Optimal Fusion Weights Form Intervals, Not Points: A Latency-Aware Study of Query-Adaptive WRRF

Anonymous ACL submission

Abstract

Reciprocal Rank Fusion (RRF) is widely used to combine sparse and dense retrievers in hybrid information retrieval, although a critical limitation is that it assigns a fixed, uniform weight to each retriever regardless of the query. Query-aware methods treat the per-query optimal weight as a single point, an assumption whose empirical validity has not been examined. We present an empirical study of dynamic fusion weight prediction across four datasets and four sparse–dense retriever pairs. We characterize the per-query optimal weight as a set-valued object and benchmark ten fusion weight-prediction methods across complexity categories. Methods include a dataset-level constant, linear regression, encoder LMs fine-tuned to predict weights from queries, and LLMs conditioned on a variety of inputs including in-context examples. Comparing these approaches to a per-query oracle that bounds achievable performance, three findings emerge. First, the per-query optimum is typically an interval rather than a point. For 10–38% of queries examined, the standard RRF weight already lies inside it, explaining why uniform fusion is a competitive baseline. Second, while the headroom above standard RRF is substantial, no evaluated method recovers more than 25% of it in most retriever configurations. Third, the latency–quality relationship is dataset-dependent rather than monotone, motivating a tiered decision framework that maps practitioner constraints, dataset characteristics, label availability, hardware, and latency budget to an appropriate method class.

1 Introduction

Hybrid information retrieval (IR) systems, which combine lexical signals from sparse retrievers with semantic embeddings from dense retrievers, have become the standard architecture for modern search applications (Ma et al., 2020; Berntson, 2023; Sawarkar et al., 2024). To merge these disparate sources, Reciprocal Rank Fusion (RRF) (Selberg and Etzioni, 1997; Cormack et al., 2009) has been adopted in practice (Ding et al., 2020; Jin et al., 2024; Rackauckas, 2024; Yu et al., 2025). RRF is effective because it operates on document ranks, bypassing the scale and range mismatches inherent in combining raw lexical and semantic scores (Bruch et al., 2023) in a training-free manner.

However, standard RRF uses a fixed, uniform weight ($w = 0.5$) as a common default, treating all retrievers as equally valuable. Alternative score-based strategies, such as the convex combination of Bruch et al. (2023), fuse raw scores rather than ranks and require both per-retriever score normalization and a learned per-dataset weight. As such, this approach is less common in practice, since rank-based RRF needs neither. Weighted RRF (WRRF) makes the fusion weight tunable, and the natural follow-up is to choose it per query. We show this framing is itself wrong: optimal WRRF weights typically form intervals, not points.

Recent work has nonetheless pursued query-aware fusion weights, adjusting the WRRF parameter per query rather than fixing it at a dataset-level optimum. Dynamic Alpha Tuning (DAT) (Hsu and Tzeng, 2025) uses a Large Language Model (LLM) to predict a query-specific fusion weight, exemplifying the query-aware weighting paradigm but introducing substantial inference latency at the fusion stage. How much of the achievable improvement above standard RRF that dynamic methods actually capture remains an open question.

Our work establishes that optimal fusion weights form intervals, not points. We characterize this interval structure across diverse datasets (MS MARCO, NQ, NFCorpus, ACORD) and retriever combinations. We then compare a spectrum of weight-prediction strategies across these combinations. These include a dataset-level constant, linear regression, fine-tuned LM encoders, in-context LLM prediction, and DAT. We evaluate these meth-

ods against the per-query oracle that bounds what any predictor can achieve. We measure absolute retrieval quality, the fraction of available headroom each method recovers above standard RRF, and per-query inference latency.

To summarize, our main contributions are:

- A characterization of the optimal fusion weight as an interval rather than a point, with structural categories (empty, RRF-easy, single interval, multi-interval) that together explain RRF's robustness and reframe per-query weight prediction as a set-valued problem.
- An evaluation showing that, despite substantial headroom above standard RRF, no evaluated method (including DAT) recovers more than a small fraction of it; the bottleneck is the predictability of the per-query optimal weight from the query alone, not the headroom itself.
- A tiered decision framework that organizes all evaluated methods along a measured latency spectrum and maps practitioner constraints (labels, GPU access, latency budget) to method classes.

2 Related Work

Multi-stage ranking systems rely on a retrieval phase and multiple re-ranking stages (Bruch et al., 2023). Hybrid IR combines sparse lexical retrievers, such as BM25 (Robertson et al., 1994; Robertson and Zaragoza, 2009), which excel at exact keyword matching, with dense retrievers (Karpukhin et al., 2020; Izacard et al., 2021; Chen et al., 2024), which leverage transformer-based embeddings to capture semantic relationships.

To merge these retrieval streams, fusion methods have historically relied on static aggregation strategies. Early approaches rooted in social choice theory, such as Borda Count (Aslam and Montague, 2001) and round-robin interleaving (Etsioni, 1995), often struggled with varying candidate pool sizes or ignored cross-system agreement. RRF (Selberg and Etzioni, 1997; Cormack et al., 2009) advanced retrieval fusion by applying a hyperbolic decay constant to reward consensus among retrievers.

Still, standard RRF can be brittle. An optimized convex combination of normalized scores can perform better (Bruch et al., 2023), but such approaches remain limited because they rely on static, dataset-level weights.

The intuition that fusion weights should adapt to individual queries has driven the development of dynamic strategies closely related to Query Performance Prediction (QPP), which aims to predict retrieval effectiveness a priori (Arabzadeh et al., 2024). Translating performance estimates into actions can follow various paths. In the context of fusion, a recent focus has been on using LLMs to calibrate retrieval parameters on the fly. AM-SRAG (Dong et al., 2026) utilizes lightweight models such as ModernBERT (Warner et al., 2025) to classify queries into complexity tiers, then adjusts both fusion weights and the smoothing constant accordingly. Similarly, DAT (Hsu and Tzeng, 2025) uses an LLM to evaluate the top result for each retriever and assign a query-specific fusion weight.

While DAT improves retrieval quality over standard RRF, it views fusion weight prediction as a point-estimation problem. Our in-context LLM approach contrasts with it in three respects: we fuse retriever ranks rather than scores (preserving RRF's freedom from per-retriever score normalization), condition the LLM on interval-valued few-shot examples drawn from similar queries rather than prompting it zero-shot, and use the query alone rather than the query plus top retrieved passages.

Beyond structural assumptions, deploying dynamic fusion methods to real-world search systems adds latency. A standard approach is to keep first-stage retrieval lightweight to reserve computational budgets for powerful downstream LLM rankers (Sun et al., 2023; Tang et al., 2024) or trained Learning to Rank (LTR) models (Liu, 2009; Li et al., 2024). So while a DAT-style LLM-in-the-loop approach to fusion may achieve better query-aware weighting, the LLM call incurs latency better spent elsewhere in the search pipeline.

We study several approaches for WRRF and compare to DAT. We chose to look at these WRRF variations because RRF is commonly used in practice and does not require standardized scores from the input retrievers. Inspired by DAT, we look at methods that are not just dataset-aware (setting the WRRF weight based on the mean optimal weight for a train split), but we also consider methods that dynamically set the WRRF for each query according to a query representation.

3 Methodology

This section describes our empirical study. We formalize WRRF and the per-query optimal weight

set $\mathcal{W}_q^*$ that underlies our characterization and the oracle ceiling used to compute headroom recovery. We then describe the spectrum of weight-prediction strategies we compare, organized from a dataset-level constant through discriminative encoders to LLM prediction. The spectrum is not a proposal for a single new method; it a set of candidates evaluated against the oracle which we evaluate across all datasets and retriever combinations.

