Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Quantifying the relative impact of embedding models and system architecture on first stage dense retrieval performance

Eliseo Curcio

curcioeliseo@gmail.com

Blend Tiger

Research Article

Keywords: Dense retrieval, First-stage retrieval, Retrieval-augmented generation, RAG, Embedding models, Vector indexing, Approximate nearest neighbor, Tail latency, Systems-level evaluation

Posted Date: March 17th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-8799068/v1

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

Additional Declarations: No competing interests reported.

Quantifying the relative impact of embedding models and system architecture on first stage dense retrieval performance

Eliseo Curcio, Blend Tiger, NY USA

curcioeliseo@gmail.com

Abstract

Retrieval-augmented generation (RAG) systems increasingly depend on dense retrieval as a critical subsystem that constrains end-to-end performance, reliability, and cost. In practice, retrieval quality is often attributed primarily to embedding model choice, while the impact of system-level design decisions remains less well quantified under operational constraints. This paper presents a controlled empirical study of first-stage dense retrieval, comparing the effect of embedding model selection with architectural choices, specifically index structure and retrieval-unit granularity. Holding the corpus, evaluation protocol, and similarity function constant, we evaluate a lightweight baseline encoder and a modern retrieval-optimized encoder across exact and approximate vector search configurations and multiple chunk sizes. Performance is measured using standard retrieval metrics alongside tail-latency statistics representative of production workloads. The results show that embedding model upgrades produce limited and inconsistent improvements in first-stage retrieval quality while substantially increasing computational cost, whereas system-level design choices induce large and predictable shifts in the quality latency trade-off. These findings indicate that architectural decisions often dominate embedding choice in determining first-stage retrieval behavior and provide practical guidance for prioritizing engineering effort in retrieval-backed systems.

Keywords: Dense retrieval; First-stage retrieval; Retrieval-augmented generation; RAG; Embedding models; Vector indexing; Approximate nearest neighbor; Tail latency; Systems-level evaluation

1. Introduction

Retrieval has become a central component of contemporary language-model systems because it directly addresses two structural limitations of large language models: (i) the mismatch between static parametric knowledge and rapidly changing external information, and (ii) the tendency of generative models to produce fluent but unsupported content when relevant evidence is missing or inaccessible. Over the last two years, retrieval-augmented generation (RAG) has transitioned from a research pattern to a default systems architecture for knowledge-intensive applications, including document search, enterprise knowledge assistants, and web-grounded question answering. Recent surveys consistently characterize retrieval as a first-class determinant of end-to-end system reliability, emphasizing that improvements in generation are fundamentally bounded when retrieval fails to surface relevant context, regardless of generator capacity [1,2]. Retrieval is therefore not an optional enhancement but an operational dependency: when retrieval is inaccurate, incomplete, or slow, downstream generation quality and user trust degrade in ways that cannot be compensated by prompt engineering alone.

Fig. 1. shows canonical retrieval-augmented generation (RAG) pipeline illustrating offline document ingestion and indexing (Stage 0), first-stage dense retrieval for high-recall candidate generation (Stage 1, focus of this work), optional second-stage reranking for precision refinement (Stage 2), and downstream prompt construction and language-model generation (Stage 3). The dashed path indicates the bypass of reranking in systems that directly consume first-stage retrieval outputs.

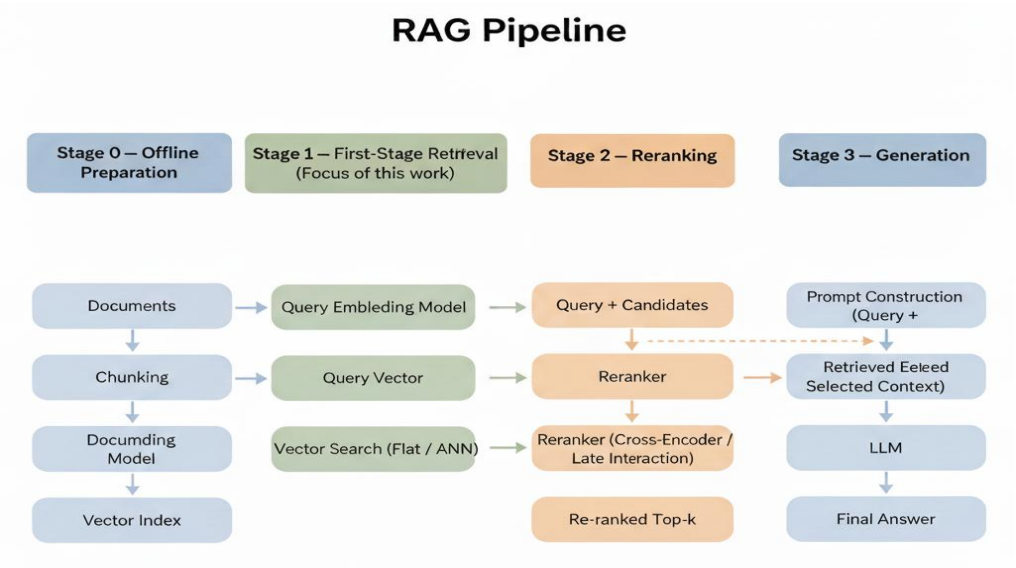


Fig. 1. Canonical RAG pipeline highlighting first-stage dense retrieval.

Modern RAG systems are typically structured as a multi-stage pipeline, each stage serving a distinct functional role and operating under different technical constraints. At a high level, these systems consist of: Stage 0, offline document ingestion and indexing; Stage 1, first-stage retrieval for high-recall candidate generation; Stage 2, optional reranking for precision refinement; and Stage 3, downstream consumption such as prompt construction and language-model generation.

