai Ding and Torsten Suel. 2011. Faster top-k document retrieval using block-max indexes. In *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, Beijing, China, July 25-29, 2011.* 993–1002.

[42] Xin Luna Dong. 2024. The Journey to a Knowledgeable Assistant with Retrieval-Augmented Generation (RAG). In *Companion of the 2024 International Conference on Management of Data, SIGMOD/PODS 2024, Santiago AA, Chile, June 9-15, 2024*. 3.

[43] Manuel Faysse, Hugues Sibille, Tony Wu, Bilel Omrani, Gautier Viaud, Céline Hudelot, and Pierre Colombo. 2025. ColPali: Efficient Document Retrieval with Vision Language Models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*.

[44] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE v2: Sparse Lexical and Expansion Model for Information Retrieval. *CoRR* abs/2109.10086 (2021).

[45] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking. In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*. 2288–2292.

[46] Cong Fu, Chao Xiang, Changxu Wang, and Deng Cai. 2019. Fast Approximate Nearest Neighbor Search With The Navigating Spreading-out Graph. *Proc. VLDB Endow.* 12, 5 (2019), 461–474.

[47] Debasis Ganguly, Dwaipayan Roy, Mandar Mitra, and Gareth J. F. Jones. 2015. Word Embedding based Generalized Language Model for Information Retrieval. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015*. 795–798.

[48] Jinyang Gao, H. V. Jagadish, Wei Lu, and Beng Chin Ooi. 2014. DSH: data sensitive hashing for high-dimensional k-nnsearch. In *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*. 1127–1138.

[49] Jianyang Gao and Cheng Long. 2023. High-Dimensional Approximate Nearest Neighbor Search: with Reliable and Efficient Distance Comparison Operations. *Proc. ACM Manag. Data* 1, 2 (2023), 137:1–137:27.

[50] Jianyang Gao and Cheng Long. 2024. RaBitQ: Quantizing High-Dimensional Vectors with a Theoretical Error Bound for Approximate Nearest Neighbor Search. *Proc. ACM Manag. Data* 2, 3 (2024), 167.

[51] 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: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*. 3030–3042.

[52] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. 1999. Similarity Search in High Dimensions via Hashing. In *VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK*. 518–529.

[53] Adrien Grand, Robert Muir, Jim Ferenczi, and Jimmy Lin. 2020. From MAXSCORE to Block-Max Wand: The Story of How Lucene Significantly Improved Query Evaluation Performance. In *Advances in Information Retrieval - 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14-17, 2020, Proceedings, Part II*, Vol. 12036. 20–27.

[54] Christophe Van Gysel, Maarten de Rijke, and Evangelos Kanoulas. 2018. Neural Vector Spaces for Unsupervised Information Retrieval. *ACM Trans. Inf. Syst.* 36, 4 (2018), 38:1–38:25.

[55] Marios Hadjieleftheriou, Nick Koudas, and Divesh Srivastava. 2009. Incremental maintenance of length normalized indexes for approximate string matching. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2009, Providence, Rhode Island, USA, June 29 - July 2, 2009*. 429–440.

[56] Faegheh Hasibi, Fedor Nikolaev, Chenyan Xiong, Krisztian Balog, Svein Erik Bratsberg, Alexander Kotov, and Jamie Callan. 2017. DBpedia-Entity v2: A Test Collection for Entity Search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*. 1265–1268.

[57] Vagelis Hristidis, Yuheng Hu, and Panagiotis G. Ipeirotis. 2010. Ranked queries over sources with Boolean query interfaces without ranking support. In *Proceedings of the 26th International Conference on Data Engineering, ICDE 2010, March 1-6, 2010, Long Beach, California, USA*. 872–875.

[58] Qiang Huang, Jianlin Feng, Yikai Zhang, Qiong Fang, and Wilfred Ng. 2015. Query-Aware Locality-Sensitive Hashing for Approximate Nearest Neighbor Search. *Proc. VLDB Endow.* 9, 1 (2015), 1–12.

[59] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. 2011. Product Quantization for Nearest Neighbor Search. *IEEE Trans. Pattern Anal. Mach. Intell.* 33, 1 (2011), 117–128.