3.1 Weighted Reciprocal Rank Fusion (WRRF)

Given a query q, let $\mathcal{R}_{\text{sparse}}(q)$ and $\mathcal{R}_{\text{dense}}(q)$ denote the ranked lists of documents returned by the sparse and dense retrievers. For a document $d \in \mathcal{R}_{\text{sparse}}(q) \cup \mathcal{R}_{\text{dense}}(q)$, denote its rank in retriever $s \in \{\text{sparse}, \text{dense}\}$ as $r_s(d|q)$, with $r_s(d|q) = \infty$ when d is absent from retriever's list.

We define the WRRF score for document d given query q as

$$\text{WRRF}(d|q) = \frac{w_q}{k + r_{\text{sparse}}(d|q)} + \frac{1 - w_q}{k + r_{\text{dense}}(d|q)} \qquad (1)$$

where $w_q \in [0, 1]$ is the fusion weight assigned to the sparse retriever for query q and k is the standard smoothing constant; we follow prior work and set $k = 60$ (Cormack et al., 2009; Bruch et al., 2023). Standard RRF corresponds to the static choice $w_q = 0.5$; the query-aware objective is to determine w_q on a per-query basis.

3.2 Empirical Optimal Weight Sets

For each query in datasets, we identify its optimal fusion weight(s) by sweeping $w \in [0, 1]$ with step size $\Delta w = 0.01$ and evaluating WRRF on the top-n retrieved candidates. The metric $\text{Metric}(q, w)$ is dataset-dependent, and n is fixed to the same value used at evaluation time. Because retrieval metrics depend on discrete rank positions, the resulting performance curve is a piecewise constant function of w, and the maximum is typically attained over a continuous range rather than at an isolated point. We therefore define the optimal weight set for query q as

$$\mathcal{W}_q^* = \left\{ w \in [0, 1] \;\middle|\; \text{Metric}(q, w) = \max_{w' \in [0,1]} \text{Metric}(q, w') \right\} \qquad (2)$$

which decomposes into a finite union of contiguous closed intervals,

$$\mathcal{I}_q = \{[a_1, b_1], [a_2, b_2], \dots\}.$$

The optimal weight set $\mathcal{W}_q^*$ may take several forms: (i) a single point, (ii) a single contiguous interval (wide or narrow), (iii) a union of multiple disjoint intervals, or (iv) the empty set $\emptyset$ in the degenerate case where no fusion weight surfaces a relevant document within the top-n (i.e., $\text{Metric}(q, w) = 0$ for all $w \in [0, 1]$), in which case no weight is optimal and the query carries no signal for distinguishing fusion weights.

The prediction strategies in § 3.3 operate on $\mathcal{I}_q$, with one exception: the dataset-level constant baseline (§ 3.3.1) uses a scalar target $\bar{w}_q$ per query. For that case, we take the midpoint of the widest interval in $\mathcal{I}_q$ and break ties with random selection.

3.3 Weight Prediction Strategies

While $\mathcal{I}_q$ provides an oracle ceiling on a labeled set, a practical system must assign fusion weights to unseen queries. We compare three families of strategies that span an adaptivity spectrum, from a single weight shared across the entire dataset, to per-query weights produced by a learned discriminative predictor, to per-query weights produced by an LLM that reasons over similar queries. Each family trades inference cost, training-data requirements, and access to retrieval signals against the adaptivity of its predictions; we treat this adaptivity–cost tradeoff as the object of study rather than committing to a single proposed method.

3.3.1 Dataset-Level Constant

Different from standard RRF, as a still-static but corpus-calibrated alternative, we replace 0.5 with the mean of $\bar{w}_q$ over the training queries, where $\bar{w}_q$ is the midpoint of the widest interval in $\mathcal{I}_q$. This baseline costs nothing at inference and persists only a single scalar per dataset.

3.3.2 Discriminative Query Predictors

To capture query-level variation, we train discriminative models that map a query q (and, optionally, the top retrieved passages as in DAT) to a predicted weight $\hat{w}_q \in [0, 1]$. We study three backbones spanning a range of capacity:

- Linear regression (Hoerl and Kennard, 1970) with an L2 (ridge) penalty over TF–IDF features of the query, as a simple learned predictor (Appendix B.1).
- RoBERTa (Liu et al., 2019) fine-tuned on a representation of the query as a transformer-encoder baseline (Appendix B.2).

- ModernBERT (Warner et al., 2025) fine-tuned in two variants: query-only and passage-conditioned. The passage-conditioned variant additionally consumes the top-ranked passage from each retriever as in DAT (Appendix B.3).

ModernBERT is exhibits the most expressive power of these approaches. For it, we vary two design choices.

Target form: point vs. interval. The simplest target is a single scalar $\bar{w}_q$, used by the linear regression baseline. But the optimal weight for q is generally an interval rather than a point, this scalar target forces the model to converge on an arbitrary summary statistic and penalizes predictions that deviate from it even when they fall within the query's optimal interval. We therefore also evaluate an interval target $[l_q, r_q] \subseteq \mathcal{I}_q$ on the transformer backbones, using the widest interval when $\mathcal{I}_q$ contains multiple disjoint pieces, consistent with the consolidation rule in § 3.2.

Loss form: MSE vs. satisficing. With a point target we use squared error. With an interval target we use an ϵ-insensitive satisficing loss (Simon, 1955) whose insensitive zone is the query's own optimal interval in the training data:

$$\mathcal{L}_{\text{sat}}(q) = \max(0, l_q - \hat{w}_q)^2 + \max(0, \hat{w}_q - r_q)^2 \quad (3)$$

The model incurs zero penalty when $\hat{w}_q \in [l_q, r_q]$ and is pushed only toward the nearest boundary otherwise.

3.3.3 In-Context LLM Prediction

At the most adaptive end of the spectrum, we use LLMs for their powerful few-shot prediction capability via a two-step in-context learning pipeline.

First, for a target query q, we retrieve the top-k most similar training queries from two independent indices over the training-query texts: a sparse lexical index and a dense index. The two top-k lists are deduplicated by query identifier to form the example set. The dual-retriever design ensures the LLM receives both lexically and semantically similar examples, details in § 4.3 and Appendix C.

Second, we construct a prompt containing each neighbor query with its optimal weight interval(s) $\mathcal{I}_{q_{\text{nbr}}}$, and ask the LLM to synthesize a single predicted weight $\hat{w}_q$ for the target query. The interval form of the exemplars conveys both the central tendency and width of acceptable fusion for similar queries; presenting a single mean weight would discard the width signal. We evaluate two LLMs of different capacity (checkpoints in § 4.3) to test whether predictor performance is bottlenecked by model scale. We also compare against DAT (Hsu and Tzeng, 2025) under matched LLMs; the design contrasts are summarized in §2.

4 Experiments

4.1 Datasets and Metrics

We conducted experiments on four datasets: MS MARCO (Nguyen et al., 2016), NQ (Kwiatkowski et al., 2019), NFCorpus (Boteva et al., 2016), and ACORD (Wang et al., 2025), spanning general web search, Wikipedia QA, medical IR, and legal contract-clause retrieval. Each dataset provides a training split used to label optimal weights and fit predictors, and a evaluation split (MS MARCO dev, NQ dev, NFCorpus test, ACORD test).

