MONOSEMANTIC RETRIEVAL

Anonymous authors
Paper under double-blind review

ABSTRACT

A retrieval system fundamentally pairs a representation method with a similarity metric that compares the representations to retrieve relevant documents. Modern retrieval is increasingly converging to using a dense vector from Large Language Model embeddings, with cosine similarity as the metric. However, polysemanticity in LLMs is a representational constraint for such systems, leaving the property that makes a document the right answer superpositioned with other features that may not be relevant to the query. At the same time, cosine similarity in high dimensions of the embeddings dilutes the weight of components that decide nuanced relevance during retrieval. Interpretability research in LLMs has emphasized methods which recover monosemantic features from polysemantic representations. We hypothesize that such methods can solve the foundational issues in using LLM based embeddings for retreival. We introduce Monosemantic Retrieval (MR), where we learn per-feature retrieval weights over over monosemantic representations from Sparse Autoencoders and use them to index documents for retrieval. We score retrieval over the few sparse features a query and document share in place of cosine similarity, thus overcoming the "curse of dimensionality". Across 66 benchmarks spanning eleven domains, we show that this sits on the quality-versus-compute Pareto frontier of retrieval, matching the performance of state-of-the-art trained dense retrieval, with $28\times$ fewer query-time operations at the median corpus, a margin that grows with collection size, and 43% of its index bytes, while improving on existing lexical and learned-sparse retrieval methods. Leveraging the interpretable capabilities of these representations, we show that MR unlocks inference time steering capabilities that dense embedders lack. We simulate a real-world deployment over a mixed multi-domain corpus and show that MR unlocks inference-time steering in retrieval systems, which improves performance over state-of-the-art dense retrievers with a relative 20.9% improvement, while retaining the frontier on both index size and inference compute.

1 INTRODUCTION

Retrieval systems have always been a pair of representation with a scoring rule. Lexical retrieval used term sets scored by exact-match overlap (Gerard Salton & Yang, 1975; Robertson & Zaragoza, 2009), learned-sparse models kept the overlap but let training choose the vocabulary weights (Thibault Formal & Clinchant, 2022), and dense retrieval models replaced both representation and scoring at once, representing every text as one vector and scoring by cosine similarity (Karpukhin et al., 2020; Nils Reimers, 2019). With frozen embedders built on large language models (Yanzhao Zhang, 2025; Wang et al., 2024; Neelakantan et al., 2022), this dense embedding approach has become the default across research and deployment, inheriting the models' internet-scale knowledge, with the field's effort going into training the vector better (Gao et al., 2021; Gautier Izacard, 2022; Hongjin Su, 2023; Orion Weller, 2025).

The represenation and scoring for such systems are both bottlenecked by fundamental problems. On the representation side, the vector is polysemantic. Superposition lets a model store far more concepts than it has dimensions by letting them share coordinates (Elhage et al., 2022), which is the wrong property for an index because the feature that makes a document the right answer shares coordinates with every other concept that feature expresses. On the scoring side, cosine similarity concentrates in high dimensions (Beyer & Shaft, 1999; Charu C. Aggarwal & Keim, 2001; Reimers & Gurevych, 2021), the anisotropy of learned embedding spaces compresses the margins further

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

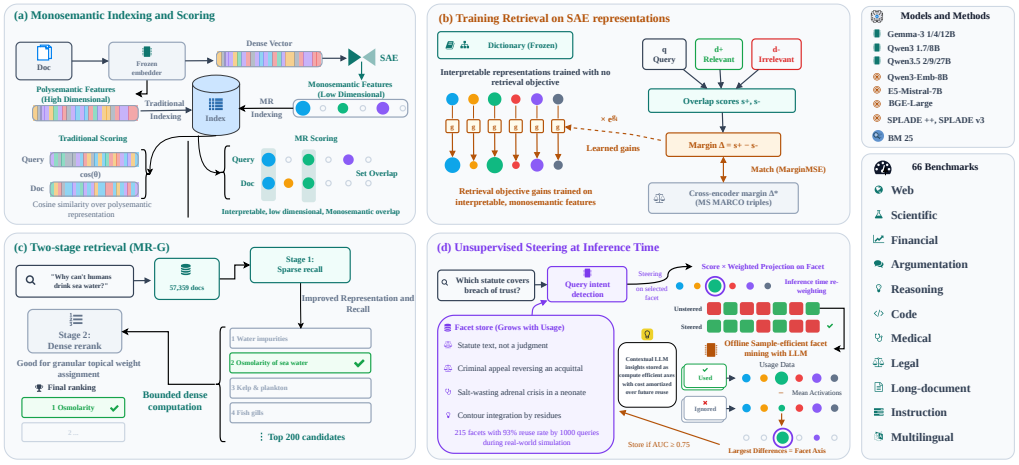

Figure 1: **MR overview.** **(a)** A frozen model's embedding entangles many concepts in one dense vector. An SAE decomposes it into named monosemantic features, and retrieval becomes weighted-Jaccard overlap between feature sets instead of cosine between opaque vectors. **(b)** The dictionary stays frozen and only a per-feature gain and bias are trained contrastively. **(c)** Two-stage retrieval on a worked biology example, where sparse feature overlap recalls the gold document that shares none of the query's words and the dense rerank then runs on 200 candidates instead of the whole collection. **(d)** Unsupervised Steering enabled at inference time using MR with no re-indexing.

(Ethayarajh, 2019; Jun Gao & Liu, 2019), and single-vector scoring has recently been argued to be limited in principle (Weller & Lee, 2025). Interpretability work finds that a concept like negation or refusal often lives along one specific direction of the latent space, expressed through a handful of active dimensions (Kiho Park & Veitch, 2024; Arditi et al., 2024; Marks & Tegmark, 2024), and a dot product over thousands of coordinates weighs such an axis at a vanishing fraction of the score no matter how decisive it is for the query, which explains why dense retrievers fail when such distinctions are highly salient (Weller & Van Durme, 2024; Weller & Soldaini, 2025).

Seen from the perspective of a retrieval system, these flaws appear as two kinds of failure. A recall error is when the right document never enters the candidate list, which nothing downstream can repair, and a ranking error is when it enters but sits too low. Both trace back to the representation. Either the deciding property is not separably encoded at all, so no scoring rule can act on it, or it is encoded but the system cannot know which facet of relevance the query intends, so the wrong properties decide the score. Rerankers (Nogueira & Cho, 2019) can only reorder what the first stage recalls, so both causes bound every system from its first step, and both follow from scoring entangled vectors by cosine similarity. A representation that separates the deciding properties improves recall on its own, and knowing which of them the query intends provides additional gains.

We propose leveraging methods from interpretability research as a strategy to overcome these bottlenecks in retrieval. Sparse autoencoders recover monosemantic features from polysemantic activations (Cunningham et al., 2024; Templeton et al., 2026; Gao & Wu, 2025), and applied to embeddings they have served to diagnose what a dense vector encodes (§2). To date, however, they have not been used to index, query and retrieve documents at the same time. Doing so helps retrieval in two ways. The representation becomes more precise, because key properties get their own interpretable coordinates instead of sharing them. The scoring rule improves as well, because sparse feature sets allow set-overlap similarity (Tversky, 1977; Broder, 1997) in place of cosine. Overlap is computed over the few features a text actually expresses, so an axis that dense scoring dilutes across thousands of coordinates becomes one or two set members that weigh in the score exactly as strongly as they fire, and the curse of dimensionality is sidestepped by scoring in the low-dimensional space of active features rather than the full embedding.

Sharing a substrate with interpretability unlocks one further capability. Since decisive relevant features become a handful of interpretable coordinates, they can be applied at query time to steer what is recalled and how candidates rank with no re-indexing. The formulation therefore addresses the

core retrieval flaws (representation through monosemantic coordinates, intent through interpretable steering, recall through both, and ranking through cheap reranking over the sparse recall stage).

In this work, we use SAE representations from dense LLMs for retrieval, instead of using the dense vectors directly (Figure 1). MR encodes each document's frozen embedding into 128 active features from a 32,768-feature SAE dictionary, retrieves by IDF-weighted Jaccard overlap over an inverted index, and optimizes retrieval while retaining interpretability by training per-feature gains over monosemantic representations during training (§3). This leads to two key contributions:

- **Pareto frontier performance in retrieval quality against indexing and compute cost across 66 retrieval benchmarks from 10 suites**. Sparse feature-overlap recall fixes representation gaps in dense models while cutting the per-query arithmetic by $28\times$ at the median corpus of 31k documents, a saving that grows with corpus size to $55\times$ on our largest corpus of 523k documents, measured on the built posting lists, and cutting the index to 43% of the trained retriever's bytes (§5).
- **An inference-time interpretable steering capability unlock during retrieval, which dense embedders lack**. We simulate a real-world retrieval over crawled data by combining nine confusable collections and two noise collections into one index, and show that Monosemantic Representation (MR) discovers steerable features based on query intent, improving on state-of-the-art dense trained retrievers by 20.9% relative nDCG@10, where the identical loop over the dense retriever does not improve at all (§6).

2 Related work

Lexical and learned-sparse retrieval. BM25 (Robertson & Zaragoza, 2009) and the learned-sparse SPLADE family (Formal & Clinchant, 2021; Thibault Formal & Clinchant, 2022; Carlos Lassance & Clinchant, 2024) score by overlap over interpretable dimensions, but those dimensions are vocabulary terms (Luyu Gao & Callan, 2021; Lin & Ma, 2021), so their ceiling is the vocabulary's ability to name what a document is about. Our index uses the same inverted index and overlap scoring as these systems, but each dimension is now a monosemantic feature from the model itself rather than a vocabulary term, so the matching happens over concepts instead of exact words.

Dense retrieval and its compression. Significant effort has been spent on optimizing the retrieval-quality against index-cost frontier of dense models. Trained embedders (Karpukhin et al., 2020; Yanzhao Zhang, 2025; Wang et al., 2024) push the quality and are our baseline. Quantization and approximate search (Douze et al., 2024; Jégou et al., 2011; Malkov & Yashunin, 2018) optimize the cost, and so we compare to quantized versions as well. Late-interaction systems (Khattab & Zaharia, 2020; Keshav Santhanam & Zaharia, 2022) spend more index for more quality on the opposite end. We differ by changing the representation itself, moving the frontier rather than trading along it.

Sparse autoencoders, and SAEs for retrieval. SAEs recover monosemantic features from activations (Cunningham et al., 2024; Gao & Wu, 2025; Templeton et al., 2026), with pretrained families now covering many open backbones (Lieberum & colleagues, 2024; He et al., 2024; Deng et al., 2026). Applied to embeddings they have served as analysis tools, diagnosing what a dense vector encodes (O'Neill & Wu, 2024; Lee et al., 2025; Jiang & Nanda, 2025), and a small concurrent line brings them toward retrieval itself. SPLARE (Formal et al., 2026) trains the autoencoder itself against the retrieval objective, so the dictionary it ends with is a new learned representation, while we keep the pretrained dictionary frozen and train only a per-feature gain and bias on top of it, which is what keeps every feature's pretraining identity available as a steering interface. Clavie & Kato (2026) extract BM25-ready sparse terms from dense retrievers, so their scoring runs through the lexical pipeline rather than over the features themselves, while we score by weighted overlap on the feature codes directly. However, neither of these works evaluates the sparse representation as a complete retrieval system across diverse domains, and neither of them uses it to steer retrieval.

Steering and controllable retrieval. Steering model behavior along learned directions is an established interpretability tool (Rimsky et al., 2024; Andy Zou et al., 2023; Arditi et al., 2024; Zhengxuan Wu & Potts, 2025). We bring the same idea to retrieval, where the object being steered is a

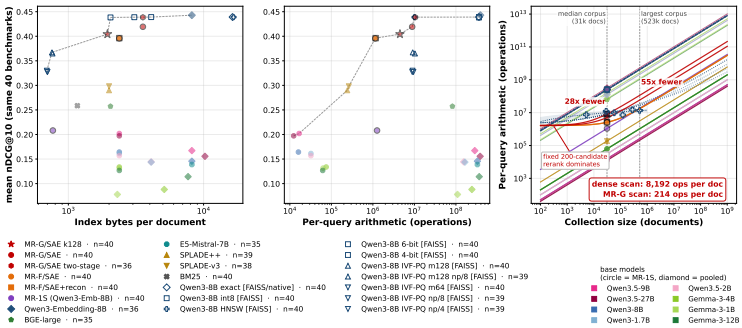

Figure 2: Quality against measured cost on the same 40 benchmarks. Mean nDCG@10 against index bytes per document and per-query arithmetic, then arithmetic against collection size. In the third panel, lines extend each system's per-document cost across index size, markers indicate values measured the median corpus, and the hollow markers with a dotted fit indicate the graph index.

ranked list and the intervention is multiplicative and document-sided rather than an additive activation edit. Instruction-following retrievers (Hongjin Su, 2023; Orion Weller, 2025) pursue related control through training. We show that specifying the intent along a monosemantic feature axis steers retrieval past the trained retriever with no retraining required.

3 MONOSEMANTIC RETRIEVAL (MR)

In MR, we take a base large language model or a dense embedding model and place a pretrained sparse autoencoder over its representations, so that every document is indexed by the monosemantic features it activates rather than by an entangled dense vector. Figure 1 shows the full pipeline at a high level, and we break it down step by step below.

We evaluate multiple configurations to separate where the monosemantic representations help. We start with base LLMs that were never trained for retrieval and use their pretrained SAEs, comparing monosemantic retrieval against the same models' dense pooled embeddings. This gives us a baseline for how much monosemantic representations help when the geometry of the base model itself has not been tuned for retrieval at all. We then repeat the construction on a trained retriever, to measure how much benefit remains when the dense embeddings have been optimized for retrieval-specific nuances. Finally, we combine the two in a two-stage setup, where the sparse index runs a compute-efficient first stage and the dense embeddings rerank the top candidates.

Sparse feature indexing. In the first step, we pass each document through the frozen base model, and the resulting representation is encoded by a pretrained sparse autoencoder, keeping the k highest feature activations from its dictionary. At indexing time, the document is represented by that sparse set of (feature id, activation) pairs and stored in an inverted index, analogous to traditional term indices (Christopher D. Manning & Schutze, 2008).

Scoring by overlap. Given the sparse feature sets q of a query and d of a document with activations $q_i, d_i \geq 0$, retrieval scores are the IDF-weighted Jaccard overlap

$$\mathrm{sim}(q, d) \;=\; \frac{\sum_i w_i \, \min(q_i, d_i)}{\sum_i w_i \, \max(q_i, d_i)}, \tag{1}$$

where w_i is the inverse document frequency of feature i over the corpus. Two texts are similar exactly when they agree on which features they express and how strongly they do so.

Retrieval training without compromising interpretability. The frozen SAE, used as is, was trained on an objective different from retrieval. Its training optimizes sparsity and reconstruction, which is what makes the features interpretable, but nothing in that objective makes the features activate optimally for retrieval. We want to retain the interpretability that comes from the original objective while still optimizing the features for retrieval, so we train a per-feature gain and bias

over the frozen dictionary (MR-G). The SAE dictionary is kept frozen to retain interpretability and monosemanticity, and for each feature in the SAE, we learn one scale and threshold with the retrieval objective. These learned features determine how strongly each feature counts toward a match and how easily it activates. We use the same training recipe as SPLADE++ (Formal & Clinchant, 2021), where the supervision comes from MS MARCO passage-ranking triples, a query with a relevant passage and a mined hard negative, scored by a cross-encoder that reads the query and passage together. This gives us supervision signal parity with the learned-sparse state of the art, so any difference against it comes from the representation and the scoring. We use the SPLADE recipe exactly on the same base model (MR-F), for an identical ablation against learned-sparse baselines (Formal et al., 2026; Thibault Formal & Clinchant, 2022). The LLM backbone and SAE directions stay frozen to ensure interpretability and monosemanticity.

Two-stage retrieval. What we find across these configurations is a natural division of labor. The dense embeddings, especially from the trained retriever, are particularly good at assigning granular topical weight to documents, while the monosemantic representations mainly help with the representation and recall issues during retrieval. We thus design a two-stage retrieval. The first stage runs over the sparse index of the trained head, which is compute-efficient because overlap scoring over posting lists costs a small fraction of a dense scan, and the saved compute lets us rerank the top 200 candidates densely with the base dense model (Nogueira & Cho, 2019), bounding the dense computation to 200 documents per query regardless of corpus size. In practice this cuts the per-query arithmetic, counting the sparse scan and the rerank together, by $28\times$ at our median corpus of 31k documents and by $55\times$ on our largest corpus of 523k documents, since a dense scan grows as N times the embedding width while our scan grows as N times the postings a query touches, a median of 214 operations per document measured on the built index (Appendix L). This is also the configuration steering operates in (§6).

We run this formulation over base LLMs from the Gemma and Qwen families with their pre-trained SAEs from the Gemma-Scope and Qwen-Scope releases (Lieberum & colleagues, 2024; Deng et al., 2026), nine base models in all (Appendix A). Our best performing configuration places a 32,768-feature dictionary with $k{=}128$ active features over the state-of-the-art embedder Qwen3-Embedding-8B (Yanzhao Zhang, 2025), which we use as the main configuration for our results.

4 Empirical overview

Benchmarks. We evaluate MR on 66 retrieval benchmarks from 10 suites. From BEIR (Thakur & Gurevych, 2021) and the MTEB retrieval tracks (Niklas Muennighoff & Reimers, 2023; Enevoldsen et al., 2025) come the web, scientific, financial and argumentation corpora. BRIGHT (Su et al., 2025) contributes reasoning-intensive retrieval, CoIR (Li et al., 2025; Husain et al., 2019; Huang et al., 2021) code, R2MED (Liu & colleagues, 2025) and TREC-PM (Roberts et al., 2016) medical, AILA (Bhattacharya & Majumder, 2019) legal, LongEmbed (Zhu & Li, 2024) long-document, FollowIR (Weller & Soldaini, 2025) instruction-following, and MIRACL (Zhang & colleagues, 2023) multilingual retrieval. Very large query sets are sampled to 2,000 queries under a pinned seed (as in the sampled MTEB variants (Enevoldsen et al., 2025)), with every system scored on the same sample, so within-benchmark comparisons are exact.

Baselines. We compare against BM25 (Robertson & Zaragoza, 2009), the learned-sparse retrievers SPLADE++ and SPLADE-v3 (Thibault Formal & Clinchant, 2022; Carlos Lassance & Clinchant, 2024), the dense retrievers BGE-large (Shitao Xiao, 2024) and E5-Mistral (Wang et al., 2024), and the trained retriever Qwen3-Embedding-8B, the base dense model MR builds on and the reference for every claim. We additionally quantize the base dense model's own index through int8, 6-bit, 4-bit, HNSW and IVF-PQ configurations (Douze et al., 2024; Malkov & Yashunin, 2018; Jégou et al., 2011), so the dense side contributes a quality–cost curve rather than a single point. Where available, we reuse baselines from their published checkpoints, and we run our own experiments with the identical evaluation harness for an apples-to-apples comparison.

Evaluation. The principal metric is nDCG@10 (native metrics elsewhere). Uncertainty is query-level bootstrap 95% confidence intervals (1,000 resamples), and head-to-head claims use paired per-query comparisons on the two systems' shared benchmarks.