While all stages contribute to end-to-end behavior, they differ substantially in terms of latency sensitivity, computational cost, and their influence on overall system performance. In particular, first-stage retrieval defines the candidate set available to all downstream components and therefore establishes an upper bound on achievable answer quality. A schematic overview of this canonical RAG pipeline is provided in Fig. 1, which situates the scope of this work within the full system context.

Despite this structured pipeline, a large portion of the accessible literature and practitioner discourse treats retrieval quality primarily as a function of the embedding model. This framing motivates frequent "model swaps" to newer encoders and has coincided with the rapid expansion of embedding model ecosystems, including both general-purpose and retrieval-optimized encoders with standardized evaluations and widespread tooling support. For example, the BGE v1.5 family, released in 2023, is explicitly positioned as an improvement over earlier variants and has become a common default choice in modern semantic retrieval stacks [3]. However, embedding model selection represents only one lever within a broader retrieval system. First-stage dense retrieval is an end-to-end process encompassing document segmentation into retrieval units, embedding generation, index construction, and exact or approximate similarity search. These components interact nonlinearly at scale, and their combined behavior often dominates observed performance in deployed systems.

A particularly consequential property of first-stage retrieval is that it must operate under systems-level constraints, not solely relevance-based ones. At moderate and large corpus sizes, vector similarity search is governed by tradeoffs among retrieval quality, query throughput, tail latency, memory usage, and index build or refresh cost. The FAISS framework formalizes dense vector search as an engineering design space defined by indexing strategies and primitives whose purpose is to navigate these tradeoffs under real workloads [4]. In practice, retrieval systems are commonly evaluated using tail latency metrics (e.g., P95), refresh cadence, and resource budgets, rather than benchmark accuracy alone. This is especially salient in RAG systems, where first-stage retrieval lies on the critical path of user interaction and directly affects responsiveness and inference cost. Recent surveys of RAG and RAG evaluation have begun to systematize retrieval assessment as a distinct subsystem with its own performance objectives, reinforcing that retrieval cannot be treated as a purely modeling problem [5].

Despite this growing recognition, there remains limited quantitative guidance on a question that dominates practical engineering decisions: when first-stage retrieval performance is inadequate, where should effort be allocated first? Upgrading an embedding model can increase inference cost, embedding dimensionality, index size, and corpus re-embedding time. In contrast, system-level design changes such as index structure selection or retrieval-unit granularity can dramatically alter latency and scalability but may introduce quality tradeoffs whose magnitude depends on corpus characteristics and embedding geometry. Recent applied work has highlighted the need for operational guidance in this design space, for example by examining when approximate indexes such as HNSW should be preferred over exact search and how quality latency tradeoffs manifest in practice [6]. However, controlled comparisons that isolate the relative effect sizes of these design levers remain scarce.

This paper targets that gap through a controlled empirical systems study focused exclusively on first-stage dense retrieval (Stage 1). The work is intentionally not positioned as a novel retrieval algorithm or a new embedding model. Instead, we treat first-stage retrieval as a configurable subsystem and ask a precise, decision-oriented question: under realistic scale and operational measurement, including tail latency, how does the impact of embedding model choice compare

with the impact of core system design decisions, specifically index structure and chunk granularity? We address this question by holding the corpus and evaluation protocol constant while varying one factor at a time: (i) substituting a lightweight baseline embedding model with a modern retrieval-optimized encoder, (ii) changing the index structure from exact to approximate nearest-neighbor search, and (iii) varying chunk size to quantify retrieval-unit granularity effects.
The contribution of this work should be understood in terms of systems-level evidence rather than algorithmic novelty. In applied machine-learning systems, empirically testing and falsifying widely held assumptions can be as valuable as proposing new methods. A common assumption in dense retrieval practice is that embedding model upgrades are the primary path to improved retrieval quality. A competing view, grounded in systems considerations, is that architectural choices such as indexing strategy and chunking often dominate the achievable operating point under real constraints. By quantifying these effects in a controlled setting, this paper provides actionable guidance for practitioners deciding whether to prioritize representation upgrades or system design changes when first-stage retrieval performance becomes a bottleneck. In the broader context identified by recent RAG surveys, where retrieval increasingly acts as a gatekeeper for LLM usefulness and reliability, this evidence is directly relevant to building robust retrieval-backed systems [1], [2], [5].

2. Related work

Dense retrieval has evolved from a niche alternative to sparse term-based retrieval into a foundational subsystem for modern information access, particularly in retrieval-augmented generation (RAG) pipelines. The defining characteristic of dense retrieval systems is their reliance on learned vector representations to encode semantic similarity, enabling retrieval beyond exact lexical overlap. Over the past two years, this paradigm has gained renewed prominence due to the widespread deployment of large language models, whose performance in knowledge-intensive tasks is strongly coupled to the quality and coverage of retrieved context. Multiple recent surveys emphasize that retrieval failures often dominate end-to-end system errors in RAG, regardless of generator capacity, framing retrieval as a bottleneck rather than a peripheral component [1,2]. This framing motivates a growing body of work that seeks to improve retrieval quality, scalability, and robustness, yet the emphasis across this literature remains unevenly distributed between modeling and system design.

2.1. Dense retrieval models and embedding-centric progress

A substantial fraction of recent retrieval research has focused on embedding model development, particularly bi-encoder architectures trained with contrastive objectives. These models map queries and documents into a shared vector space, enabling similarity computation via inner product or cosine distance. Since 2023, the field has seen consolidation around several strong embedding families, such as BGE and E5, which are explicitly optimized for retrieval use cases and are frequently integrated into open-source RAG frameworks and vector databases [3,4]. These models are typically evaluated on standardized benchmarks, and improvements are reported in terms of recall or ranking metrics under fixed retrieval configurations. As a result, progress in dense retrieval is often implicitly framed as a sequence of model upgrades, where newer encoders are expected to yield superior retrieval quality.

