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

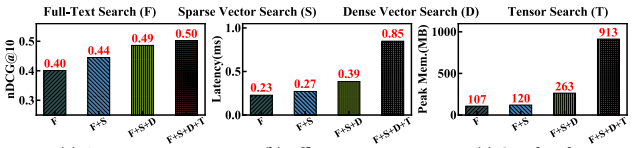
Hybrid search, the integration of lexical and semantic retrieval, has become a cornerstone of modern information retrieval systems, driven by demanding applications like RAG. The design space for these systems is complex, yet a systematic understanding of the trade-offs among their retrieval paradigms, combination schemes, and re-ranking methods is still lacking. To address this, we present the first experimental analysis of advanced hybrid search architectures. Our framework integrates four retrieval paradigms—full-text search, sparse vector search, dense vector search, and tensor search—and evaluates their combinations and re-ranking strategies across 11 real-world datasets. Our results reveal three key findings: (1) A "weakest link" phenomenon, where a weak path can substantially degrade overall accuracy, highlighting the need for path-wise quality assessment before fusion. (2) A data-driven map of performance trade-offs, demonstrating that optimal configurations depend heavily on resource constraints and data characteristics, precluding a one-size-fits-all solution. (3) The identification of tensor-based re-ranking fusion as an 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 adaptive, scalable hybrid search systems and identify key directions for future research.

PVLDB Reference Format:
Mengzhao Wang, Boyu Tan, Yunjun Gao, Hai Jin, Yingfeng Zhang, Xiangyu Ke, Xiaoliang Xu, and Yifan Zhu. Balancing the Blend: An Experimental Analysis of Trade-offs in Hybrid Search. PVLDB, 19(8): 1715 - 1730, 2026.
doi:10.14778/3811243.3811246

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

1 INTRODUCTION

Modern information retrieval tasks increasingly require database systems to move beyond single-paradigm search [111, 144]. Hybrid search architectures, blending multiple retrieval techniques, have emerged as a powerful solution, driven by applications like Retrieval-Augmented Generation (RAG) [39, 49, 142]. However, designing these systems for optimal performance is challenging, as the trade-offs between retrieval quality, efficiency, and cost are complex [69, 125, 149]. This challenge is particularly acute for advanced architectures combining three or more paradigms, as their synergistic benefits and interference effects remain unstudied. This raises an important question: **how can we systematically navigate this complex design space of hybrid search to make evidence-based architectural decisions?**



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

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

The study of retrieval has a long history [119], stemming from decades of work in both the Database (DB) [34, 122, 126, 133] and Information Retrieval (IR) [27, 60, 134, 138] communities. This research has produced two complementary approaches: lexical and semantic search. Lexical methods, such as Full-Text Search (FTS) [48, 123] and Sparse Vector Search (SVS) [29, 52], excel at exact keyword matching, but often fail to capture contextual meaning [61]. In contrast, semantic approaches like Dense Vector Search (DVS) [76, 127] and Tensor Search (TenS) [71, 109] employ neural models to understand context [124, 130], though they may lack the keyword-specific ability. Hybrid search combines these paradigms to synthesize their complementary strengths [40, 83, 135]. While this blended approach can improve retrieval accuracy, it introduces a fundamental tension between effectiveness and system cost. As shown in Figure 1, each additional retrieval path, while boosting accuracy, can increase query latency and memory consumption. This creates a complex 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 hybrid search studies [38, 82, 144] typically focus on pairwise combinations, a systematic understanding of how three or more paradigms perform and interfere in a single system is lacking. Second, common re-ranking strategies rely on simple fusion methods like Reciprocal Rank Fusion (RRF) [28, 45], whose effectiveness is unverified for complex scenarios where candidate lists from multiple paradigms must be consolidated. Finally, the community lacks a framework to systematically evaluate the trade-offs

Proceedings of the VLDB Endowment, Vol. 19, No. 8 ISSN 2150-8097.
doi:10.14778/3811243.3811246

of all four major paradigms (full-text [122], sparse [147], dense [56], and tensor [93]), which hinders reproducible research. Given the proven, yet distinct, value of each paradigm [56, 93, 104, 111, 147], such a study is essential for understanding their intrinsic trade-offs and guiding future system design.

To bridge these gaps, this paper conducts an experimental analysis of hybrid search. We built a modular evaluation framework, based on the `Infinity` database, supporting the systematic evaluation of arbitrary combinations of the four retrieval paradigms. Using this framework, we conduct an experimental study across 11 real-world datasets to create a data-driven map of the hybrid search performance landscape. Our investigation is guided by three research questions: (**RQ1**) Does integrating additional retrieval paths consistently improve search accuracy? (**RQ2**) What criteria guide the selection of appropriate combination schemes for different scenarios? (**RQ3**) How can candidates from different retrieval paths be effectively re-ranked within a database?

Our analysis yields three key findings that provide practical guidelines and new directions. **First**, we identify a "weakest link" phenomenon in multi-path search. While combining paths can improve accuracy, our results provide the systematic evidence that a weak path can *substantially* degrade overall performance. This highlights the need for path-wise quality assessment before fusion. **Second**, we provide a data-driven map of performance trade-offs, demonstrating that no single hybrid method excels across accuracy, efficiency, and resource cost simultaneously. Rather, the ideal configuration is dependent on specific constraints (e.g., latency, memory) and data characteristics; our analysis provides quantitative evidence to guide these design choices. **Third**, we demonstrate that Tensor-based Re-ranking Fusion (TRF) is a practical re-ranking strategy. It outperforms mainstream fusion methods like Reciprocal Rank Fusion (RRF), offering the semantic power of a full tensor search at a fraction of the computational and memory cost.

Our major contributions are as follows:

- **A Systematic Survey of Retrieval Paradigms**. We present the systematic survey of four retrieval paradigms. We analyze their historical evolution, technical underpinnings, and respective trade-offs, establishing the foundation for hybrid search.
- **A Novel Open-Source Evaluation Framework**. We design, build, and open-source a modular framework for evaluating advanced hybrid search. It is the first publicly available tool supporting the flexible combination and evaluation of all four major retrieval paradigms and multiple re-ranking strategies.
- **A Comprehensive Evaluation and Data-Driven Analysis**. We conduct a rigorous evaluation of retrieval combinations and re-ranking strategies across 11 real-world datasets. Our analysis quantifies the trade-offs in accuracy, efficiency, and overhead, offering evidence-based guidelines for practitioners.
- **Identification of Key Challenges and Future Directions**. Based on our empirical findings, we provide observations to guide scenario-specific choices and articulate key challenges and promising research directions for hybrid search.

The remainder of this paper is organized as follows. We review retrieval paradigms in Section 2 and detail our evaluation framework in Section 3. We then describe the experimental setup in Section 4 and present our experimental results in Section 5. We discuss our findings, their implications, and future work in Section 6. Section 7 concludes the paper.

2 BACKGROUND

This section reviews the technical underpinnings of the four major paradigms that constitute modern hybrid search.

2.1 Lexical Search Paradigms

Lexical search determines relevance via keyword frequency and statistical signals. While efficient and interpretable, these paradigms suffer from the "vocabulary mismatch" problem [54]. By treating text as an unordered bag of words, they fail to capture contextual nuance. This section details two dominant lexical methods: traditional Full-Text Search (FTS) and learned Sparse Vector Search (SVS), as depicted 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 [64, 102, 105], to statistical weighting schemes like TF-IDF that introduced the concept of term importance [23, 96, 99]. This formed the basis for modern probabilistic models, with the Okapi BM25 (Best Matching 25) algorithm now the de-facto standard for relevance ranking in production systems [62, 108, 117, 118], such as Elasticsearch [6] and OpenSearch [12]. BM25 enhances earlier frameworks by incorporating two key heuristics: non-linear term frequency saturation, which prevents overly frequent terms from dominating the relevance score, and document length normalization, which adjusts scores 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 relevance. This reliance on exact term statistics defines FTS behavior. As illustrated in Figure 2 (top left), for a query like "red sports shoes", FTS excels at precise phrase matching. This precision is a key strength, but the reliance on exact terms also reveals its primary limitation: a lack of semantic understanding, causing it to fail on synonyms like "sneakers" for "shoes".

