A Comparative Analysis of Sparse, Dense, and Hybird Ranking Pipelines for MS MARCO

Aaron Bengochea, Timothy Cao
(ab6503, tc4088)@nyu.edu
CS 6913: Web Search Engines – Fall 2025
NYU Tandon School of Engineering

November 4, 2025

Project Repository: https://github.com/timothycao/search-systems

Abstract

Modern information retrieval systems increasingly rely on hybrid architectures that combine sparse lexical retrieval, dense vector search, and neural reranking. In this work, we conduct a systematic comparison of several such components to quantify their relative and complementary strengths. Our baseline systems consist of a BM25 inverted index and an HNSW approximate nearest neighbor graph, each producing an initial ranked list of candidate passages. On top of these retrieval outputs, we evaluate two rank fusion based reranking strategies, Linear Score Fusion (LSF) and Reciprocal Rank Fusion (RRF), which merge rankings from the sparse and dense retrieval pipelines.

To further assess the impact of semantic similarity modeling, we implement a BERT based bi-encoder reranker that performs cascading reranking over the outputs of all previous systems: BM25, HNSW, LSF, and RRF. This allows us to isolate and measure the incremental benefit provided by neural embeddings when applied on top of both single system rankings and fusion derived rankings. All systems are evaluated on MS MARCO development and evaluation subsets using standard retrieval metrics, and we additionally analyze performance as a function of query length.

Our results demonstrate clear performance differences across retrieval paradigms and highlight the value of layered reranking in hybrid pipelines. The comparative analysis provides insight into when sparse, dense, fusion-based, and neural rerankers excel, offering practical guidance for designing high-performance retrieval architectures.

1 Introduction

Information retrieval has evolved from purely lexical term-matching approaches to hybrid architectures that integrate sparse, dense, and neural components. Classical systems such as BM25 offer strong performance on keyword-oriented queries and provide efficient large scale retrieval, yet they struggle in cases where lexical overlap is weak. Conversely, dense vector retrieval methods, such as those powered by neural encoders and approximate nearest-neighbor (ANN) search, capture semantic similarity but may underperform when exact term specificity is required. As a result, modern retrieval pipelines increasingly combine multiple ranking signals to balance lexical precision with semantic recall.

This work investigates the effectiveness of such hybrid strategies within the passage retrieval setting. We begin by implementing two baseline retrieval systems: a BM25 inverted index and an HNSW-based dense retrieval engine. These form our primary sources of candidate passages, each contributing complementary retrieval characteristics. Building on these baselines, we explore two fusion based reranking approaches, Linear Score Fusion (LSF) and Reciprocal Rank Fusion (RRF), which merge BM25 and HNSW rankings to better exploit lexical and semantic evidence. Beyond fusion methods, we integrate a BERT based bi-encoder reranker that performs cascading reranking over the outputs of the BM25, HNSW, LSF, and RRF systems in order to evaluate the incremental benefit of neural semantic modeling.

A second focus of this study concerns query characteristics. Because query length often correlates with user intent and linguistic ambiguity, we analyze retrieval performance across short, medium, and long queries to understand how different ranking paradigms behave under varying query complexity.

Across both development and evaluation subsets of the MS MARCO dataset, we systematically measure the performance of each retrieval and reranking approach using standard IR metrics. Our contributions are as follows:

1. We implement a modular, retrieval pipeline supporting sparse, dense, fusion, and neural reranking methods.
2. We provide a controlled comparison of BM25 and HNSW baselines, highlighting their complementary strengths.
3. We evaluate two fusion rerankers in LSF and RRF to quantify the gains achievable through rank and score combination.
4. We integrate a bi-encoder neural reranker and assess its cascading impact when applied on top of all prior ranking systems.

Overall, this work offers an empirical perspective on the interactions among sparse retrieval, dense retrieval, fusion based methods, and neural rerankers. The findings aim to provide a practical analysis for the designing high performance retrieval pipelines that balance efficiency, semantic robustness, and reranking effectiveness.