[60] Rohan Jha, Bo Wang, Michael Günther, Saba Sturua, Mohammad Kalim Akram, and Han Xiao. 2024. Jina-ColBERT-v2: A General-Purpose Multilingual Late Interaction Retriever. *CoRR* abs/2408.16672 (2024).

[61] Wenqi Jiang, Marco Zeller, Roger Waleffe, Torsten Hoefler, and Gustavo Alonso. 2024. Chameleon: a Heterogeneous and Disaggregated Accelerator System for Retrieval-Augmented Language Models. *Proc. VLDB Endow.* 18, 1 (2024), 42–52.

[62] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. 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, EMNLP 2020, Online, November 16-20, 2020*. 6769–6781.

[63] Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*. 39–48.

[64] Carlos Lassance, Hervé Déjean, Thibault Formal, and Stéphane Clinchant. 2024. SPLADE-v3: New baselines for SPLADE. *CoRR* abs/2403.06789 (2024).

[65] Conglong Li, Minjia Zhang, David G. Andersen, and Yuxiong He. 2020. Improving Approximate Nearest Neighbor Search through Learned Adaptive Early Termination. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*. 2539–2554.

[66] Minghan Li, Sheng-Chieh Lin, Barlas Oguz, Asish Ghoshal, Jimmy Lin, Yashar Mehdad, Wen-tau Yih, and Xilun Chen. 2023. CITADEL: Conditional Token Interaction via Dynamic Lexical Routing for Efficient and Effective Multi-Vector Retrieval. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*. 11891–11907.

[67] Wen Li, Ying Zhang, Yifang Sun, Wei Wang, Mingjie Li, Wenjie Zhang, and Xuemin Lin. 2020. Approximate Nearest Neighbor Search on High Dimensional Data - Experiments, Analyses, and Improvement. *IEEE Trans. Knowl. Data Eng.* 32, 8 (2020), 1475–1488.

[68] Xiangyang Li, Kuicai Dong, Yi Quan Lee, Wei Xia, Yichun Yin, Hao Zhang, Yong Liu, Yasheng Wang, and Ruiming Tang. 2024. CoIR: A Comprehensive Benchmark for Code Information Retrieval Models. *CoRR* abs/2407.02883 (2024).

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

[70] Yingfan Liu, Jiangtao Cui, Zi Huang, Hui Li, and Heng Tao Shen. 2014. SK-LSH: An Efficient Index Structure for Approximate Nearest Neighbor Search. *Proc. VLDB Endow.* 7, 9 (2014), 745–756.

[71] Antoine Louis, Vageesh Kumar Saxena, Gijs van Dijck, and Gerasimos Spanakis. 2025. ColBERT-XM: A Modular Multi-Vector Representation Model for Zero-Shot Multilingual Information Retrieval. In *Proceedings of the 31st International Conference on Computational Linguistics, COLING 2025, Abu Dhabi, UAE, January 19-24, 2025*. 4370–4383.

[72] Kejing Lu, Mineichi Kudo, Chuan Xiao, and Yoshiharu Ishikawa. 2021. HVS: Hierarchical Graph Structure Based on Voronoi Diagrams for Solving Approximate Nearest Neighbor Search. *Proc. VLDB Endow.* 15, 2 (2021), 246–258.

[73] Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2021. Sparse, Dense, and Attentional Representations for Text Retrieval. *Trans. Assoc. Comput. Linguistics* 9 (2021), 329–345.

[74] Ji Ma, Ivan Korotkov, Keith B. Hall, and Ryan T. McDonald. 2020. Hybrid First-stage Retrieval Models for Biomedical Literature. In *Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum, Thessaloniki, Greece, September 22-25, 2020 (CEUR Workshop Proceedings)*, Vol. 2696.

[75] Ji Ma, Ivan Korotkov, Yinfei Yang, Keith B. Hall, and Ryan T. McDonald. 2021. Zero-shot Neural Passage Retrieval via Domain-targeted Synthetic Question Generation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*. 1075–1088.

[76] Joel Mackenzie, Antonio Mallia, Alistair Moffat, and Matthias Petri. 2022. Accelerating Learned Sparse Indexes Via Term Impact Decomposition. In *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*. Association for Computational Linguistics, 2830–2842.