The evaluation metric is driven by the relevance characteristics of each corpus. For MS MARCO and NQ, which feature a single relevant passage per query, we report MRR@10. For NFCorpus and ACORD, which contain multiple graded-relevance documents per query, we report nDCG@10.

4.2 Retrievers

We pair two sparse retrievers with two dense retrievers, yielding four hybrid configurations that span varying "balance of power" between the modalities.

Sparse. BM25 (Robertson and Zaragoza, 2009), implemented via Pyserini (Lin et al., 2021), and RM3 (Abdul-Jaleel et al., 2004; Lavrenko and Croft, 2017), a pseudo-relevance feedback variant that performs query expansion using the top-ranked documents from an initial BM25 pass.

Dense. all-MiniLM-L6-v2 (Sentence-Transformers, 2021) and Qwen3-Embedding-8B (Zhang et al., 2025), an efficient bi-encoder and a high-capacity LLM-based embedder.

For both grid-search labeling and evaluation, we retrieve the top 200 candidates per retriever before fusion. This number was chosen based on our preliminary study, in which increasing the cutoff to 400 yielded no meaningful improvement.

4.3 Predictors and Comparators

Discriminative predictors. The linear regression baseline is implemented with scikit-learn; the neural backbones are RoBERTa-large (Liu et al., 2019) and ModernBERT-large (Warner et al.,

2025) (query-only and passage-conditioned variants), both fine-tuned end-to-end. Target–loss pairings follow Section 3.3.2; details in Appendix B.

In-context LLMs. The two-step pipeline of §3.3.3 uses Qwen3-32B (Zhang et al., 2025) and Ministral-3-14B-Instruct-2512 (Liu et al., 2026), with BM25 and Qwen3-Embedding-8B as the sparse and dense example-retrieval indices following a lightly modified KATE (Liu et al., 2022) (top $k = 5$ each, training split only). Prompt template: Figure 4; details in Appendix C.

Comparator: DAT. DAT (Hsu and Tzeng, 2025) is run under matched LLMs (Qwen3-32B, Ministral-3-14B-Instruct-2512).

4.4 Latency Measurement

We measure per-query inference latency for every method in § 3.3 on an AWS `ml.g6e.2xlarge` instance (48 GB VRAM). LLM calls (§ 3.3.3) are routed through Amazon Bedrock (the timed block includes the full round-trip). All timings use batch size 1 with no local caching, after a warm-up pass. Each method is run over the full evaluation split of every (dataset, retriever-pair) configuration.

We aggregate per-configuration mean latencies within a sub-tier and report mean $\pm$ 95% CI using a Student's t-interval with $n - 1$ degrees of freedom: $n = 16$ for single-method sub-tiers; $n = 32$ for sub-tiers pooling two methods. Each input is itself a mean over many within-dataset queries, so by the Central Limit Theorem, the sampling noise in each input is approximately normal; the t-interval then accounts for the small-sample uncertainty in estimating the standard deviation from only n values. We aggregate per-tier latencies the same way (mean $\pm$ 95% CI).

5 Results

We organize the results around three questions. §5.1 quantifies how much room exists above standard RRF for any per-query weight predictor, and how much of that room current methods recover. §5.2 characterizes the structure of the per-query optimal weight set $\mathcal{W}_q^*$ and shows the same characterization explains where the unrecovered headroom resides. §5.3 reports the per-query optimal weight distribution and cross-dataset patterns. The decision framework in §6 situates each method within the cost–quality trade-off these results imply.

5.1 Headroom and Recovery

For each query q in the evaluation split we computed the per-query oracle, $\mathrm{Oracle}(q) = \max_{w \in [0,1]} \mathrm{Metric}(q, w)$, by sweeping w on the same grid used during training-set labeling (§3.2). Averaged over the evaluation set, the oracle is the upper bound on what any weight predictor that assigns a single weight per query can achieve.

Table 1 reports the oracle, the headroom above standard RRF, and the performance of all evaluated methods. The headroom is substantial on MS MARCO and NQ (0.30–0.40 metric points across all four retriever pairs), and smaller on ACORD and NFCorpus (0.05–0.09), foreshadowing the flat latency–quality regime documented in §6. The larger absolute oracle on MS MARCO and NQ partly reflects an in-domain advantage: all-MiniLM-L6-v2 and Qwen3-Embedding-8B include MS MARCO and NQ in their training corpora, so the dense channel surfaces a relevant document more reliably on these datasets than on ACORD or NFCorpus. Translating these absolute differences into a comparable scale, Table 2 reports each method's recovery: the fraction of headroom it captures, $(\text{Method} - \text{RRF})/(\text{Oracle} - \text{RRF})$.

Two observations stand out. First, no method recovers more than 25% of the headroom on any BM25-based MS MARCO and NQ configuration; the maximum across all configurations is 38.3% (Ministral3 FS on NQ RM3+Qwen3), and the median across the 144 (method, configuration) cells in Table 2 is in the single digits. Second, headroom and recovery do not move together: MS MARCO and NQ BM25+MiniLM offer the largest absolute headroom ($\sim$0.35) yet every method recovers under 10% on each, while on small-headroom configurations (NFCorpus and ACORD, oracle–RRF gap of 0.05–0.08) a small miss of the optimal interval can drag a method below standard RRF in aggregate, most prominently on NFCorpus.

5.2 Anatomy of the Optimal Weight Set

The recovery numbers above are aggregate statistics over heterogeneous queries. To understand where headroom resides and why it is hard to capture, we partition each query's optimal weight set into four structural categories: empty (no $w \in [0, 1]$ achieves a nonzero metric), single point (one isolated optimal weight), single (1) interval, and multiple disjoint intervals. Table 3 reports the breakdown on the evaluation split.

Table 1: Absolute performance across datasets and retriever pairs. *Oracle* is the per-query upper bound $\frac{1}{|Q|}\sum_q \max_{w\in[0,1]} \mathrm{Metric}(q,w)$; *Headroom* is Oracle$-$RRF and bounds the achievable improvement of any per-query weight predictor. The metric is MRR@10 for MS MARCO and NQ, nDCG@10 for ACORD and NFCorpus.