2 Background and Related Work

Modern passage retrieval systems are increasingly constructed from a combination of sparse, dense, and neural ranking components. Each paradigm captures different aspects of query document relevance, and extensive prior work has shown that the strengths of one approach often compensate for the weaknesses of another. This section reviews the foundational retrieval methods relevant to our hybrid architecture.

2.1 Sparse Retrieval and BM25

Sparse lexical retrieval methods represent documents as sets of discrete terms and rely on exact token overlap between queries and documents. Among these methods, BM25 [7] remains one of the most widely used ranking functions due to its balance of simplicity, interpretability, and retrieval effectiveness. BM25 relies on term frequency, inverse document frequency, and length normalization to produce relevance scores. Despite its strong performance on keyword dense queries, sparse retrieval fails to capture semantic similarity when relevant passages lack direct lexical overlap with the query.

2.2 Dense Vector Retrieval and HNSW

Dense retrieval methods address the semantic gap by encoding queries and documents into continuous vector spaces using neural embedding models [2]. Similarity between a query and a document is typically computed via cosine similarity or dot product. However, performing exact nearest-neighbor search over millions of vectors is computationally expensive. To mitigate this, ANN structures such as Hierarchical Navigable Small World (HNSW) graphs [3] enable fast search with high recall. HNSW search is guided by a multi layer graph structure that approximates global similarity while maintaining efficient local exploration. Dense retrieval tends to outperform sparse methods on paraphrased or semantically complex queries but may struggle with specificity and rare terms.

2.3 Fusion-Based Reranking

Because sparse and dense retrieval methods exhibit complementary strengths, hybrid fusion techniques have become an effective approach for improving retrieval robustness. Rank based fusion methods, such as Reciprocal Rank Fusion (RRF) [1], combine rankings from multiple systems using the reciprocal of their rank positions, rewarding documents that appear consistently near the top of multiple ranking lists. Score based methods, such as Linear Score Fusion (LSF), combine raw similarity scores through weighted interpolation. Fusion has the ability to improve both precision and recall by aggregating independent retrieval signals, and prior studies have shown that even simple fusion strategies often outperform more complex single model systems.

2.4 Neural Reranking: Bi-Encoders and Cross-Encoders

Beyond initial retrieval, neural reranking models further refine candidate lists by directly comparing queries and passages. Cross-encoders [5] process the query and document jointly and typically achieve the strongest results but are computationally expensive. Bi-encoders [6], in contrast, encode queries and documents independently and compute similarity via vector operations. Although bi-encoders trade some accuracy for efficiency, they are significantly faster and well suited for reranking the top-k candidates produced by sparse or dense retrieval systems. Their effectiveness in hybrid architectures has been demonstrated extensively, especially when applied in a cascading rerank pipeline.

2.5 Hybrid Retrieval Architectures

Recent research has shown that hybrid retrieval pipelines combining sparse, dense, and neural components consistently achieve SOTA performance on benchmarks such as MS MARCO [4]. Sparse retrieval provides lexical precision, dense retrieval contributes semantic generalization, fusion creates a more stable ranking across heterogeneous signals, and neural reranking offers final stage semantic refinement. Our work follows this hybrid design philosophy, implementing and comparing each component to quantify its contribution to overall ranking performance.

3 System Implementation and Execution Framework

This section describes how the retrieval and reranking systems in our study are constructed, executed, and evaluated. Rather than revisiting the conceptual foundations of sparse, dense, and neural retrieval. We focus here on the practical workflow for running each component of the pipeline. All systems are accessed through a unified command-line interface that allows for consistent experimentation across the development (dev) and evaluation (eval1 and eval2) subsets of the MS MARCO dataset. The dev set provides binary relevance labels, while the eval sets contain graded relevance. These differences inform which evaluation metrics are applicable in later sections.

3.1 Building the Retrieval Indexes

Before querying or reranking can be performed, the BM25 and HNSW indexes must be constructed. This is handled through the `scripts.build` entry point, which prepares the inverted index for sparse retrieval and the dense vector index for approximate nearest-neighbor search.