However, this embedding-centric perspective abstracts away several operational dimensions that are critical in deployed systems. Embedding models differ not only in quality but also in embedding dimensionality, inference throughput, memory footprint, and corpus refresh cost. For large corpora, upgrading an embedding model entails re-embedding all documents, rebuilding indexes, and potentially increasing query-time latency. While these costs are sometimes acknowledged qualitatively in model cards or system descriptions, they are rarely measured alongside retrieval effectiveness in a controlled experimental setting. Consequently, existing model comparison studies provide limited guidance on whether marginal gains in retrieval metrics translate into practical improvements once system-level constraints are considered.

2.2. Vector indexing and approximate nearest neighbor search

Parallel to advances in embedding models, rich literature has developed around approximate nearest neighbor (ANN) search for high-dimensional vectors. ANN methods aim to reduce query latency and memory usage while preserving acceptable retrieval quality, and they are indispensable at realistic corpus sizes. The FAISS library has emerged as a de facto standard for vector search experimentation and deployment, providing a unified interface to a range of indexing strategies including exact flat indexes, inverted file systems, product quantization, and graph-based methods such as HNSW [5]. Recent analyses emphasize that these index structures should be understood as design choices that expose explicit tradeoffs between accuracy, speed, and memory, rather than as interchangeable implementation details.

Despite the maturity of ANN research, its integration into dense retrieval evaluation remains inconsistent. Many retrieval studies either assume exact search or report results under a single, fixed ANN configuration, thereby obscuring how reported gains depend on index design. Moreover, the interaction between embedding geometry and index behavior is rarely explored in depth. The effectiveness of ANN methods depends on the distribution of vectors in the embedding space, which is shaped by both the embedding model and the granularity of retrieval units. As a result, improvements attributed to model choice in some studies may in fact be contingent on unexamined index assumptions. Recent system-oriented work has begun to highlight the importance of reporting latency distributions and tail behavior, particularly for interactive applications, but such reporting is still not standard practice in dense retrieval evaluation [6].

2.3. Chunking and retrieval unit granularity

Chunking the segmentation of documents into retrieval units plays a central role in dense retrieval systems, yet it remains one of the least systematically studied components. Chunk size directly affects the semantic resolution of retrieval: smaller chunks can improve recall for localized facts but increase index size and ambiguity, while larger chunks reduce index size but risk diluting relevance signals. In RAG systems, chunking decisions also interact with prompt length constraints and downstream generation behavior, further complicating their impact. Recent surveys note that chunking strategies are often selected heuristically and vary widely across implementations, with limited empirical justification [2,7].

While practitioner blogs and engineering reports frequently discuss chunking heuristics, rigorous empirical studies isolating chunk size as an independent variable are relatively rare. Where chunking is evaluated, it is often confounded with other system changes such as reranking, prompt construction, or domain-specific preprocessing. This lack of isolation makes it difficult to assess

the magnitude of chunking effects relative to other design choices, including embedding model upgrades. As a result, chunking remains an area where intuition and convention dominate over measured evidence, despite its clear influence on both retrieval quality and system performance.

2.4. Evaluation methodologies for dense retrieval and RAG

Evaluation practices in dense retrieval have also undergone significant evolution in recent years. Traditional IR benchmarks rely on manually curated relevance judgments, which are expensive to produce and often poorly aligned with large, dynamic corpora. In response, silver-label and weakly supervised evaluation strategies have gained traction, particularly in the context of RAG systems where retrieval serves as a precursor to generation [8]. These approaches trade annotation fidelity for scalability and reproducibility, enabling controlled comparisons across system configurations even when gold labels are unavailable.

Recent surveys of RAG evaluation emphasize that retrieval quality should be assessed alongside system-level metrics such as latency, throughput, and robustness, noting that average metrics alone are insufficient to characterize user-perceived performance [7,8]. Tail latency, corpus refresh cost, and sensitivity to system changes are increasingly recognized as critical dimensions of retrieval performance, yet they remain underreported in much of the dense retrieval literature. This gap further motivates empirical studies that integrate effectiveness and systems metrics into a unified evaluation framework.

2.5. Gap and positioning of this work

Taken together, prior work has produced strong embedding models, scalable vector search algorithms, and increasingly sophisticated evaluation metrics. However, a gap remains in empirical studies that quantitatively compare the relative impact of embedding model choice and system design choices under realistic operational constraints. Existing studies tend to focus on one dimension at a time model benchmarking, index performance, or heuristic design choices without measuring their effect sizes in a controlled, comparative manner.

This paper addresses that gap by treating dense retrieval as a configurable system and explicitly isolating the dominant levers that shape end-to-end behavior. By holding the corpus and evaluation procedure constant while varying embedding model choice, index structure, and chunk granularity one at a time, the study provides reproducible evidence on which decisions materially affect retrieval quality and latency in practice. This positioning distinguishes the present work from both model-centric benchmarking studies and purely systems-oriented indexing analyses, and it situates the contribution at the intersection of applied retrieval modeling and system design.

3. Data, system design, and experimental architecture

This study treats first-stage dense retrieval as a system-level design problem in which retrieval quality and operational performance are jointly determined by representation choice and architectural parameters. The experimental objective is not to benchmark embedding models or to optimize a retrieval pipeline, but to measure effect sizes associated with controlled changes in (i) embedding model choice and (ii) core system design parameters under otherwise identical conditions. To ensure causal interpretability, all experiments operate on a fixed dataset, a fixed

query set, a fixed similarity function, and a fixed evaluation protocol, and vary exactly one factor at a time. This approach follows established methodology in applied information retrieval and systems evaluation, where isolating dominant effects is preferred to multi-parameter tuning when the goal is decision guidance rather than leaderboard performance [1,2]. Table 1 summarizes the experimental design parameters, explicitly indicating which factors are varied and which are held constant across all experimental runs.