5 RETRIEVAL RESULTS

Analysis across benchmarks. MR-G improves on SPLADE++ on 48 of 56 shared benchmarks (mean nDCG@10 0.433 against 0.307), on SPLADE-v3 on 46 of 55, on BM25 on 51 of 60 (0.462 against 0.313), and on BGE-large and E5-Mistral on 48 and 51 of 51. Since part of the margin over SPLADE++ comes from the backbone, MR-F serves as the matched control, applying the SPLADE++ recipe unchanged to our backbone and retraining the full dictionary. MR-G, which freezes the dictionary and trains only per-feature gains and biases, still improves on it (0.419 against 0.396 on 40 shared benchmarks), so at fixed backbone and supervision, keeping the monosemantic features adds quality while preserving interpretability and steerability. Against the trained retriever on 53 shared benchmarks, the sparse index alone reaches 0.445 against 0.473 from 43% of the index bytes, and a dense rerank of its top 200 candidates closes the gap to within 0.005 at $28\times$ to $55\times$ less query-time arithmetic, with the index at 68% of the dense bytes when a 4-bit copy is stored for the rerank. We conduct two ablations to separate what the dictionary and the training each contribute (Appendix I). A random basis in place of the SAE dictionary, at the same sparsity, roughly halves quality (0.165 against 0.319 over 28 paired substrate–layer configurations, winning none), so the monosemanticity of the features themselves matter, consistent with embedding-side SAE analyses (O'Neill & Wu, 2024; Lee et al., 2025).On the main configuration, removing the training entirely drops MR-G from 0.460 to 0.282 on 59 shared benchmarks, and training only the gains with the biases frozen recovers just 0.346, so both how strongly and how easily each feature fires matter. Adding a frozen-decoder reconstruction loss to this training leaves quality unchanged, so the codes can be held faithful to the model's own representation at no retrieval cost. Finally, replacing weighted Jaccard with Tversky or a sparse inner product shifts means only slightly: as §1 predicts, all three score over the few hundred features a query and document share, thus sidestepping the curse of dimensionality. On ten held-out MTEB retrieval tasks never used for tuning, the frozen recipe's two-stage configuration matches or improves on the dense scan on all ten (Appendix D).

The quality–cost frontier. MR-G has the **biggest impact on inference-time compute**. A dense retriever scores every document with a full 8,192-operation dot product, and quantization does not change this, since it cheapens each multiply-accumulate without reducing their count. An MR postings scan touches only the posting lists of the features a query activates at a median of 214 operations per document on the built index. In MR-G, dense scoring is confined to reranking a fixed 200 candidates, whose cost is the same at any collection size, so only the sparse scan grows with the collection. Thus, MR-G performs $28\times$ fewer operations than the dense retriever at the median corpus and $55\times$ fewer at the largest, without compromising quality. **MR-G stays on the frontier for index size as well**, with postings at 43% of the fp16 dense index's bytes. Quantization of the dense retriever reaches 25% at 4 bits with quality unchanged, but these savings hide practical constraints. Their calibration is fitted to the indexed data, so when the collection drifts it must be refit and the index re-encoded. Their arithmetic savings depend on hardware support (4- and 6-bit quantizations have no native kernels for retrieval), and it does not shrink the scan, so at scale it must be paired with a candidate-pruning index. The the two common pruning methods have their own trade-offs. IVF-PQ keeps arithmetic comparable to MR-G's and stores 0.8 KB per document, but loses substantial quality (0.07 nDCG@10) and its codebooks are likewise trained on the indexed data. A graph index like HNSW matches reranked MR-G in quality and arithmetic at the median corpus, but keeps every vector and its adjacency lists in memory at $4.7\times$ the bytes per document of MR-G's postings, cannot delete or update documents in place, and loses recall as the collection grows unless efSearch is raised (which raises the query cost and degrades performance). MR-G's first stage holds up at scale instead, still recovering 97.6% to 99.9% of the dense top-10 at depth 200 on the five largest corpora, up to 523k documents. On the other hand, replacing MR-G's first stage with BM25 at the same depth recovers only 57.3% of the dense top-10. MR-G avoids these costs because it is independent of the indexed collection. Its dictionary, head and encoder are fixed before indexing, so new documents are encoded and inserted as postings with no re-indexing, and the scan runs on the same inverted-index infrastructure that serves lexical search from disk. MR-G is also the only system in the figure that is interpretable and steerable (§6). Appendix C gives the measurements behind each of these costs and extends the advantage to base LLMs never trained for retrieval, where the monosemantic index beats the model's own pooled dense embeddings with fewer bytes, far less arithmetic and no trainable parameters. The MR approach thus unlocks interpretability, steerability and an optimized compute margin. This margin is the frontier once index size, inference compute at query time and the training the systems require are considered together.

6 STEERING RETRIEVAL WITH FEATURE FACETS

The retrieval results establish MR as a competitive substrate, and we now show that it **unlocks the capability for inference-time steering learned from user feedback**.

Search and retrieval settings in the real world often scrape and index a massive collection of information (for e.g. web crawl or internal organization documents) with basic metadata. A common failure mode for such retrieval systems in the wild is recall: the systems retrieve semantically similar documents from a different domain which are not aligned with the intent of the query (for example, retrieving cases when query intent is for statutes) or from the right domain at the wrong level of nuance (Appendix K). As an example, take the query "Why can't humans drink sea water?". Dense retrievers return "Saltwater science" blogs and ocean-ecosystem passages, which are semantically and topically similar but do not answer the query, while the correct document explains how water above blood osmolarity dehydrates humans. If we could infer the intent of a query at retrieval time and compute which features of a document express relevance to it, we could steer retrieval along those features for more meaningful results.

This allows us to define what we call ***feature facets***. A feature facet is a property of documents that a group of monosemantic features consistently fires on, like a document "explaining a physiological mechanism of dehydration". Since every document is already stored as a set of interpretable features, a facet is nothing more than a small group of those features a category of documents shares, and it can be computed from the index itself. Once we know which facet a query wants, we can promote the candidates that express those features and suppress the ones that express a competing facet, without recomputing the index. These are capabilities which only an interpretable, monosemantic retrieval system can unlock.

Dense representations from polysemantic models cannot support such steering, and modern retrieval works around this by training representation along pre-defined axes (Weller & Van Durme, 2024; Weller & Soldaini, 2025; Orion Weller, 2025) or by engineering lexical matching rules (Robertson & Zaragoza, 2009; Dai & Callan, 2020; Formal & Clinchant, 2021), both of which require strong human priors and cannot adapt when an unseen intent arrives at inference time (Weller & Soldaini, 2025; Weller & Lee, 2025). Retrieval under these conditions is increasingly being handled by frontier LLM agents that navigate a knowledge store themselves (Yao et al., 2023; Asai et al., 2024; Gutiérrez et al., 2024; Jin et al., 2025; Zheng et al., 2025; OpenAI, 2025). These systems work well because a capable LLM is good at the two things this setting actually demands: recognising what a query is really asking for (query intent classification) and naming the property of a document that makes it the right kind of answer (document facet mining). The second part is expensive, since the model reads hundreds to thousands of candidates over several calls (for the sea-water query reading through the saltwater-science blogs until it finds the one explaining osmolarity), and this work is discarded once the answer is produced (Asai et al., 2024; Jin et al., 2025), so the next query with the same intent pays for it again. This represents two extremes to retrieval: an agent that gets contextual retrieval right at a token cost and latency that scale with traffic, or cheap dense models that inherit their representation's problems.

MR offers a balanced solution. We keep the LLM for query intent detection, which is comparatively cheap, and hand the intensive step to MR. The model mines each facet offline from usage feedback alone, reading a few dozen documents once (26 at the median) instead thousands per query, and defines it at a granularity that later queries with the same intent can reuse. MR converts the facet into a steerable axis over the index, so the reweighting across thousands of documents runs without the model, and the one-time mining cost is amortized over every later query that shares the intent.

6.1 REAL-WORLD RETRIEVAL SIMULATION

To test whether this allows meaningful inference-time steering, we simulate a real world setting. We collate diverse domains from multiple benchmarks into a single index to serve as a messy, real-world collection of scraped information. Such a knowledge base generally contains high level metadata, which is not enough to steer systems or filter data at the dynamic, adaptive level we described without large amount of curated engineering effort. Thus, all they contain instead are *implicit facets*: properties that genuinely separate its documents along query intents, but not registered as usable metadata or a field which the system can use. We assume the intent of a query is not known either, which is generally the case in such real-world retrieval systems, since it depends on user goals and

session context which the system does not observe (Broder, 2002; Beitzel et al., 2007; Cao et al., 2009). What a deployed system does observe is usage feedback (which returned documents a user selected), and this accumulates with traffic for free. So the question we put to the experiment is whether a system can start with an empty facet store, watch its own traffic, and grow the facets and the intent routing it needs on its own, such that a query arriving later with an intent it has seen before is steered correctly and efficiently.

Building the mixed index. In a mixed corpus, the failure that dominates is the recall error of §1: candidates of the right topic but the wrong kind crowd out the relevant documents, which no reranker can recover. The facets along which these errors run are implicit, so they cannot be enumerated ahead of time, and a benchmark on which inference-time steering can be tested has to be built to contain them. We built ours to be as messy as a real scrape. Its documents come from collections that are close enough in topic to confuse a retriever, such as statutes along with the judgments that apply them, theorem statements next to the problems that need them, and several kinds of clinical writing about the same conditions, so a query's candidate list contains an error signal of the right topic in the wrong kind of document. Many of its queries are reasoning-intensive, so the intent and the relevant document connect through a property which cannot be computed by semantic or lexical match (for e.g. retrieving a technique to prove a theorem or patient case to condition matching). An unrelated code corpus is added as noise, since a real index holds plenty of documents irrelevant to any query. The built index contains 431,090 documents across nine collections with 100 queries per collection. Appendix E details the index construction. Our hypothesis is that this is the setting where steering helps most in improving over the two-stage baseline and SoTA trained retrievers.

Query stream and intent routing. A stream of 978 queries arrives over the mixed index in batches of 50, and the facet store starts empty. Each query is read by a frontier-LLM together with the facet current store, and may return the intended set of facets for the query or abstain, in which case the query runs unsteered. To simulate unknown dynamic query intent, no information about which collection a query or document comes from is provided to the agent. We run two experiments on the router's output: steering by the single top ranked facet or by the composition of every relevant facet.

Facet mining from feedback. The only signal the system observes is which returned documents a user ended up using, simulated with the benchmark gold labels for that round and revealed only after the round is served. These gold labels stand in as proxy for the denoised relevance signal that deployed systems derive from raw interactions through click models and position-bias correction (Joachims et al., 2017; Chuklin et al., 2022). After each round a frontier-LLM miner is handed the used documents and the ignored ones as a contrastive set. It decides whether this set exposes a property worth steering on, names it at a granularity of its discretion, and picks its own positive and negative examples from the round's candidate pool. Whatever separates the two sides has to be recovered from this mined set alone. The named facet is then tied to monosemantic features through its examples. Writing $\bar{z}_{P,i}$ and $\bar{z}_{N,i}$ for the mean activation of feature i over the positive and the negative examples, the axis is $A_f = \mathrm{top}_{12}\,(\bar{z}_{P,i} - \bar{z}_{N,i})$ with loadings $w_i = \bar{z}_{P,i} - \bar{z}_{N,i}$, and a document's projection on it is $p_d = \sum_{i \in A_f} w_i\, z_{d,i}$. A facet enters the store only if this axis separates a held-out split of its own examples at an AUC of 0.75 or better (median 1.000 across the 215 admitted facets). (Appendix G details the full setting and the store round by round).

Inference-time steering. In future queries, the intent router can detect that a reusable facet can be used to steer retrieval. Each candidate's first-stage score is then multiplied by $\max(0,\, 1 + s\,\tilde{p}_d)$, where $\tilde{p}_d$ is the document's projection on the axis standardized over the index and s a fixed strength, so documents expressing the intended facet are promoted and documents expressing a competing one suppressed at inference time, without affecting the index. The strength and normalization of the steering is discovered through controlled experiments. Using collection information as proxy for query intent, we study steering in an isolated setting to inform these parameters (Appendix F). The normalised strength is set to 0.5 standard deviations for every facet. This normalization of each facet's projection by its median and spread over the index allows a fixed setting to behave comparably across facets with diverse purity, granularity, domains and class imbalance.

The facet store and reuse. Every admitted facet is cached as an axis for future steering. By the end of the loop, it contains 215 facets. 17 of them broadly name a collection genre (the coarsest

Table 1: Inference-time steering results (978 queries, 21 rounds, paired bootstrap 95% CIs).

	nDCG@10	vs. trained retriever	95% CI	p
Trained retriever (dense)	0.1796	—	—	—
Two-stage, unsteered	0.1802	+0.0006	—	—
MR steered, single facet	**0.2171**	**+0.0375**	$[0.0256, 0.0498]$	<0.0001
MR steered, composed	**0.2171**	**+0.0375**	$[0.0253, 0.0494]$	<0.0001
Dense-space steered, single facet	0.1783	-0.0013	$[-0.0156, 0.0129]$	0.57
Dense-space steered, composed	0.1716	-0.0080	$[-0.0234, 0.0075]$	0.85

granularity discovered in the initial rounds) and 198 subtype and topic properties ranging from the posture of a criminal appeal to a proof technique, that accumulate once the coarse separation is already stored. The model is still called per query for intent classification, which reads only the query and the facet names, while the costly step of reading candidate documents to mine what makes them relevant is paid once per facet and reused thereafter. Across the 1,729 facet assignments the router made, 93.0% reuse a facet already in the store at least two rounds earlier. Thus, those queries **no model reads a single candidate document, amortizing the steering computation to MR instead** (Appendix G details the reuse and abstention counts).

Results. Table 1 reports the steering results. The single-facet experiment performs at 0.2171 nDCG@10 against the trained retriever's 0.1796 on the same 978 queries, a gain of +0.0375 and a **20.9% relative improvement over SoTA**, with the composed experiment at the same 0.2171. The unsteered two-stage system performs at 0.1802, showing that all of the improvements comes from the facets the system found on its own. Since the online loop discovers facets and infers intent at the same time, a failure in either confounds the other, so on its own it cannot say why steering helps where it does. We therefore run controlled experiments (Appendix F) that hold one of these fixed at a time to systematically observe their behaviours. These results give an intuition for when steering helps. The relevant documents for a query have to share a discoverable facet along which they separate from the rest. There has to be a recall bottleneck along this facet, meaning the retriever's errors are of the wrong kind rather than off topic, so that a facet can tell them apart from the gold. Finally, there has to be headroom: those wrong-kind documents must actually rank above the gold, since demoting documents that already sit below it changes nothing (Appendix H).

Can a dense retriever do this? To investigate if meaningful steering can be achieved with dense polysemantic representations, we perform an ablation where we remove MR entirely. In this ablation, the dense retriever directly serves the query stream and its results are used for facet mining by the same LLM agent. A mined facet becomes the mean-difference direction of its examples in the embedding space (Rimsky et al., 2024; Nina Panickssery & Turner, 2024), with the reweighting of the cosine scores is calibrated identically to the original setting (Appendix I.1). As seen in Table 1, the dense steering ends within noise of the unsteered retriever, even though the dense run had as much signal to steer from as MR, if not more, since its result pools held relevant documents for 67% of queries against 62% for MR's first stage, and it ended with twice as many facet definitions. The issue stems from the steering process itself. In a polysemantic representation, the mean-difference direction encodes every attribute on which the mined positives and negatives differ, even if it is not relevant to the query intent. Thus, an irrelevant document that shares those attributes projects on the facet and is steered along with the right ones (Appendix I.1 works this through per collection). Inference-time steering therefore requires MR for two reasons. A dense retriever cannot promote a facet without promoting whatever co-varies with it and the LLM cannot compensate for this. Its intents and examples are equally sound in both runs. The fundamental error mode is in utilizing these insights in the representation where the facet is expressed. The monosemantic index is also cheaper, requiring about 5×10^5 operations compared to 3.5×10^9 operations for dense.

7 Conclusion

We introduce monosemantic retrieval as a new paradigm to resolve fundamental bottlenecks in the representation and scoring of retrieval systems. MR achieves the pareto frontier of retrieval, providing SoTA performance at a fraction of the index size and inference compute, an outperforming all lexical and learned-sparse baselines. Through a real-world simulation, we then demonstrate that MR unlocks inference-time steering that dense embedders lack, with a 20.9% relative improvement over SoTA retrievers while retaining the frontier on index size and inference compute. We see MR as a first step toward interpretable, steerable and self-improving retrieval systems, paving the way for a new paradigm in compute efficient retrieval.

ETHICS STATEMENT

This work uses only publicly released benchmark collections (BEIR, MTEB, BRIGHT, CoIR, R2MED, LongEmbed, FollowIR, MIRACL, AILA and TREC-PM) and publicly released models and dictionaries, under their respective licenses, and involves no human subjects, no personal data collection and no annotation by people. The mixed index of §6 is assembled from those public corpora, and the feedback signal in the simulation is the benchmarks' own relevance judgments, not user data. We have read and adhere to the ICLR Code of Ethics.

REPRODUCIBILITY STATEMENT

Implementation details for every configuration, including the dictionary training recipe and its data sources, the head training objective and hyperparameters, the measured sparsity of each arm, the scoring rules, the benchmark protocol with its pinned query sampling, the baseline settings and the cost-accounting conventions, are given in Appendix L. Per-benchmark results for every system appear in Appendix A, the shared-benchmark frontier and the measured operation counts in Appendix C, the mixed-index construction in Appendix E, the controlled steering experiments in Appendix F, the round-by-round facet store of the online loop in Appendix G, and the router and miner prompts verbatim in Appendix M. All benchmarks and base models, and the Gemma-Scope and Qwen-Scope dictionaries, are publicly available. The supplementary material contains the anonymized code used to run every experiment in the paper, with a README giving one command per stage, from dictionary training through the online loop and the operation-count measurements, under a single configurable output root. The dictionary we trained over Qwen3-Embedding-8B and the trained heads will be released with the camera-ready version.

AI USE STATEMENT

In this work, we used generative AI tools, specifically an LLM-based coding assistant, to implement parts of the methods, to clean and reformat datasets (the construction of the mixed index from public benchmark collections), to assemble details in parts of the appendix and to create and modify some scientific figures. We did not use generative AI tools to generate synthetic datasets, to develop theoretical models, to formulate or prove mathematical claims, or to support qualitative or thematic analysis of human data; translation is not applicable to this work. We have reviewed all AI-assisted work. AI-generated code was reviewed by the authors and its outputs were verified before reporting. We take responsibility for the final content of this work, including text, claims and artifacts produced with the aid of generative AI.

REFERENCES

Sarah Chen-James Campbell Phillip Guo Richard Ren Alexander Pan Xuwang Yin Mantas Mazeika Ann-Kathrin Dombrowski Andy Zou, Long Phan et al. Representation engineering: A top-down approach to ai transparency. arXiv preprint arXiv:2310.01405, 2023.

Andy Arditi, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. Refusal in language models is mediated by a single direction. Advances in Neural Information Processing Systems, 2024.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. 2024.