Method	**MS MARCO (MRR@10)**				**NQ (MRR@10)**				**ACORD (nDCG@10)**				**NFCorpus (nDCG@10)**			
	BM25(M)	**BM25(Q)**	**RM3(M)**	**RM3(Q)**	**BM25(M)**	**BM25(Q)**	**RM3(M)**	**RM3(Q)**	**BM25(M)**	**BM25(Q)**	**RM3(M)**	**RM3(Q)**	**BM25(M)**	**BM25(Q)**	**RM3(M)**	**RM3(Q)**
Oracle (per-query best w)	0.634	0.620	0.627	0.618	0.720	0.689	0.689	0.677	0.203	0.221	0.204	0.231	0.405	0.455	0.423	0.473
Headroom (Oracle $-$ RRF)	0.353	0.336	0.358	0.347	0.347	0.303	0.400	0.376	0.071	0.083	0.078	0.088	0.055	0.067	0.068	0.077
Standard RRF	0.281	0.284	0.269	0.271	0.373	0.386	0.289	0.301	0.132	0.138	0.126	0.143	0.350	0.388	0.355	0.396
Mean Optimal Weight	0.295	0.314	0.289	0.309	0.381	0.417	0.332	0.378	0.135	0.137	0.132	0.143	0.348	0.397	0.355	0.403
Ridge Regressor (MBW)	0.296	0.317	0.289	0.310	0.383	0.416	0.330	0.377	0.138	0.137	0.130	0.145	0.349	0.396	0.356	0.401
RoBERTa Regression (IW)	0.304	0.323	0.298	0.319	0.391	0.421	0.335	0.388	0.126	0.154	0.126	0.154	0.350	0.402	0.356	0.403
ModernBERT (IW)	0.305	0.326	0.297	0.318	0.387	0.423	0.333	0.393	0.125	0.156	0.125	0.155	0.350	0.394	0.359	0.400
ModernBERT Passage Cond.	0.304	0.327	0.296	0.325	0.405	0.426	0.340	0.396	0.143	0.155	0.135	**0.159**	0.349	0.399	0.359	0.404
DAT Qwen3	0.299	0.309	0.285	0.299	0.401	0.427	0.320	0.358	0.141	0.148	0.132	0.147	0.346	0.399	0.362	**0.410**
DAT Ministral3	0.304	0.322	0.292	0.313	0.404	0.439	0.362	0.406	0.125	0.132	0.135	0.150	0.333	0.393	0.346	0.408
Qwen3 FS (IW)	0.304	0.335	0.295	0.328	0.391	0.433	0.346	0.410	0.137	0.143	0.126	0.142	0.350	0.403	0.355	0.403
Ministral3 FS (IW)	0.303	**0.350**	0.299	0.346	0.386	**0.446**	0.371	0.445	0.139	0.151	0.132	0.157	0.349	0.404	0.347	0.399

Table 2: Headroom recovery (%): (Method $-$ RRF)/(Oracle $-$ RRF) $\times$ 100. Standard RRF is 0% and the oracle is 100% by construction. Values below zero indicate underperformance relative to standard RRF. Best per column in **bold**.

Method	**MS MARCO**				**NQ**				**ACORD**				**NFCorpus**			
	BM25(M)	**BM25(Q)**	**RM3(M)**	**RM3(Q)**	**BM25(M)**	**BM25(Q)**	**RM3(M)**	**RM3(Q)**	**BM25(M)**	**BM25(Q)**	**RM3(M)**	**RM3(Q)**	**BM25(M)**	**BM25(Q)**	**RM3(M)**	**RM3(Q)**
Mean Optimal Weight	4.0	8.9	5.6	10.9	2.3	10.2	10.8	20.5	4.2	−1.2	7.7	0.0	−3.6	13.4	0.0	9.1
Ridge Regressor (MBW)	4.2	9.8	5.6	11.2	2.9	9.9	10.3	20.2	8.5	−1.2	5.1	2.3	−1.8	11.9	1.5	6.5
RoBERTa Regression (IW)	6.5	11.6	8.1	13.8	5.2	11.6	11.5	23.1	−8.5	19.3	0.0	12.5	0.0	20.9	1.5	9.1
ModernBERT (IW)	**6.8**	12.5	7.8	13.5	4.0	12.2	11.0	24.5	−9.9	**21.7**	−1.3	13.6	0.0	9.0	5.9	5.2
ModernBERT Passage Cond.	6.5	12.8	7.5	15.6	**9.2**	13.2	12.8	25.3	**15.5**	20.5	**11.5**	**18.2**	−1.8	16.4	5.9	10.4
DAT Qwen3	5.1	7.4	4.5	8.1	8.1	13.5	7.8	15.2	12.7	12.1	7.7	4.5	−7.3	16.4	**10.3**	**18.2**
DAT Ministral3	6.5	11.3	6.4	12.1	8.9	17.5	18.3	27.9	−9.9	−7.2	**11.5**	8.0	−30.9	7.5	−13.2	15.6
Qwen3 FS (IW)	6.5	15.2	7.3	16.4	5.2	15.5	14.2	29.0	7.0	6.0	0.0	−1.1	0.0	22.4	0.0	9.1
Ministral3 FS (IW)	6.2	**19.6**	**8.4**	**21.6**	3.8	**19.8**	**20.5**	**38.3**	9.9	15.7	7.7	15.9	−1.8	**23.9**	−11.8	3.9

Table 3: Structure of the per-query optimal weight set $\mathcal{W}_q^*$ on the evaluation split. Percentages within each (dataset, retriever-pair) row sum to 100%.

Dataset	**Retriever pair**	**Empty (%)**	**Single point (%)**	**1 interval (%)**	**Multi-interval (%)**
MS MARCO	BM25 + MiniLM	30.2	0.7	67.1	2.0
	BM25 + Qwen3	26.2	0.7	70.9	2.1
	RM3 + MiniLM	30.3	1.0	66.9	1.8
	RM3 + Qwen3	26.7	0.7	70.5	2.0
NQ	BM25 + MiniLM	27.1	0.5	71.0	1.3
	BM25 + Qwen3	20.4	0.7	77.3	1.7
	RM3 + MiniLM	29.7	1.0	68.5	0.9
	RM3 + Qwen3	21.5	0.4	77.0	1.0
ACORD	BM25 + MiniLM	0.0	21.1	78.9	0.0
	BM25 + Qwen3	0.0	19.3	77.2	3.5
	RM3 + MiniLM	0.0	28.1	71.9	0.0
	RM3 + Qwen3	0.0	26.3	70.2	3.5
NFCorpus	BM25 + MiniLM	22.6	6.2	70.3	0.9
	BM25 + Qwen3	18.9	5.3	74.9	0.9
	RM3 + MiniLM	23.2	7.1	69.0	0.6
	RM3 + Qwen3	18.3	3.4	77.1	1.2

Unreachable queries ($\mathcal{W}_q^* = \emptyset$). For 18–30% of evaluation queries on MS MARCO, NQ, and NFCorpus, no fusion weight in $[0,1]$ surfaces a relevant document within the top-10. These queries contribute zero to standard RRF, to the oracle, and to every dynamic method; they do not affect the recovery percentage (which is invariant to whether they are included in the average), but they cap the absolute oracle. On MS MARCO BM25+MiniLM, the absolute oracle (0.634) is bounded above by 0.698 ($= 1 - 0.302$) simply because 30.2% of queries score zero in every column; of the 0.698 achievable, the oracle attains 0.634, meaning the average per-query oracle on reachable queries is approximately 0.91 MRR@10. Fusion alone, with the right weight chosen per query, would place a relevant document near rank one. The unreachable queries indicate a limitation of the retrieval stage rather than the fusion stage: no choice of w can recover documents that neither retriever ranks in the top-200. ACORD is the lone exception, with zero unreachable queries across all four retriever pairs, consistent with its smaller, curated corpus.

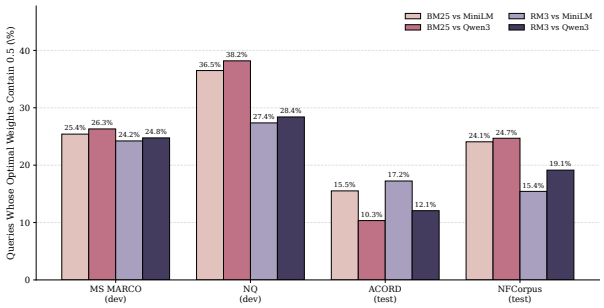

Figure 1: Fraction of evaluation queries for which $0.5 \in \mathcal{I}_q$. On these queries standard RRF attains the per-query oracle by construction; a dynamic predictor that selects $\hat{w}_q \neq 0.5$ can only match or underperform RRF.