For large-scale retrieval, FTS engine efficiency and effectiveness rely heavily on advanced dynamic pruning [46, 60, 88, 116, 119] and flexible query capabilities [1, 107]. Pruning algorithms such as WAND (Weak AND) [27] and its modern successor, Block-Max WAND [48], enable early termination by calculating real-time score bounds, effectively filtering low-value candidates without full scoring. Beyond ranking, FTS supports rule-based query operations (e.g., phrase searches, wildcards, term weighting) that enable highly controllable retrieval. These capabilities, combined with optimizations like inverted index compression [89, 101], form the technical stack enabling modern FTS engines to handle massive corpora.

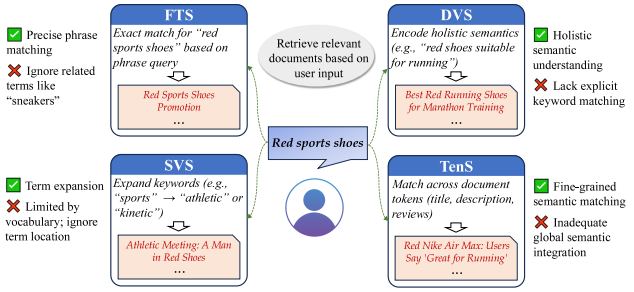


Figure 2: A toy example of different retrieval paths.

2.1.2 Sparse Vector Search (SVS). SVS is a modern lexical search approach that enhances traditional term-based methods with learned term weights from deep neural models [38, 78]. Unlike FTS, which relies on raw corpus statistics, SVS employs learning models like SPLADE [51, 52, 72] to transform text into high-dimensional sparse vectors. In these vectors, each dimension corresponds to a term in an expanded vocabulary, and non-zero values represent the learned contextual importance of that term. Relevance between a query Q and a document D is computed as the inner product (IP) of their sparse vectors, $\boldsymbol{q}$ and $\boldsymbol{d}$:

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

A higher IP 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 "sports" to related terms like "athletic", mitigating the vocabulary mismatch problem to a degree. However, SVS remains a lexical method. It is constrained by its vocabulary and disregards term position, which limits its ability to capture complex phrasal or high-level semantics.

The high dimensionality of sparse vectors makes efficient search challenging, a problem addressed using techniques analogous to those in FTS. SVS systems build inverted indexes on the non-zero vector dimensions, applying dynamic pruning algorithms to accelerate query processing [30–32, 85, 86, 94, 137]. Advanced methods such as Block-Max Pruning (BMP) [91] precompute maximum term scores for document blocks, allowing the system to prune large portions of the search space without full evaluation. The efficacy of this method is demonstrated by its adoption in industrial systems like Pinecone [21]. By combining learned relevance with fast pruning, SVS offers a powerful 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 rely on inverted indexes and pruning, their foundational differences in relevance modeling create distinct trade-offs. The core distinction is the origin of term weights: FTS derives relevance from unsupervised corpus statistics (e.g., BM25), making it robust and training-free. In contrast, SVS uses pre-trained neural models to learn contextual term importance, capturing nuances statistical methods miss. This leads to a key practical trade-off: FTS provides highly controllable, rule-based operations (e.g., precise phrase and wildcard searches), while SVS excels at learned term expansion, mitigating vocabulary mismatch at the cost of direct control. From a systems perspective, this also introduces a significant difference in operational overhead: FTS indexes raw text directly, whereas SVS requires a computationally intensive inference step for vector generation before indexing. Ultimately, the choice balances the statistical robustness and controllability of FTS against the contextual matching of SVS, highlighting their complementary nature.

2.2 Semantic Search Paradigms

Unlike lexical methods, semantic search utilizes deep neural networks to generate contextual representations. While excelling at capturing latent relationships and high-level intent, this semantic depth typically sacrifices keyword precision and incurs significant computational and domain-adaptation costs. This section details the two dominant semantic approaches: DVS, utilizing a single-vector global representation, and TenS, employing a multi-vector representation, as contrasted in Figure 2 (right).

2.2.1 Dense Vector Search (DVS). DVS represents a text's entire semantic meaning (e.g., a sentence or document) as a single, fixed-dimensional dense vector, or embedding. Embeddings are typically generated using bi-encoder architectures [70, 84, 115], where models like Sentence-BERT [106] aggregate token-level representations into a holistic summary, such as by using a special token's output (e.g., `[CLS]` [38, 44]) or mean-pooling all token hidden states [103]. Semantic similarity between a query Q and a document D is then computed as the inner product of their dense vectors; a higher score indicates greater similarity. This global representation enables 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 without the exact keywords. However, this aggregation process obscures fine-grained details, leading to reduced precision for queries dependent on exact terms.

Exact search over millions or billions of dense vectors is computationally prohibitive. Consequently, DVS systems rely on Approximate Nearest Neighbor Search (ANNS) algorithms [25, 33, 97, 100, 132, 150] to achieve low-latency, high-throughput retrieval. ANNS algorithms include tree-based [112, 141], hashing-based [55, 59, 66, 79], quantization-based [22, 24, 57, 67], and graph-based approaches [53, 114, 127]. Graph-based algorithms, particularly HNSW (Hierarchical Navigable Small World graph) [87], have recently emerged as the state-of-the-art, offering a superior balance of accuracy and efficiency. These techniques form the core engine of most modern vector databases [35, 124, 139], including SingleStore [35], Chroma [11], and Qdrant [13].

2.2.2 Comparison and Synthesis: SVS vs. DVS. While both SVS and DVS employ vector representations, they embody fundamentally different philosophies of semantic modeling. The primary distinction is representation granularity. 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, evolving traditional "bag-of-words" models. In contrast, DVS pursues holistic semantic summarization, compressing a text's entire meaning into a single dense vector where individual dimensions lack explicit meaning, allowing it to capture high-level intent. This representational difference

dictates their underlying architectures: SVS typically utilizes the classic inverted index and dynamic pruning stack from FTS, while DVS relies on an entirely different class of ANNS algorithms. 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. For instance, vector databases like Milvus [16] and Weaviate [15] integrate SVS alongside their native DVS capabilities specifically to improve retrieval accuracy on queries that require precise keyword matching.

2.2.3 Tensor Search (TenS). TenS represents a shift from single-vector representations, instead modeling a text as a set of contextualized embeddings for each of its tokens. Pioneered by models like ColBERT [2, 42, 68, 71, 80, 110], TenS employs a "late-interaction" architecture. Rather than aggregating a text's meaning into one vector, it retains a tensor of token embeddings for both query and 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 for each query token against all document tokens:

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

where $\boldsymbol{q}_i$ and $\boldsymbol{d}_j$ are the token embeddings of the query and document, respectively. This token-level interaction 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 the query "red sports shoes", it could match "red" in a title, a related token like "Nike" in its description, and "running" in a user review. This aggregation of best-token matches captures fine-grained semantic relationships. However, this focus on local token alignment is also its primary weakness: inadequate global semantic integration. Summing individual token scores may fail to capture the holistic context of a sentence or paragraph.

The primary drawback of TenS is its significant computational and memory overhead [58, 75, 98, 110], stemming from the need to store and process an embedding for every token in the corpus. A naive implementation can result in index sizes orders of magnitude larger than for DVS. Consequently, substantial research has focused on mitigating this high cost. Techniques range from representation compression (e.g., residual representations in ColBERTv2 [110]) to multi-stage filtering pipelines like PLAID [109] and quantization-based methods like EMVB [93] that use hardware acceleration (e.g., SIMD) for efficiency. Such techniques have enabled its use in industrial systems like Vespa [10] where accuracy is paramount. Despite these advancements, the high cost of TenS remains a significant barrier to 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 use neural embeddings, they differ fundamentally in their semantic granularity and resulting retrieval architectures. DVS employs 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 local details [70, 103]. In contrast, TenS uses a fine-grained multi-vector approach, retaining a contextualized embedding for each token. This "late-interaction" mechanism can capture nuanced relationships but may struggle to form a coherent understanding of the text's overall global meaning [71]. This core difference in representation dictates their architectural trade-offs. DVS, with its single-vector-per-document model, is well-suited for the mature ecosystem of ANNS algorithms, enabling efficient search over massive datasets. TenS, however, presents a far greater retrieval challenge due to the sheer volume 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 general-purpose semantic workhorse, while TenS represents a resource-intensive paradigm that trades scalability for fine-grained semantics. Their distinct strengths make them complementary.