Steven M Beitzel, Eric C Jensen, David D Lewis, Abdur Chowdhury, and Ophir Frieder. Automatic classification of web queries using very large unlabeled query logs. *ACM Transactions on Information Systems (TOIS)*, 25(2):9–es, 2007.

Goldstein J. Ramakrishnan R. Beyer, K. and U Shaft. When is nearest neighbor meaningful? International Conference on Database Theory, 1999.

Ghosh K. Ghosh S.-Pal A. Mehta P. Bhattacharya A. Bhattacharya, P. and P Majumder. Fire 2019 aila track: Artificial intelligence for legal assistance. Proceedings of the 11th Forum for Information Retrieval Evaluation, 2019. URL https://doi.org/10.1145/3368567.3368587.

Andrei Broder. A taxonomy of web search. In *ACM Sigir forum*, volume 36, pp. 3–10. ACM New York, NY, USA, 2002.

Andrei Z. Broder. On the resemblance and containment of documents. Proceedings of the Compression and Complexity of Sequences, 1997.

Huanhuan Cao, Derek Hao Hu, Dou Shen, Daxin Jiang, Jian-Tao Sun, Enhong Chen, and Qiang Yang. Context-aware query classification. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pp. 3–10, 2009.

Thibault Formal Carlos Lassance, Herve Dejean and Stephane Clinchant. Splade-v3: New baselines for splade. arXiv preprint, 2024. URL https://arxiv.org/abs/2403.06789.

Alexander Hinneburg Charu C. Aggarwal and Daniel A. Keim. On the surprising behavior of distance metrics in high dimensional spaces. Database Theory — ICDT 2001, 2001.

Prabhakar Raghavan Christopher D. Manning and Hinrich Schutze. Introduction to information retrieval. Cambridge University Press, 2008.

Aleksandr Chuklin, Ilya Markov, and Maarten De Rijke. *Click models for web search*. Springer Nature, 2022.

Lee S. Shakir A. Clavie, B. and M. P Kato. Latent terms: Dense retrievers contain trivially extractable bm25-ready zipfian vocabularies, 2026.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. In *The Twelfth International Conference on Learning Representations*, 2024.

Zhuyun Dai and Jamie Callan. Context-aware term weighting for first stage passage retrieval (deepct). pp. 1533–1536, 2020.

Boyi Deng, Xu Wang, Yaoning Wang, Yu Wan, Yubo Ma, Baosong Yang, Haoran Wei, Jialong Tang, Huan Lin, Ruize Gao, et al. Qwen-scope: Turning sparse features into development tools for large language models. *arXiv preprint arXiv:2605.11887*, 2026.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The Faiss library. *arXiv preprint arXiv:2401.08281*, 2024.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022.

Kenneth Enevoldsen, Isaac Chung, Imene Kerboua, Márton Kardos, Ashwin Mathur, David Stap, Jay Gala, Wissam Siblini, Dominik Krzemiński, Genta Indra Winata, et al. Mmteb: Massive multilingual text embedding benchmark. 2025.

Joshua Engels, Eric J Michaud, Isaac Liao, Wes Gurnee, and Max Tegmark. Not all language model features are one-dimensionally linear. In *The Thirteenth International Conference on Learning Representations*, 2025.

Kawin Ethayarajh. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. 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.

Piwowarski B. Formal, T. and S Clinchant. Splade: Sparse lexical and expansion model for first stage ranking. Special Interest Group on Information Retrieval, 2021.

Thibault Formal, Maxime Louis, Hervé Déjean, and Stéphane Clinchant. Learning retrieval models with sparse autoencoders. *arXiv preprint arXiv:2603.13277*, 2026. URL https://arxiv.org/abs/2603.13277.

la Tour T. D.-Tillman H. Goh G. Troll R. Radford A. Sutskever I. Leike J. Gao, L. and J Wu. Scaling and evaluating sparse autoencoders. International Conference on Learning Representations, 2025.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6894–6910, 2021.

Lucas Hosseini-Sebastian Riedel Piotr Bojanowski Armand Joulin Edouard Grave Gautier Izacard, Mathilde Caron. Unsupervised dense information retrieval with contrastive learning. Transactions on Machine Learning Research, 2022.

Anita Wong Gerard Salton and Chung-Shu Yang. A vector space model for automatic indexing. Communications of the ACM, 1975.

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: neurobiologically inspired long-term memory for large language models. volume 37, 2024.

Zhengfu He, Wentao Shu, Xuyang Ge, Lingjie Chen, Junxuan Wang, Yunhua Zhou, Frances Liu, Qipeng Guo, Xuanjing Huang, Zuxuan Wu, et al. Llama Scope: Extracting millions of features from Llama-3.1-8B with sparse autoencoders. In *arXiv preprint arXiv:2410.20526*, 2024.

Jungo Kasai-Yizhong Wang Yushi Hu Mari Ostendorf Wen-tau Yih Noah A. Smith Luke Zettlemoyer Tao Yu Hongjin Su, Weijia Shi. One embedder, any task: Instruction-finetuned text embeddings. Findings of the Association for Computational Linguistics: ACL 2023, 2023.

Junjie Huang, Duyu Tang, Linjun Shou, Ming Gong, Ke Xu, Daxin Jiang, Ming Zhou, and Nan Duan. Cosqa: 20,000+ web queries for code search and question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2021.

Hamel Husain, Ho-Hsiang Wu, Tiferet Gazit, Miltiadis Allamanis, and Marc Brockschmidt. CodeSearchNet challenge: Evaluating the state of semantic code search. *arXiv preprint arXiv:1909.09436*, 2019.

Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011.

Sun X. Dunlap L.-Smith L. Jiang, N. and N Nanda. Interpretable embeddings with sparse autoencoders: A data analysis toolkit, 2025.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Conference on Language Modeling (COLM)*, 2025.

Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. Unbiased learning-to-rank with biased feedback. In *Proceedings of the tenth ACM international conference on web search and data mining*, pp. 781–789, 2017.

Xu Tan-Tao Qin Liwei Wang Jun Gao, Di He and Tie-Yan Liu. Representation degeneration problem in training natural language generation models. Proceedings of the Seventh International Conference on Learning Representations (ICLR), 2019.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. Conference on Empirical Methods in Natural Language Processing, 2020.

Jon Saad-Falcon Christopher Potts Keshav Santhanam, Omar Khattab and Matei Zaharia. Colbertv2: Effective and efficient retrieval via lightweight late interaction. Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), 2022.

O. Khattab and M Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over bert. Special Interest Group on Information Retrieval, 2020.

Yo Joong Choe Kiho Park and Victor Veitch. The linear representation hypothesis and the geometry of large language models. Proceedings of the 41st International Conference on Machine Learning, 2024.

Seongwan Lee, Sol Park, and Dongha Lee. Decoding dense embeddings: Sparse autoencoders for interpreting and discretizing dense retrieval. *arXiv preprint arXiv:2506.00041*, 2025.

Xiangyang Li, Kuicai Dong, Yi Quan Lee, Wei Xia, Hao Zhang, Xinyi Dai, Yasheng Wang, and Ruiming Tang. CoIR: A comprehensive benchmark for code information retrieval models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025. URL https://arxiv.org/abs/2407.02883. arXiv preprint arXiv:2407.02883 (2024); accepted to ACL 2025 Main.

Rajamanoharan S. Conmy A.-Smith L. Lieberum, T. and colleagues. Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2. BlackboxNLP Workshop, 2024.

Jimmy Lin and Xueguang Ma. A few brief notes on deepimpact, coil, and a conceptual framework for information retrieval techniques. arXiv preprint arXiv:2106.14807, 2021.

W. Liu and colleagues. R2med: A benchmark for reasoning-driven medical retrieval, 2025.

Zhuyun Dai Luyu Gao and Jamie Callan. Coil: Revisit exact lexical match in information retrieval with contextualized inverted list. Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), 2021.

Yury A. Malkov and Dmitry A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs (HNSW). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4):824–836, 2018.

Samuel Marks and Max Tegmark. The geometry of truth: Emergent linear structure in large language model representations of true/false datasets. In *Conference on Language Modeling (COLM)*, 2024.

Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, et al. Text and code embeddings by contrastive pre-training. *arXiv preprint arXiv:2201.10005*, 2022.

Loic Magne Niklas Muennighoff, Nouamane Tazi and Nils Reimers. Mteb: Massive text embedding benchmark. Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics (EACL), 2023.

Iryna Gurevych Nils Reimers. Sentence-bert: Sentence embeddings using siamese bert-networks. 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.

Julian Schulz-Meg Tong Evan Hubinger Nina Panickssery, Nick Gabrieli and Alex M Turner. Steering llama 2 via contrastive activation addition. arXiv preprint arXiv:2312.06681, 2024.

Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with bert. arXiv preprint arXiv:1901.04085, 2019.

Ye C. Iyer K. O'Neill, C. and J. F Wu. Disentangling dense embeddings with sparse autoencoders, 2024.

OpenAI. Introducing deep research. In *OpenAI Blog*, 2025. URL https://openai.com/index/introducing-deep-research/.

Dawn Lawrie Ashwin Paranjape Yuhao Zhang Jack Hessel Orion Weller, Benjamin Van Durme. Promptriever: Instruction-trained retrievers can be prompted like language models. Proceedings of the Thirteenth International Conference on Learning Representations (ICLR), 2025.

N. Reimers and I Gurevych. The curse of dense low-dimensional information retrieval for large index sizes. Association for Computational Linguistics, 2021.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Matt Turner. Steering llama 2 via contrastive activation addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15504–15522, 2024.

Kirk Roberts, Dina Demner-Fushman, Ellen M. Voorhees, and William R. Hersh. Overview of the trec 2016 clinical decision support track. In *Proceedings of the Twenty-Fifth Text REtrieval Conference (TREC)*, 2016.

S. Robertson and H Zaragoza. The probabilistic relevance framework: Bm25 and beyond. Foundations and Trends in Information Retrieval, 2009.

Peitian Zhang Niklas Muennighoff Defu Lian Jian-Yun Nie Shitao Xiao, Zheng Liu. C-pack: Packed resources for general chinese embeddings. Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2024.

Hongjin Su, Howard Yen, Mengzhou Xia, Weijia Shi, Niklas Muennighoff, Han-yu Wang, Haisu Liu, Quan Shi, Zachary S. Siegel, Michael Tang, et al. Bright: A realistic and challenging benchmark for reasoning-intensive retrieval. 2025.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, et al. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *arXiv preprint arXiv:2605.29358*, 2026.

Reimers N. Ruckle A. Srivastava A. Thakur, N. and I Gurevych. Beir: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. Neural Information Processing Systems Datasets and Benchmarks, 2021.

Benjamin Piwowarski Thibault Formal, Carlos Lassance and Stephane Clinchant. From distillation to hard negative sampling: Making sparse neural ir models more effective (splade++). Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2022.

Amos Tversky. Features of similarity. Psychological Review, 1977.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Improving text embeddings with large language models (e5-mistral). *arXiv preprint arXiv:2401.00368*, 2024.

Boratko M. Naim I. Weller, O. and J Lee. On the theoretical limitations of embedding-based retrieval. International Conference on Learning Representations, 2025.

Chang B. MacAvaney S. Lo K. Cohan A. Van Durme B.-Lawrie D. Weller, O. and L Soldaini. Followir: Evaluating and teaching information retrieval models to follow instructions. North American Chapter of the Association for Computational Linguistics, 2025.

Lawrie D. Weller, O. and B Van Durme. Nevir: Negation in neural information retrieval. Findings of the Association for Computational Linguistics, 2024.

Dingkun Long Xin Zhang Huan Lin Baosong Yang Pengjun Xie An Yang Dayiheng Liu Junyang Lin Fei Huang Jingren Zhou Yanzhao Zhang, Mingxin Li. Qwen3 embedding: Advancing text embedding and reranking through foundation models. arXiv preprint arXiv:2506.05176, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

Thakur N. Ogundepo O. Zhang, X. and colleagues. Miracl: A multilingual retrieval dataset covering eighteen diverse languages. Transactions of the Association for Computational Linguistics, 2023.

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. DeepResearcher: Scaling deep research via reinforcement learning in real-world environments. In *arXiv preprint arXiv:2504.03160*, 2025.

Atticus Geiger Zheng Wang Jing Huang Dan Jurafsky Christopher D Manning Zhengxuan Wu, Aryaman Arora and Christopher Potts. Axbench: Steering llms? even simple baselines outperform sparse autoencoders. Proceedings of the 42nd International Conference on Machine Learning, 2025.

Wang L. Yang N. Song Y. Wu W. Wei F. Zhu, D. and S. Li. Longembed: Extending embedding models for long context retrieval. Conference on Empirical Methods in Natural Language Processing, 2024.

A FULL PERFORMANCE TABLE

The table blocks below report nDCG@10 for every presented system on every benchmark. The first column of each block names the benchmark and the second gives its document count (ndocs). The header rows give, per system, the trainable parameters, the index bytes per document, and the dense multiply-accumulates per query as a function of collection size N and embedding dimension d. Deeper shading marks higher scores within a benchmark row. The sweep runs under a tier-based prioritization, where we first complete the benchmarks that give domain diversity at moderate computational cost and then fill the larger benchmarks for the comparisons that matter most, so some configurations carry fewer entries than others, and missing entries render grey.

Benchmark	**ndocs**	MR-G/SAE gains on frozen d	MR-G/SAE k128 native spars	MR-F/SAE full head from di	MR-F/SAE+recon + frozen-de	MR-1S (Qwen3-Emb-8B)	MR-1S (Gemma-3-4B base)	MR-1S (Qwen3.5-27B base)	MR-1S: Gemma-3-12B	MR-1S: Gemma-3-1B	MR-1S: Qwen3-1.7B	MR-1S: Qwen3-8B	MR-1S: Qwen3.5-2B	MR-1S: Qwen3.5-9B	MR-G/SAE-nh gains, bias AL
trainable params		65.5K	65.5K	134M	134M	0	0	0	0	0	0	0	0	0	32.8K
index bytes/doc		3558	1944	2381	2381	768	2381	2381	2381	2381	2381	2381	2381	2381	20070
dense MACs/query		0	0	0	0	0	0	0	0	0	0	0	0	0	0
legal/aila-statutes	82	0.760	0.739	0.708	0.704	0.202	0.202	0.199	0.199	0.197	0.196	0.187	0.171	0.175	0.624
mteb/FinanceBenchRetrieval	145	0.783	0.773	0.761	0.759	0.327	0.213	0.316	0.194	0.204	0.287	0.327	0.272	0.309	0.707
legal/consumer-contracts	154	0.812	0.801	0.774	0.776	0.562	0.462	0.562	0.332	0.457	0.511	0.497	0.466	0.495	0.695
legal/aila-casedocs	186	0.443	0.436	0.450	0.458	0.303	0.282	0.303	0.279	0.282	0.247	0.288	0.253	0.269	0.349
longembed/2wikimqa	300	0.920	0.906	0.829	0.833	0.528	0.528	0.514	0.388	0.420	0.378	0.474	0.389	0.493	0.850
legal/corporate-lobbying	319	0.946	0.941	0.941	0.941	0.883	0.766	0.883	0.722	0.740	0.827	0.814	0.806	0.850	0.883
longembed/narrativeqa	355	0.666	0.647	0.595	0.596	0.144	0.127	0.164	0.161	0.151	0.081	0.144	0.097	0.152	0.544
mteb/FinQARetrieval	380	0.582	0.557	0.501	0.498	0.331	0.155	0.244	0.220	0.134	0.298	0.310	0.291	0.331	0.469
longembed/needle	800	0.224	0.214	0.191	0.194	0.191	0.157	0.191	0.118	0.160	0.178	0.198	0.169	0.198	0.221
longembed/passkey	800	0.430	0.435	0.378	0.380	0.502	0.495	0.499	0.502	0.499	0.499	0.499	0.500	0.495	0.435
mteb/CorebC2CRetrieval	1 744	0.428	0.409	0.309	0.309	0.343	0.144	0.343	0.057	0.079	0.105	0.090	0.106	0.256	0.432
mteb/CorebT2CRetrieval	1 744	0.373	0.364	0.310	0.309	0.250	0.084	0.106	0.094	0.105	0.154	0.133	0.195	0.250	0.285
mteb/FeedbackQARetrieval	2 364	0.709	0.700	0.671	0.673	0.424	0.269	0.276	0.228	0.266	0.305	0.298	0.320	0.424	0.588
mteb/NanoClimateFeverRetrieval	3 408	0.386	0.373	0.372	0.386	0.234	0.158	0.162	0.129	0.186	0.150	0.154	0.144	0.234	0.312
mteb/OBLIQBenchMathRetrieval	3 508	0.115	0.111	0.110	0.111	0.132	0.084	0.132	0.089	0.076	0.076	0.096	0.099	0.122	0.040
beir/nfcorpus	3 633	0.399	0.389	0.359	0.358	0.297	0.146	0.297	0.027	0.133	0.160	0.179	0.190	0.262	0.370
mteb/FreshStackRetrieval	3 804	0.409	0.394	0.375	0.378	0.190	0.182	0.165	0.144	0.156	0.164	0.167	0.152	0.190	0.230
mteb/NanoFiQA2018Retrieval	4 598	0.698	0.665	0.639	0.646	0.202	0.097	0.122	0.126	0.119	0.124	0.121	0.109	0.202	0.546
mteb/NanoFEVERRetrieval	4 996	0.955	0.955	0.948	0.948	0.483	0.483	0.235	0.328	0.394	0.266	0.254	0.211	0.255	0.923
mteb/NanoMSMARCORetrieval	5 043	0.664	0.647	0.618	0.629	0.348	0.124	0.296	0.106	0.107	0.303	0.315	0.290	0.348	0.579
beir/scifact	5 183	0.768	0.760	0.748	0.752	0.634	0.469	0.634	0.512	0.451	0.481	0.553	0.503	0.590	0.732
bright/pony	7 894	0.010	0.013	0.016	0.017	0.138	0.051	0.107	0.086	0.032	0.114	0.102	0.124	0.138	0.000
beir/arguana	8 674	0.513	0.508	0.497	0.500	0.421	0.421	0.310	0.326	0.384	0.382	0.392	0.386	0.420	0.421
coir/apps	8 765	0.646	0.608	0.623	0.627	0.213	0.028	0.223	0.032	0.019	0.012	0.026	0.022	0.066	0.347
mteb/EuroPIRQRetrieval	10 000	0.914	0.888	0.883	0.870	0.912	0.571	0.894	0.620	0.588	0.907	0.908	0.912	0.880	0.867
r2med/iiyi-clinical	10 449	0.232	0.224	0.258	0.252	0.135	0.122	0.131	0.115	0.119	0.127	0.124	0.135	0.128	0.122
followir/core17	19 899	0.495	0.472	0.501	0.498	0.326	0.255	0.326	0.231	0.200	0.257	0.237	0.259	0.303	0.323
coir/cosqa	20 604	0.355	0.339	0.339	0.341	0.206	0.110	0.190	0.077	0.105	0.127	0.157	0.138	0.186	0.299
mteb/CQADupstackAndroidRetriev	22 998	0.553	0.528	0.522	0.525	0.230	0.016	0.170	0.205	0.032	0.198	0.188	0.194	0.230	0.441
bright/theoremqa_theorems	23 839	0.217	0.197	0.263	0.257	0.166	0.047	0.166	0.042	0.044	0.045	0.050	0.050	0.134	0.054
beir/scidocs	25 657	0.264	0.244	0.231	0.232	0.124	0.046	0.097	0.017	0.053	0.070	0.065	0.073	0.124	0.177
mteb/R2MEDPMCTreatmentRetrieva	28 809	0.416	0.408	0.404	0.413	0.092	0.011	0.048	0.058	0.023	0.067	0.086	0.056	0.092	0.241
r2med/pmc-treatment	28 809	0.415	0.407	0.405	0.413	0.166	0.063	0.166	0.072	0.062	0.067	0.086	0.056	0.107	0.241
followir/news21	30 921	0.499	0.484	0.499	0.493	0.354	0.354	0.268	0.097	0.315	0.305	0.283	0.260	0.326	0.347
mteb/CQADupstackProgrammersRet	32 176	0.493	0.467	0.466	0.466	0.154	0.003	0.137	0.008	0.008	0.112	0.108	0.106	0.154	0.355
r2med/medical-sciences	34 792	0.416	0.404	0.395	0.387	0.289	0.225	0.297	0.171	0.189	0.167	0.211	0.186	0.289	0.226
mteb/CQADupstackEnglishRetriev	40 221	0.548	0.527	0.519	0.521	0.157	0.007	0.138	0.016	0.012	0.143	0.143	0.135	0.157	0.399
r2med/bioinformatics	47 471	0.452	0.439	0.396	0.391	0.205	0.173	0.309	0.157	0.123	0.205	0.237	0.189	0.269	0.352
followir/robust04	47 492	0.504	0.486	0.435	0.438	0.350	0.285	0.318	0.240	0.245	0.219	0.267	0.255	0.350	0.347
bright/economics	50 220	0.157	0.157	0.155	0.161	0.164	0.112	0.092	0.089	0.105	0.106	0.091	0.124	0.164	0.013
bright/psychology	52 835	0.212	0.216	0.216	0.222	0.124	0.076	0.096	0.042	0.062	0.072	0.056	0.092	0.124	0.030
r2med/medqa-diag	56 194	0.187	0.177	0.168	0.164	0.026	0.026	0.015	0.005	0.017	0.015	0.012	0.017	0.028	0.098
bright/biology	57 359	0.199	0.183	0.160	0.163	0.228	0.147	0.190	0.102	0.141	0.110	0.133	0.164	0.228	0.062
beir/fiqa	57 638	0.567	0.536	0.519	0.518	0.081	0.052	0.019	0.007	0.027	0.064	0.053	0.081	0.034	0.355
mteb/R2MEDPMCClinicalRetrieval	60 406	0.398	0.371	0.324	0.317	0.048	–	–	–	–	0.038	0.000	0.030	0.048	0.313
r2med/pmc-clinical	60 406	0.398	0.371	0.324	0.317	0.193	0.164	0.151	0.117	0.157	0.193	0.124	0.106	0.105	0.313