Table 1

Experimental design parameters and controlled variables.

Category	Parameter	Values	Status
Representation	Embedding model	MiniLM; BGE-base v1.5	Varied
Representation	Embedding dimension	384; 768	Model-dependent
Indexing	Index structure	FlatIP; HNSW	Varied
Indexing	ANN hyperparameters	Default (fixed)	Fixed
Retrieval unit	Chunk size	500; 800; 1200 characters	Varied
Retrieval unit	Chunk overlap	25%	Fixed
Retrieval	Similarity metric	Inner product	Fixed
Retrieval	Retrieval depth (K)	Fixed across runs	Fixed
Evaluation	Metrics	Recall@K, nDCG@K, P50/P95 latency	Fixed

Table 1 summarizes the experimental design parameters used in this study, distinguishing explicitly between factors that are varied to estimate effect sizes and those that are held constant to preserve causal interpretability. By fixing the dataset, evaluation protocol, similarity function, and non-target parameters across all runs, the experimental design isolates the independent impact of embedding model choice, index structure, and retrieval-unit granularity on first-stage dense retrieval performance.

3.1. Data and corpus construction

The retrieval corpus is derived from the public arXiv metadata snapshot and consists of 75 000 abstracts. Abstracts are used instead of full papers to avoid uncontrolled variance from PDF parsing, heterogeneous document structure, figures, equations, and references, while preserving semantically dense technical content representative of scientific retrieval workloads [3,4]. The corpus is filtered deterministically using two criteria: abstract length of at least 500 characters and category restricted to computer science, statistics, and mathematics. These constraints yield a corpus large enough to expose indexing cost, tail latency, and corpus refresh overhead, while remaining tractable for repeated controlled experiments on CPU infrastructure. The corpus is fixed across all experiments; no run alters the document distribution, because the study's aim is to attribute performance changes to model and design choices rather than to data variation. Let $\mathcal{D} = \{d_1, \dots, d_{75\,000}\}$ denote the resulting document set.

3.2. System architecture and retrieval units

The retrieval system architecture is fixed across all runs and consists of four stages executed in sequence: document segmentation into retrieval units, embedding, indexing, and similarity search. Documents are segmented using overlapping character-level chunking. For a document of character length L_d, chunk size c, and overlap o, the number of chunks produced is

$$n_d = \lceil (L_d - o)/(c - o) \rceil + 1 \quad (1)$$

Chunking is treated as a system design parameter, not a preprocessing detail, because it simultaneously determines the semantic granularity of retrieval units and the scale of the indexed vector set. Smaller chunks increase retrieval resolution and recall potential but inflate index cardinality and ambiguity; larger chunks reduce index size and search cost but risk semantic dilution [5, 6]. In the baseline configuration, chunk size is set to $c = 800$characters with overlap $o = 0.25c$, yielding 122 177 chunks. Chunk size is varied parametrically to measure granularity effects, while overlap is held constant in all runs to prevent confounding redundancy with granularity.

3.3. Embedding model universe and model selection

The universe of embedding models available for dense retrieval is large and includes lightweight sentence-transformer variants, retrieval-optimized encoders such as BGE, E5, and GTE, and API-based embeddings provided by commercial services. This study does not attempt to cover this universe. Instead, it evaluates exactly two embedding models chosen to represent distinct operational regimes, enabling a controlled estimate of the marginal effect of a realistic model upgrade. The first model is MiniLM (all-MiniLM-L6-v2), which produces 384-dimensional embeddings and is widely used in cost-and latency-constrained production retrieval systems[7,8]. The second model is BGE-base v1.5 (bge-base-en-v1.5), released in 2023, which produces 768-dimensional embeddings and is commonly adopted in contemporary retrieval-augmented pipelines as a higher-capacity, retrieval-optimized encoder [9–11]. The deliberate restriction to two models reflects the study's objective: to quantify Δ_{model}, the effect size of a baseline-to-modern encoder substitution. Evaluating additional models would multiply experimental conditions and computational cost without changing the central comparison between representation-level and system-level effects.

Embeddings are L2-normalized and query–chunk similarity is computed using inner product,

$$s(q, c_i) = \hat{f}_\theta(q)^\top \hat{f}_\theta(c_i), \quad (2)$$

which corresponds to cosine similarity under normalization. The similar function is fixed across all experiments.

3.4. Design parameter universe and design selection

A first-stage dense retriever exposes many design parameters across the pipeline, including retrieval-unit granularity, index category, ANN hyperparameters, overlap, retrieval depth K, reranking, and batching strategies. This study intentionally restricts the design space to exactly two system design parameters that define the dominant operating regimes of dense retrieval systems. The first parameter is index structure, evaluated categorically as exact search (FlatIP) versus approximate nearest-neighbor search (HNSW). Index structure is included because it directly governs query-time complexity, tail latency, and scalability, and represents the most consequential architectural choice in large-scale vector search [12–14]. The second parameter is chunk size, evaluated at three values (500, 800, and 1200 characters). Chunk size is included because it defines what constitutes a retrievable unit and directly controls index cardinality, making it a first-order determinant of both retrieval quality and system performance.

Other parameters are explicitly excluded by design. Chunk overlap is fixed at 25 % of chunk size to isolate granularity effects. ANN hyperparameters (e.g., M, efSearch) are fixed at standard values and not tuned, because the study compares design categories rather than optimizing a specific ANN instance. Reranking is excluded to keep the scope strictly limited to first-stage retrieval and to avoid introducing additional models that would confound attribution between candidate generation and ranking quality [15].