2.3 Hybrid Search Architectures.

Lexical and semantic paradigms are complementary but individually insufficient. Hybrid search fuses these approaches to deliver both keyword precision and semantic robustness that single-path systems cannot match. The central challenges of this architecture, however, lie in orchestrating multi-path retrieval and merging disparate candidate lists. This section details these two pillars: path combination schemes and candidate re-ranking strategies.

2.3.1 Retrieval-Path Combination Schemes. Current hybrid search implementations [29, 82, 140] are predominantly dual-path schemes, categorized by the primary system they extend. The vector-centric approach [5, 15, 16] integrates SVS into vector databases to augment native DVS capabilities, addressing their weakness in handling precise keyword queries. Conversely, the text-centric approach integrates DVS into established FTS engines like Elasticsearch [6], retrofitting them with semantic capabilities [7, 36, 136]. Integration strategies for these dual-path systems vary: some create a unified index for joint pruning [144], while others use separate indexes managed by a multi-index engine [41]. However, these approaches are limited to two paradigms. Comprehensive studies effectively integrating three or more paradigms remain scarce.

2.3.2 Candidate Re-ranking Strategies. Merging candidate lists from multiple retrieval paths into a single result set requires a re-ranking strategy. A prevalent lightweight technique in modern systems is Reciprocal Rank Fusion (RRF) [17, 45]. RRF is a score-agnostic method that synthesizes ranked lists by prioritizing candidates appearing near the top of multiple lists; this is effective when raw scores (e.g., BM25, vector similarity) are not directly comparable [40]. The RRF score for a candidate c is:

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

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

An alternative, Weighted Sum (WS) [18, 28], linearly combines normalized scores from each path:

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

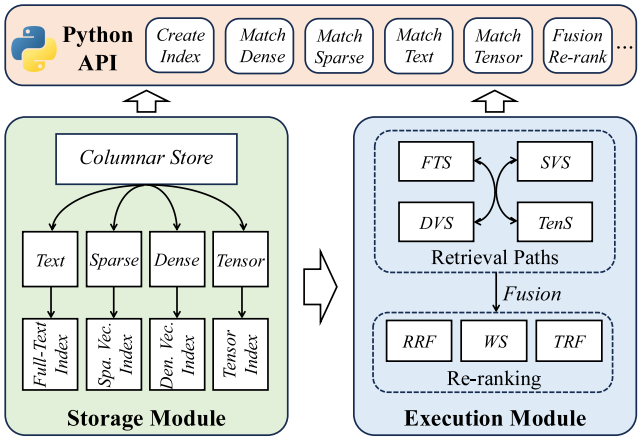


Figure 3: Overview of the evaluation framework.

where w_i is the pre-defined weight and $\text{score}_i(c)$ is the candidate's score. Unlike RRF, WS is score-aware, suitable when scores are normalized and their relative importance can be estimated [128].

Remarks. These lightweight fusion methods contrast with heavyweight learning-based re-rankers, which are architecturally distinct and typically reserved for second-stage processing [14, 19]. Prominent heavyweight models use *cross-encoder* architectures [71, 145], performing a computationally intensive *joint encoding* of concatenated query and document tokens at query time. This deep interaction yields superior accuracy but is impractical for in-database deployment on large candidate sets due to high latency and hardware (e.g., GPU) requirements [106]. The *late-interaction* architecture [68, 71, 110] of tensor models is fundamentally different. It decouples the encoding process: document token embeddings are generated offline and indexed, while at query time, only a lightweight computation (e.g., MaxSim) is performed against the query's tokens. This design, separating expensive document encoding from online interaction, makes late-interaction a viable strategy for fine-grained re-ranking *within the database*. Our study focuses on this critical first stage of efficient in-database fusion. We evaluate both lightweight methods and the feasibility of using late-interaction models for this purpose, an approach whose trade-offs in complex hybrid systems are not yet well understood.

3 EVALUATION FRAMEWORK

We developed the evaluation framework on top of the open-source `Infinity` database.

3.1 Overview

As shown in Figure 3, this framework is realized via a two-module architecture: The *Storage Module* employs a decoupled, columnar design. During ingestion, each document is partitioned into columns (Text, Sparse, Dense, Tensor), each managed by a specialized indexing fleet (e.g., HNSW for DVS, optimized inverted indexes for FTS and SVS). This architectural choice is critical for modularity and performance, as each path uses its own state-of-the-art algorithm without compromise—a key limitation of unified index approaches—and ensures extensibility. The *Execution Module* orchestrates hybrid search by transforming a query into a pipelined

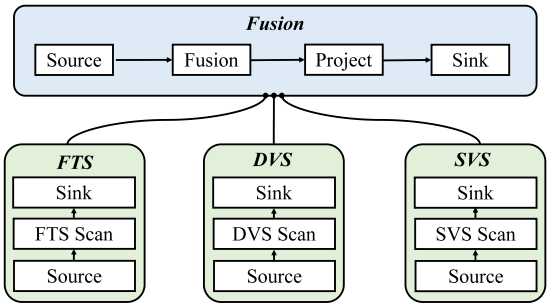


Figure 4: Illustration of the pipelined DAG execution model for a three-path (FTS+DVS+SVS) hybrid query.

Directed Acyclic Graph (DAG) of physical operators (see Figure 4). Each retrieval path executes in parallel, streaming results to a final fusion operator that applies the chosen re-ranking strategy (e.g., RRF). This model is applied consistently across all combinations, ensuring a fair performance comparison. The framework is exposed via a user-friendly Python API[1] for reproducibility.

3.2 Storage Module

All indexes are persisted to disk and accessed via a buffer manager that stages data between disk and memory, allowing the system to scale beyond main memory capacity. This disk-oriented architecture entails specific trade-offs for each paradigm:

- **FTS and SVS:** These inverted-index-based paradigms are well-suited for disk storage, as their sequential access patterns within posting lists are efficiently handled by the buffer manager and OS page cache.
- **DVS (HNSW [87]):** The disk-based design leverages on-demand paging to fetch graph nodes, thereby reducing memory consumption at the cost of increased I/O latency.
- **TenS:** The large storage footprint of TenS makes a disk-based approach a practical necessity. Our EMVB implementation is specifically designed to mitigate the high I/O costs of this heavyweight paradigm.

3.3 Execution Module

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 to the next operator as it becomes available. This approach reduces idle time and is highly effective for streaming search queries. Each operator in the pipeline is designed to execute concurrently. The framework manages a dedicated thread pool, strategically assigning operator tasks to available cores to maximize parallelism while mitigating resource contention. This combination of DAG-based planning and a concurrent, push-based engine allows the framework to efficiently coordinate complex, multi-path queries with minimal overhead.

Example 1. *As illustrated in Figure 4, a hybrid query is compiled into a query pipeline based on a Directed Acyclic Graph (DAG) of physical operators. The DAG structure enables parallel execution, where independent* `Scan` *operators for each retrieval path, such as an*

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

inverted index scan for FTS or a graph traversal for DVS, run concurrently. As these operators generate candidates, results are streamed to a final `Fusion` *operator. This operator incrementally applies a specified re-ranking strategy (e.g., RRF) to merge the incoming candidate lists. This pipelined model allows the fusion computation to overlap with the ongoing scan operations, minimizing end-to-end query latency.*

3.4 Algorithm Implementations

Our framework's modularity supports specialized, state-of-the-art implementations for each component. We implemented our FTS engine from scratch for tight integration with our storage and execution models, inspired by production-grade designs like `Lucene`[2] and `Tantivy`[3]. It uses the BM25 ranking function, with query performance accelerated by a Block-Max WAND (BMW) [48] dynamic pruning algorithm. The implementation also supports configurable tokenizers [8, 9] and document ID reassignment to improve data locality and pruning efficiency. For SVS, we implement Block-Max Pruning (BMP) [91], a highly-efficient dynamic pruning algorithm. Our implementation includes its core components of block filtering [90] and candidate block evaluation, optimized with SIMD instructions [47] and a hybrid inverted-forward index structure to accelerate top-k selection. The DVS component uses a state-of-the-art HNSW graph index [87]. Our version further integrates Locally-adaptive Vector Quantization (LVQ) [22], a two-stage quantization process that significantly reduces the index footprint (minimizing disk I/Os) and accelerates distance computations during graph traversal. To manage the high overhead of TenS, we implement the EMVB technique [93]. 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) [67] 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 designed to answer the research questions posed in Section 1. We describe the core experimental components, define our evaluation metrics, and specify the implementation details and hardware environment.