Benchmark	ndocs	MR-G/SAE gains on frozen d	MR-G/SAE k128 native spars	MR-F/SAE full head from di	MR-F/SAE+recon + frozen-de	MR-1S (Qwen3-Emb-8B)	MR-1S (Gemma-3-4B base)	MR-1S (Qwen3.5-27B base)	MR-1S: Gemma-3-12B	MR-1S: Gemma-3-1B	MR-1S: Qwen3-1.7B	MR-1S: Qwen3-8B	MR-1S: Qwen3.5-2B	MR-1S: Qwen3.5-9B	MR-G/SAE-nb gains, bias AL
trainable params		65.5K	65.5K	134M	134M	0	0	0	0	0	0	0	0	0	32.8K
index bytes/doc		3558	1944	2381	2381	768	2381	2381	2381	2381	2381	2381	2381	2381	20070
dense MACs/query		0	0	0	0	0	0	0	0	0	0	0	0	0	0
bright/sustainable_living	60 792	0.155	0.140	0.127	0.127	0.046	0.025	–	–	0.002	0.032	0.035	0.025	0.046	0.024
r2med/medxpertqa	61 379	0.128	0.112	0.076	0.075	0.025	0.025	0.028	0.017	0.016	0.016	0.023	0.019	0.023	0.059
bright/robotics	61 961	0.144	0.149	0.143	0.150	0.087	0.087	0.081	0.054	0.083	0.054	0.061	0.057	0.066	0.017
mteb/OBLIQBenchTwitterRetrieva	72 122	0.047	0.035	0.020	0.022	0.002	0.001	0.002	–	–	0.002	0.000	0.001	0.001	0.010
coir/synthetic-text2sql	105 851	0.722	0.715	–	–	0.533	0.553	0.534	0.387	0.538	0.616	0.529	0.569	0.548	–
bright/stackoverflow	107 081	0.187	0.160	0.167	0.164	0.051	0.009	0.051	0.005	–	–	–	–	–	0.070
bright/earth_science	121 249	0.319	0.297	0.275	0.281	0.128	0.128	0.203	0.090	0.115	0.121	0.121	0.116	0.106	0.145
miracl/sw	131 924	0.665	0.648	0.605	0.606	0.128	0.051	–	0.012	0.049	0.124	0.128	0.010	0.068	0.430
beir/treccovid	171 332	–	–	–	–	0.884	0.020	–	–	–	0.001	–	–	–	–
mteb/QuoraRetrievalHardNegativ	177 163	0.861	0.862	0.853	0.854	0.124	0.124	–	–	–	–	–	–	–	0.776
bright/aops	188 002	0.029	0.024	0.028	0.032	0.007	0.007	–	–	–	–	–	–	–	0.028
bright/theoremqa_questions	188 002	0.195	0.179	0.178	0.175	0.091	0.091	–	–	0.073	–	–	–	–	0.108
coir/codesearchnet	280 310	0.896	0.887	0.882	0.884	0.468	–	–	–	–	–	–	–	0.483	0.428
miracl/bn	297 265	0.755	0.729	–	–	0.049	0.049	–	–	0.018	–	–	–	–	–
trec-pm/clinicaltrials-2019	306 238	0.509	0.463	–	–	–	–	–	–	–	–	–	–	–	–
beir/touche2020	382 545	0.301	0.291	–	–	0.017	–	–	–	–	–	0.017	–	0.013	–
bright/leetcode	413 932	0.170	0.156	0.168	0.168	–	–	–	–	–	–	–	–	–	0.099
miracl/te	518 079	0.823	0.819	0.768	0.767	–	–	–	–	–	–	–	–	–	0.687
beir/quora	522 931	0.875	0.869	–	–	0.761	–	–	–	0.772	–	–	–	–	–
· B/doc basis	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Benchmark	ndocs	Qwen3-Embedding-8B	Qwen3-8B exact [FAISS fp32]	Qwen3-8B int8 [FAISS]	Qwen3-8B 6-bit [FAISS]	Qwen3-8B 4-bit [FAISS]	Qwen3-8B HNSW [FAISS]	Qwen3-8B IVF-PQ m128 [FAISS]	Qwen3-8B IVF-PQ m128 np/8 [FAI	Qwen3-8B IVF-PQ m64 [FAISS]	Qwen3-8B IVF-PQ m64 np/4 [FAIS	Qwen3-8B IVF-PQ m64 np/8 [FAIS	BGE-large	E5-Mistral-7B	SPLADE++
trainable params		8B	8B	8B	8B	8B	8B	8B	8B	8B	8B	8B	335M	7B	110M
index bytes/doc		8192	16384	4096	3072	2048	16656	800	800	545	545	545	2048	8192	2000
dense MACs/query		$N\,d$	$N\,d$	$N\,d$	$N\,d$	$N\,d$	sub-lin.	sub-lin.	sub-lin.	sub-lin.	sub-lin.	sub-lin.	$N\,d$	$N\,d$	0
legal/aila-statutes	82	0.841	–	–	–	–	–	–	–	–	–	–	0.231	0.136	0.277
mteb/FinanceBenchRetrieval	145	0.846	–	–	–	–	–	–	–	–	–	–	0.594	0.173	0.657
legal/consumer-contracts	154	–	–	–	–	–	–	–	–	–	–	–	–	–	–
legal/aila-casedocs	186	0.460	–	–	–	–	–	–	–	–	–	–	0.260	0.084	0.325
longembed/2wikimqa	300	0.934	–	–	–	–	–	–	–	–	–	–	0.440	0.062	0.627
legal/corporate-lobbying	319	–	–	–	–	–	–	–	–	–	–	–	–	–	–
longembed/narrativeqa	355	0.700	–	–	–	–	–	–	–	–	–	–	0.219	0.034	0.295
mteb/FinQARetrieval	380	0.626	–	–	–	–	–	–	–	–	–	–	0.304	0.115	0.615
longembed/needle	800	0.231	–	–	–	–	–	–	–	–	–	–	0.210	0.030	0.227
longembed/passkey	800	0.443	–	–	–	–	–	–	–	–	–	–	0.388	0.189	0.265
mteb/CorebC2CRetrieval	1 744	0.452	–	–	–	–	–	–	–	–	–	–	0.130	0.007	0.059
mteb/CorebT2CRetrieval	1 744	0.415	–	–	–	–	–	–	–	–	–	–	0.158	0.025	0.203
mteb/FeedbackQARetrieval	2 364	0.744	–	–	–	–	–	–	–	–	–	–	0.501	0.288	0.655
mteb/NanoClimateFeverRetrieval	3 408	0.422	–	–	–	–	–	–	–	–	–	–	0.242	0.215	0.311
mteb/OBLIQBenchMathRetrieval	3 508	0.115	–	–	–	–	–	–	–	–	–	–	0.080	0.042	0.069
beir/nfcorpus	3 633	0.419	–	–	–	–	–	–	–	–	–	–	0.381	0.176	0.352
mteb/FreshStackRetrieval	3 804	0.445	–	–	–	–	–	–	–	–	–	–	0.253	0.027	0.220
mteb/NanoFiQA2018Retrieval	4 598	0.708	0.708	0.708	0.708	0.701	0.708	0.656	0.656	0.580	0.580	0.580	0.320	0.185	0.507
mteb/NanoFEVERRetrieval	4 996	0.962	0.962	0.962	0.962	0.962	0.962	0.904	0.912	0.911	0.916	0.916	0.835	0.712	0.928
mteb/NanoMSMARCORetrieval	5 043	0.702	0.702	0.697	0.702	0.698	0.702	0.608	0.608	0.572	0.572	0.572	0.455	0.388	0.685
beir/scifact	5 183	0.788	0.788	0.788	0.787	0.787	0.788	0.705	0.729	0.657	0.665	0.665	0.746	0.358	0.708
bright/pony	7 894	0.011	0.011	0.011	0.011	0.011	0.011	0.011	0.011	0.009	0.009	0.009	0.020	0.000	0.134
beir/arguana	8 674	0.530	0.530	0.530	0.530	0.529	0.530	0.525	0.525	0.509	0.509	0.509	0.645	0.255	0.508
coir/apps	8 765	0.782	0.783	0.783	0.782	0.784	0.782	0.626	0.638	0.565	0.570	0.570	0.076	0.003	0.086
mteb/EuroPIRQRetrieval	10 000	0.902	0.894	0.894	0.895	0.898	0.894	0.823	0.827	0.786	0.786	0.786	0.799	0.603	0.940
r2med/iiyi-clinical	10 449	0.282	0.282	0.283	0.285	0.282	0.281	0.239	0.239	0.229	0.229	0.229	0.147	0.018	0.124
followir/core17	19 899	–	0.420	0.420	0.421	0.425	0.420	0.377	0.377	0.324	0.324	0.324	–	–	0.346
coir/cosqa	20 604	0.402	0.454	0.454	0.451	0.469	0.481	0.456	0.456	0.427	0.427	0.427	0.344	0.016	0.185
mteb/CQADupstackAndroidRetriev	22 998	0.595	0.595	0.594	0.594	0.593	0.595	0.523	0.523	0.477	0.477	0.477	0.311	0.156	0.392
bright/theoremqa_theorems	23 839	0.216	0.216	0.216	0.212	0.210	0.212	0.179	0.179	0.163	0.163	0.163	0.069	0.110	0.061
beir/scidocs	25 657	0.327	0.286	0.286	0.287	0.285	0.286	0.264	0.264	0.251	0.251	0.251	0.226	0.041	0.158
mteb/R2MEDPMCTreatmentRetrieva	28 809	0.498	0.498	0.498	0.497	0.499	0.499	0.304	0.304	0.235	0.235	0.235	0.215	0.104	0.177
r2med/pmc-treatment	28 809	–	0.498	0.498	0.497	0.499	0.499	0.309	0.309	0.224	0.224	0.224	0.215	0.104	0.177
followir/news21	30 921	–	0.370	0.370	0.369	0.367	0.370	0.362	0.362	0.313	0.313	0.313	–	–	0.241
mteb/CQADupstackProgrammersRet	32 176	0.528	0.528	0.527	0.528	0.526	0.528	0.446	0.447	0.409	0.410	0.410	0.142	0.124	0.339
r2med/medical-sciences	34 792	0.422	0.422	0.422	0.422	0.421	0.420	0.342	0.342	0.295	0.295	0.295	0.277	0.152	0.312
mteb/CQADupstackEnglishRetriev	40 221	0.576	0.576	0.576	0.577	0.576	0.573	0.467	0.472	0.404	0.406	0.406	0.305	0.179	0.408
r2med/bioinformatics	47 471	0.497	0.497	0.497	0.498	0.503	0.497	0.403	0.403	0.350	0.350	0.350	0.270	0.147	0.230
followir/robust04	47 492	0.480	0.465	0.466	0.465	0.469	0.465	0.393	0.393	0.353	0.353	0.353	–	–	0.317

Benchmark	**ndocs**	Qwen3-Embedding-8B	Qwen3-8B exact [FAISS fp32]	Qwen3-8B int8 [FAISS]	Qwen3-8B 6-bit [FAISS]	Qwen3-8B 4-bit [FAISS]	Qwen3-8B HNSW [FAISS]	Qwen3-8B IVF-PQ m128 [FAISS]	Qwen3-8B IVF-PQ m128 np/8 [FAI	Qwen3-8B IVF-PQ m64 [FAISS]	Qwen3-8B IVF-PQ m64 np/4 [FAIS	Qwen3-8B IVF-PQ m64 np/8 [FAIS	BGE-large	E5-Mistral-7B	SPLADE++
trainable params		8B	8B	8B	8B	8B	8B	8B	8B	8B	8B	8B	335M	7B	110M
index bytes/doc		8192	16384	4096	3072	2048	16656	800	800	545	545	545	2048	8192	2000
dense MACs/query		$N\ d$	$N\ d$	$N\ d$	$N\ d$	$N\ d$	sub-lin.	sub-lin.	sub-lin.	sub-lin.	sub-lin.	sub-lin.	$N\ d$	$N\ d$	0
bright/economics	50 220	0.159	0.159	0.159	0.158	0.156	0.159	0.109	0.109	0.083	0.083	0.083	0.158	0.014	0.135
bright/psychology	52 835	0.238	0.238	0.238	0.235	0.234	0.239	0.190	0.190	0.159	0.159	0.159	0.153	0.022	0.134
r2med/medqa-diag	56 194	0.235	0.235	0.236	0.236	0.231	0.235	0.190	0.190	0.165	0.165	0.165	0.083	0.046	0.057
bright/biology	57 359	0.173	0.171	0.171	0.171	0.170	0.169	0.162	0.162	0.154	0.154	0.154	0.117	0.035	0.182
beir/fiqa	57 638	0.616	0.616	0.616	0.614	0.615	0.616	0.509	–	0.452	–	–	0.450	0.117	0.356
mteb/R2MEDPMCClinicalRetrieval	60 406	0.423	0.426	0.426	0.424	0.419	0.426	0.291	0.291	0.277	0.277	0.277	0.143	0.019	0.190
r2med/pmc-clinical	60 406	–	0.426	0.426	0.425	0.419	0.426	0.294	0.294	0.275	0.275	0.275	0.143	0.019	0.190
bright/sustainable_living	60 792	0.156	0.157	0.156	0.159	0.155	0.157	0.126	0.126	0.093	0.093	0.093	0.134	0.038	0.138
r2med/medxpertqa	61 379	0.178	0.178	0.178	0.178	0.174	0.178	0.113	0.113	0.096	0.096	0.096	0.041	0.040	0.028
bright/robotics	61 961	0.149	0.149	0.149	0.149	0.152	0.149	0.065	0.065	0.053	0.053	0.053	0.111	0.003	0.149
mteb/OBLIQBenchTwitterRetrieva	72 122	0.048	0.047	0.047	0.047	0.046	0.048	0.036	0.036	0.032	0.032	0.032	0.007	0.003	0.003
coir/synthetic-text2sql	105 851	0.601	0.849	0.849	0.850	0.849	0.849	0.674	–	0.564	–	–	0.505	0.127	–
bright/stackoverflow	107 081	0.178	0.170	0.169	0.169	0.169	0.165	0.155	0.155	0.133	0.133	0.133	0.109	0.006	0.137
bright/earth_science	121 249	0.329	0.334	0.334	0.333	0.330	0.334	0.267	0.267	0.236	0.236	0.236	0.227	0.002	0.252
miracl/sw	131 924	0.725	0.723	0.725	0.723	0.720	0.723	0.544	0.544	0.469	0.469	0.469	–	–	0.438
beir/treccovid	171 332	0.926	–	–	–	–	–	–	–	–	–	–	–	–	0.711
mteb/QuoraRetrievalHardNegativ	177 163	0.874	0.874	0.874	0.874	0.875	0.873	0.810	0.810	0.754	0.754	0.754	0.563	0.713	0.834
bright/aops	188 002	0.033	0.028	0.028	0.027	0.027	0.028	0.021	0.021	0.018	0.018	0.018	0.021	0.004	0.039
bright/theoremqa_questions	188 002	0.288	0.216	0.215	0.216	0.213	0.195	0.129	0.129	0.102	0.102	0.102	0.130	0.123	0.084
coir/codesearchnet	280 310	0.935	0.935	0.935	0.934	0.934	0.933	0.754	0.754	0.606	0.606	0.606	–	–	–
miracl/bn	297 265	–	0.783	0.783	0.782	0.781	0.783	0.672	0.672	0.596	0.596	0.596	–	–	–
trec-pm/clinicaltrials-2019	306 238	–	–	–	–	–	–	–	–	–	–	–	–	–	0.371
beir/touche2020	382 545	0.334	–	–	–	–	–	–	–	–	–	–	0.248	0.052	0.247
bright/leetcode	413 932	0.200	0.206	–	–	–	0.203	–	–	0.083	–	–	0.166	0.000	0.147
miracl/te	518 079	0.840	–	–	–	–	–	–	–	–	–	–	–	–	–
beir/quora	522 931	0.878	–	–	–	–	–	–	–	–	–	–	0.890	0.756	0.835
· B/doc basis	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Benchmark	**ndocs**	SPLADE-v3	BM25	Pooled: Gemma-3-12B	Pooled: Gemma-3-1B	Pooled: Gemma-3-4B	Pooled: Qwen3-1.7B	Pooled: Qwen3-8B	Pooled: Qwen3.5-27B	Pooled: Qwen3.5-2B	Pooled: Qwen3.5-9B
trainable params		110M	0	0	0	0	0	0	0	0	0
index bytes/doc		2000	1166	7680	2304	5120	4096	8192	10240	4096	8192
dense MACs/query		0	0	$N\ d$	$N\ d$	$N\ d$	$N\ d$	$N\ d$	$N\ d$	$N\ d$	$N\ d$
legal/aila-statutes	82	0.224	0.172	0.251	0.157	0.192	0.142	0.148	0.175	0.154	0.154
mteb/FinanceBenchRetrieval	145	0.686	0.259	0.174	0.171	0.191	0.134	0.152	0.268	0.140	0.178
legal/consumer-contracts	154	–	0.663	0.328	0.284	0.415	0.395	0.326	0.408	0.359	0.332
legal/aila-casedocs	186	0.312	0.220	0.146	0.135	0.128	0.255	0.235	0.278	0.265	0.239
longembed/2wikimqa	300	0.621	0.969	0.372	0.194	0.309	0.292	0.311	0.322	0.257	0.308
legal/corporate-lobbying	319	–	0.900	0.662	0.593	0.701	0.715	0.675	0.766	0.692	0.748
longembed/narrativeqa	355	0.292	0.708	0.183	0.076	0.072	0.054	0.084	0.092	0.082	0.123
mteb/FinQARetrieval	380	0.638	0.696	0.098	0.082	0.094	0.172	0.139	0.096	0.135	0.219
longembed/needle	800	0.235	0.434	0.101	0.100	0.124	0.080	0.071	0.105	0.085	0.107
longembed/passkey	800	0.266	0.494	0.070	0.086	0.120	0.110	0.051	0.122	0.058	0.110
mteb/CorebC2CRetrieval	1 744	0.086	0.211	0.116	0.041	0.218	0.080	0.036	0.371	0.089	0.273
mteb/CorebT2CRetrieval	1 744	0.158	0.157	0.139	0.032	0.092	0.026	0.045	0.040	0.060	0.087
mteb/FeedbackQARetrieval	2 364	0.676	0.480	0.260	0.132	0.332	0.280	0.257	0.067	0.231	0.297
mteb/NanoClimateFeverRetrieval	3 408	0.310	0.309	0.042	0.132	0.153	0.142	0.126	0.078	0.122	0.165
mteb/OBLIQBenchMathRetrieval	3 508	0.073	0.045	0.046	0.036	0.071	0.043	0.069	0.070	0.055	0.077
beir/nfcorpus	3 633	0.363	0.322	0.125	0.089	0.153	0.104	0.153	0.203	0.123	0.175
mteb/FreshStackRetrieval	3 804	0.210	0.236	0.040	0.091	0.057	0.096	0.105	0.018	0.137	0.120
mteb/NanoFiQA2018Retrieval	4 598	0.514	0.349	0.159	0.058	0.063	0.149	0.126	0.182	0.089	0.230
mteb/NanoFEVERRetrieval	4 996	0.913	0.836	0.126	0.227	0.284	0.239	0.218	0.187	0.300	0.248
mteb/NanoMSMARCORetrieval	5 043	0.700	0.472	0.061	0.123	0.082	0.339	0.329	0.194	0.314	0.272
beir/scifact	5 183	0.717	0.679	0.352	0.272	0.305	0.421	0.515	0.604	0.432	0.553
bright/pony	7 894	0.143	0.119	0.163	0.045	0.167	0.084	0.034	0.045	0.166	0.064
beir/arguana	8 674	0.489	0.397	0.310	0.237	0.343	0.412	0.450	0.240	0.473	0.482
coir/apps	8 765	–	0.043	0.043	0.003	0.032	0.014	0.031	0.051	0.016	0.018
mteb/EuroPIRQRetrieval	10 000	0.942	0.937	0.330	0.294	0.345	0.667	0.490	0.602	0.544	0.633
r2med/iiyi-clinical	10 449	0.129	0.106	0.023	0.020	0.033	0.071	0.069	0.061	0.069	0.056
followir/core17	19 899	0.328	0.229	–	–	–	–	–	–	0.157	–
coir/cosqa	20 604	0.217	0.149	0.099	0.066	0.094	0.106	0.133	0.187	0.100	0.145
mteb/CQADupstackAndroidRetriev	22 998	0.414	0.359	0.108	–	0.079	0.228	0.160	0.049	0.178	0.199
bright/theoremqa_theorems	23 839	0.055	0.021	0.074	0.013	0.024	0.021	0.032	0.151	0.034	0.079
beir/scidocs	25 657	0.156	0.149	0.065	0.033	0.039	0.058	0.072	0.080	0.058	0.100
mteb/R2MEDPMCTreatmentRetrieva	28 809	0.163	0.141	0.007	0.000	0.005	0.016	0.054	0.000	0.023	0.003
r2med/pmc-treatment	28 809	0.163	0.185	0.033	0.007	0.032	0.016	0.054	0.099	0.032	0.054
followir/news21	30 921	0.261	0.222	–	–	–	–	–	–	–	–