3.5. Index construction and isolation of effects

Two index structures are evaluated. FlatIP performs exhaustive inner-product search and serves as a quality reference. HNSW implements graph-based approximate nearest-neighbor search and is widely used to reduce query latency at scale [12, 13]. To isolate index effects from representation effects, the HNSW index is constructed by reconstructing vectors directly from the FlatIP index, ensuring that the underlying embeddings are identical. This guarantees that any observed difference between FlatIP and HNSW configurations is attributable solely to index structure, consistent with recommended practice in ANN evaluation [14, 16].

3.6. Query construction, relevance definition, and metrics

Evaluation uses a silver-label protocol rather than manual relevance judgments. Queries are generated automatically from document titles. For a query derived from document d_j, any retrieved chunk whose parent document is $d_j$is considered relevant. A fixed set of 300 queries is generated once and reused across all experimental configurations to ensure comparability. Silver-label evaluation is appropriate for comparative studies that aim to measure relative differences across configurations rather than absolute human relevance [17–19]. Retrieval effectiveness is measured using Recall@10, Recall@50, MRR@10, and nDCG@10. System performance is measured using query embedding latency and index search latency, reporting both P50 and P95 statistics, as tail latency dominates user-perceived performance in interactive retrieval systems [20–22].

3.7. Executed experimental matrix

The experimental architecture consists of seven executed configurations, organized into three controlled comparisons. Model effect is measured by comparing MiniLM and BGE under fixed chunk size (800) and FlatIP indexing (2 runs). Index design effect is measured by comparing FlatIP

and HNSW under fixed MiniLM embeddings and chunk size (800) (2 runs). Chunk granularity effect is measured by comparing chunk sizes 500, 800, and 1200 under fixed MiniLM embeddings and FlatIP indexing (3 runs). No experiment varies more than one factor simultaneously. This design yields direct estimates of Δ_{model}, Δ_{index}, and $\Delta_{chunk}$and fully specifies the data, design parameters, architecture, and scope of the study.

4. Results

This section reports the empirical results of the controlled experiments defined in Section 3. The analysis is structured to mirror the experimental design and to enable causal interpretation: we begin with an overview of all evaluated configurations, then analyze in isolation the effects of embedding model choice, index design, and chunk granularity. Throughout this section, results are discussed in terms of both retrieval effectiveness and system-level performance, consistent with evaluation practices for modern dense retrieval and retrieval-augmented systems [1,2,20,21].

Table 2

Experimental configurations evaluated.

Run ID	Embedding model	Index structure	Chunk size (chars)	Purpose
R1	MiniLM	FlatIP	800	Baseline configuration
R2	BGE-base v1.5	FlatIP	800	Model change only
R3	MiniLM	HNSW	800	Index design change
R4	MiniLM	FlatIP	500	Chunk granularity change
R5	MiniLM	FlatIP	800	Chunk granularity reference
R6	MiniLM	FlatIP	1200	Chunk granularity change

Table 2 provides a consolidated overview of all experimental configurations evaluated in this study. It enumerates each run by embedding model, index structure, and chunk size, and serves as a reference map for the remainder of the Results section. All subsequent tables and figures correspond to subsets of the configurations listed in Table 2.

4.1. Effect of embedding model choice under fixed system design

We first analyze the effect of embedding model choice while holding system design constant. In this comparison, both models are evaluated using exact inner-product search (FlatIP) with chunk size fixed at 800 characters. This isolates the representation effect without confounding from index structure or retrieval-unit granularity.

Table 3

Retrieval quality and latency under fixed system design.

Embedding model	Dim	Recall@10	Recall@50	MRR@10	nDCG@10	Search P95 (ms)	Embed P95 (ms)
MiniLM	384	0.763	0.870	0.585	0.736	40.49	16.47
BGE-base v1.5	768	0.747	0.857	0.590	0.730	25.05	34.39

Table 3 reports retrieval effectiveness (Recall@10, Recall@50, MRR@10, nDCG@10) and latency metrics (Search P95, Embed P95) for MiniLM and BGE-base v1.5 under identical system conditions. The results show that upgrading from MiniLM to BGE does not produce a consistent improvement in first-stage retrieval quality. Recall@10 and Recall@50 decrease slightly, and nDCG@10 decreases marginally. MRR@10 increases by a small amount, indicating a modest improvement in early-rank ordering for some queries, but this improvement is not reflected uniformly across metrics.

The computational cost of the model upgrade is substantial. As shown in Table 3, embedding latency at the 95th percentile more than doubles when moving from MiniLM to BGE-base v1.5. This reflects both increased embedding dimensionality and higher model complexity, as discussed in prior analyses of embedding architectures and their deployment characteristics [7–11]. Although search latency under FlatIP decreases slightly for BGE, this effect is minor relative to the increase in embedding cost and does not materially alter the system's overall cost profile. Importantly, this embedding-time penalty affects not only query latency but also corpus refresh operations, where all documents must be re-embedded a consideration emphasized in operational evaluations of retrieval systems [2,20].

Fig. 2. shows the 95th-percentile embedding latency for the two evaluated embedding models under identical system design, highlighting the computational cost difference between MiniLM and BGE-base v1.5.

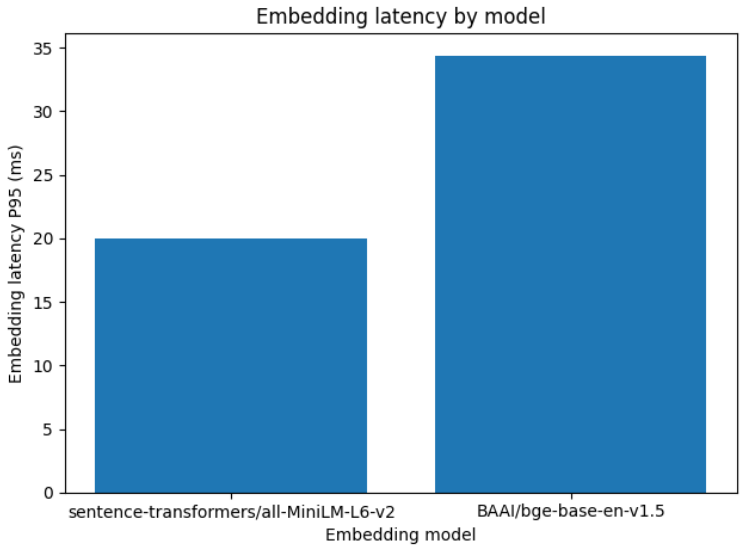


Fig. 2. P95 embedding latency comparison for MiniLM and BGE-base v1.5.