4.1 Experimental Components

4.1.1 Datasets. To ensure generalizability, we evaluate all methods on 11 real-world datasets, chosen for variety in domain, task, and scale (Table 1). Most are from the widely-used BEIR benchmark [115], supplemented by two multilingual datasets from MTEB [92] and a Chinese news question answering dataset from AIR-Bench [37] to enhance diversity. The collection covers 9 distinct tasks (e.g., fact-checking) across 7 domains, with corpus sizes from 5.2K to 8.8M. With the exception of MSMA(en), all datasets are out-of-domain. This provides a test of zero-shot retrieval performance, as the embedding models were not fine-tuned on them.

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

4.1.2 Embedding Models. We use mainstream models to generate DVS, SVS, and TenS embeddings. For DVS and SVS, we use `BGE-M3` [38], which supports long inputs (up to 8,192 tokens) and concurrently produces high-quality dense and sparse representations. While `BGE-M3` can generate token tensors, its 1,024-dimensional embeddings are impractical for large-scale TenS; we observed embedding <50% of the MLDR(en) dataset would require 1.1TB. To address this, we use the highly efficient `answerai-colbert-small-v1` [2] for English TenS. With 33M parameters (vs. `BGE-M3`'s 560M), it produces compact 96-dimensional representations. For Chinese datasets, we use `Jina-ColBERT-v2` [68], selecting its most compact 64-dimensional variant for space efficiency. To validate the generalizability of our findings, we extended our evaluation to include three leading embedding models from the MTEB leaderboard[4]: `NV-Embed-v2` [73], `Qwen3-Embedding-8B` (`Qwen3-8B`) [146], and `KaLM-Embedding-Gemma3-12B-2511` (`Gemma3-12B`) [65, 148].

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 (Section 3.4). From these, we construct and evaluate all 11 possible hybrid combinations: all six two-path, all four three-path, and the single four-path configuration. For hybrid methods, we primarily 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. To broaden our comparison, we also compare against two widely-used heavyweight learning-based re-rankers, `gte-multilingual-reranker-base` (GTE) [145] and `bge-reranker-v2-m3` (BGE) [38], as external baselines.

4.2 Evaluation Metrics

We evaluate each method along 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 $k = 10$ by default. As our primary metric, nDCG@k is standard in information retrieval (IR) benchmarks [77, 115, 129] due to its ability to handle graded relevance judgments and properly account for document rank. Theoretical analyses also confirm its comparability across methods [131]. These properties make it more robust and informative than rank-unaware measures like Precision/Recall or those limited to binary relevance, such as Mean Reciprocal Rank (MRR) and Mean Average Precision (MAP) [115]. To confirm the robustness of our principal findings, we also report Recall@10 and MRR@10 in our analysis (Section 5.1).

4.2.2 Efficiency and Overhead. We evaluate system performance on both online query processing and offline index construction. *Query Performance*: We measure online efficiency via Query Latency (both average and P99 tail latency) and Throughput (Queries Per Second, QPS, under a multi-threaded batch workload). *Indexing Performance*: We measure offline efficiency via total Indexing Time (including data import and construction) and storage overhead via

[4]https://huggingface.co/spaces/mteb/leaderboard

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) [95]	Miscellaneous	Passage Retrieval	8,841,823	43	56	2.9GB	13GB	36GB	254GB
DBPE(en) [63]	Wikipedia	Entity Retrieval	4,635,922	400	50	1.4GB	6.9GB	18.2GB	58GB
MCCN(zh) [37]	News	Question Answering	935,162	339	1,263	2.3GB	6.9GB	3.8GB	148GB
TOUC(en) [26]	Miscellaneous	Argument Retrieval	382,545	49	292	184MB	1.6GB	1.5GB	56GB
MLDR(zh) [38]	Wiki., Wudao	Long-Document Retrieval	200,000	800	4,249	3.1GB	5.2GB	791MB	186GB
MLDR(en) [38]	Wikipedia	Long-Document Retrieval	200,000	800	3,308	3.1GB	4.4GB	791MB	94GB
TREC(en) [120]	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) [43]	Scientific	Citation Prediction	25,657	1,000	176	30MB	89MB	106MB	2.4GB
SCIF(en) [121]	Scientific	Fact Checking	5,183	809	214	7.5MB	23MB	24MB	676MB

on-disk Index Size. *Resource Overhead*: We measure Peak Memory Consumption during both online querying 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 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 [74, 81]. The LVQ enhancement is applied as detailed in its original work [22]. **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 [24, 67, 143]. **Re-ranking**: For RRF, the smoothing parameter κ is set to 60, a widely-used value [40, 45]. For WS, the weight of each path is set to its nDCG@10 score from its single-path evaluation on the respective dataset. TRF re-ranks the candidate pool by computing the MaxSim score (Equation (3)) between the query's tokens and each document's pre-computed token tensors. For all hybrid methods, the candidate list size per path (k_0) is 10 by default, a value chosen based on initial analysis to balance effectiveness and efficiency (see Section 5.3). Unless otherwise specified, RRF is the default re-ranking strategy.

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.

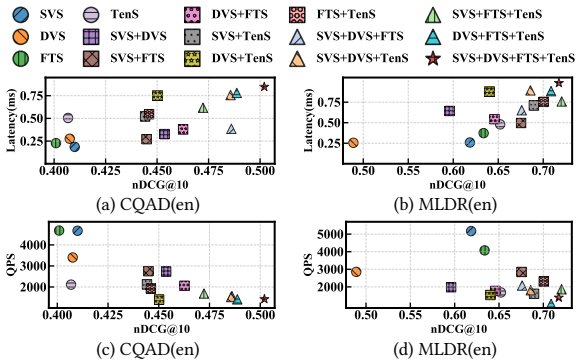


Figure 5: Accuracy vs. Efficiency of combination schemes.

5 EXPERIMENTAL RESULTS

This section presents our experimental evaluation, analyzing results across 11 real-world datasets to answer the research questions posed in Section 1. We report the principal findings here; a more exhaustive set of results is available in [20].

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.

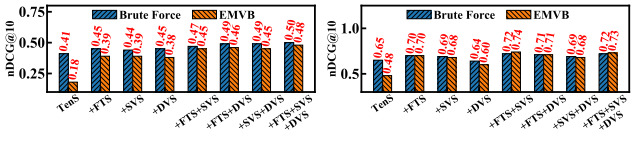
The underlying principle of hybrid search is *complementarity*: the assumption that fusing retrieval paths with different strengths will yield synergistic gains in accuracy. Our results confirm this on many datasets. For instance, on the DBPE(en) dataset, combining an FTS path (nDCG@10 of 0.565) with an SVS path (0.479) under RRF improves the score to 0.608 (Table 2). However, this synergy is not guaranteed. A low-accuracy path can pollute the candidate pool with irrelevant documents, causing the hybrid system to underperform its best constituent path. We term this the **"weakest link"** phenomenon. This effect is demonstrated on the TOUC(en) dataset, where fusing a strong FTS path (0.650) with a weak DVS path (0.390) degrades the final RRF score to 0.604.

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**	.677	.455	.495	.675	.690	.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	.920	.471	.508	.463	.465	.347	**.354**	.748	.761
	DVS+SVS	.814	.887	.674	.711	.518	.622	.566	.596	.365	.448	.596	.666	.803	.899	.506	.510	.454	.463	.320	.353	.716	.746
Three-Path	FTS+SVS+DVS	.819	.888	.692	.713	.474	**.624**	.654	.627	.451	**.496**	.676	**.693**	.865	**.924**	.490	.496	**.486**	.465	.345	.353	.739	**.762**

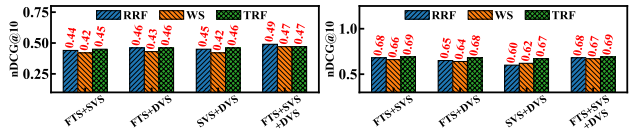
Table 3: Effectiveness on the MLDR(en) dataset measured by Recall@10 and MRR@10.