RRF-easy queries ($0.5 \in \mathcal{I}_q$). For a substantial fraction of reachable queries the standard RRF weight of 0.5 already lies within the query's optimal interval. For these queries, standard RRF attains the per-query oracle by construction, and any dynamic predictor that selects $\hat{w}_q \notin \mathcal{I}_q$ can only match or underperform RRF. Figure 1 reports this fraction across all configurations on the evaluation split. The fraction is largest on NQ (27.4–38.2%, peaking at 38.2% on BM25+Qwen3) and on MS MARCO (24.2–26.3%), moderate on NFCorpus (15.4–24.7%), and smallest on ACORD (10.3–17.2%). The presence of this subpopulation explains why standard RRF, despite using a fixed query-agnostic weight, remains a competitive baseline: a substantial portion of the test distribution does not require dynamic weighting at all.

Interval-valued optima dominate over single points. Among queries with $\mathcal{W}_q^* \neq \emptyset$, the op-

timal set is far more often an interval than a single point. On MS MARCO and NQ, single-point optima are rare ($< 1\%$ of queries); more than 65% of queries have a contiguous interval of equally optimal weights, and a small but consistent 1–2% of queries have multiple disjoint optimal intervals. This pattern reverses on ACORD, where 19–28% of queries exhibit single-point optima—likely a coarseness artifact of a small corpus with ∼50 graded-relevance documents per query and only 57 evaluation queries, yielding a correspondingly coarse metric landscape over w. The interval-dominant structure has methodological consequences: a predictor trained against a single scalar target $\bar{w}_q$ is penalized for predictions that fall within $\mathcal{I}_q$ but away from the chosen summary statistic, which is a misalignment between the training objective and the underlying task. The ϵ-insensitive satisficing loss (§3.3.2) removes this misalignment but, as Table 2 shows, removes only a small fraction of the recovery gap; the main bottleneck is not the loss form but the predictability of $\mathcal{I}_q$ from the query alone.

5.3 Per-Query Optimal Weight Distribution

Figure 2 reports the distribution of the per-query mean optimal weight for the BM25+Qwen3-Embedding-8B retriever pair, averaging across all $w \in \mathcal{I}_q$ for queries with $\mathcal{W}_q^* \neq \emptyset$; the full set across all four retriever pairs appears in Figure 5 of Appendix D. The cross-dataset pattern tracks the in-domain status of the dense retriever. On MS MARCO and NQ, the distribution is heavily skewed toward low values of w (favoring the dense channel). MS MARCO and NQ are both in-domain for Qwen3-Embedding-8B (and for all-MiniLM-L6-v2, the second dense retriever evaluated in the appendix figure), so the dense channel is unusually strong on these benchmarks. The skew partially explains why the dataset-level mean optimal weight baseline (§3.3.1) already recovers a comparable fraction of the per-query oracle to more expressive predictors on these datasets: when the per-query optimum is consistently low, a single low constant approximates it well. On ACORD, no systematic skew is visible: the distribution is approximately even across $[0, 1]$ with concentrations at the endpoints, roughly 17–25% of queries are optimal at $w = 0$ (dense-only fusion) and 8–12% at $w = 1$ (sparse-only), consistent with ACORD's out-of-domain status for both retrievers in this legal IR setting, where neither channel significantly dominates. On NFCorpus under Qwen3-Embedding-8B, about 15–25% of queries concentrate near $w = 0$ or $w = 0.5$, suggesting two query subpopulations (dense-only optima and balanced-fusion optima); the shape is sensitive to the choice of dense retriever, and under MiniLM the mass spreads more uniformly across the interval (Appendix D).

Table 4: Three latency tiers motivated by inference latency clusters; training requirements or data used for supervision define sub-tiers. Latency reported as mean ± with 95% CI over n within-dataset means. See §4.4 for latency measurement details.

Tier	Sub-tier	Method	Latency (95% CI)	Infra	Training Data	Query-aware	Top-1 Retrieval-aware
T0	–	RRF ($w = 0.5$)	$< 1\,\mu s$	CPU	not required	no	no
T1	T1a	Mean Optimal Weight	$< 1\,\mu s$	CPU	required	no	no
	T1b	Linear Regression	$265 \pm 3\,\mu s$	CPU	required	yes	no
T2	T2a	Small Encoder LM	5.238 ± 0.035 ms	GPU (self-host)	required	yes	no
	T2b	Small Encoder LM	8.461 ± 0.020 ms	GPU (self-host)	required	yes	yes
T3	T3a	DAT (zero-shot)	205.496 ± 8.550 ms	GPU (self-host)/LLM API	not required	yes	yes
	T3b	Few-shot LLM	225.372 ± 8.369 ms	GPU (self-host)/LLM API	required	yes	no

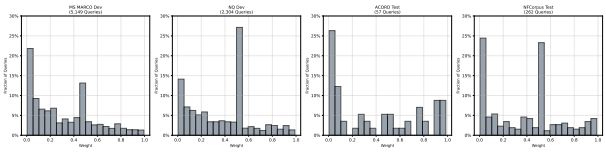

Figure 2: Distribution of the per-query mean optimal weight on the evaluation split for the BM25+Qwen3-Embedding-8B retriever pair, $\frac{1}{|\mathcal{I}_q|}\int_{\mathcal{I}_q} w \, \mathrm{d}w$ for queries with $\mathcal{W}_q^* \neq \emptyset$. The full set across all four retriever pairs is reported in Figure 5 (Appendix D).

6 A Decision Framework for Fusion Method Selection

Per-query inference latencies (§4.4) span sub-μs to 280 ms and cluster into three regimes: Tier 0 and Tier 1 methods incur no per-query forward pass; Tier 2 invokes a fine-tuned encoder ($[5.866, 6.759]$ ms, 95% CI); Tier 3 invokes an LLM ($[209.104, 221.764]$ ms, 95% CI). Figure 3 plots quality versus latency across datasets. We organize the discussion around *three latency tiers* observable in the figure, separating methods by inference runtime and corresponding infrastructure. Within each tier, sub-tiers share similar latency and infrastructure but differ in terms of training requirements or data used for supervision (Table 4).

6.1 Two Prerequisite Checks

Two practical questions determine which methods are suitable for a given setting. *Dataset sensitivity:* a lightweight diagnostic, motivated by §5.1 and §5.2, is to evaluate $\frac{\text{Headroom (Oracle}-\text{RRF)}}{\text{Oracle}}$ (Table 1) and compare it against a threshold δ, the minimum improvement that justifies the cost of a higher-tier method. *Training data availability:* T1, T2, and

T3b all require labeled queries. In the absence of training data, the candidate set reduces to T0 or T3a.

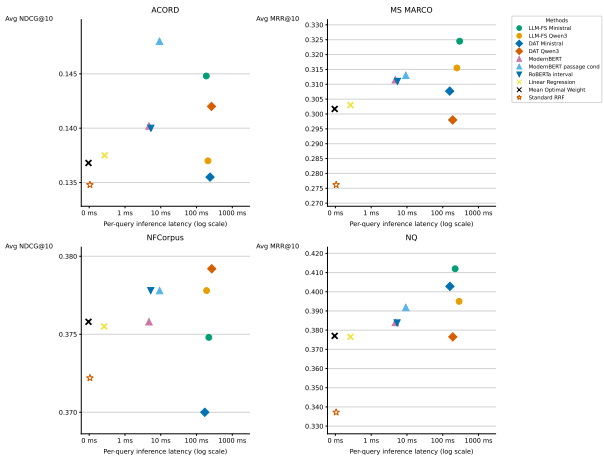

Figure 3: Per-query latency vs. ranking quality across four datasets (MRR@10 for MS MARCO and NQ, nDCG@10 for ACORD and NFCorpus). Each point averages over 2×2 retriever combinations per dataset. See §4.4 for latency measurement details.