Benchmark	**ndocs**	SPLADE-v3	BM25	Pooled: Gemma-3-12B	Pooled: Gemma-3-1B	Pooled: Gemma-3-4B	Pooled: Qwen3-1.7B	Pooled: Qwen3-8B	Pooled: Qwen3.5-27B	Pooled: Qwen3.5-2B	Pooled: Qwen3.5-9B
trainable params		110M	0	0	0	0	0	0	0	0	0
index bytes/doc		2000	1166	7680	2304	5120	4096	8192	10240	4096	8192
dense MACs/query		0	0	$N\,d$	$N\,d$	$N\,d$	$N\,d$	$N\,d$	$N\,d$	$N\,d$	$N\,d$
mteb/CQADupstackProgrammersRet	32 176	0.339	0.268	0.078	–	0.065	0.140	0.102	0.116	0.093	0.115
r2med/medical-sciences	34 792	0.336	0.176	0.262	0.113	0.139	0.073	0.103	0.174	0.149	0.166
mteb/CQADupstackEnglishRetriev	40 221	0.424	0.334	0.072	0.076	0.076	0.171	0.145	0.125	0.163	0.148
r2med/bioinformatics	47 471	0.238	0.198	0.163	0.100	0.157	0.142	0.172	0.200	0.154	0.176
followir/robust04	47 492	0.294	0.192	0.249	0.134	0.028	0.093	0.113	0.158	0.142	0.174
bright/economics	50 220	0.157	0.165	0.067	0.035	0.086	0.102	0.113	0.119	0.097	0.155
bright/psychology	52 835	0.151	0.134	0.084	0.031	0.042	0.086	0.078	0.084	0.058	0.067
r2med/medqa-diag	56 194	0.063	0.017	0.015	0.005	0.013	0.009	0.019	0.022	0.014	0.028
bright/biology	57 359	0.210	0.182	0.204	0.053	0.096	0.092	0.162	0.161	0.136	0.171
beir/fiqa	57 638	0.380	0.236	0.078	0.025	0.039	0.083	0.095	–	0.079	0.101
mteb/R2MEDPMCClinicalRetrieval	60 406	0.181	0.192	0.034	0.030	0.034	–	–	0.112	–	–
r2med/pmc-clinical	60 406	0.181	0.209	0.063	0.034	0.057	–	0.107	0.112	0.086	0.127
bright/sustainable_living	60 792	0.150	0.133	0.096	–	0.031	0.069	0.085	–	0.051	0.112
r2med/medxpertqa	61 379	0.018	0.014	0.016	0.008	0.013	0.016	0.025	0.016	0.013	0.023
bright/robotics	61 961	0.158	0.109	0.037	0.027	0.016	0.038	0.037	0.079	0.038	0.044
mteb/OBLIQBenchTwitterRetrieva	72 122	0.004	0.002	0.011	0.005	0.004	0.003	–	–	0.007	–
coir/synthetic-text2sql	105 851	–	0.367	0.023	0.066	0.064	0.163	0.117	0.159	0.117	–
bright/stackoverflow	107 081	0.128	0.166	0.016	0.051	0.002	0.059	0.036	0.000	0.063	0.056
bright/earth_science	121 249	0.267	0.279	0.154	0.052	0.104	0.122	0.098	0.201	0.106	–
miracl/sw	131 924	0.431	0.475	0.027	0.005	0.052	0.134	0.099	0.052	–	–
beir/treccovid	171 332	0.731	–	0.249	0.186	0.218	0.158	0.253	0.324	0.226	0.241
mteb/QuoraRetrievalHardNegativ	177 163	0.815	0.714	0.449	0.448	0.219	0.676	0.627	0.710	0.663	0.616
bright/aops	188 002	0.030	0.032	0.037	0.044	0.038	0.035	0.034	0.048	0.049	0.050
bright/theoremqa_questions	188 002	0.071	0.056	0.059	0.051	0.024	0.044	0.074	0.071	0.046	0.053
coir/codesearchnet	280 310	–	0.664	–	0.001	–	–	–	–	–	–
miracl/bn	297 265	–	0.449	0.061	0.021	0.088	0.029	–	–	0.049	–
trec-pm/clinicaltrials-2019	306 238	0.408	0.503	0.066	0.034	0.057	0.072	–	–	0.093	–
beir/touche2020	382 545	0.312	0.442	0.060	0.057	0.065	0.086	0.048	–	0.055	0.025
bright/leetcode	413 932	0.152	0.138	0.115	0.049	0.034	–	–	–	–	–
miracl/te	518 079	–	0.449	0.007	0.009	0.016	–	–	–	–	–
beir/quora	522 931	0.814	0.727	0.425	0.493	0.387	–	–	–	–	–
· B/doc basis	–	–	–	–	–	–	–	–	–	–	–

B Detailed per-benchmark analysis

By domain. Reading the full table of Appendix A by suite, the clearest pattern is that the feature index is strongest exactly where retrieval depends on what kind of document is wanted rather than on fine topical similarity. On the legal suite MR-G leads every sparse and lexical family outright (0.740 suite mean against BM25's 0.489 and SPLADE++'s 0.301 on the benchmarks each covers) and sits 0.049 behind the trained retriever on the two legal benchmarks both cover (0.602 against 0.650), and on instruction-following retrieval it leads the trained retriever (0.504 against 0.480 on the shared benchmark). On reasoning-intensive retrieval the two sit close (0.166 against 0.177 suite mean) and several of our gains over the trained retriever live there. The biology gain holds for MR-G (0.199 against 0.173) and is largest for the untrained dictionary on the same embedder (0.228, paired bootstrap $p{=}0.002$), theorem retrieval is level for MR-G (0.217 against 0.217) with MR-F ahead (0.263, one-sided $p{=}0.04$), and stack overflow sits ahead within noise (0.187 against 0.178, $p{=}0.22$). The web, scientific, financial and medical suites sit close behind the trained retriever (within 0.024 to 0.044 of suite mean), while code and multilingual retrieval are its strongholds without steering (0.540 against 0.597 and 0.744 against 0.783).

Against the sparse families. MR-G leads SPLADE++ and BM25 on every suite mean except long-document retrieval, and the exception is instructive. The long-document suite contains synthetic exact-match constructions (a needle sentence or a passkey planted in a long distractor document), and for those the literal term identity is the entire task, so BM25 operates at its ceiling (0.651 suite mean against our 0.560). The mirror case runs the other way, since SPLADE++ was trained on the same cross-encoder distillation data as MR-G yet collapses where its vocabulary cannot name the content, to 0.133 on code and 0.165 on medical retrieval where we hold 0.450 and 0.338 on the same benchmarks, which is the vocabulary ceiling of a term index measured directly against the same supervision on model-internal features.

Base LLMs, and where monosemanticity helps most. For backbones never trained for retrieval, the monosemantic index leads the same model's pooled dense embeddings on every suite, and the margins say why. On the strongest base LLM the largest gap is long-document retrieval (0.478 against 0.160), where pooling a long document into one dense vector averages the deciding span away while the sparse features keep it as a discrete event that overlap scoring can still find. Instruction-following (0.436 against 0.314) and code (0.296 against 0.162) follow the same logic, since the deciding property is a localized signal rather than the document's average topic. The margins grow with scale within a family, and the ordering of bases matches the per-substrate frontier of Appendix C.

Against the trained retriever, and the success mode. The individual wins against the trained retriever concentrate where the gold is defined by document kind rather than by topic, reasoning-heavy web forums and instruction-constrained relevance among them, and this is the failure family the steering section targets, relevant documents of the wrong kind crowding the candidates. Where the trained retriever wins, the deciding property is fine-grained topical similarity inside one homogeneous corpus, code solutions to a described problem, numeric financial passages, scientific citation matching, and multilingual retrieval, all places where a purpose-trained dense geometry is exactly the right tool. Even there the comparison does not run away from us, since the same benchmarks are within 0.06 to 0.14 while the index holds 43% of the bytes and the dense arithmetic stays bounded, and the clearest counterexample closes the argument, because code retrieval, our weakest unsteered domain, is the domain the language facets of §F.3 lift past every trained system. This reframes the deficits as a usage nuance rather than an outright loss, since where fine-grained topical retrieval is what decides, the right feature-facet description lets steering outperform the base retrieval, and we hypothesize the same holds for the other trained-retriever-dominated domains once their facets are described well, and the error analysis of Appendix H sets out the conditions under which it does. Where pure quality favors the trained retriever, the frontier position still pays in index size, inference compute and training cost, and where the deciding property is a facet, the same index steers past it.

C MATCHED-SUBSET AND PER-SUBSTRATE FRONTIERS

Figure 2 places every system on the same 40 benchmarks, the set on which the quantized dense ladder was measured, so every point there is a mean over identical evaluations. Figure 3 is the same figure with each system on every benchmark it covers, 66 in all, with the number of benchmarks printed beside each system. Means over different benchmark sets are not comparable across systems, so that figure is read within a system's own coverage, and it is the version behind the per-benchmark tables of Appendix A. The ordering of §5 is unchanged on it, with MR-G at 0.462 over its 60 benchmarks against the trained retriever's 0.488 over its 57, and the learned-sparse and lexical families well below.

What the arithmetic buys. Table 5 puts the measured costs side by side. Scalar quantization changes the precision of each multiply-accumulate and not their number, so the 4-bit, 6-bit and int8 arms all scan every document at 8,192 operations, as the fp16 index does, and the two-stage MR-G query is $28\times$ cheaper than any of them at the median corpus and $55\times$ cheaper at the largest while matching their quality within 0.001. The inverted-file arm reduces the count by probing a fixed fraction of its lists, which brings it near MR-G's arithmetic at 0.8 KB per document, at a cost of 0.07 nDCG@10. The graph arm reaches the dense ladder's quality at a near-constant 1.2×10^7 to 1.4×10^7 operations per query, $1.4\times$ MR-G at the median corpus and $0.2\times$ at the largest, from an index $4.7\times$ larger per document than the postings.

Index bytes. The 43% figure is an upper bound, because we count 6 bytes per stored pair uncompressed. Production inverted indexes routinely compress postings with delta and variable-length coding, which dense vectors cannot use. MR-G and the quantized dense indexes are both well ahead of the learned-sparse baselines, which occupy the same range of index sizes but score much lower. MR-F, our own learned-sparse head on the same base model, shows why. Retraining the full dictionary scans even more cheaply, at 30 operations per document, but costs 0.02 nDCG@10 against MR-G (0.396 against 0.419 on the 40 benchmarks) and gives up the frozen feature meanings that make the index interpretable and steerable. The k128 head gives MR-G a size setting of its own, at 1,944 bytes per document and 0.404 on the same benchmarks, whereas each step down the scalar

ladder is a coarser precision the hardware may not run natively, and 6-bit and 4-bit have no native kernels.

Arithmetic. The 214-operation figure is itself an upper bound, because we counted every posting a query touches, while production inverted indexes skip entries that cannot change the top-k. It is a median over 80 collections, with per-collection values from 140 to 423 that fall as collections grow, since a query's active features cover a shrinking share of a larger corpus, and Table 5 reports the two collections behind the $28\times$ and $55\times$ figures. The postings are the only stored index. IVF-PQ and HNSW are pruning structures laid over a separate vector store, and each is built from the indexed data: scalar calibration, PQ codebooks and graph edges all degrade as that data shifts. MR-G's dictionary, head and encoder are fixed before indexing, so a new domain indexes exactly as the old one did, and shards can be built independently and merged.

Graph indexes. On the 40 shared benchmarks HNSW at efSearch 64 reaches 0.439 at 1.0×10^7 operations per query, level with reranked MR-G at 1.1×10^7. Its distance count is nearly flat in collection size, between 876 and 1,801 across the six corpora we built graphs on, from 5k to 518k documents, so it is cheaper than MR-G's linear scan above roughly 50k documents and more expensive below. What that bound costs is elsewhere. A flat-vector graph holds 16.7 KB per document resident in memory, 8.6 GB for the 518k corpus and 1.7 TB at 10^8 documents, where a postings index of 3,558 bytes per document is served from disk. FAISS offers no deletion from a graph, and implementations that do mark deleted nodes as tombstones that degrade search until a rebuild. And the count does not hold as the collection fills: recall@10 at efSearch 64 fell from 0.999 at 29k documents to 0.967 at 518k, and restoring it to 0.974 required efSearch 128 at $1.7\times$ the distance computations, a cost paid on every query.

Table 5: Quality against index bytes and per-query operations on the 40 benchmarks shared with the FAISS ladder. Sparse-arm operations are measured on the built posting lists, with the two-stage row adding the fixed 200-candidate rerank of $2 \cdot 200 \cdot 4{,}096$ operations and averaging the 36 of the 40 benchmarks the rerank run covers. A dense scan counts $8{,}192$ operations per document. HNSW (M=32, efSearch 64) counts its distance computations at $8{,}192$ each, measured at 28.8k and 518k documents, with recall@10 against exact search of 0.999 and 0.967. IVF-PQ follows the probe model $2\,n_{\text{list}}d + 2 \cdot 256\,d + (n_{\text{probe}}/n_{\text{list}})\,n\,m$ with $n_{\text{list}} = 4\sqrt{n}$, $n_{\text{probe}} = n_{\text{list}}/32$ and $m = 128$. Operations are given at the median corpus of 31k documents and the largest of 523k.

System	nDCG@10	Bytes/doc	Ops at 31k	Ops at 523k	Ops vs MR-G two-stage
MR-G, two-stage	0.439	3,558	9.0×10^6	7.8×10^7	$1\times$ / $1\times$
MR-G, sparse index only	0.419	3,558	7.3×10^6	7.7×10^7	$0.8\times$ / $1.0\times$
MR-G k128, sparse index only	0.404	1,944	3.8×10^6	3.7×10^7	$0.4\times$ / $0.5\times$
MR-F, sparse index only	0.396	2,381	9.9×10^5	7.5×10^6	$0.1\times$ / $0.1\times$
Dense, fp16	0.439	8,192	2.5×10^8	4.3×10^9	$28\times$ / $55\times$
Dense, int8	0.439	4,098	2.5×10^8	4.3×10^9	$28\times$ / $55\times$
Dense, 6-bit	0.439	3,074	2.5×10^8	4.3×10^9	$28\times$ / $55\times$
Dense, 4-bit	0.438	2,050	2.5×10^8	4.3×10^9	$28\times$ / $55\times$
Dense, HNSW	0.439	16,656	1.2×10^7	1.4×10^7	$1.4\times$ / $0.2\times$
Dense, IVF-PQ ($m = 128$)	0.367	757	8.0×10^6	2.7×10^7	$0.9\times$ / $0.3\times$

Figure 4 then breaks the frontier out by autoencoder substrate, the nine bases across the Gemma-Scope and Qwen-Scope families, and asks whether the recipe depends on one architecture or SAE type. It does not. Every base traces the same shape, and the untrained monosemantic index sits above the same backbone's pooled dense embeddings for every one of the nine, which is the base-LLM claim of §5 holding per substrate. Within the Qwen3.5 family quality rises from the 2B base (0.222 mean nDCG@10 over the 50 benchmarks every base covers) to the 9B (0.264), with the 27B at 0.258, and within the Gemma-3 family the 4B base (0.204) sits above both the 1B (0.191) and the 12B (0.173), so scale helps up to a point and the dictionary matters more than the backbone's size beyond it. The Qwen bases sit above the Gemma bases at comparable sizes, and the state-of-the-art embedder base leads them all (0.285), which is the measured reason the main configuration builds on it (§3).