	Single-Path			Multi-Path (RRF)				Multi-Path (TRF)			
	FTS	SVS	DVS	FTS+SVS	FTS+DVS	SVS+DVS	FTS+SVS+DVS	FTS+SVS	FTS+DVS	SVS+DVS	FTS+SVS+DVS
Recall@10	0.755	0.735	0.614	0.794	0.766	0.739	0.798	0.806	0.790	0.771	0.814
MRR@10	0.595	0.582	0.450	0.637	0.608	0.552	0.637	0.652	0.643	0.629	0.653



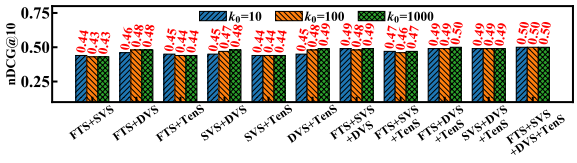
(a) CQAD(en) (b) MLDR(en)

Figure 6: Accuracy comparison of brute-force and EMVB.

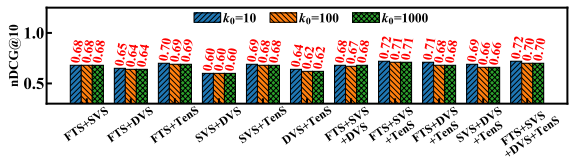


(a) CQAD(en) (b) MLDR(en)

Figure 7: Accuracy comparison of RRF, WS, and TRF.



(a) CQAD(en)



(b) MLDR(en)

Figure 8: Accuracy under different numbers of candidates.

5.1.1 Generalizability Analysis. We examine the generalizability of this observation by verifying its consistency across multiple experimental dimensions. The observed phenomenon is independent of the evaluation metric (Table 3) and embedding model (Table 4), and persists irrespective of the algorithmic implementation (Figure 6). Crucially, our results indicate that this degradation, originating in the candidate generation phase, is difficult to mitigate at the re-ranking stage. The effect holds true irrespective of the fusion strategy's complexity, from lightweight RRF (Figure 7) to heavyweight learning-based re-rankers (Table 5). Moreover, simply increasing the candidate list size (k_0) fails to resolve the issue (Figure 8). This underscores a key principle: the initial candidate quality imposes a hard accuracy ceiling, and once a weak path introduces irrelevant documents, the damage is difficult to reverse.

5.1.2 Analysis of Influencing Factors. We analyze the "weakest link" effect by two primary factors: the re-ranking mechanism and query characteristics. First, the failure mode depends on the re-ranking strategy. Rank-based methods like RRF are susceptible to high ranks from a weak path, irrespective of relevance. In contrast, TRF's semantic verification makes it vulnerable to "hard negatives"—highly similar but irrelevant documents that achieve a high MaxSim score. Our case study in Section 5.4 provides a concrete analysis of these distinct failure modes. Second, which path becomes the "weakest link" is strongly correlated with query length. For datasets with long, descriptive queries such as MCCN(zh), the semantic DVS path holds a substantial advantage over the lexical FTS path (nDCG@10 of 0.543 vs. 0.222). Conversely, for datasets with short keyword queries like TOUC(en), the trend reverses, with the precise FTS path significantly outperforming DVS (0.650 vs. 0.390). This demonstrates a clear pattern: semantic paths thrive on the rich context provided by longer queries, whereas lexical paths excel at matching the exact keywords typical of shorter ones.

5.2 Combination Scheme Selection (RQ2)

Finding 2: The optimal hybrid search architecture is a context-dependent trade-off among accuracy, query performance, and resource overhead; no single configuration excels universally.

We analyze the multi-dimensional trade-offs inherent in hybrid search architectures across three key dimensions: accuracy-latency, resource overhead, and context-specific factors.

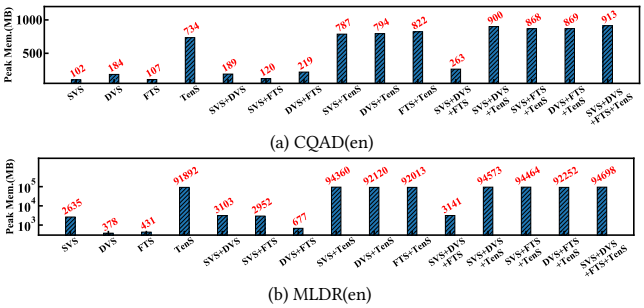
5.2.1 Accuracy-Latency Trade-off. Adding retrieval paths improves accuracy at the cost of higher latency, a pattern consistent across datasets (Figures 5 and 12). On CQAD(en), for instance, the four-path FTS+DVS+SVS+TenS (red ★) achieves an nDCG@10 of 0.50, but its latency is 3.7× higher than the FTS-only path (green ●). This

Table 4: Effectiveness (nDCG@10) of hybrid search configurations with state-of-the-art embedding models.

Dataset	Model	Single-Path	Multi-Path (RRF)			Multi-Path (TRF)		
		DVS	DVS+FTS	DVS+SVS	All Three	DVS+FTS	DVS+SVS	All Three
CQAD(en)	NV-Embed-v2	0.481	0.537	0.564	0.547	0.506	0.505	0.486
	Qwen3-8B	0.573	0.533	0.555	0.539	0.504	0.504	0.486
	Gemma3-12B	0.535	0.516	0.539	0.533	0.498	0.503	0.483
MLDR(en)	NV-Embed-v2	0.489	0.511	0.427	0.616	0.662	0.641	0.686
	Qwen3-8B	0.421	0.606	0.572	0.659	0.671	0.657	0.689
	Gemma3-12B	0.200	0.557	0.484	0.636	0.666	0.645	0.691

Table 5: Effectiveness (nDCG@10) of heavyweight re-rankers on datasets with short (CQAD) and long (MLDR) documents.

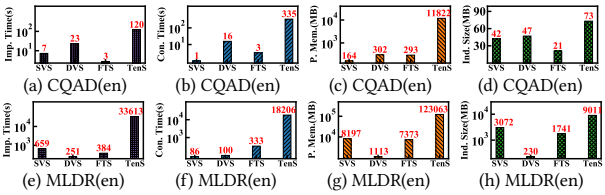
CQAD(en)				
Methods	**FTS+SVS**	**FTS+DVS**	**DVS+SVS**	**FTS+SVS+DVS**
GTE	0.487	0.512	0.516	0.512
BGE	0.486	0.514	0.515	0.511
MLDR(en)				
Methods	**FTS+SVS**	**FTS+DVS**	**DVS+SVS**	**FTS+SVS+DVS**
GTE	0.471	0.454	0.466	0.438
BGE	0.488	0.472	0.480	0.456



(a) CQAD(en)

(b) MLDR(en)

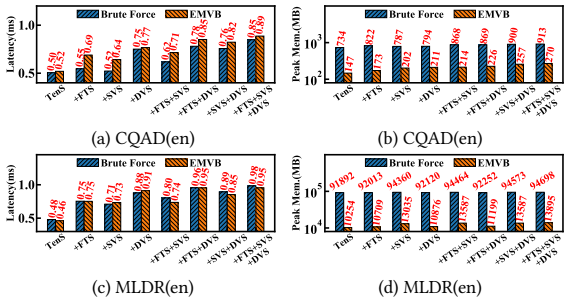
Figure 9: Memory consumption across retrieval methods.



(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 10: Indexing overhead of different retrieval paths.



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

(c) MLDR(en) (d) MLDR(en)

Figure 11: Online overhead of brute-fore and EMVB.

latency cost is inherent to the parallel execution model, where the final fusion step must await all path completions. Thus, a system's P99 tail latency is dictated by its slowest component (Table 6). For example, the FTS+DVS combination inherits the DVS path's high tail latency, making it substantially slower than FTS alone. This trade-off is complicated by query characteristics, as the slowest path can change: DVS latency is constant, while FTS and SVS latency may increase with query length, demanding query-aware choices.