6.2 Tier Properties

T0 is appropriate when no labels and no LLM are available, or as a zero-latency method in hybrid retrieval. **T1** applies a learned weight on CPU: T1a captures a small but consistent gain on responsive datasets (see Mean Optimal Weight in Table 2), and T1b performs comparably to T1a. Thus, T1 is a suitable low-latency choice when no GPU is available or when simple model is preferred for deployment. **T2** shares the same latency tier across sub-tiers (6.313 ± 0.446 ms per query), and on responsive datasets recovers much of the achievable gain at a fraction of the LLM inference latency. T2b is competitive across datasets, the strongest Tier 2 method overall, and is a latency-effective choice when a GPU and top-1 passages are available at fusion time. **T3** invokes an LLM: T3a is the only method above T0 that requires no labeled training data, while T3b is the strongest method on NQ and MS MARCO. Within-tier variation in T3 is substantial (see LLM rows of Table 2). With sampling $temperature > 0$, results vary run-to-run, a concern if deterministic behaviour is expected. T3 fits best when fusion is the final stage and the latency budget allows the LLM call; otherwise, the same budget may serve a downstream re-ranker better. Non-retrieval-aware methods (T0–T2a, T3b) depend only on the query.

7 Conclusion

Optimal fusion weights in hybrid retrieval are typically intervals, not points, and the standard RRF weight often already lies inside the per-query optimal interval—a structural explanation for why uniform fusion remains competitive. Substantial headroom above standard RRF remains uncaptured on BM25-based MS MARCO and NQ configurations, even by LLM-based predictors, indicating that the bottleneck is the predictability of the per-query optimum from the query alone. The evaluated methods organize into a tiered decision framework that maps practitioner constraints (labels, GPU access, latency budget) to an appropriate method class.

Limitations

We note the following scope conditions, which point to natural extensions of our study.

- We evaluate on English benchmarks, where sparse and dense retrievers are both mature; the specific weight intervals may shift in multilingual or low-resource settings, though the set-valued characterization remains language-agnostic.
- We study one sparse–dense pair (k=2); the formulation extends directly to k>2, with the optimum becoming a region in the $(k-1)$-simplex, which we leave to future work.
- We do not enforce greedy or temperature-zero decoding, so absolute LLM-method numbers may shift slightly across runs; averaging over the full query set keeps the relative ordering of method classes stable.

References

Nasreen Abdul-Jaleel, James Allan, W Bruce Croft, Fernando Diaz, Leah Larkey, Xiaoyan Li, Mark D Smucker, and Courtney Wade. 2004. Umass at trec 2004: Novelty and hard.

Negar Arabzadeh, Chuan Meng, Mohammad Aliannejadi, and Ebrahim Bagheri. 2024. Query performance prediction: Techniques and applications in modern information retrieval. In *Proceedings of the 2024 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*, SIGIR-AP 2024, page 291–294, New York, NY, USA. Association for Computing Machinery.

Javed A Aslam and Mark Montague. 2001. Models for metasearch. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 276–284.

Alec Berntson. 2023. Azure ai search: Outperforming vector search with hybrid retrieval and reranking. *Microsoft Community Hub. Accessed on August*, 19:2025.

Vera Boteva, Demian Gholipour, Artem Sokolov, and Stefan Riezler. 2016. A full-text learning to rank dataset for medical information retrieval. In *European Conference on Information Retrieval*, pages 716–722. Springer.

Sebastian Bruch, Siyu Gai, and Amir Ingber. 2023. An analysis of fusion functions for hybrid retrieval. *ACM Trans. Inf. Syst.*, 42(1).

Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In *Findings of the association for computational linguistics: ACL 2024*, pages 2318–2335.

Gordon V. Cormack, Charles L A Clarke, and Stefan Buettcher. 2009. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, page 758–759, New York, NY, USA. Association for Computing Machinery.

Keyan Ding, Kede Ma, Shiqi Wang, and Eero P Simoncelli. 2020. Image quality assessment: Unifying structure and texture similarity. *IEEE transactions on pattern analysis and machine intelligence*, 44(5):2567–2581.

Wenxuan Dong, Mingguang Diao, and Meiqi Yang. 2026. An adaptive multi-source retrieval-augmented generation framework integrating query complexity awareness and confidence-aware fusion. *Applied Sciences*, 16(5):2495.

ESaO Etsioni. 1995. Multi-service search and comparison using the metacrawler. In *4th International World Wide Web Conference*.

Arthur E. Hoerl and Robert W. Kennard. 1970. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67.

Hsin-Ling Hsu and Jengnan Tzeng. 2025. Dat: Dynamic alpha tuning for hybrid retrieval in retrieval-augmented generation. *arXiv preprint arXiv:2503.23013*.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*.

Qiao Jin, Zifeng Wang, Charalampos S Floudas, Fangyuan Chen, Changlin Gong, Dara Bracken-Clarke, Elisabetta Xue, Yifan Yang, Jimeng Sun, and Zhiyong Lu. 2024. Matching patients to clinical trials with large language models. *Nature communications*, 15(1):9074.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pages 6769–6781.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, and 1 others. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Victor Lavrenko and W Bruce Croft. 2017. Relevance-based language models. In *ACM SIGIR Forum*, volume 51, pages 260–267. ACM New York, NY, USA.

Yongqi Li, Nan Yang, Liang Wang, Furu Wei, and Wenjie Li. 2024. Learning to rank in generative retrieval. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press.

Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: A Python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proceedings of the 44th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021)*, pages 2356–2362.

Alexander H Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, and 1 others. 2026. Ministral 3. *arXiv preprint arXiv:2601.08584*.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, Dublin, Ireland and Online. Association for Computational Linguistics.

Tie-Yan Liu. 2009. Learning to rank for information retrieval. *Found. Trends Inf. Retr.*, 3(3):225–331.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ji Ma, Ivan Korotkov, Keith B Hall, and Ryan T McDonald. 2020. Hybrid first-stage retrieval models for biomedical literature. In *CLEF (Working Notes)*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, and 1 others. 2011. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830.

Zackary Rackauckas. 2024. Rag-fusion: a new take on retrieval-augmented generation. *arXiv preprint arXiv:2402.03367*.

S Robertson, Steve Walker, Susan Jones, and MHB GATFORD. 1994. Okapi at 3. In *Proceedings of the 3rd Text REtrieval Conference (-3)*, pages 109–126.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 4(1-2):1–174.

Kunal Sawarkar, Abhilasha Mangal, and Shivam Raj Solanki. 2024. Blended rag: Improving rag (retriever-augmented generation) accuracy with semantic search and hybrid query-based retrievers. In *2024 IEEE 7th International Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 155–161.

Erik Selberg and Oren Etzioni. 1997. The metacrawler architecture for resource aggregation on the web. *IEEE expert*, 12(1):11–14.

Sentence-Transformers. 2021. all-minilm-l6-v2: Sentence embeddings model. https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2.

Herbert A Simon. 1955. A behavioral model of rational choice. *The quarterly journal of economics*, pages 99–118.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. Is chatgpt good at search? investigating large language models as re-ranking agents. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 14918–14937.