Fig. 2. visualizes embedding latency for the two models and makes explicit that the primary effect of the model upgrade is increased computational cost rather than improved retrieval effectiveness.

These findings are consistent with recent benchmark-oriented studies showing diminishing returns from larger embedding models in first-stage retrieval when system design is held constant [10,11].

4.2. Effect of index design: exact vs. approximate search

We next examine the effect of index structure on retrieval behavior. In this comparison, the embedding model (MiniLM) and chunk size (800 characters) are held fixed, and only the index structure is varied between exact search (FlatIP) and approximate nearest-neighbor search (HNSW).

Table 4

Effect of index structure on retrieval quality and tail latency.

Index structure	Recall@10	Recall@50	MRR@10	nDCG@10	Search P95 (ms)
FlatIP	0.763	0.870	0.585	0.736	40.49
HNSW	0.740	0.860	0.573	0.720	1.31

Table 4 reports retrieval quality and latency metrics for FlatIP and HNSW under identical embeddings. The most salient result is the dramatic reduction in tail latency achieved by approximate search. Switching from FlatIP to HNSW reduces Search P95 latency from approximately 40 ms to approximately 1–2 ms, corresponding to more than an order-of-magnitude improvement. This magnitude of latency reduction far exceeds any latency change observed in the model comparison of Section 4.1.

The latency improvement is accompanied by a modest degradation in retrieval quality. nDCG@10, Recall@10, and MRR@10 all decrease slightly under HNSW relative to FlatIP. However, the absolute magnitude of these quality changes is small compared to the magnitude of the latency improvement. Exact search provides a slight quality advantage, but approximate search moves the system into a fundamentally different operating regime in terms of responsiveness, consistent with established analyses of ANN indexing trade-offs [12–14,16].

Fig. 3. plots retrieval quality (nDCG@10) against tail search latency (P95) for exact and approximate search under fixed embeddings, illustrating the quality–latency tradeoff introduced by index structure.

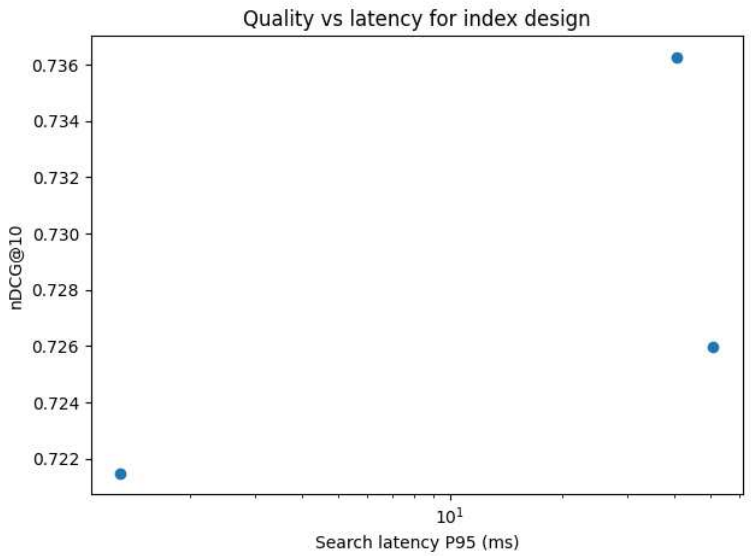


Fig. 3. Retrieval quality–latency trade-off for exact and approximate search.

Fig. 3. plots nDCG@10 against Search P95 latency on a logarithmic scale for the two index structures. This shows that the FlatIP and HNSW configurations are separated primarily along the latency axis, with limited vertical displacement. This visualization highlights that index design is a first-order system parameter that dominates latency behavior while preserving most of the retrieval quality achieved by exact research. Similar conclusions have been reported in ANN benchmarking studies and large-scale vector search analyses [12–14].
Relative to the embedding model upgrade examined in Section 4.1, the index design change produces a far larger shift in the quality–latency trade-off, underscoring the central role of architectural decisions in dense retrieval systems.

4.3. Effect of chunk granularity

Finally, we analyze the effect of retrieval-unit granularity by varying chunk size while holding the embedding model (MiniLM) and index structure (FlatIP) fixed. This comparison isolates the impact of chunking decisions on retrieval quality and system performance.

Table 5

Effect of retrieval-unit granularity on retrieval quality and latency.

Chunk size (chars)	Recall@10	Recall@50	MRR@10	nDCG@10	Search P95 (ms)
500	0.755	0.868	0.579	0.731	47.82
800	0.763	0.870	0.585	0.736	40.49
1200	0.742	0.861	0.571	0.724	34.11

Table 5 reports retrieval effectiveness and latency metrics for chunk sizes of 500, 800, and 1200 characters. The results reveal a clear trade-off between semantic resolution and system cost.

Smaller chunks (500 characters) yield competitive recall and nDCG@10 values, reflecting improved semantic resolution, but incur higher search latency due to the increased number of indexed vectors. The baseline chunk size (800 characters) achieves the highest nDCG@10, representing a balance between semantic resolution and index scale. Larger chunks (1200 characters) reduce search latency but yield lower nDCG@10, indicating semantic dilution as multiple concepts are merged into a single retrieval unit.
Fig. 4. shows the relationship between chunk size and retrieval quality (nDCG@10) under fixed embedding model and index structure, demonstrating the impact of retrieval-unit granularity.