5.2.2 Resource Overhead. Hybrid search architectures introduce critical trade-offs in resource overhead, encompassing both online memory consumption and offline index construction. Resource usage generally scales with the number and type of paths combined, as shown in Figures 9 and 10. To illustrate the severity of this trade-off, we can examine the TenS paradigm as an example. Its resource cost is disproportionately high, with a memory footprint that can be orders of magnitude larger than other paradigms. This is a direct consequence of its per-token embedding representation; unlike DVS where storage is constant per document, the storage footprint of TenS scales linearly with document length. This linear scaling is also reflected in its offline indexing costs, requiring substantially more time and peak memory to construct its index of token embeddings. These results demonstrate that the resource profile of each paradigm is a critical design consideration, as a single resource-intensive path can dominate a system's overall cost.

5.2.3 Context-Dependent Trade-offs. The optimal trade-off point is not static but shifts based on algorithmic and data characteristics. For instance, an optimized algorithm like EMVB can dramatically reduce the resource cost of TenS, but at the expense of retrieval accuracy (Figure 11). Similarly, data characteristics alter the balance. On corpora with long documents, the fixed-size vectors of DVS are highly efficient to index, but may suffer from information loss; in contrast, the representations for FTS and SVS retain more term-level detail at the cost of larger index sizes. These factors reinforce our central finding: selecting a hybrid architecture is not a search for a single best configuration, but a complex optimization problem that must be tailored to specific algorithms, data characteristics, and application requirements.

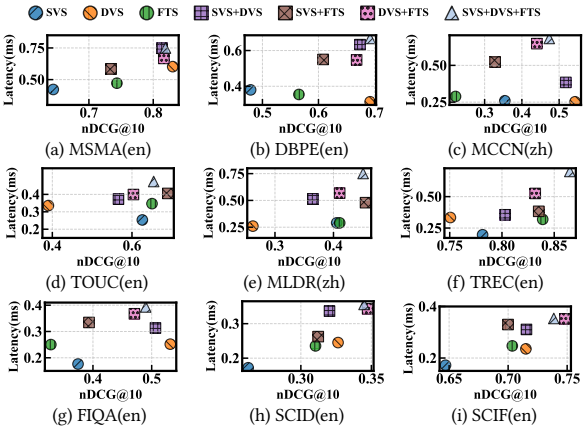
5.3 Cross-Path Re-ranking (RQ3)

Finding 3: While lightweight fusion methods like RRF are efficient, TRF offers a path to significantly higher accuracy by leveraging fine-grained semantics at a fraction of the cost of a full TenS path.

Our experiments show that the choice of re-ranking strategy is critical to a hybrid system's final accuracy. We position TRF by comparing it against lightweight fusion, heavyweight learning-based re-rankers, and a full tensor retrieval path.

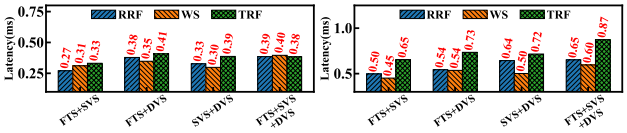
Table 6: Mean and P99 Latency (ms) comparison on the MLDR(en) dataset.

	Single-Path			Multi-Path (RRF)				Multi-Path (TRF)			
	FTS	SVS	DVS	FTS+SVS	FTS+DVS	SVS+DVS	FTS+SVS+DVS	FTS+SVS	FTS+DVS	SVS+DVS	FTS+SVS+DVS
Mean (ms)	0.37	0.26	0.26	0.50	0.54	0.64	0.65	0.65	0.73	0.72	0.87
P99 (ms)	0.53	0.45	0.63	0.58	0.85	0.83	0.91	0.77	0.95	1.00	1.01



(a) MSMA(en) (b) DBPE(en) (c) MCCN(zh)
(d) TOUC(en) (e) MLDR(zh) (f) TREC(en)
(g) FIQA(en) (h) SCID(en) (i) SCIF(en)

Figure 12: Cross-dataset latency-accuracy trade-offs.



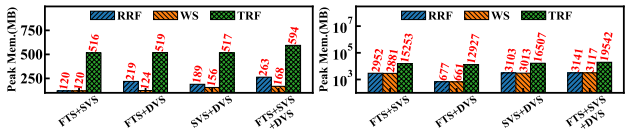
(a) CQAD(en) (b) MLDR(en)

Figure 13: Retrieval latency under RRF, WS, and TRF.

5.3.1 TRF vs. Lightweight Fusion. TRF generally demonstrates a significant accuracy advantage over widely-used lightweight methods like RRF and WS (see Figure 7, Table 2, and Table 4). While RRF and WS are limited to coarse document-level signals such as ranks or scores, TRF performs fine-grained semantic verification. It leverages the MaxSim computation to model token-to-token interactions between the query and each candidate. 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, as we demonstrate with a query-level analysis in our case study (Section 5.4).

5.3.2 TRF vs. Learning-based Re-rankers. Compared to heavyweight learning-based re-rankers like GTE and BGE, TRF offers a more practical solution for in-database fusion. While heavyweight models excel on short-document datasets like CQAD(en), our results reveal a critical weakness on long-document corpora like MLDR(en), where their performance is substantially lower than even RRF (Table 5). This failure stems from the fixed input token limits of cross-encoders (e.g., 512 tokens), which force document truncation and lead to critical information loss. Furthermore, these models incur over 100× more ranking time than TRF, rendering them impractical for the in-database fusion scenarios that are the focus of our work.

5.3.3 TRF vs. TenS. While TRF incurs higher latency and memory overhead than lightweight fusion (Figures 13 and 14), its primary advantage lies in its cost-effectiveness compared to a full TenS path. The operational scope is the key difference: TRF re-ranks a small set of candidates, whereas a full TenS path must search a massive



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

Figure 14: Memory overhead across re-ranking methods.

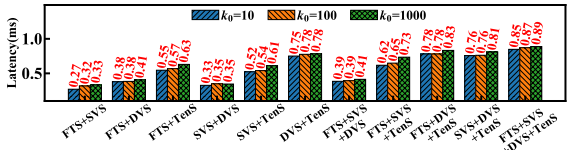


Figure 15: Latency under different numbers of candidates.

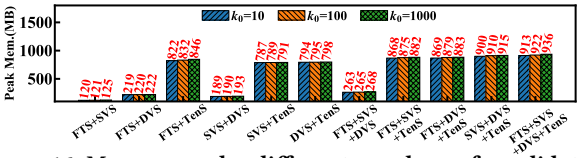


Figure 16: Memory under different numbers of candidates.

index. This architectural design yields dramatic resource savings; for example, on MLDR(en), using TRF on the FTS+DVS candidate set reduces peak memory usage by 86% compared to adding a full FTS+DVS+TenS combination. This highlights a key insight: late interaction is more efficiently harnessed as a re-ranking mechanism on a pre-filtered candidate set than as a primary retrieval method.

5.3.4 Sensitivity to Candidate Set Size (k_0). The performance overhead of TRF is a direct function of the number of candidates (k_0) being re-ranked, which provides a clear tuning parameter. Our analysis demonstrates that increasing k_0 yields diminishing returns for accuracy (Figure 8) while incurring a near-linear increase in latency and memory for TRF (Figures 15 and 16). This allows system designers to directly balance accuracy requirements against latency and memory budgets by adjusting the candidate set size.

5.4 Query-level Case Study

We analyze query `INEX_LD-2009061` ("france second world war normandy") from the DBPE(en) dataset to illustrate our findings, with results detailed in Table 7. The query's intent (military events) is lexically ambiguous, with terms that also relate to memorials.

The RRF fusion of FTS and DVS yields an nDCG@10 of 0.604, a score lower than the 0.784 achieved by the stronger DVS path alone. This degradation exemplifies the "weakest link" phenomenon and stems from RRF's rank-based heuristic. As Table 7 shows, FTS ranks the irrelevant `<...Bayeux_Cemetery>` first (lexical matches), while DVS ranks the irrelevant `<...American_Cemetery>` second (semantic score). RRF misinterprets the high ranks from both paths as agreement and incorrectly promotes these two irrelevant documents. The FTS path thus acts as a "weakest link", introducing high-ranking noise that degrades the final result.

Table 7: Detailed Top-10 results for query `INEX_LD-2009061` ("france second world war normandy") on the DBPE(en) dataset. Results show shortened document titles (e.g., `<...Normandy_Campaign>`). Document relevance from qrels is shown in parentheses (Rel: 1 or 0). Bolded documents are relevant.