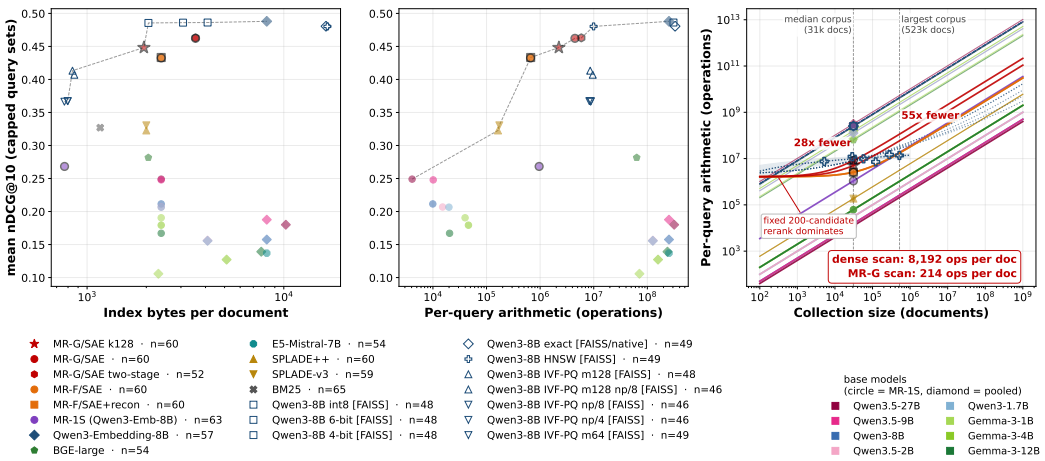

Figure 3: The frontier of Figure 2 with each system on every benchmark it covers, 66 in all, and its number of benchmarks beside each system. Operations for the measured sparse arms and the graph index are the same measurements as in Figure 2. Means over different benchmark sets are not comparable across systems.

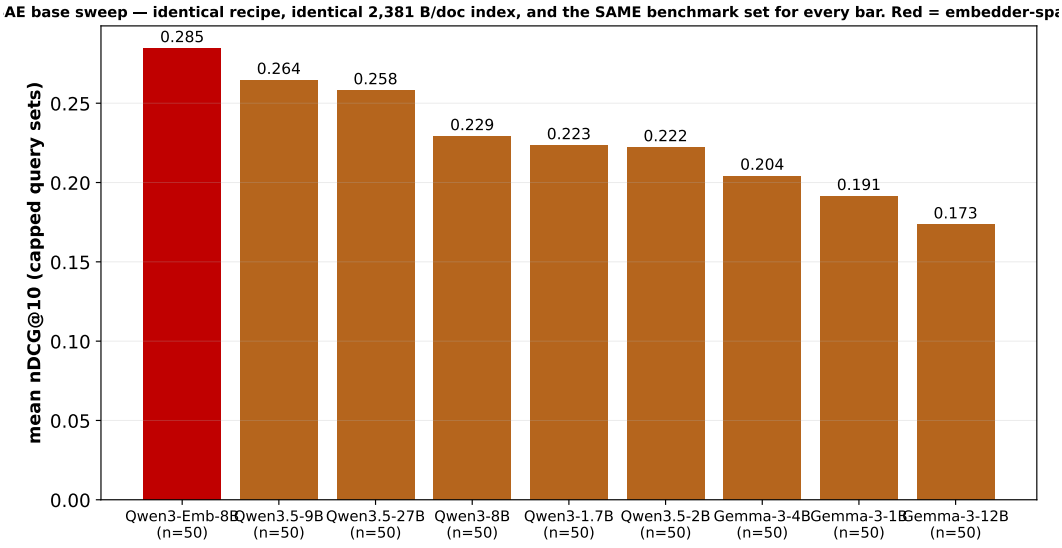

Figure 4: The frontier broken out by autoencoder substrate (nine bases across the Gemma-Scope and Qwen-Scope families), showing the recipe is not specific to one dictionary.

D HELD-OUT HEAD-TO-HEAD

Every design choice in the main configuration (layer, dictionary, gains, depth) was fixed on the 66 benchmarks of §4, so we also test the frozen recipe, in the zero-shot spirit BEIR established (Thakur & Gurevych, 2021), on the MTEB retrieval tasks that took no part in any tuning decision. As the main benchmark set grew over the course of the work most of the original held-out tasks were absorbed into it, and ten remain outside it, listed in Table 6. The two-stage configuration matches or improves on the dense full scan on all ten (mean nDCG@10 0.717 against 0.715), so the sparse index inherits whatever generality the frozen embedder has.

Table 6: Held-out MTEB retrieval tasks, nDCG@10 of the dense full scan against the two-stage configuration with the frozen recipe, no tuning on any of them.

Task	Group	Documents	Dense scan	Two-stage	Δ
NanoArguAna	NanoBEIR (subsampled)	3,635	0.756	0.756	+0.000
NanoDBPedia	NanoBEIR (subsampled)	6,045	0.701	0.701	+0.000
NanoHotpotQA	NanoBEIR (subsampled)	5,090	0.883	0.886	+0.002
NanoNFCorpus	NanoBEIR (subsampled)	2,953	0.376	0.387	+0.011
NanoNQ	NanoBEIR (subsampled)	5,035	0.781	0.781	+0.000
NanoQuora	NanoBEIR (subsampled)	5,046	0.951	0.951	+0.000
NanoSCIDOCS	NanoBEIR (subsampled)	2,210	0.525	0.525	+0.000
NanoSciFact	NanoBEIR (subsampled)	2,919	0.804	0.804	+0.000
NanoTouche2020	NanoBEIR (subsampled)	5,745	0.641	0.641	+0.001
Multilingual NanoSciFact	Scientific	2,919	0.736	0.742	+0.006

E THE MIXED INDEX

Constructing the mixed index. For every pair of collections in our benchmark pool we merged their corpora and measured, for one collection's queries, the share of the frozen trained retriever's top-ten slots taken by the other collection's documents, which we call the pair's confusability. Pairs at 0.4 or more in either direction were admitted, which gives nine collections across legal, mathematical and medical retrieval, statutes next to case documents and several kinds of clinical documents. We then build one mixed index from the nine (431,090 documents in total, 100 queries per collection under a fixed seed), so the index carries the recall problem just as a real-world setting would. On top of the nine, we also add an unrelated code-retrieval corpus purely as noise, with no queries scored against it, to simulate a real-world index where plenty of documents are relevant to no query at all. Duplicated corpora are indexed once, since two copies of the same document under different names would poison any facet measurement. The collection a document came from is used here only to construct the index. The inference-time steering loop is never given it. This confusability through collection merging serves as a way to introduce complexity of real world data scrapes into the target benchmark, to ensure it is as difficult as possible for the real world simulation with the mixed index.

Selecting the mixed index. Table 7 reports the confusability of every collection pair we measured, on the merged corpus of the pair and before any steering is run, and the decision each pair received. The financial pair, for example, falls below the 0.4 threshold and is not included. The code pair's symmetric 0.50 is the signature of a duplicated corpus rather than of mutual confusion, the two benchmarks share byte-identical document matrices and their pair axis classifies at chance, so the corpus is indexed once and serves as the noise mass. One further pair repackages corpora already admitted, and one sits outside the cached protocol because of corpus size.

Table 7: Confusability over collection pairs, the share of one collection's top-ten slots the frozen retriever fills with the other's documents on their merged corpus. A pair enters the mixed index when it is 0.4 or more in some direction.

Collection pair	Confusability	Decision
statutes ↔ case documents	0.87	admitted
pmc-treatment ↔ pmc-clinical	0.79	admitted
iiyi ↔ medxpertqa	0.73	admitted
medxpertqa ↔ pmc-clinical	0.69	admitted
medqa-diag ↔ pmc-clinical	0.55	admitted
iiyi ↔ medqa-diag	0.52	admitted
theoremqa questions ↔ theorems	0.49 / 0.45	admitted
code C2C ↔ T2C	0.50 / 0.50	duplicate corpus, indexed once as noise
pmc repackage	0.78	duplicates admitted corpora
hotpotqa ↔ nq	0.40	outside the cached protocol (corpus size)
FinanceBench ↔ FinQA	0.17 / 0.16	below threshold, not included

F CONTROLLED STEERING EXPERIMENTS WITH KNOWN QUERY INTENT

The two experiments in this appendix are not simulations of a deployment. They are using proxies from available data we have on hand to (i) isolate error signals coming from intent classification so that we can study the behaviour of steering strength and normalization in a controlled setting (ii) ablate on different methods which can be used to do automatic intent classification in the inference-time setting. Both hand the system the facet and, in one regime, the query intent as well, using the collection a document came from as its facet and the collection a query wants as its intent. That isolates the error that intent detection would otherwise add to the measurement, so one variable can move at a time, which is what lets us measure the design choices that the inference-time loop of §6.1 then has to make on its own without being given any of this.

Three of those choices come from here. The reweighting strength and its dependence on how pure a facet's features are comes from the strength responses below, and it is why the online loop fixes a calibrated normalised strength rather than tuning a raw one. Whether to normalise the projection comes from the comparison of the raw and the normalised reweighting under noisy intent. And the choice of a frontier LLM as the intent detector comes from the three detectors of §F.2, where the two cheaper detectors that read the query's own features or the unsteered results are measured and found wanting, so the online loop's routing agent is a validated choice rather than an assumed one.

F.1 FACET DEFINITION AND AXIS CONSTRUCTION IN THE CONTROLLED EXPERIMENTS

The controlled experiments define their facets from labels we hold. These labels are not true facets, just collection level identities which we use as proxies to isolate error signals coming from query intent and focus on the other aspects of steering. which the inference-time run of Appendix G never sees, and this subsection is the construction they use. It is separate from the construction of Appendix G on purpose, since the two differ in where the examples come from, how many there are and how an axis is admitted.

Facet discovery and labeling. The collection facets we used as proxies of known query intent are used in the control experiments to study the behaviour of the steering strength and normalization. They need no labeling, since collection membership comes free with the benchmark. The finer facets are discovered and labeled before any analysis quantity is computed. We first cluster every collection's full document matrix (spherical k-means with ten clusters over the cached representations, covering the whole corpus rather than a sample), read exemplar documents per cluster, and let a frontier-LLM agent propose the candidate facets the collection factors into. We then draw a cluster-stratified sample of 160 documents per collection and label it with independent LLM labeling agents, each given a blind batch with collection identifiers stripped, 1,440 labels over the nine collections in total, and two independent labeling campaigns agree on 200 of 200 overlapping documents. The labeled sample serves both the axis construction below (50 documents per side) and the analysis quantities that follow it. Labels then extend to the full corpus by the cheapest mechanism that validates, carrying cluster labels where the clusters align with the facet, applying a deterministic surface rule where one validates at 0.85 agreement or better, or training an axis on ten or more labeled documents per side and predicting the rest. The programming language facets are the exception, since their labels come from corpus metadata and are exact.

Constructing a facet axis from labeled documents. For each facet, 50 labeled documents that carry it and 50 that do not are drawn from the labeled sample, and the axis is the twelve features whose mean activation differs most between the two sides, with that difference as each feature's loading, the same construction activation steering uses (Rimsky et al., 2024; Andy Zou et al., 2023). A document's projection on the axis is the loading-weighted sum of its activations over those features, and at query time each of the 200 candidates has its first-stage score multiplied by $\max(0,\, 1 + s \cdot p_d)$ at a raw strength s, or by the same factor with p_d standardized by the corpus median and spread per axis in the normalised variant. Strengths are scanned over 4–64 for the collection facets and up to 1,024 for the language facets, at recall depths of 50, 100 and 200, and the figures of this appendix report the scans. The axes reach held-out facet-classification AUC of 0.954–0.999. As a check on the labeling budget we also rebuilt them using half of each corpus as labeled data, where the held-out separation does not improve, so the one-time 50-per-side labeling is a sample-efficient way to find definitive feature facets.

Computing the analysis quantities. Confusability is the share of the frozen retriever's top-ranked slots filled by wrong-facet documents, measured on the merged corpus for the pair table and conditioned on own-collection candidates for the within-collection version. Gold concentration is the fraction of queries whose gold documents all carry the same facet label. The fire contrast of an axis is its top feature's fire rate on labeled in-facet documents minus its fire rate on the labeled rest. Transfer quality is the held-out accuracy of the label-extension mechanism against labels it never saw. All four are computed from the labeled sample and the frozen retriever's own rankings.

Within-collection calibration. For the finer facets of §H we ran the full dose-response inside every admitted collection and recorded, per collection, the within-collection confusability, the gold concentration, the top feature's fire contrast (its fire rate on in-facet documents minus its fire rate on the rest), the label-transfer quality, and the steering increment of the composed axis over the collection-only axis with its bootstrap 95% CI (Table 8). The only significant positive increment sits at the largest within-collection confusability, every value at or below 0.30 is null, and the two significant negatives are the two collections whose label transfer falls furthest below the floor, which is the evidence behind the label-quality condition. All prose facets carry fire contrasts of 0.41–0.72 and inverted-U strength responses, while the language facets reach a contrast of 1.00 and the plateau of §F.3, so the contrast separates code from prose while confusability separates the winners inside prose.

Table 8: Within-collection calibration. Increment is composed axis over collection-only axis at each configuration's best strength and depth, paired bootstrap 95% CI. The language row has no collection-only incumbent and reports the gain over the unsteered system.

Collection	Confus.	Gold conc.	Fire contrast	Transfer	Increment	95% CI
theoremqa-theorems	0.54	0.82	0.63	0.85	$+0.035$	$[+0.000, +0.071]$
medxpertqa	0.30	0.60	0.61	0.66	$+0.006$	$[-0.012, +0.026]$
theoremqa-questions	0.25	1.00	0.67	1.00	$+0.002$	$[-0.018, +0.024]$
pmc-clinical	0.22	0.80	0.64	0.70	-0.017	$[-0.037, +0.002]$
iiyi-clinical	0.17	0.71	0.62	0.42	-0.006	$[-0.021, +0.008]$
pmc-treatment	0.15	0.75	0.72	0.65	-0.178	$[-0.252, -0.109]$
medqa-diag	0.15	0.82	0.41	≈ 0	-0.075	$[-0.105, -0.047]$
aila-casedocs	0.08	0.86	0.59	0.89	-0.001	$[-0.002, +0.000]$
aila-statutes	0.00	0.56	–	0.96	$+0.000$	degenerate
code language	0.74–0.88	0.85 / 0.40	1.00	1.00	$+0.233$	$[+0.190, +0.279]$

Argument retrieval and the headroom condition. Steering argument retrieval, a collection outside the mixed index whose gold is concentrated on one facet, gives a composed-over-collection increment of $+0.0025$ with a bootstrap 95% CI of $[+0.0003, +0.0057]$ (p=0.016), and $+0.0108$ over the unsteered system (p=0.0004). The magnitude is what exposed the headroom condition of §H, because argument retrieval's wrong-facet candidates sit below its gold, while every large win came from displaced gold, statutes at 0.17 on the mixed index against 0.84 isolated, and text-to-code at 0.31 against the 0.66 steering reaches.

Intent detectors. Query intent is the harder half, and we tried three detectors of increasing capability, each with its own prior. The first prior is that the intent is already expressed in the index's own vocabulary, so the simplest detector scores the query against each facet axis the same way we score documents, on the expectation that the query itself would at least partially express the features of the facet it wants, at no cost beyond the projection. This reaches only 25% accuracy on the mixed index, because queries express how people ask about a document rather than what the document looks like. The second prior is the classical one behind relevance feedback, that the unsteered results, however confused, still lean toward the desired facet, so the second detector takes the majority facet over the top retrieved candidates. The risk is that in the worst case we would be steering toward the wrong facet, and on a confusable index both concerns turn out true, so it reaches 46% and taught us that a detector needs a way to abstain. Since we are simulating a real-world setting, the third detector is an LLM classifier, on the prior that intent detection needs the query read as a request rather than as a bag of features, and that such a reader stays robust when new intents, facets or domains are added (a smaller dedicated intent model can replace it under compute constraints). We give it the abstention option the second detector motivated, with abstaining queries running unsteered. This reaches

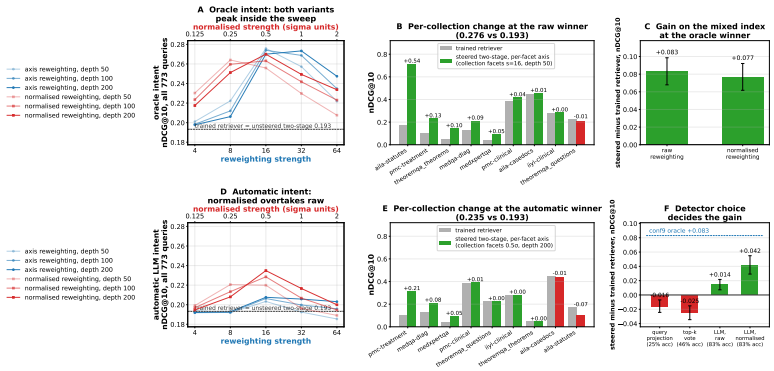

Figure 5: Collection-level steering on the nine-collection mixed index (431k documents including unscored distractor mass). Top row, known intent. Bottom row, automatic LLM intent with abstention. Columns show the strength response for the raw and normalised reweighting, per-collection nDCG@10 at each regime's best approach, and the change against the trained retriever with bootstrap 95% CIs, for the raw and normalised reweighting (top) and for the three intent detectors (bottom). Gains appear on the collections the retriever confuses most, survive the distractor mass, and drop to about half under automatic intent.

83% accuracy at 24% abstention, the abstentions concentrating where the same query text appears in several collections and only the gold defines which is wanted. On the language facets the same three detectors reach 24%, 24% and 41%, and the LLM classifier abstains on all 1,123 text-to-code queries because their text never names a language, which is the measured basis for presenting the language experiment under specified intent.