[77] Joel Mackenzie, Matthias Petri, and Alistair Moffat. 2022. Anytime Ranking on Document-Ordered Indexes. *ACM Trans. Inf. Syst.* 40, 1 (2022), 13:1–13:32.

[78] Yury A. Malkov and Dmitry A. Yashunin. 2020. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* 42, 4 (2020), 824–836.

[79] Antonio Mallia, Giuseppe Ottaviano, Elia Porciani, Nicola Tonellotto, and Rossano Venturini. 2017. Faster BlockMax WAND with Variable-sized Blocks. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*. 625–634.

[80] Antonio Mallia, Michal Siedlaczek, and Torsten Suel. 2019. An Experimental Study of Index Compression and DAAT Query Processing Methods. In *Advances in Information Retrieval - 41st European Conference on IR Research, ECIR 2019, Cologne, Germany, April 14-18, 2019, Proceedings, Part I (Lecture Notes in Computer Science)*, Vol. 11437. 353–368.

[81] Antonio Mallia, Michal Siedlaczek, and Torsten Suel. 2021. Fast Disjunctive Candidate Generation Using Live Block Filtering. In *WSDM '21, The Fourteenth ACM International Conference on Web Search and Data Mining, Virtual Event, Israel, March 8-12, 2021*. 671–679.

[82] Antonio Mallia, Torsten Suel, and Nicola Tonellotto. 2024. Faster Learned Sparse Retrieval with Block-Max Pruning. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024.* 2411–2415.

[83] Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2023. MTEB: Massive Text Embedding Benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2023, Dubrovnik, Croatia, May 2-6, 2023.* 2006–2029.

[84] Franco Maria Nardini, Cosimo Rulli, and Rossano Venturini. 2024. Efficient Multi-vector Dense Retrieval with Bit Vectors. In *Advances in Information Retrieval - 46th European Conference on Information Retrieval, ECIR 2024, Glasgow, UK, March 24-28, 2024, Proceedings, Part II*, Vol. 14609. 3–17.

[85] Thong Nguyen, Sean MacAvaney, and Andrew Yates. 2023. A Unified Framework for Learned Sparse Retrieval. In *Advances in Information Retrieval - 45th European Conference on Information Retrieval, ECIR 2023, Dublin, Ireland, April 2-6, 2023, Proceedings, Part III (Lecture Notes in Computer Science)*, Vol. 13982. 101–116.

[86] 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. In *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016 (CEUR Workshop Proceedings)*, Vol. 1773.

[87] Jiaul H. Paik. 2013. A novel TF-IDF weighting scheme for effective ranking. In *The 36th International ACM SIGIR conference on research and development in Information Retrieval, SIGIR '13, Dublin, Ireland - July 28 - August 01, 2013.* 343–352.

[88] James Jie Pan, Jianguo Wang, and Guoliang Li. 2024. Survey of vector database management systems. *VLDB J.* 33, 5 (2024), 1591–1615.

[89] Cheoneum Park, Seohyeong Jeong, Minsang Kim, KyungTae Lim, and Yong-Hun Lee. 2025. SCV: Light and Effective Multi-Vector Retrieval with Sequence Compressive Vectors. In *Proceedings of the 31st International Conference on Computational Linguistics: Industry Track.* 760–770.

[90] Derek Paulsen, Yash Govind, and AnHai Doan. 2023. Sparkly: A Simple yet Surprisingly Strong TF/IDF Blocker for Entity Matching. *Proc. VLDB Endow.* 16, 6 (2023), 1507–1519.

[91] Yun Peng, Byron Choi, Tsz Nam Chan, Jianye Yang, and Jianliang Xu. 2023. Efficient Approximate Nearest Neighbor Search in Multi-dimensional Databases. *Proc. ACM Manag. Data* 1, 1 (2023), 54:1–54:27.

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

[93] Stefan Pohl, Alistair Moffat, and Justin Zobel. 2012. Efficient Extended Boolean Retrieval. *IEEE Trans. Knowl. Data Eng.* 24, 6 (2012), 1014–1024.