Rank	**FTS** (nDCG@10: 0.624)	**DVS** (nDCG@10: 0.784)	**FTS+DVS** (RRF, nDCG@10: 0.604)	**FTS+DVS** (TRF, nDCG@10: 0.880)
1	<...Bayeux...Cemetery> (0)	**<...military...Normandy...>** (1)	<...American_Cemetery...> (0)	**<...Normandy_Campaign>** (1)
2	**<...Hillman_Site>** (1)	<...American_Cemetery...> (0)	<...Bayeux...Cemetery> (0)	**<...Invasion_of_Normandy>** (1)
3	<...American_Cemetery...> (0)	<...Operation_Newton> (0)	**<...military...Normandy...>** (1)	<...American_Cemetery...> (0)
4	<...Order_of_battle...Epsom> (0)	<...Orglandes...cemetery> (0)	**<...Hillman_Site>** (1)	<...Normandy> (0)
5	**<...Bény-sur-Mer...War...>** (1)	**<...Falaise_pocket>** (1)	<...Operation_Newton> (0)	<...Order_of_battle...Epsom> (0)
6	<...Duke_of_Normandy_(pigeon)> (0)	<...Normandie-Niemen> (0)	<...Order_of_battle...Epsom> (0)	<...Bayeux...Cemetery> (0)
7	<...Operation_Cobra...battle> (0)	**<...Normandy_Campaign>** (1)	<...Orglandes...cemetery> (0)	**<...military...Normandy...>** (1)
8	<...Ranger_Assault_Group> (0)	<...Normandy> (0)	**<...Bény-sur-Mer...War...>** (1)	<...Operation_Cobra...battle> (0)
9	<...Saint-Jean-de-Daye_Airfield> (0)	<...Order_of_battle...airborne...> (0)	**<...Falaise_pocket>** (1)	<...Orglandes...cemetery> (0)
10	<...Second_Army_(UK)> (0)	**<...Invasion_of_Normandy>** (1)	<...Duke_of_Normandy_(pigeon)> (0)	**<...Falaise_pocket>** (1)

In contrast, TRF re-scores the same candidate set and achieves a significantly higher nDCG@10 of 0.880. Its effectiveness stems from its fine-grained semantic verification using the MaxSim score. This token-level computation allows TRF to disambiguate the query's intent. Consequently, it demotes irrelevant documents like the cemeteries, as their constituent tokens (e.g., "cemetery", "memorial") are a poor semantic match for the query's military intent. Conversely, TRF elevates relevant documents such as `<...Normandy_Campaign>` and `<...Invasion_of_Normandy>`, which DVS had ranked only 7th and 10th. It promotes them because their tokens (e.g., "campaign", "invasion", "battle") align precisely with the query's semantics. The TRF method is not flawless, as it still places the irrelevant `<...American_Cemetery>` at rank 3, treating it as a hard negative. Nonetheless, this example highlights the difference in principles: RRF's reliance on rank-based fusion makes it susceptible to noise from a single weak path, whereas TRF's token-level verification provides greater resilience against such rank interference.

6 DISCUSSION AND FUTURE WORK

This section synthesizes our findings and analyzes their implications. Table 8 summarizes the trade-offs of hybrid search architectures qualitatively.

6.1 Principal Findings and Their Implications

6.1.1 The "Weakest Link" Effect: The Critical Role of Path Quality. Table 8 suggests that more paths often lead to higher accuracy. However, our results also show a hybrid system's accuracy is constrained by its least effective component, sometimes underperforming its best constituent path. Our work provides systematic, quantitative evidence of this "weakest link" phenomenon in complex hybrid search, challenging the common "more is better" assumption.

The primary implication is that hybrid architectures are sensitive systems, not simple ensembles where strengths automatically aggregate. This highlights the symbiosis between candidate generation and re-ranking: a re-ranker's effectiveness is fundamentally constrained by its inputs. It cannot recover documents missed during initial retrieval, making candidate quality a hard ceiling on final accuracy. A critical design principle is therefore to validate component performance before integration. Dynamic path selection is a potential mitigation, which we will revisit in Section 6.2.

Table 8: A qualitative summary of hybrid search architecture trade-offs. Ratings are shown as a 5-unit bar (e.g., ■■■□□), where more black indicates better performance (higher accuracy or higher efficiency/lower cost). We use F, S, D, and T as abbreviations for FTS, SVS, DVS, and TenS, respectively.

#Paths	Type	Accuracy	Efficiency	Memory	Indexing
1	F	■□□□□	■■■■■	■■■■■	■■■■■
	S	■□□□□	■■■■■	■■■■■	■■■■■
	D	■□□□□	■■■■■	■■■■■	■■■■■
	T	■■□□□	■□□□□	■□□□□	■□□□□
2	F+S	■■□□□	■■■□□	■■■□□	■■■□□
	F+D	■■□□□	■■■□□	■■■□□	■■■□□
	F+T	■■■□□	■□□□□	■□□□□	■□□□□
	S+D	■■□□□	■■■□□	■■■□□	■■■□□
	S+T	■■■□□	■□□□□	■□□□□	■□□□□
	D+T	■■■□□	■□□□□	■□□□□	■□□□□
3	F+S+D	■■■□□	■■□□□	■■□□□	■■□□□
	F+S+T	■■■■□	■□□□□	■□□□□	■□□□□
	F+D+T	■■■■□	■□□□□	■□□□□	■□□□□
	S+D+T	■■■■□	■□□□□	■□□□□	■□□□□
4	F+S+D+T	■■■■□	■□□□□	■□□□□	■□□□□

6.1.2 A Data-Driven Map of Performance Trade-offs. Table 8 shows that architectures excel in specific dimensions (e.g., TenS-inclusive paths for accuracy, single DVS for speed). This reveals a fundamental trade-off: no single architecture excels across accuracy, efficiency, and resource overhead simultaneously. Currently, navigating these trade-offs has been challenging, with choices often based on intuition or ad-hoc pairwise studies. Our work provides a more systematic foundation, enabling quantitative assessment of a configuration's specific performance profile.

The practical implication is that a static architecture is likely suboptimal. Our findings instead point to the need for systems that adaptively select path combinations based on query type, data, or performance requirements. The trade-off map also reveals practical patterns, such as the FTS+SVS+DVS combination, which emerges as a highly balanced "sweet spot": it achieves high accuracy without the severe resource demands of TenS-inclusive configurations.

6.1.3 TRF: A Practical Approach for In-Database Fine-Grained Re-ranking. We find that Tensor-based Re-ranking Fusion (TRF) is a practical and highly effective architecture for in-database re-ranking. It fills a key gap in the re-ranking landscape: our results (Section 5.3) show it consistently outperforms coarse-grained fusion

methods like RRF and WS. Unlike a full TenS path, TRF operates only on a small candidate set, making it far more resource-efficient. Unlike heavyweight cross-encoders [71, 145] requiring separate GPU processing, TRF is designed as an in-database operation.

The implication is that TRF offers a clear upgrade path from traditional fusion methods. Practitioners can gain a significant accuracy boost for a moderate increase in latency and memory, making TRF attractive for accuracy-critical applications. We highlight that these strong results were achieved with a baseline, unoptimized TRF implementation; its high performance, coupled with optimization potential (e.g., tensor compression), marks it as a promising direction for future database retrieval systems, as we discuss next.

6.2 Challenges and Future Directions

Our findings illuminate key challenges and open promising research directions. We articulate five: developing adaptive architectures, managing index maintenance costs, optimizing tensor-based re-ranking, extending hybrid search to multi-modal documents, and performing end-to-end evaluations within RAG systems.

6.2.1 Towards Adaptive Hybrid Search. Our findings—that a "weakest link" can degrade performance and that no single architecture is universally optimal in performance trade-offs—reveal a critical limitation of current static systems. A fixed combination of retrieval paths is unlikely to be optimal for all queries or data subsets. This highlights a major research direction: *adaptive hybrid search architectures*. Future systems could incorporate a cost-based query optimizer to predict the effectiveness and resource cost of each available path, enabling the dynamic selection of an optimal path set based on user-defined constraints (e.g., a latency budget) and pruning a predicted "weak link" at runtime. Moving from static to query-aware execution is a crucial step for building robust hybrid search systems. For example, our analysis (Section 5.1) indicates query length is a simple yet effective heuristic: short, keyword-centric queries are often best served by FTS's low-latency precision, whereas longer, descriptive queries provide the context for DVS to achieve superior semantic matching. A query-aware routing mechanism analyzing these characteristics could dynamically select an optimal path combination, mitigating the "weakest link" problem.