F.2 Collection-level facets under known and automatic intent

Figure 5 reports the mixed-index experiment under two regimes. With the intended collection known per query (the pseudo-grounded setting), the reweighted two-stage system reaches 0.276 nDCG@10 against the trained retriever's 0.193, a gain of +0.083 and a 43% relative improvement over the state-of-the-art trained retriever. Per-collection gains track the measured confusability (the most confused collections improve the most). The strength response is an inverted U peaking at s=16, and can be explained intuitively from the behavior of the feature facets. Documents belonging to a facet project more strongly onto its features than documents that do not, but the projection is rarely perfect. The best statutes feature, for example, fires on 77% of the statute documents and on 4% of everything else, where an ideal facet would fire on 100% and 0%, in which case the strength would not matter at all. Because the facet is noisy, a strong enough reweighting separates the scores until the intended facet dominates the candidate list, while past the sweet spot the false negatives of the axis begin suppressing relevant documents and bend the curve down. We will see the ideal case later, since the code-language facets of §F.3 come close to the 100-to-0 profile and let the strength grow arbitrarily high.

Query intent is the harder half. The three detectors of Appendix F.1 reach 25%, 46% and 83% accuracy here, and the LLM classifier with abstention retains +0.042 nDCG@10 over the trained retriever, about half the grounded gain. The raw and normalised variants split along two lines here. For a heavily imbalanced index normalising is the safer choice (the corpus median and spread absorb the imbalance) while for a roughly balanced mix like ours tuning the raw strength suffices, a distinction that matters in the wild where the class balance of an index is rarely known ahead of time. Intent certainty is the second line, and the less expected one. Under noisy intent the raw reweighting is the brittle approach (its gain drops from +0.033 to +0.014 when the distractor mass is added) while the normalised approach is unchanged, so normalisation is the more reliable choice whenever intent is automatic.

F.3 GRANULAR TASK-DEFINED FACETS

Collection membership is the coarsest facet a corpus offers, but there can be multiple levels of granularity at which facets are defined. For task-specific settings, it is possible to define more granular and meaningful facets than collection membership alone, which we test using the code-retrieval benchmarks, since their corpus is factored by construction. The corpus contains programming problems, and every problem appears implemented in five programming languages. A query in these benchmarks asks for a specific problem in a specific language, so the topic of the retrieval (which problem) is fully separated from the facet (which language), and getting the topic right is not enough, since the same problem in the wrong language is a wrong answer. We are not just looking for a code implementation here, we are looking for that code implementation in a particular programming language. The frozen retriever confuses exactly this, filling 74% of its top-ten slots with the wrong language for code-to-code queries and 88% for text-to-code queries. What we study through this is how the quality of the feature facets impacts steering, and how well the facets can be defined when the task supplies them. Task-specific retrieval can sit inside a real-world mixed index or run on its own dedicated corpus, so we test both.

Figure 6 reports both benchmarks separately, on the mixed index and standalone on their own corpus. With the target language specified at query time, code-to-code retrieval moves from 0.453 to 0.558 on the mixed index and to 0.590 standalone, and text-to-code moves from 0.306 to 0.662 mixed and from 0.414 to 0.680 standalone, a pooled standalone gain of +0.242 with a bootstrap 95% CI of $[0.228, 0.256]$ over all 1,401 queries, and relative improvements of 30% and 64% over the state-of-the-art trained retriever on the two benchmarks. There are cases where documents in several languages can all be accurate for a given query, and for those our system automatically runs unsteered based on the intent detection, since we know we do not know the query intent for them. The strength response also connects back to the facet behavior of §F.2. The language features are extremely clean, with the Go feature firing on 100% of Go documents and none of the rest, so instead of the inverted U of the noisy collection facets the raw reweighting saturates into a plateau (strengths 256 through 1,024 are flat and equal to the normalised plateau), acting at saturation as a clean filter that cannot overshoot, which is why the strength can grow arbitrarily high here. Steering in this fashion improves performance for task-specific code retrieval, with steered text-to-code at 0.68 against the trained retriever's 0.415, SPLADE++'s 0.203 and BGE-large's 0.158 on the same benchmark, and the corresponding code-to-code numbers 0.59 against 0.452, 0.059 and 0.130.

While such a setting is may not be realistic, since such facets might possibly be filtered by some sort of metadata, this control was designed to allows us to study the behaviour of pure facets which can perfectly separate gold documents from irrelevant ones, and confirm that steering works across multiple granularities of facet definitions. It also helps us study the different settings and behaviours of the intent detection module, which can help us better understand when and where steering can help, and to what extent. The text of a text-to-code query never names its language (an LLM classifier abstains on 1,123 of 1,123), and the language a code-to-code query is written in matches the language it wants only 41% of the time, so steering on detected intent would not be possible. Intent that is not written down cannot be detected, only specified, and the finer the facet, the more often specification, not detection, is the workable interface.

What the strength responses say about monosemanticity. Beneath the deltas, these controlled, query-intent-isolated steering results measure a granularity property of representations, because the dense retriever provably contains the language facet (a probe on its embedding recovers it at ceiling accuracy) yet cosine cannot be told to care about it, while the feature index exposes the same information as addressable coordinates. Every axis classifies at the same held-out AUC, yet a single Go feature fires on 100% of Go documents and 0% of the rest while prose facets need committees of softer features (echoing that not every concept is carried one-dimensionally (Engels et al., 2025)), and this per-feature selectivity, not the AUC, is what predicts how hard a facet can be pushed. A dedicated feature converts a facet from classifiable into steerable-to-saturation, and the plateau-versus-U shapes follow from it. This allows us to define the selection rule for when to admit a facet into the facet store for the inference-time run: there is a threshold and a limit for how much separability a feature facet should contain for it to be a high quality steering axis, which is what we use to define the held-out set analysis and admission rule protocol during the inference-time real-world simulation.

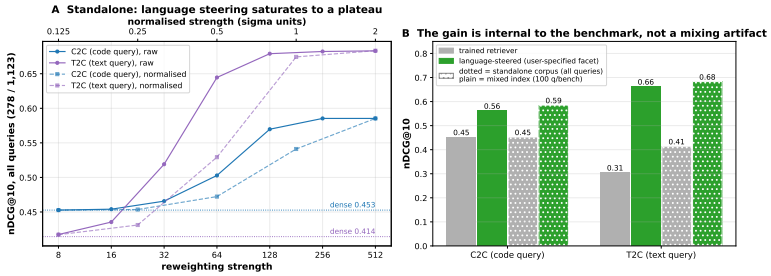

Figure 6: Language-facet steering on code retrieval, code-to-code and text-to-code reported separately. Left, the strength response of the raw and normalised reweighting standalone on the deduplicated corpus, saturating to a filter rather than turning over because the language axes are nearly exact. Right, dense against steered nDCG@10 on the mixed index and standalone (dotted overlay), with all queries evaluated and multi-language-gold queries running unsteered. A 12-feature named axis takes text-to-code retrieval from 0.41 to 0.68, above every trained system we measure on these benchmarks.

G THE FACET STORE ROUND BY ROUND

Axis construction in the inference-time run. No labels enter this construction. The positive and negative examples of a facet are the ones the miner picked from the round's own candidate pool, with at least eight on each side, and the axis is the twelve features whose mean activation differs most between them, with that difference as each feature's loading, so that with $\bar{z}_{P,i}$ and $\bar{z}_{N,i}$ the mean activation of feature i over the two sides, $A_f = \mathrm{top}_{12}\,(\bar{z}_{P,i} - \bar{z}_{N,i})$, $w_i = \bar{z}_{P,i} - \bar{z}_{N,i}$ and $p_d = \sum_{i \in A_f} w_i\, z_{d,i}$. The axis is calibrated on the same examples, recording the median and spread of p_d over them so that the normalised projection $\tilde{p}_d = (p_d - \mathrm{med}(p))/\,\mathrm{sd}(p)$ is defined before the facet is ever applied, and it is admitted to the store only if it separates a 70/30 held-out split of its own examples at an AUC of 0.75 or better. Every admitted facet is then applied at the one fixed setting the loop carries, the normalised strength of 0.5 standard deviations and a recall depth of 200, through $\hat{s}_d = s_d \cdot \max(0,\, 1 + s\,\tilde{p}_d)$, and nothing is scanned or tuned per facet.

The loop in full. The simulation loop runs as follows. A stream of 978 queries arrives over the mixed index in rounds of 50. The facet store starts empty, and the only thing the system ever observes is which documents a user ended up using out of the ones it returned, which we simulate with the benchmark gold labels for the queries of that round, revealed only after the round has been served. The first round therefore runs unsteered, because an empty store has nothing to steer with, and every later round is served with the state of the facet store at that time.

In the offline mining step after each round, the documents the user used and the documents the system returned but the user ignored are handed to a frontier-LLM agent (the router and miner prompts are reproduced in Appendix M) as a contrast. The agent decides whether that contrast exposes a property worth steering on at all, and it is free to abstain if no meaningful facet is discoverable. When it does find a property, it also chooses the granularity, which we do not constrain or suggest in any way, so a facet can come back as broad as the genre of a collection or as narrow as a single proof technique. It selects its own positive and negative examples from that round's candidate pool, which is what keeps the mining sample-efficient, and the axis is then computed from those examples by the same construction. A facet enters the store only through a held-out AUC gate, and facets already stored are ignored (the gate and the statistics of the 215 facets the run admitted follow below).

Query intent based facet selection is handled by another frontier-LLM agent that reads the current store and the text of each incoming query, with fresh context per batch and no access to anything but the store's facet names and descriptions. It may return several facets for one query, and it may abstain, in which case that query runs unsteered. Using a frontier model for this rather than something cheaper is itself a measured choice, since the two cheaper detectors we tried fall well short of it (Appendix F.2). We run two experiments on the router's output. The single-facet experiment steers each query by the one facet the router ranked first, while the composed experiment sums the loading vectors of every facet the router selected for that query into one axis.

The reweighting strength is fixed at a calibrated normalised 0.5 standard deviations for every facet in the run, chosen from the controlled experiments of Appendix F, where the facet and the intent are known and the strength can be moved on its own. One setting is possible at all because the shape of the strength response is governed by how pure a facet's features are, and an online store, which mines facets of every purity without knowing which it has, takes the normalised form at a fixed strength and tunes nothing per facet.

Because the facet definition is unconstrained, the store ends up carrying facets at every granularity, with 17 of the 215 naming a collection genre, 100 an intermediate subtype and 98 a specific topic or technique, and the coarse ones are minted in the first rounds while the narrow ones accumulate later once the remaining separation is within a collection rather than across collections (they are listed round by round below).

Table 9 is the whole of the inference-time run of §6.1, one row per round, in the order it happened. It is here because the aggregate numbers hide the thing the experiment is actually about, which is a system with nothing in it at the start that ends up with a working steering vocabulary purely from watching its own traffic.

A facet is admitted to the store only if it carries at least eight examples per side and separates a 70/30 held-out split at an AUC of 0.75 or better. Of the 215 facets the run admitted, the median held-out AUC is 1.000 and 80% sit at 0.95 or above, from a median of twelve positive and fourteen negative examples each. Across the 1,729 facet assignments the router made, 93.0% point at a facet already in the store at least two rounds earlier, over the 21 mining rounds 792 of the miner's decisions reused a facet that already existed against 239 proposals for new ones, so the store does not grow unbounded as user data scales, and 895 of the 978 queries (91.5%) end up steered, with the remainder being the first round and the router's abstentions.

Two things in the table are worth reading directly. Round 1 is served unsteered because the store is empty, and it is the only round that has to be. And the proposals column settles rather than falling to zero, at roughly ten new facets per round over the second half of the run against 28 in the opening round, so after twenty rounds the miner is still finding properties the store does not carry, while four out of five of its decisions in any round point at a facet that already exists.

Table 9: The inference-time run round by round. Queries per round is 50 except for the last round of each stream segment. Steered counts the queries the router did not abstain on. Dense is the trained retriever on that round's queries, and single and composed are the two steering experiments. Proposed are the facets the miner returned as new, admitted are those that passed the held-out AUC gate, and reuse counts the miner's pointers to facets that already existed.

Round	Queries	Steered	Dense	Single	Composed	Proposed	Admitted	Reuse	Store
1	50	unsteered	0.122	0.122	0.122	28	16	4	16
2	50	43	0.234	0.302	0.291	5	5	36	21
3	50	46	0.181	0.162	0.180	9	9	41	30
4	50	41	0.112	0.150	0.127	11	10	36	40
5	50	45	0.256	0.358	0.351	7	7	39	47
6	50	48	0.147	0.195	0.206	19	15	37	62
7	50	46	0.238	0.260	0.243	15	14	35	76
8	50	48	0.191	0.248	0.255	15	15	36	91
9	50	50	0.236	0.292	0.279	8	7	46	98
10	50	50	0.163	0.147	0.191	8	8	50	106
11	50	50	0.147	0.178	0.172	17	17	41	123
12	50	50	0.135	0.194	0.185	10	10	46	133
13	50	50	0.214	0.261	0.266	12	11	52	144
14	50	50	0.229	0.279	0.265	12	11	55	155
15	50	50	0.184	0.173	0.172	12	12	45	167
16	23	23	0.200	0.298	0.311	4	4	23	171
17	50	50	0.108	0.118	0.129	17	17	44	188
18	50	50	0.193	0.213	0.216	9	9	48	197
19	50	50	0.194	0.211	0.227	11	9	40	206
20	50	50	0.116	0.220	0.201	10	9	38	215
21	5	5	0.202	0.254	0.254	0	0	0	215

The 215 facets separate into 17 that name a collection genre, such as *court case judgments* and *statute sections*, 100 at an intermediate subtype level, such as *bibliography and citation-list fragments (runs of literature references with no expository prose)* and *criminal-appeal judgments in reversal-of-acquittal posture*, and 98 that are specific to a topic or a technique, such as *contour-integral evaluation via Cauchy's integral formula* and *neonatal salt-wasting adrenal crisis*. The coarse facets were minted in the first rounds, when the contrasts the agent saw were dominated by which collection a document came from, and the narrow ones accumulate later, once the coarse ones are already in the store and the remaining separation is within a collection rather than across collections. This is the granularity ladder of Appendix F.3 appearing on its own, from feedback, rather than being supplied by us. `claude-fable-5` was used throughout as the frontier LLM agent during the simulation for both the roles.

H When does steering help? An error analysis

Across all of the steering runs, three conditions separate the queries and collections where steering helped from the ones where it did not, and none of them depends on how the facet was obtained. The first is a recall bottleneck along a facet, meaning that the candidates crowding out the relevant documents are of the wrong kind rather than merely off topic, which is what the confusability measurements of Appendix F.1 record for the collection pairs and, within a collection, for the finer facets. The second is that the relevant documents for a query share one facet that can be mined, which we measure as gold concentration, the fraction of queries whose gold documents all carry the same facet label. Note that this is the more defining factor for steering, as a properly defined facet can allow for arbitrarily granular steering and classification of documents. Our control experiments serve to study the behaviours of the systems at different granularities, and how other parts of the system depend on it, while the actual inference-time real-world simulation allows this arbitrary granular control of the system to a capable frontier LLM. In our control settings, for example, text-to-code carries 85% single-language gold and steers accordingly, code-to-code only 40% (most of its queries accept an implementation in any language, and such queries simply run unsteered), and nfcorpus is the clean counter-example, confusable at 0.32 but at 17% concentration because relevance there ignores the facet entirely, so steering it would be pointless however good the axes. These show, as an example, how gold concentration on a defined facet can affect steering in a controlled study. The key insights generalize beyond these control settings. The third is headroom, because wrong-facet candidates only matter if they sit above the gold, and pushing down documents that already rank below it changes nothing, which is what bounded the argument-retrieval gain of Appendix F.1 to $+0.0025$. Label quality is a floor underneath all three, since forcing steering through 65%-accurate labels loses up to 0.18 nDCG@10, so a facet whose labels cannot be validated at roughly 75% accuracy is not steered at all, the same abstention logic the intent classifier applies per query. The per-facet measurements behind this are tabulated in Table 8 of Appendix F.1.

I Ablations and controls

Each moving part of our experiments carries a measured control. Swapping the SAE dictionary for a matched random basis at the same sparsity, pooling and scoring roughly halves retrieval quality (mean nDCG@10 0.319 against 0.165 over 28 paired substrate–layer configurations, with a shuffled dictionary at 0.160 and the random basis winning none of the 28), so the content of the dictionary, not merely its sparse format, matters for retrieval. The training side decomposes the same way in the main results, where the trained gains reach 0.460 mean nDCG@10 on their 59 benchmarks against the untrained variant's 0.282 on those 59 (0.268 over its 63, §5), collapse to 0.346 when the encoder bias is frozen along with the dictionary, and gain nothing from adding a frozen-decoder reconstruction objective (0.431 against 0.431 for the distilled head), so what training buys is a reweighting of which features matter rather than reconstruction fidelity. Halving the stored activations costs 0.014 (0.446 at 1,944 bytes per document against 0.460 at 3,558). We also test multiple aggregation and scoring variants (max, sum and attention-weighted pooling, Tversky, inner product) and find that they shift means by small margins. This is the expected outcome, because the argument of §1 is about the space the score is computed in rather than the rule applied within it. The dilution of decisive axes is a property of full-dimensional distances, where every coordinate contributes and the contrast between near and far neighbours vanishes as dimension grows (Beyer & Shaft, 1999; Charu

C. Aggarwal & Keim, 2001; Weller & Lee, 2025). All three rules score over the few features a query and a document both activate, a few hundred coordinates out of 32,768 rather than that in the dense vector, so each of them sidesteps that dilution. Across the 61 collections a query shares a median of 186 active features with its relevant document and 108 with a random one, so the score is decided on a support of a few hundred features, not on the order of thousands of coordinates. Weighted Jaccard and Tversky do so by set overlap (Broder, 1997; Tversky, 1977) and the inner product does so because a sparse dot product is a sum over the shared support alone (Formal & Clinchant, 2021). What separates them is how the shared features are weighted, which is the small margin the ablation measures.

First-stage control. The two-stage system's quality comes from the dense rerank, so the question is whether the sparse first stage is a better candidate generator than a lexical one at the same depth. Replacing it with BM25 and holding the rerank, the depth and the metric fixed answers this on 25 benchmarks. At depth 200, BM25 candidates recover 57.3% of the dense retriever's own top-10 and MR-G candidates 98.4%, and the reranked results reach 0.359 against 0.400 mean nDCG@10, with the dense full scan at 0.411 on the same benchmarks. MR-G recalls more on 24 of the 25 and its reranked result is higher on 15.

Stages. On the 57 benchmarks where both stages are measured at native context, single-stage MR-G reaches 0.481 mean nDCG@10, its 200 candidates contain 99.6% of the dense top-10, and the rerank lifts the result to 0.506 against the dense scan's 0.509. The first stage sets recall, the rerank sets the final quality, and the reranked result sits within 0.003 of the dense scan.

On the steering side, three controls bound what the reweighting is and is not doing. The dense-space loop of §6, the same inference-time run with the SAE removed from every stage, ends within noise of the unsteered retriever, so the gain comes from the monosemantic index rather than from the LLM roles or the multiplicative gate. The label-noise results of Appendix H bound label quality, with 65%-accurate labels losing up to 0.18 nDCG@10. Within the inference-time run, steering by the single facet the router ranked first and composing every facet it returned are indistinguishable (paired difference +0.0000, 95% CI $[-0.0082, +0.0084]$ over the 978 queries), so the gain does not depend on how the routed facets are combined.