[94] Giovanni Puccetti, Alessio Miaschi, and Felice Dell'Orletta. 2021. How Do BERT Embeddings Organize Linguistic Knowledge?. In *Proceedings of Deep Learning Inside Out: The 2nd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures, DeeLIO@NAACL-HLT 2021, Online, June 10 2021.* 48–57.

[95] Yifan Qiao, Yingrui Yang, Haixin Lin, and Tao Yang. 2023. Optimizing Guided Traversal for Fast Learned Sparse Retrieval. In *Proceedings of the ACM Web Conference 2023, WWW 2023, Austin, TX, USA, 30 April 2023 - 4 May 2023.* 3375–3385.

[96] Tadeusz Radecki. 1988. Trends in research on information retrieval – The potential for improvements in conventional Boolean retrieval systems. *Inf. Process. Manag.* 24, 3 (1988), 219–227.

[97] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019.* 3980–3990.

[98] Antoinette Renouf, Andrew Kehoe, and Jayeeta Banerjee. 2007. WebCorp: an integrated system for web text search. In *Corpus linguistics and the web.* 47–67.

[99] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at TREC-3. In *Proceedings of The Third Text REtrieval Conference, TREC 1994, Gaithersburg, Maryland, USA, November 2-4, 1994 (NIST Special Publication)*, Vol. 500-225. National Institute of Standards and Technology (NIST), 109–126.

[100] Keshav Santhanam, Omar Khattab, Christopher Potts, and Matei Zaharia. 2022. PLAID: An Efficient Engine for Late Interaction Retrieval. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management, Atlanta, GA, USA, October 17-21, 2022.* 1747–1756.

[101] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022. ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022.* 3715–3734.

[102] Kunal Sawarkar, Abhilasha Mangal, and Shivam Raj Solanki. 2024. Blended RAG: Improving RAG (Retriever-Augmented Generation) Accuracy with Semantic Search and Hybrid Query-Based Retrievers. In *7th IEEE International Conference on Multimedia Information Processing and Retrieval, MIPR 2024, San Jose, CA, USA, August 7-9, 2024.* 155–161.

[103] Chanop Silpa-Anan and Richard I. Hartley. 2008. Optimised KD-trees for fast image descriptor matching. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008), 24-26 June 2008, Anchorage, Alaska, USA.*

[104] R. Smith. 2007. An Overview of the Tesseract OCR Engine. In *9th International Conference on Document Analysis and Recognition (ICDAR 2007), 23-26 September, Curitiba, Paraná, Brazil.* IEEE Computer Society, 629–633.

[105] Suhas Jayaram Subramanya, Devvrit, Harsha Vardhan Simhadri, Ravishankar Krishnaswamy, and Rohan Kadekodi. 2019. DiskANN: Fast Accurate Billion-point Nearest Neighbor Search on a Single Node. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada.* 13748–13758.

[106] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. [n.d.]. BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2).*

[107] Nicola Tonellotto, Craig Macdonald, and Iadh Ounis. 2018. Efficient Query Processing for Scalable Web Search. *Found. Trends Inf. Retr.* 12, 4-5 (2018), 319–500.

[108] Andrew Trotman and David Keeler. 2011. Ad hoc IR: not much room for improvement. In *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, Beijing, China, July 25-29, 2011.* 1095–1096.

[109] Andrew Trotman, Antti Puurula, and Blake Burgess. 2014. Improvements to BM25 and Language Models Examined. In *Proceedings of the 2014 Australasian Document Computing Symposium, ADCS 2014, Melbourne, VIC, Australia, November 27-28, 2014.* 58.

[110] Howard R. Turtle and James Flood. 1995. Query Evaluation: Strategies and Optimizations. *Inf. Process. Manag.* 31, 6 (1995), 831–850.

[111] Ellen M. Voorhees, Tasmeer Alam, Steven Bedrick, Dina Demner-Fushman, William R. Hersh, Kyle Lo, Kirk Roberts, Ian Soboroff, and Lucy Lu Wang. 2020. TREC-COVID: constructing a pandemic information retrieval test collection. *SIGIR Forum* 54, 1 (2020), 1:1–1:12.