6.2.2 Managing Index Maintenance Costs. A further evaluation may consider not only read-path performance but also the cost of maintaining multiple indexes. Our evaluation (Figure 10) quantifies the initial, offline construction costs, detailing import time, construction time, peak memory, and on-disk size for each paradigm. This provides a view of static deployment costs. However, real-world data is rarely static. A critical area is the analysis of *dynamic* maintenance costs, such as ingestion throughput and update/delete latency. Architectural choices benefiting read performance may introduce significant write-path complexities; for instance, maintaining consistency across four separate indexes during high-volume updates is a non-trivial challenge. A systematic study of these dynamic trade-offs is essential for understanding the total cost of ownership of advanced hybrid search systems.

6.2.3 Optimizing the Cost-Performance of Tensor-based Re-ranking. Our finding identifies TRF as a high-efficacy method that outperforms traditional fusion. Its broader adoption, however, is limited by higher computational and memory costs compared to RRF. This presents a valuable research direction: *optimizing the cost* of TRF. Tensor compression is a promising avenue; 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, computation acceleration for the MaxSim operation is critical, perhaps by using hardware-specific instructions (e.g., SIMD) or developing more efficient token-level interaction algorithms. Such optimizations could eliminate the primary barriers to TRF's adoption, positioning it as a new standard for high-accuracy re-ranking.

6.2.4 Hybrid Search for Multi-Modal Documents. While our work provides a foundation for textual hybrid search, a significant future direction is extending these principles to *multi-modal documents*. This requires designing fusion mechanisms that can effectively weigh and combine signals from visual modalities (e.g., semantic embeddings from models like ColPali [50]) with those from textual modalities (e.g., exact keyword matches from Optical Character Recognition (OCR) [113]) to improve retrieval performance.

6.2.5 End-to-End Evaluation within RAG Systems. Another vital direction is evaluating hybrid search *within an end-to-end Retrieval-Augmented Generation (RAG) framework*, not just as standalone components. The retrieval module's performance is fundamentally coupled with other pipeline components. A system-level evaluation must therefore analyze the interplay between the chosen hybrid strategy and other factors, such as document chunking policies and query intent recognition. Understanding how these architectural interactions impact the quality and factuality of the final generated output is essential for building optimized RAG systems.

7 CONCLUSION

We conducted the first systematic study of advanced hybrid search architectures. Our novel evaluation framework comprehensively evaluates four major retrieval paradigms and their 15 combinations across 11 real-world 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, showing 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 our results provide a crucial foundation for practitioners building next-generation retrieval systems and for researchers exploring adaptive hybrid search.

ACKNOWLEDGMENTS

The authors are deeply grateful to colleagues from the `Infinity` team for their foundational contributions and insightful discussions. Special thanks go to Zhichang Yu, Yushi Shen, Zhiqiang Yang, Ling Qin, and Yi Xiao for their invaluable efforts in advancing the `Infinity` project. This work was supported in part by the NSFC (Grant Nos. 62025206, U23A20296, 62502434), Zhejiang Province's "Lingyan" R&D Project (Grant No. 2024C01259), and Yongjiang Talent Introduction Programme (Grant No. 2022A-237-G). Yunjun Gao is the corresponding author of this paper.

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] 2024. We Make AI Work. https://vespa.ai/. [Online; accessed 21-September-2025].

[11] 2025. Chroma is the open-source search and retrieval database for AI applications. https://www.trychroma.com/. [Online; accessed 21-September-2025].

[12] 2025. Find the meaning in your data. https://opensearch.org/. [Online; accessed 21-September-2024].

[13] 2025. High-Performance Vector Search at Scale. https://qdrant.tech/. [Online; accessed 21-September-2025].

[14] 2025. Hybrid Search. https://turbopuffer.com/docs/hybrid. [Online; accessed 20-September-2025].

[15] 2025. Hybrid Search Explained. https://weaviate.io/blog/hybrid-search-explained. [Online; accessed 06-February-2025].

[16] 2025. Hybrid Search with Milvus. https://milvus.io/docs/hybrid_search_with_milvus.md. [Online; accessed 06-February-2025].

[17] 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].

[18] 2025. Reranking. https://milvus.io/docs/reranking.md. [Online; accessed 20-March-2025].

[19] 2025. A Review of Hybrid Search in Milvus. https://zilliz.com/blog/a-review-of-hybrid-search-in-milvus. [Online; accessed 20-September-2025].

[20] 2025. Supplementary Materials: Detailed Experimental Results. https://github.com/whenever5225/infinity/tree/main/exps/results. [Online; accessed 28-July-2025].

[21] 2025. The vector database for scale in production. https://www.pinecone.io/. [Online; accessed 21-September-2025].

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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).

[33] 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.

[34] 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.

[35] 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.

[36] 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).

[37] 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).

[38] 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).

[39] 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.

[40] 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.

[41] 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.

[42] 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).

[43] 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.

[44] 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.

[45] 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.

[46] 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.

[47] 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.

[48] 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.

[49] 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.

[50] 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*.

[51] 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).

[52] 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.

[53] 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.

[54] 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.

[55] 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.

[56] 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.

[57] 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.

[58] 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.

[59] 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.

[60] 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.

[61] 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.

[62] 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.

[63] 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.

[64] 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.

[65] Xinshuo Hu, Zifei Shan, Xinping Zhao, Zetian Sun, Zhenyu Liu, Dongfang Li, Shaolin Ye, Xinyuan Wei, Qian Chen, Baotian Hu, Haofen Wang, Jun Yu, and Min Zhang. 2025. KaLM-Embedding: Superior Training Data Brings A Stronger Embedding Model. *CoRR* abs/2501.01028 (2025).

[66] 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.

[67] 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.

[68] 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).

[69] 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.

[70] 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.

[71] 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.

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

[73] Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2025. NV-Embed: Improved Techniques for Training LLMs as Generalist Embedding Models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*.

[74] 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.

[75] 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.

[76] 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.

[77] 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).

[78] 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).

[79] 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.

[80] 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.

[81] 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.

[82] 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.

[83] 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.

[84] 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.

[85] 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.

[86] 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.

[87] 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.

[88] 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.

[89] 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.

[90] 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.

[91] 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.

[92] 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.

[93] 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.

[94] 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.

[95] 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.

[96] 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.

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

[98] 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.

[99] 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.

[100] 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.

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

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

[103] 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.

[104] 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.

[105] 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.

[106] 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.

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

[108] 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.

[109] 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.

[110] 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.

[111] 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.

[112] 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*.

[113] 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.

[114] 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.

[115] 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)*.

[116] 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.

[117] 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.

[118] 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.

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

[120] 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.

[121] 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.

[122] 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.

[123] 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.

[124] 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.

[125] 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.

[126] 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.

[127] 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.

[128] 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.

[129] 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).

[130] 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.

[131] 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.

[132] 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.

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

[134] 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.

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

[136] 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.

[137] 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).

[138] 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.

[139] 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.

[140] 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.

[141] 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.

[142] 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.

[143] 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.

[144] 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).

[145] 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.

[146] Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025. Qwen3 Embedding: Advancing Text Embedding and Reranking Through Foundation Models. *CoRR* abs/2506.05176 (2025).

[147] 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.

[148] Xinping Zhao, Xinshuo Hu, Zifei Shan, Shouzheng Huang, Yao Zhou, Xin Zhang, Zetian Sun, Zhenyu Liu, Dongfang Li, Xinyuan Wei, Youcheng Pan, Yang Xiang, Meishan Zhang, Haofen Wang, Jun Yu, Baotian Hu, and Min Zhang. 2025. KaLM-Embedding-V2: Superior Training Techniques and Data Inspire A Versatile Embedding Model. *CoRR* abs/2506.20923 (2025).

[149] 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.

[150] 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.