Raphael Tang, Crystina Zhang, Xueguang Ma, Jimmy Lin, and Ferhan Ture. 2024. Found in the middle: Permutation self-consistency improves listwise ranking in large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2327–2340, Mexico City, Mexico. Association for Computational Linguistics.

Steven H Wang, Maksim Zubkov, Kexin Fan, Sarah Harrell, Yuyang Sun, Wei Chen, Andreas Plesner, and Roger Wattenhofer. 2025. Acord: An expert-annotated retrieval dataset for legal contract drafting. *arXiv preprint arXiv:2501.06582*.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Griffin Thomas Adams, Jeremy Howard, and Iacopo Poli. 2025. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2526–2547, Vienna, Austria. Association for Computational Linguistics.

Shi Yu, Chaoyue Tang, Bokai Xu, Junbo Cui, Junhao Ran, Yukun Yan, Zhenghao Liu, Shuo Wang, Xu Han, Zhiyuan Liu, and Maosong Sun. 2025. Visrag: Vision-based retrieval-augmented generation on multi-modality documents. In *International Conference on Learning Representations*, volume 2025, pages 21074–21098.

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

A Dataset Statistics

Table 5 reports the number of queries in each split, the total passage-collection size, and the graded-relevance scale used in the qrels for each dataset in our evaluation (Section 4).

B Training Details

B.1 Linear Regression

We implement the linear regression baseline (Section 3.3.2) with scikit-learn (Pedregosa et al., 2011). Queries are vectorized using TfidfVectorizer with word-level unigrams through 4-grams ngram_range=(1,4)), English stop-word removal, sublinear term-frequency scaling (sublinear_tf=True), and document-frequency cutoffs `min_df` $= 2$ and `max_df` $= 0.95$; the vocabulary

Dataset	#Train	#Dev	#Test	#Passages	Relevance scale
MS MARCO	502,939	6,980	43	8,841,823	$\{0, 1\}$ (train/dev); $\{0, 1, 2, 3\}$ (test)
NFCorpus	2,590	324	323	3,633	$\{0, 1, 2\}$
ACORD	51	6	57	3,931	$\{0, 1, 2, 3, 4\}$
NQ	307,373	7,830	7,842	3,040,270	$\{0, 1\}$ (train); $\{0, 1, 2, 3, 4\}$ (dev/test)

Table 5: Per-dataset statistics: number of queries in each split, total passage-collection size, and graded-relevance scale used in the qrels.

size is unbounded. The ridge regressor (Ridge) is trained with regularization strength $\alpha = 100$, a fitted intercept, and scikit-learn's default solver selection. Predictions are clipped to $[0.01, 1.0]$ at inference time.

B.2 RoBERTa

We fine-tune `roberta-large` with the Hugging Face Transformers Trainer. The regression head is a single linear layer over the pooler output, with dropout 0.01 on both the attention and pooled representations. Queries are tokenized to a maximum length of 64. Training uses AdamW at learning rate 2×10^{-5} with a linear schedule, 500 warmup steps, weight decay 0.001, gradient-norm clipping at 0.5, per-device batch size 256, and `fp16` mixed precision, for up to 10 epochs. We evaluate every 200 steps, retain the best checkpoint by eval loss, and apply early stopping with patience 10. The interval variant optimizes the satisficing loss $\mathcal{L}_{\text{sat}}$ (Section 3.3.2) against the target interval $[l_q, r_q]$; the point variant uses MSE against the scalar target $\bar{w}_q$ and shares all other hyperparameters. The random seed is 42.

B.3 ModernBERT

We fine-tune `answerdotai/ModernBERT-large` in two input variants. The *query-only* variant takes the query alone, mirroring the RoBERTa setup. The *passage-conditioned* variant takes `[CLS] Query [SEP] Sparse Top-1 [SEP] Dense Top-1 [EOS]` (Section 4.3), with the sparse and dense top-1 passages selected from the same retriever indices used at evaluation time. Both wrap the encoder with a single linear regression head over the `[CLS]` representation (with dropout 0.01) and are trained with the satisficing loss $\mathcal{L}_{\text{sat}}$ against the interval target $[l_q, r_q]$. Optimization, schedule, and early-stopping settings match RoBERTa (Appendix B.2): AdamW at 2×10^{-5} with a linear schedule and 500 warmup steps, weight decay 0.001, gradient-norm clipping at 0.5, `fp16` mixed precision, up to 10 epochs, evaluation every 200 steps, best-by-eval-loss checkpoint with early-stopping patience 10, and seed 42. The two variants differ only in sequence length and batch size: the query-only variant uses maximum length 64 and per-device batch size 256, while the passage-conditioned variant uses maximum length 1024 to accommodate the query plus two passages and reduces the per-device batch size to 8 to fit in memory.

C In-Context LLM Prediction Details

C.1 Exemplar Retrieval

For a target query q, we retrieve similar training queries from two independent indices built over the training-split query texts and union the two top-k lists.

Dense index. Training queries are encoded with `Qwen/Qwen3-Embedding-8B` (Zhang et al., 2025) via the SentenceTransformers interface (`trust_remote_code=True`, `normalize_embeddings=True`); the corpus matrix is kept on GPU in `bfloat16`. At query time, q is encoded with the same model and identical normalization, and we score every training-query embedding by cosine similarity, returning the top $k = 5$. Because both sides are L2-normalized, cosine similarity reduces to the inner product.

Sparse index. The same training-query texts are indexed with the `bm25s` library[1] at default tokenization (lowercasing and whitespace splitting) and default BM25 parameters, and we return the top $k = 5$ for q.

Union and exemplar payload. The two top-5 lists are deduplicated by `query_id` (yielding up to 10 unique exemplars). For each retained exemplar we attach (i) its optimal weight set $\mathcal{I}_{q_{\text{nbr}}}$ in interval form and (ii) the corresponding metric value (MRR@10 or nDCG@10) achieved by that exemplar's best-weight configuration on the training split. The pool is restricted to the training split for every evaluation.

[1] https://github.com/xhluca/bm25s

C.2 LLM Generation

We use Qwen3-32B and Ministral-3-14B-Instruct-2512. Both are loaded with Hugging Face `device_map="auto"` and rendered prompts via the model-specific chat template (`add_generation_prompt=True`); tokenizers are configured for batched inference with left padding and `pad_token = eos_token`.

Qwen3-32B. Loaded with AutoModelForCausalLM and AutoTokenizer. We use 4-bit weight quantization via `BitsAndBytesConfig (load_in_4bit=True, bnb_4bit_compute_dtype=torch.bfloat16)` so that matmuls run in bf16. The chat template is invoked with `enable_thinking=False` to suppress Qwen3's reasoning trace, since the system prompt forbids non-numeric output. Decoding uses `max_new_tokens=10`, `temperature=0.1`, `do_sample=True`, and `pad_token_id=eos_token_id`.

Ministral-3-14B-Instruct-2512. Loaded with the multimodal Mistral3ForConditionalGeneration class paired with the MistralCommonBackend tokenizer. The checkpoint ships in FP8; we load it with `FineGrainedFP8Config(dequantize=True)`. Because the model accepts multimodal inputs, we filter the tokenizer output to retain only input_ids and attention_mask before calling `generate()` (no pixel inputs are supplied). Decoding uses `max_new_tokens=10` with `temperature=0.1` and no explicit `do_sample` flag, which falls back to greedy decoding under Hugging Face defaults.