[112] David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. 2020. Fact or Fiction: Verifying Scientific Claims. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020.* 7534–7550.

[113] Jianguo Wang, Chunbin Lin, Ruining He, Moojin Chae, Yannis Papakonstantinou, and Steven Swanson. 2017. MILC: Inverted List Compression in Memory. *Proc. VLDB Endow.* 10, 8 (2017), 853–864.

[114] Jianguo Wang, Chunbin Lin, Yannis Papakonstantinou, and Steven Swanson. 2017. An Experimental Study of Bitmap Compression vs. Inverted List Compression. In *Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD Conference 2017, Chicago, IL, USA, May 14-19, 2017.* 993–1008.

[115] Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, Kun Yu, Yuxing Yuan, Yinghao Zou, Jiquan Long, Yudong Cai, Zhenxiang Li, Zhifeng Zhang, Yihua Mo, Jun Gu, Ruiyi Jiang, Yi Wei, and Charles Xie. 2021. Milvus: A Purpose-Built Vector Data Management System. In *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021.* 2614–2627.

[116] Mengzhao Wang, Haotian Wu, Xiangyu Ke, Yunjun Gao, Xiaoliang Xu, and Lu Chen. 2024. An Interactive Multi-modal Query Answering System with Retrieval-Augmented Large Language Models. *Proc. VLDB Endow.* 17, 12 (2024), 4333–4336.

[117] Mengzhao Wang, Weizhi Xu, Xiaomeng Yi, Songlin Wu, Zhangyang Peng, Xiangyu Ke, Yunjun Gao, Xiaoliang Xu, Rentong Guo, and Charles Xie. 2024. Starling: An I/O-Efficient Disk-Resident Graph Index Framework for High-Dimensional Vector Similarity Search on Data Segment. *Proc. ACM Manag. Data* 2, 1 (2024), 14:1–14:27.

[118] Mengzhao Wang, Xiaoliang Xu, Qiang Yue, and Yuxiang Wang. 2021. A Comprehensive Survey and Experimental Comparison of Graph-Based Approximate Nearest Neighbor Search. *Proc. VLDB Endow.* 14, 11 (2021), 1964–1978.

[119] Shuai Wang, Shengyao Zhuang, and Guido Zuccon. 2021. BERT-based Dense Retrievers Require Interpolation with BM25 for Effective Passage Retrieval. In *ICTIR '21: The 2021 ACM SIGIR International Conference on the Theory of Information Retrieval, Virtual Event, Canada, July 11, 2021.* 317–324.

[120] Xiaoyue Wang, Jianyou Wang, Weili Cao, Kaicheng Wang, Ramamohan Paturi, and Leon Bergen. 2024. BIRCO: A Benchmark of Information Retrieval Tasks with Complex Objectives. *CoRR* abs/2402.14151 (2024).

[121] Yifan Wang, Haodi Ma, and Daisy Zhe Wang. 2022. LIDER: An Efficient High-dimensional Learned Index for Large-scale Dense Passage Retrieval. *Proc. VLDB Endow.* 16, 2 (2022), 154–166.

[122] Yining Wang, Liwei Wang, Yuanzhi Li, Di He, and Tie-Yan Liu. 2013. A Theoretical Analysis of NDCG Type Ranking Measures. In *COLT 2013 - The 26th Annual Conference on Learning Theory, June 12-14, 2013, Princeton University, NJ, USA (JMLR Workshop and Conference Proceedings)*, Vol. 30. 25–54.

[123] Zeyu Wang, Qitong Wang, Xiaoxing Cheng, Peng Wang, Themis Palpanas, and Wei Wang. 2024. Steiner-Hardness: A Query Hardness Measure for Graph-Based ANN Indexes. *Proc. VLDB Endow.* 17, 13 (2024), 4668–4682.

[124] Manuel Widmoser, Daniel Kocher, and Nikolaus Augsten. 2024. Scalable Distributed Inverted List Indexes in Disaggregated Memory. *Proc. ACM Manag. Data* 2, 3 (2024), 171.

[125] Shiguang Wu, Wenda Wei, Mengqi Zhang, Zhumin Chen, Jun Ma, Zhaochun Ren, Maarten de Rijke, and Pengjie Ren. 2024. Generative Retrieval as Multi-Vector Dense Retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024.* 1828–1838.