BM25 Index Construction. The BM25 system constructs a positional inverted index from the MS MARCO passage collection, normalizes document lengths, and serializes term statistics needed for scoring. The index is built via:

```
python -m scripts.build --system bm25
```

HNSW Index Construction. The HNSW system generates dense embeddings for all passages using a pretrained encoder and adds them to a Hierarchical Navigable Small World graph. Vector normalization and FAISS indexing parameters are applied during construction. The index is built via:

```
python -m scripts.build --system hnsw
```

In both cases, once the indexes are built, they remain static and are reused across all experiments.

3.2 Running BM25 and HNSW Retrieval

After building the indexes, top-k retrieval results may be generated for any query set. The `scripts.run` interface loads the appropriate index and applies the system's scoring function.

BM25 Retrieval

```
python -m scripts.run --system bm25 --qrels dev    --save bm25_dev
python -m scripts.run --system bm25 --qrels eval1  --save bm25_eval1
python -m scripts.run --system bm25 --qrels eval2  --save bm25_eval2
```

HNSW Retrieval

```
python -m scripts.run --system hnsw --qrels dev    --save hnsw_dev
python -m scripts.run --system hnsw --qrels eval1  --save hnsw_eval1
python -m scripts.run --system hnsw --qrels eval2  --save hnsw_eval2
```

The `--qrels` flag determines whether queries come from the dev set or one of the eval sets, ensuring that retrieval outputs align with the relevance labels used in downstream evaluation.

3.3 Fusion Based Reranking: LSF and RRF

Once BM25 and HNSW runs are available, we may then apply fusion based reranking. Both fusion methods take the two ranked lists as input and produce a single reranked output.

RRF Rerank

```
python -m scripts.run --system rrf --qrels dev   --targets bm25_dev hnsw_dev     --save rrf_dev
python -m scripts.run --system rrf --qrels eval1 --targets bm25_eval1 hnsw_eval1 --save rrf_eval1
python -m scripts.run --system rrf --qrels eval2 --targets bm25_eval2 hnsw_eval2 --save rrf_eval2
```

LSF Rerank

```
python -m scripts.run --system lsf --qrels dev   --targets bm25_dev hnsw_dev     --save lsf_dev
python -m scripts.run --system lsf --qrels eval1 --targets bm25_eval1 hnsw_eval1 --save lsf_eval1
python -m scripts.run --system lsf --qrels eval2 --targets bm25_eval2 hnsw_eval2 --save lsf_eval2
```

The `--targets` argument specifies the two runs to be fused, and the pipeline automatically loads the corresponding BM25 and HNSW outputs.

3.4 Neural Reranking with a Bi-Encoder

The final stage of the pipeline employs a BERT based bi-encoder to rerank the top-k documents returned by any retrieval or fusion system. The bi-encoder computes semantic similarity between query and passage embeddings and produces a new ranking.

Neural rerank using BM25 run:

```
python -m scripts.run --system biencoder --qrels dev   --targets bm25_dev   --save bi_bm25_dev
python -m scripts.run --system biencoder --qrels eval1 --targets bm25_eval1 --save bi_bm25_eval1
python -m scripts.run --system biencoder --qrels eval2 --targets bm25_eval2 --save bi_bm25_eval2
```

Neural rerank using HNSW run:

```
python -m scripts.run --system biencoder --qrels dev   --targets hnsw_dev   --save bi_hnsw_dev
python -m scripts.run --system biencoder --qrels eval1 --targets hnsw_eval1 --save bi_hnsw_eval1
python -m scripts.run --system biencoder --qrels eval2 --targets hnsw_eval2 --save bi_hnsw_eval2
```

Neural rerank using RRF run:

```
python -m scripts.run --system biencoder --qrels dev   --targets rrf_dev   --save bi_rrf_dev
python -m scripts.run --system biencoder --qrels eval1 --targets rrf_eval1 --save bi_rrf_eval1
python -m scripts.run --system biencoder --qrels eval2 --targets rrf_eval2 --save bi_rrf_eval2
```