Invalid-output handling. The system message constrains the model to a single float in $[0, 1]$, but we still post-process defensively. The generated string is searched with the regular expression `\d*\.\d+` for a decimal literal; if none is found we fall back to a bare integer match (`\b[01]\b`). Successful matches are clipped to $[0, 1]$ and rounded to two decimals to match the resolution of the grid search. Any parse failure defaults to $\hat{w}_q = 0.50$.

D Per-Query Optimal Weight Distribution: Full Results

Figure 5 reports the distribution of the per-query mean optimal weight on the evaluation split across all four retriever pairs (BM25 and RM3 paired with all-MiniLM-L6-v2 and Qwen3-Embedding-8B). The main-text figure (Figure 2) shows the BM25+Qwen3-Embedding-8B row; the remaining three rows are included here.

E Licenses and Terms of Use

All datasets, models, and software used in this work are publicly released for research use and, to our knowledge, do not contain personally identifying information about private individuals or offensive content. Our use is consistent with their stated licenses and terms.

Datasets. MS MARCO is distributed by Microsoft under non-commercial research-only terms; the dataset is provided "as is" and we use it solely to study retrieval methods.[2] Natural Questions is released under CC BY-SA 3.0. NFCorpus is publicly released for non-commercial research use by its authors. ACORD is released under CC BY 4.0.

Retrieval software. BM25 and RM3 are obtained via Pyserini (Apache 2.0). The `bm25s` library used for exemplar retrieval is released under the MIT license.

Models. The dense retrievers all-MiniLM-L6-v2 and Qwen3-Embedding-8B are both released under Apache 2.0. The predictor backbones RoBERTa-large (MIT) and ModernBERT-large (Apache 2.0) are fine-tuned end-to-end. The in-context LLMs Qwen3-32B and Ministral-3-14B-Instruct-2512 are both released under Apache 2.0 and accessed via local inference. All model checkpoints are obtained from the Hugging Face Hub and used in accordance with their model cards.

Supporting libraries. scikit-learn (BSD-3-Clause), Hugging Face Transformers, Datasets, and SentenceTransformers (all Apache 2.0), and PyTorch (BSD-3-Clause) are used under their respective open-source licenses.

Our experimental use of every artifact above is consistent with its intended research purpose and license terms.

F AI Assistance Disclosure

We used general-purpose AI assistants in a limited, supervised capacity: grammar and phrasing polish during writing, brainstorming during research discussions.

[2] https://microsoft.github.io/msmarco/

LLM Weight Prediction Prompt Template

```
System:

You are an expert search system optimizer.
Your task is to predict the optimal weight
for combining a sparse retriever and a dense
retriever for a given search query.


STRICT CONSTRAINT:

You must output ONLY a single float number
between 0.00 and 1.00. Do not include any
reasoning, markdown formatting, explanation,
or conversational text. Just the number.


### Context (Similar Historical Queries):

For each historical query, you are provided
its score and its optimal weights (a list
of weights that yielded the best score,
representing the weight given to the sparse
retriever).


[1] Query: "[Example Query 1]"
Optimal Weights: [0.12, 0.25]
Score (nDCG@10): 0.842


...


[10] Query: "[Example Query 10]"
Optimal Weights: [0.75, 0.90]
Score (nDCG@10): 0.913


### Task:

Based on the optimal weights of the similar
context queries, predict a single optimal
weight for the Target Query.


### Target Query:

Query: {{target_query}}

Predicted Optimal Weight:
```

Figure 4: Few-shot prompt for LLM-based weight prediction.

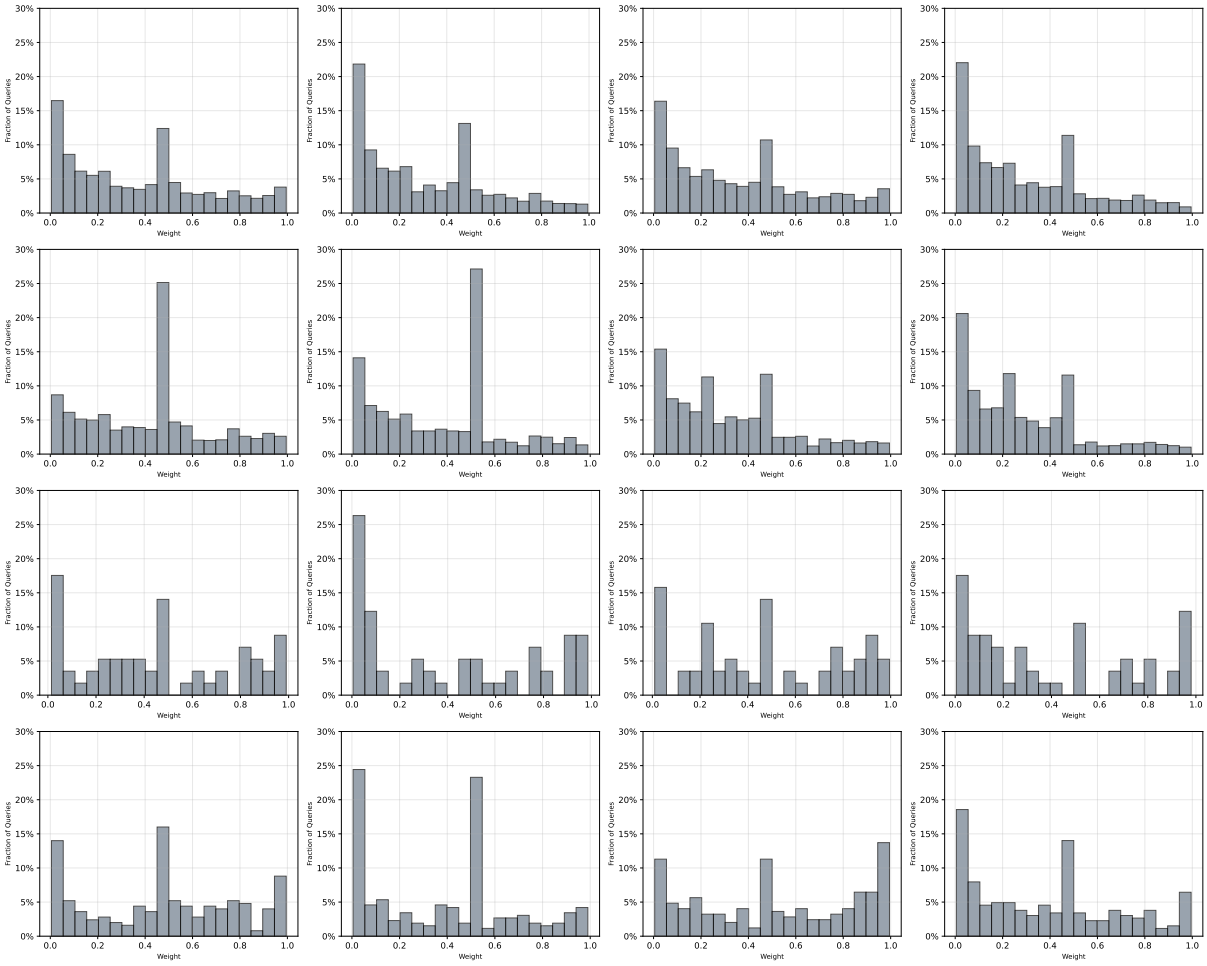

Figure 5: Distribution of the per-query mean optimal weight on the evaluation split across all four retriever pairs, $\frac{1}{|\mathcal{I}_q|}\int_{\mathcal{I}_q} w \, \mathrm{d}w$ for queries with $\mathcal{W}_q^* \neq \emptyset$.