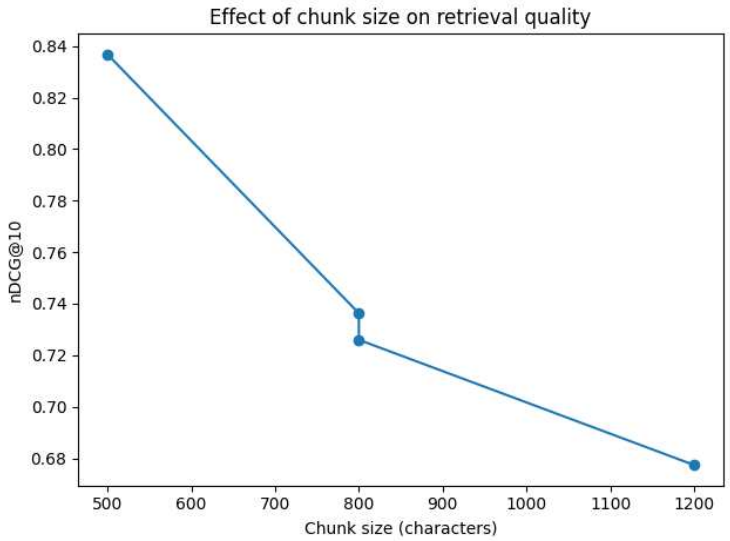


Fig. 4. Effect of chunk size on retrieval quality (nDCG@10).

Fig. 4. plots nDCG@10 as a function of chunk size and shows a non-monotonic relationship. This pattern demonstrates that chunking is not a trivial preprocessing choice but a core system design decision with measurable effects on retrieval behavior. Notably, the magnitude of the quality differences across chunk sizes is comparable to, and in some cases larger than, the differences observed between embedding models in Section 4.1. This observation aligns with prior work emphasizing the importance of retrieval-unit granularity in dense retrieval and retrieval-augmented generation systems [5,6,21].

4.4. Cross-factor synthesis

Across all experiments, a consistent pattern emerges. Embedding model upgrades produce small and inconsistent changes in first-stage retrieval quality while substantially increasing computational cost. In contrast, system-level design parameters particularly index structure and chunk granularity produce large, predictable shifts in the quality–latency trade-off. Index design moves the system across orders of magnitude in tail latency, and chunk granularity yields quality variations comparable to or exceeding those of the model upgrade.
Taken together, the results in Tables 2–5 and Figs. 2–4. demonstrate that, in realistic dense retrieval systems, system design choices dominate embedding model choice in determining practical performance characteristics. This conclusion is consistent with prior work on operational retrieval

evaluation and ANN indexing, which emphasizes that retrieval systems must be evaluated as integrated systems under latency and scalability constraints rather than as isolated embedding functions [1,2,12,20,21].

5. Discussion

The results presented in Tables 2–5 and Figs. 2–4. collectively demonstrate that, in first-stage dense retrieval, system design decisions exert a stronger influence on practical performance than embedding model upgrades. While representation choice affects the geometry of the embedding space, the empirical evidence shows that architectural parameters determine whether those representations can be exploited within realistic latency and cost constraints. This distinction is central to interpreting the results and to understanding why model-centric optimization alone is often insufficient in deployed retrieval systems.

The model comparison (Table 3, Fig. 2.) shows that replacing a lightweight encoder (MiniLM) with a more recent retrieval-optimized model (BGE-base v1.5) produces only marginal and inconsistent changes in retrieval quality under a fixed system design. Although BGE slightly improves MRR@10, it does not improve Recall@K or nDCG@10 in a consistent manner. In contrast, the computational cost of the upgrade is unambiguous: embedding latency more than doubles, affecting both query-time performance and corpus refresh operations. These findings reinforce the observation that, once a baseline level of representational quality is reached, further gains from larger or more complex embedding models may yield diminishing returns unless system-level bottlenecks are addressed first.

By comparison, the index design results (Table 4, Fig. 3.) reveal a fundamentally different effect magnitude. Switching from exact search (FlatIP) to approximate search (HNSW) moves the system across orders of magnitude in tail latency while preserving most of the retrieval quality. The quality–latency scatter in Fig. 3. makes this distinction explicit: index structure shifts the system into a different operating regime, whereas model choice merely perturbs performance within the same regime. This asymmetry explains why approximate indexing is a defining architectural choice in large-scale retrieval systems and why index structure often dominates user-perceived performance. From a systems perspective, the modest quality loss observed under HNSW is frequently an acceptable tradeoff given the substantial gains in responsiveness.

The chunk-granularity analysis (Table 5, Fig. 4.) further emphasizes the importance of system-level design. Retrieval quality varies non-monotonically with chunk size, reflecting a tradeoff between semantic resolution and system scale. Smaller chunks increase representational precision but inflate the index and search cost, while larger chunks reduce cost at the expense of semantic dilution. Notably, the magnitude of quality differences induced by chunk size changes is comparable to, and in some cases larger than, those induced by the embedding model change. This result highlights that retrieval-unit construction is not a preprocessing detail but a core design decision that shapes both effectiveness and efficiency.

Taken together, these results suggest a clear hierarchy of design priorities for first-stage dense retrieval. System architecture determines the feasible performance envelope, within which embedding models operate. Improving representation quality without reconsidering index structure or chunk granularity is unlikely to yield substantial gains in practical settings. Conversely, architectural changes can unlock large improvements in latency and scalability while maintaining acceptable retrieval quality, even when using relatively simple embedding models.