[126] Xiang Wu, Ruiqi Guo, David Simcha, Dave Dopson, and Sanjiv Kumar. 2019. Efficient Inner Product Approximation in Hybrid Spaces. *CoRR* abs/1903.08690 (2019).

[127] Jasper Xian, Tommaso Teofili, Ronak Pradeep, and Jimmy Lin. 2024. Vector Search with OpenAI Embeddings: Lucene Is All You Need. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining, WSDM 2024, Merida, Mexico, March 4-8, 2024.* ACM, 1090–1093.

[128] Zhichao Xu, Fengran Mo, Zhiqi Huang, Crystina Zhang, Puxuan Yu, Bei Wang, Jimmy Lin, and Vivek Srikumar. 2025. A Survey of Model Architectures in Information Retrieval. *CoRR* abs/2502.14822 (2025).

[129] Peilin Yang, Hui Fang, and Jimmy Lin. 2017. Anserini: Enabling the Use of Lucene for Information Retrieval Research. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017.* 1253–1256.

[130] Wen Yang, Tao Li, Gai Fang, and Hong Wei. 2020. PASE: PostgreSQL Ultra-High-Dimensional Approximate Nearest Neighbor Search Extension. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020.* 2241–2253.

[131] Yingrui Yang, Parker Carlson, Shanxiu He, Yifan Qiao, and Tao Yang. 2024. Cluster-based Partial Dense Retrieval Fused with Sparse Text Retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024.* 2327–2331.

[132] Peter N. Yianilos. 1993. Data Structures and Algorithms for Nearest Neighbor Search in General Metric Spaces. In *Proceedings of the Fourth Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms, 25-27 January 1993, Austin, Texas, USA.* 311–321.

[133] Runjie Yu, Weizhou Huang, Shuhan Bai, Jian Zhou, and Fei Wu. 2025. AquaPipe: A Quality-Aware Pipeline for Knowledge Retrieval and Large Language Models. *Proc. ACM Manag. Data* 3, 1 (2025), 11:1–11:26.

[134] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021. Jointly Optimizing Query Encoder and Product Quantization to Improve Retrieval Performance. In *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021.* 2487–2496.

[135] Haoyu Zhang, Jun Liu, Zhenhua Zhu, Shulin Zeng, Maojia Sheng, Tao Yang, Guohao Dai, and Yu Wang. 2024. Efficient and Effective Retrieval of Dense-Sparse Hybrid Vectors using Graph-based Approximate Nearest Neighbor Search. *CoRR* abs/2410.20381 (2024).

[136] Xin Zhang, Yanzhao Zhang, Dingkun Long, Wen Xie, Ziqi Dai, Jialong Tang, Huan Lin, Baosong Yang, Pengjun Xie, Fei Huang, Meishan Zhang, Wenjie Li, and Min Zhang. 2024. mGTE: Generalized Long-Context Text Representation and Reranking Models for Multilingual Text Retrieval. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: EMNLP 2024 - Industry Track, Miami, Florida, USA, November 12-16, 2024.* Association for Computational Linguistics, 1393–1412.

[137] Xi Zhao, Zhonghan Chen, Kai Huang, Ruiyuan Zhang, Bolong Zheng, and Xiaofang Zhou. 2024. Efficient Approximate Maximum Inner Product Search Over Sparse Vectors. In *40th IEEE International Conference on Data Engineering, ICDE 2024, Utrecht, The Netherlands, May 13-16, 2024.* 3961–3974.

[138] Xinyang Zhao, Xuanhe Zhou, and Guoliang Li. 2024. Chat2Data: An Interactive Data Analysis System with RAG, Vector Databases and LLMs. *Proc. VLDB Endow.* 17, 12 (2024), 4481–4484.

[139] Chaoji Zuo, Miao Qiao, Wenchao Zhou, Feifei Li, and Dong Deng. 2024. SeRF: Segment Graph for Range-Filtering Approximate Nearest Neighbor Search. *Proc. ACM Manag. Data* 2, 1 (2024), 69:1–69:26.