I.1 Steering a dense retriever with the same loop

The loop of §6.1 is rerun with MR removed from every stage. The trained dense retriever serves the stream, its own top 100 forms the miner's feedback pool, and a mined facet becomes the unit mean-difference direction of its positive and negative examples in the 4,096-dimensional embedding space, the construction of activation steering (Andy Zou et al., 2023; Rimsky et al., 2024). The direction is admitted by the same held-out AUC gate, calibrated by the same corpus median and spread, and applied through the same gate $\max(0,\, 1 + s\,\tilde{p}_d)$ to the cosine scores. Its strength is the normalised optimum the calibration procedure of Appendix F gives for the dense scan, 0.125 standard deviations, against 0.5 for MR. The frontier model, the contracts and addenda, the round structure and the 978 queries are the reported run's, with fresh routing and mining decisions, and the aggregate result is the last two rows of Table 1.

Neither the feedback nor the store separates the two runs. The dense retriever's pools held a relevant document for 67% of queries against 62% for MR's first stage. The dense loop admitted 433 facets (26 collection-level, 282 subtype, 125 topic) against MR's 215, with a mean held-out AUC of 0.979 against 0.969 and four gate rejections, since a mean difference over 4,096 entangled dimensions separates almost any two example sets. Its router steered 899 of the 928 queries of rounds 2 to 21, and 93.1% of its assignments reused a facet at least two rounds old, matching MR's 93.0%.

Table 10 shows where the two loops part. Both gain on the collections the retriever confuses with a sibling, statutes against the judgments that apply them and theorem statements against the problems that need them. The dense-space loop then gives its gains back on the collections the retriever was already serving well. Where statutes sit beside judgments, steering toward statutes raises statute queries from 0.171 to 0.416 but drags case-document queries, already answered at 0.446, down to 0.258, and the clinical case reports fall from 0.385 to 0.282 under facets meant for the neighbouring clinical collections. A direction in the embedding space moves every document that co-varies with the property it was fitted on, whether or not that document carried the intent, so the same reweighting

Table 10: The dense-retriever loop against the MR loop per collection, single-facet arm, nDCG@10 with the change from the unsteered dense retriever in parentheses.

Collection	Queries	Unsteered dense	Dense-space loop	MR loop
AILA-STATUTES	50	0.171	0.416 (+0.245)	0.528 (+0.357)
AILA-CASEDOCS	50	0.446	0.258 (−0.189)	0.438 (−0.009)
THEOREMQA-THEOREMS	76	0.045	0.125 (+0.080)	0.092 (+0.047)
THEOREMQA-QUESTIONS	194	0.161	0.177 (+0.016)	0.176 (+0.015)
PMC-TREATMENT	150	0.108	0.080 (−0.027)	0.200 (+0.092)
PMC-CLINICAL	114	0.385	0.282 (−0.103)	0.399 (+0.014)
IIYI-CLINICAL	129	0.253	0.238 (−0.015)	0.226 (−0.028)
MEDQA-DIAG	118	0.116	0.131 (+0.016)	0.118 (+0.002)
MEDXPERTQA	97	0.039	0.067 (+0.028)	0.046 (+0.007)

that promotes the wanted kind demotes what the retriever had right. Under MR's facets of twelve named features the case-document queries move by -0.009 and the clinical case reports by $+0.014$, because a document that expresses none of a facet's features keeps its place relative to every other such document. The agent is not what differs between the runs. It detects the same intents and mines from richer pools, and the signal it hands the dense retriever is what cannot be acted on.

The monosemantic facet is also the cheaper one. Projecting the index on it reads the posting lists of its twelve features, 5.1×10^5 entries on average over the 215 facets of the store, where a dense direction needs a 4,096-wide product against every document, 3.5×10^9 operations, about $7{,}000\times$ more per facet, and the facet itself is twelve feature ids and loadings rather than a 431k-entry projection to cache.

J Discussion

The steering results are, to us, the clearest operational payoff of monosemanticity we have measured. On an index built to be as confusable as a real scrape, a system that is told nothing about where its documents came from or what its queries want recovers, from feedback alone, a vocabulary of facets that runs from the genre of a collection to the posture of a criminal appeal and the technique a proof needs, routes queries to it, and reweights candidates in the space the index is already stored in, and doing so improves on the state-of-the-art trained retriever with no change to the index. The error analysis of Appendix H says when to expect this, when the candidates crowding out the relevant documents are of the wrong kind, the relevant documents share a facet that can be mined, and the facet can push those candidates below the gold, and the controlled experiments of Appendix F, which fix the facet and the intent to isolate the reweighting itself, are where its strength and normalised form come from.

K Qualitative examples

The per-benchmark analysis of Appendix B fixes which examples are representative, and Tables 11 to 13 show one worked example per success mode, each drawn from the stored rankings of the base-LLM comparisons (the monosemantic index against the same substrate's pooled dense embeddings), with the exact query, the exact top-ranked document of each system marked by its relevance grade, and a comment connecting the example back to the analysis. Tables 14 and 15 extend the same format beyond the base-LLM setting, to the learned-sparse head-to-head and the trained retriever, each recomputed from the cached representations of the runs the main text reports.

L Implementation details

Encoding and indexing. Every document is encoded once by the frozen base model at its native context, up to 4,096 tokens with longer documents processed in 2,048-token chunks, and queries carry the instruction prompt the embedder was released with. The resulting embedding h is L2-normalised. For the untrained configuration, the pretrained SAE encoder produces $\mathrm{relu}(W_{enc}\, h + b_{enc})$ and we keep the $k{=}128$ highest activations. For the trained configurations,

Table 11: Instruction-constrained relevance (instruction-following retrieval, robust04). nDCG@10 for this query, 0.68 monosemantic against 0.10 pooled dense.

Query	"How is Attention Deficit Disorder (ADD) diagnosed and treated in young children? Relevant documents specifically address diagnosis and/or treatment of ADD. Simple definitions and descriptions of ADD are not relevant."
Monosemantic top-1 **(relevant)**	"Training or technical assistance for an individual with disabilities, or, where appropriate, the family of an individual with disabilities … (c) Attention deficit disorder (ADD). As used to define [students receiving treatment services]"
Pooled dense top-1 **(not relevant)**	"DYSLEXIA CONFUSION … Question: I'm confused about the term dyslexia. It seems to be applied now to every child who has trouble learning … even experts disagree on the definition of dyslexia and its cause."
Comment	The instruction excludes definitional documents, and the pooled vector returns exactly one, topically adjacent as a childhood learning condition. The feature overlap lands on the required document kind, the treatment-services context, which is the document-kind success mode of the analysis.

Table 12: Reasoning-style medical retrieval (medical-sciences). nDCG@10 for this query, 1.00 monosemantic against 0.00 pooled dense.

Query	"What diseases can be cured by laughing? I'm helping my sister to write [an] article about 'laughing'. I believe laughing and happiness have positive effects on our health … I wondered whether laughing can cure diseases."
Monosemantic top-1 **(relevant)**	"Laughter may even help you to live longer. A study in Norway found that people with a strong sense of humor outlived those who don't laugh as much. The difference was particularly notable for those battling cancer … Boosts immunity, lowers stress hormones, decreases pain."
Pooled dense top-1 **(not relevant)**	"Research confirms that this can be achieved through numerous mechanisms, including relaxation, breathing exercises, fitness exercises, imagery, meditation, yoga, qigong, tai chi, biofeedback …"
Comment	The deciding property is one specific concept (laughter) inside a crowded wellness region, and the pooled vector retrieves the region while the feature overlap retrieves the concept, which is the pattern behind the suite-level margin on medical retrieval.

Table 13: Code without shared vocabulary (code-contest retrieval). nDCG@10 for this query, 1.00 monosemantic against 0.00 pooled dense.

Query	"Takahashi is a user of a site that hosts programming contests … Let the current rating of the user be a. Suppose that the performance of the user in the contest is b. Then, the new rating of the user will be the average of a and b."
Monosemantic top-1 **(relevant)**	A five-line Python solution whose comments are written entirely in Japanese, reading the current and goal ratings and printing `(G * 2) - R`, the exact performance value the problem asks for.
Pooled dense top-1 **(not relevant)**	"# Question B. Road to Cinema … `def roadToCinema(V, S, T, stations)` … whether this [vehicle] can travel within the time limit", a solution to a different contest problem about fuel stations.
Comment	The query and its gold share almost no vocabulary in any term sense, since the gold's comments are Japanese and its identifiers are two letters, so only the algorithmic content connects them. The feature overlap carries that content while a term index has nothing to match, which is the vocabulary ceiling of the analysis measured on a single query.

the code is $c(h) = \log\big(1 + \mathrm{relu}(Wh + b) \odot e^{g}\big)$, where W is the SAE dictionary held as a frozen buffer and g is the per-feature gain. Because a TopK autoencoder's encoder never had to produce sparsity by itself, we calibrate the dictionary before training with one shared scale on all rows and one shared bias shift, chosen so the initial sparsity matches a randomly initialised head at the same bias. This calibration is direction-preserving, every feature points exactly where it did, so the auto-interpretability meanings and the steering axes of the dictionary survive verbatim. In MR-G the gain and bias train (65.5K parameters), in the bias-frozen variant only the gain trains (32.8K), and in MR-F the full W trains (134M). Nonzero features become posting lists, and index bytes count the stored (feature id, activation) pairs at 6 bytes each.

Table 14: The vocabulary ceiling measured on the code query of Table 13. SPLADE++ representations of the query and its gold share almost no informative active terms, and SPLADE++ scores 0.086 nDCG@10 on this benchmark overall.

Query, strongest terms	takahashi, contest, rating, average, constraints, host
Gold, strongest terms	int, r, g, input, print, target
Shared strong terms	the single-letter variable names g and r, and generic tokens such as input, print and output
Comment	The query describes the problem in prose and the gold implements it in code, so the only vocabulary they share is variable letters and boilerplate input-output tokens. A term index has nothing informative to match, while the monosemantic features of Table 13 carry the algorithmic content itself.

Table 15: MR-G/SAE against the trained retriever on a reasoning-forum query (BRIGHT biology), recomputed from the cached representations under the published scoring protocol. Per-query nDCG@10, 0.61 for MR-G/SAE, 0.24 for the trained retriever.

Query	"Why can't humans drink sea water? It would seem to be a huge evolutionary advantage … Could someone explain to me any likely reasons that this is the case?"
MR-G/SAE top-1 **(relevant)**	The osmolarity of sea water entry (1000 mOsm/l), the quantity the explanation actually rests on, since water above blood osmolarity dehydrates rather than hydrates.
Trained retriever top-1 to top-3 **(all not relevant)**	A bare "Saltwater Science" blog-link fragment, then two general ocean-ecosystem passages about kelp, plankton and how fish obtain oxygen.
Comment	The query never names osmolarity. Its relevance runs through a reasoning chain from drinking sea water to salt concentration, which is exactly what the BRIGHT suite is built to test. The trained retriever fills its top slots with passages that are topically about sea water and say nothing about why it cannot be drunk, while the feature overlap promotes the document expressing the concentration concept itself. This is the pattern behind the gains on reasoning-heavy retrieval in Appendix B.

Training. The backbone stays frozen and its embeddings are cached once, which is what makes training at tens of thousands of steps tractable. Supervision comes from MS MARCO passage-ranking triples with mined hard negatives and cross-encoder margins from a BERT ensemble, the same distillation data that produced SPLADE++ (Thibault Formal & Clinchant, 2022). The loss distills the student's sparse-dot margin onto the cross-encoder margin, adds an in-batch and hard-negative InfoNCE term, and regularizes sparsity with the FLOPS penalty $\sum_j (\mathbb{E}_i\, s_{ij})^2$ under a quadratic ramp (Formal & Clinchant, 2021).

Dictionary provenance. The 32,768-feature dictionary over Qwen3-Embedding-8B is one we trained, since the public SAE releases cover the residual streams of base language models and not the pooled output of an embedding model. It is a TopK autoencoder with k=128 on the L2-normalised 4,096-d embedding, trained for 40 epochs on 400,000 passages drawn without labels from the document sides of QuoraRetrieval, Touché-2020, ArguAna, SciFact and AskUbuntu (batch 4,096, learning rate 4×10^{-4}, auxiliary loss on the 512 deadest features at weight $1/32$, seed 0), which takes under four minutes on one H100. It ends at a fraction of variance unexplained of 0.080 with 77 dead features. The training uses their documents only, never a query or a relevance label. This is an inference-time fit on the document text, similar to the IDF weights, quantizer calibration and codebooks that every indexed system fits on its own corpus. The dictionary itself is frozen before any retrieval training. On the 62 benchmarks whos corpora is never used for training, the performance trends remain the same.

Sparsity. The k=128 of the untrained index is a hard cap. The trained heads use a ReLU threshold and their sparsity is set by the FLOPS penalty, so the number of active features per document is a measured quantity rather than a setting. Figure 7 shows its distribution over the 4.99M documents of the 80 native-context collections. MR-G stores a median of 578 features per document, with per-collection medians from 300 to 1,122, the k128 head, whose calibration targets 128, stores 278, and the untrained index stores exactly 128. Query codes are similar, with a median near 480 for

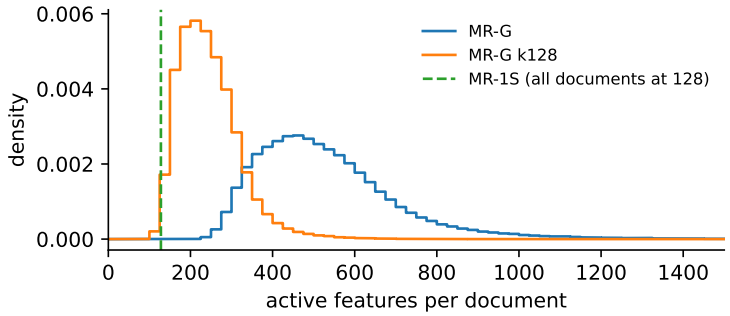

Figure 7: Active features per document for the trained heads, pooled over the 80 native-context collections (4.99M documents). The untrained index is capped at 128, the trained heads are set by the FLOPS penalty, and MR-G stores a median of 578.

MR-G. The scan cost follows from these counts and from how the postings are distributed over features, because a query walks only the posting lists of its own active features. Measured on the actual posting lists, counting two operations per entry, the MR-G scan touches 214 entries per indexed document at the median collection (140 to 423 across collections), the k128 head 109 and the untrained index 35. The most frequent 1% of features hold 28% of MR-G's postings.

Interpretability of features. Learned sparse baselines such as SPLADE are trained only on the objective of retrieval and ranking. Their training objective does not require sparsity in the context of inducing monosemantic representations from an activation reconstruction setting. SAE training, on the other hand is trained under a reconstruction objective under sparsity constraints and its activations are a measurement of the models own representation. MR-F ablations show this directly, as we are able to recover the SPLADE-style quality on our backbone but at the cost of interpretability of the sparse features.

Scoring. The IDF weight of feature i is its inverse document frequency over the indexed corpus. The Tversky and inner-product variants score over the same posting lists. In the two-stage configuration the sparse stage returns 200 candidates, and the rerank is cosine between the query embedding and the candidates' embeddings from the same frozen base model, computed on the fly so nothing dense is stored.

Base-LLM substrates. For backbones that are not embedding models, the representation is the residual stream at a chosen layer, pooled over token positions (max, sum or attention-weighted pooling), and encoded by that backbone's pretrained SAE from the Gemma-Scope-2 or Qwen-Scope releases at the release's default layer.

Benchmark protocol. Benchmarks with very large query sets are evaluated on a fixed uniform sample of 2,000 query ids drawn under a pinned seed, identical for every system, and nDCG@10 follows the standard trec evaluation protocol. The mixed steering index caps each collection at 100 queries under a fixed seed.

Baseline settings. Baselines run inside the same harness rather than being copied from published tables, so that quality, index bytes and per-query arithmetic are measured on identical corpora and query samples. E5-Mistral runs through the official mteb loader with its released instruction template applied to queries, last-token pooling, fp16 weights and the 4,096-token context cap that every system shares. The comparison is against the quality-versus-cost frontier of this harness and not against values obtained under other settings, and a baseline that scores lower here than in its own report is reported as it scored.

Mixed-index construction. Document ids and qrels are namespaced by collection. Before the mixed index is built, every collection must reproduce its exact-search reference score on its own iso-

lated rows, and a failed check is an error rather than a skip. Byte-identical corpora appearing under two benchmark names are detected and indexed once. The codes come from the same dictionary the axes were discovered in, so stored feature ids apply verbatim at steering time.

Cost accounting. Index bytes are counted from the stored postings or dense vectors at their stored precision, per-query dense arithmetic follows the formulas of Appendix A, and latency is served through the production FAISS and Lucene paths. The index-size figures count the postings. If a deployment stores a dense copy for the rerank instead of recomputing the 200 candidates, the 4-bit quantised copy of Table 5 costs 2,050 bytes per document and scores within 0.001 of fp16 on a full scan, so the two-stage index would hold 5,608 bytes per document, 68% of the fp16 dense index, at unchanged rerank quality.

M ROUTER AND MINER PROMPTS

The two LLM roles of the inference-time loop (§6.1) run under fixed contracts, reproduced here verbatim. Each prompt is the contract, followed from round 7 onward by the addendum, followed by the round's batch (the facet store with ids, names and descriptions, and the queries or feedback pools of the batch). The placeholder `%OUT%` is the path the agent writes its answer to. We ensured that the IDs were opaque and no prefixes which could leak the intent or facet labels were provided in the prompts.

Router contract.

> You are routing retrieval queries to steering facets. For each query below, choose the facet(s) from the store whose described property the query's intended documents must have. Order multiple facets by confidence.
>
> - Abstain (empty list) unless a facet clearly applies: steering with a wrong facet is worse than not steering. The measured system abstains on roughly a quarter of queries.
> - Never invent facet ids.
> - Output: write a single JSON object {"<qid>": {"facets": ["f0003", ...]}, ...} with an empty list for abstention, covering EVERY qid below exactly once.
>
> Write that JSON (and nothing else) to the file `%OUT%` using the Write tool, and also return it as your final message.

Miner contract.

> You are mining steering facets from retrieval feedback. For each query below you get the query text, its gold document ids, and the top-100 unsteered candidate pool as [{doc_id, is_gold, snippet}]. The gold documents are what the user ended up using; the rest of the pool was retrieved but not used.
>
> - Decide whether that contrast exposes a document property worth steering on. Abstain freely (empty list) when the pool and the gold do not separate on any nameable property, or when relevance is purely topical.
> - When a property exists: select POSITIVE examples (pool documents expressing the property, gold included) and NEGATIVE examples (retrieved-but-unused documents that lack it). At least 8 per side, from the pool only.
> - Name the facet at the granularity the contrast supports, specifically enough to be reusable. Add a one-sentence description a later query-router will read, and a granularity tag (collection | subtype | topic-specific).
> - If an equivalent facet already exists in the store names below, return {"reuse": "<facet_id>"} instead of redefining it.
> - Up to 3 facets per query when genuinely different contrastive splits of the same pool support them.
> - Output: a single JSON object {"<qid>": [{"name":..., "description":..., "granularity":..., "pos":[doc_id...], "neg":[doc_id...]} | {"reuse":"<facet_id>"} , ...], ...} with an empty list for abstention, covering EVERY qid below exactly once.
>
> Write that JSON (and nothing else) to the file `%OUT%` using the Write tool, and also return only a one-line count summary as your final message.