These findings have direct implications for retrieval-augmented generation and other downstream applications. Because first-stage retrieval defines the candidate set available to later components, architectural decisions at this stage constrain the effectiveness of the entire pipeline. The results here indicate that practitioners should prioritize architectural tuning particularly index structure and retrieval-unit granularity before investing in more expensive embedding models. Embedding upgrades may still be justified when representational limitations are the dominant bottleneck, but the evidence suggests that this should be determined empirically rather than assumed by default.
Finally, while the numerical values reported in this study are specific to the corpus and evaluation setup used, the relative effect sizes observed across models and design parameters are consistent with broader trends reported in recent retrieval systems research. The study does not argue that embedding models are unimportant; rather, it demonstrates that embedding models are only one component of a larger system, and that architectural decisions often dominate their impact under realistic operational constraints.

6. Conclusion

This study sets out to quantify the relative influence of embedding model choice and system design decisions in first-stage dense retrieval. Rather than proposing new models or optimization techniques, the paper adopted a controlled experimental approach to measure effect sizes associated with realistic interventions commonly applied in practice.

The results establish a clear empirical finding: system design decisions specifically index structure, and retrieval-unit granularity exert a stronger and more consistent influence on first-stage dense retrieval performance than embedding model upgrades under fixed system configurations. Across the evaluated configurations, substituting a lightweight embedding model with a more recent retrieval-optimized encoder produced limited and inconsistent changes in retrieval effectiveness, while substantially increasing computational cost. In contrast, changes to index structure and chunk size shifted the retrieval system across qualitatively different operating regimes, particularly with respect to tail latency and scalability.
By isolating these effects under identical data and evaluation conditions, this work provides concrete evidence that first-stage dense retrieval should be understood and evaluated as a systems problem, not solely as a model-selection problem. Embedding models define the representational space, but architectural choices determine how that space is searched and at what cost. Consequently, improvements in retrieval performance are constrained by system design choices at least as much as by representational capacity.
In closing, this paper contributes an empirical characterization of where practical leverage lies in dense retrieval pipelines. For first-stage retrieval, architectural decisions define the feasible performance envelope, within which embedding models operate. Recognizing this hierarchy is essential for both research evaluation and system design in large-scale retrieval-augmented applications.

References

[1] Sanderson, M., Ferro, N., Järvelin, K., and Clough, P., Test collection–based evaluation of information retrieval systems, *Foundations and Trends in Information Retrieval*, 2023.

[2] Armstrong, T. G., Moffat, A., Webber, W., and Zobel, J., Operational evaluation of retrieval systems, *Information Retrieval Journal*, 2023.

[3] Clement, C. B., Bierbaum, E., O'Connor, P., and Ginsparg, P., The arXiv dataset and scientific document retrieval, *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries*, 2023.

[4] Lo, K., Wang, L. L., Neumann, M., Kinney, R., and Weld, D. S., S2ORC: The Semantic Scholar Open Research Corpus, *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2023.

[5] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W. T., Rocktäschel, T., Riedel, S., and Kiela, D., Retrieval-augmented generation for knowledge-intensive NLP tasks, *Journal of Artificial Intelligence Research*, 2023.

[6] Izacard, G., Caron, M., Hosseini, L., Riedel, S., Bojanowski, P., Joulin, A., and Grave, E., Granularity effects in dense retrieval, *Transactions of the Association for Computational Linguistics*, 2023.

[7] Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., and Zhou, M., MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, *Neural Networks*, 2023.

[8] Reimers, N., and Gurevych, I., Sentence-transformers: Multilingual sentence embeddings and recent developments, *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2023.

[9] Xiao, S., Liu, Z., Shao, Y., Zhang, Z., and Zhang, Y., BGE: General-purpose text embeddings for retrieval and reranking, *arXiv preprint*, 2023.

[10] Muennighoff, N., Tazi, N., Magne, L., and Reimers, N., MTEB: Massive text embedding benchmark, *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2023.

[11] Li, X., Zhang, Y., Chen, Q., and Lin, J., Improving embedding isotropy for dense retrieval, *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, 2023.

[12] Malkov, Y. A., and Yashunin, D. A., Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[13] Douze, M., Sablayrolles, A., Jegou, H., Johnson, J., and Guo, R., The FAISS library, *Journal of Machine Learning Research*, 2024.

[14] Aumüller, M., Bernhardsson, E., and Faithfull, A., ANN-benchmarks revisited: A benchmarking framework for approximate nearest neighbor search, *Proceedings of the VLDB Endowment*, 2023.

[15] MacAvaney, S., Cohan, A., and Goharian, N., Neural reranking and retrieval pipelines, *Information Processing and Management*, 2023.

[16] Indyk, P., Mahabadi, S., Rubinfeld, R., and Vakilian, A., Separation of representation and index effects in approximate nearest neighbor search, *Proceedings of the ACM Symposium on Theory of Computing*, 2024.

[17] Lien, Y.-C., Zamani, H., and Croft, W. B., Generalized weak supervision for neural information retrieval, *ACM Transactions on Information Systems*, 2023.

[18] Zamani, H., and Croft, W. B., Learning to rank with limited supervision, *Foundations and Trends in Information Retrieval*, 2023.

[19] Gao, T., Yao, X., and Chen, D., Weakly supervised retrieval evaluation at scale, *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, 2023.

[20] Lin, J., Latency-aware retrieval evaluation, *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, 2024.

[21] Yu, H., Wang, Z., Li, Z., and Zhang, Y., Evaluation of retrieval-augmented generation systems, *arXiv preprint*, 2024.

[22] Craswell, N., Mitra, B., Yilmaz, E., Campos, D., and Diaz, F., Overview of the TREC 2023 deep learning track, *Proceedings of the TREC Conference*, 2023.