Neural rerank using LSF run:

```
python -m scripts.run --system biencoder --qrels dev   --targets lsf_dev   --save bi_lsf_dev
python -m scripts.run --system biencoder --qrels eval1 --targets lsf_eval1 --save bi_lsf_eval1
python -m scripts.run --system biencoder --qrels eval2 --targets lsf_eval2 --save bi_lsf_eval2
```

This cascading design enables us to assess the incremental value of neural reranking on top of both single-system retrieval and fusion-derived retrieval.

3.5 Comprehensive Evaluation

Our framework includes a dedicated tool for analyzing ranking performance at both the global level and as a function of query length. This script, `bucket_evaluation.py`, extends the standard evaluation process by performing two forms of assessment: overall aggregated evaluation across all queries, and bucketed evaluation based on query length categories. This enables us to identify whether different retrieval and reranking systems exhibit distinct strengths on short, medium, or long queries, and whether certain hybrid combinations offer more robust performance across query types.

3.5.1 Dataset Handling

All experiments are conducted on three MS MARCO subsets, each with distinct relevance formats and associated evaluation metrics:

- **dev:** Contains binary relevance labels (0 or 1).
- **eval1** and **eval2:** Contain graded relevance labels (0–3).

The evaluation script automatically loads the appropriate query set and relevance file based on the `--qrels` argument, ensuring consistent metric computation across systems.

3.5.2 Query Length Bucketing

To support fine grained analysis, each query is tokenized and assigned to one of three buckets:

- **Short queries**: 1–3 tokens,
- **Medium queries**: 4–6 tokens,
- **Long queries**: 7 or more tokens.

After a retrieval or reranking system produces its output, the `bucket_evaluation.py` script performs:

1. **Aggregated evaluation** across all queries in the set, matching the standard metrics used throughout our study.
2. **Bucketed evaluation** that computes the same metrics separately for short, medium, and long queries, revealing and storing query length specific performance.

3.5.3 Evaluation Metrics

The specific metrics applied depend on whether binary or graded relevance labels are available:

- **For the dev set (binary relevance):**
  - MRR@10
  - Recall@100
  - MAP
- **For the eval1 and eval2 sets (graded relevance 0–3):**
  - MRR@10
  - Recall@100
  - NDCG@10
  - NDCG@100

3.5.4 Typical Evaluation Invocations

Once a system has produced its retrieval or reranking output, the `bucket_evaluation.py` script may be invoked to compute overall aggregated and bucketed performance metrics. Below we show typical usage examples for each retrieval and reranking system.

BM25

```
python -m scripts.bucket_evaluation --system bm25 --qrels dev   --run bm25_dev   --save res_bm25_dev
python -m scripts.bucket_evaluation --system bm25 --qrels eval1 --run bm25_eva1  --save res_bm25_eval1
python -m scripts.bucket_evaluation --system bm25 --qrels eval2 --run bm25_eval2 --save res_bm25_eval2
```

HNSW

```
python -m scripts.bucket_evaluation --system hnsw --qrels dev   --run hnsw_dev   --save res_hnsw_dev
python -m scripts.bucket_evaluation --system hnsw --qrels eval1 --run hnsw_eva1  --save res_hnsw_eval1
python -m scripts.bucket_evaluation --system hnsw --qrels eval2 --run hnsw_eval2 --save res_hnsw_eval2
```

LSF

```
python -m scripts.bucket_evaluation --system lsf --qrels dev   --run lsf_dev   --save res_lsf_dev
python -m scripts.bucket_evaluation --system lsf --qrels eval1 --run lsf_eva1  --save res_lsf_eval1
python -m scripts.bucket_evaluation --system lsf --qrels eval2 --run lsf_eval2 --save res_lsf_eval2
```

RRF

```
python -m scripts.bucket_evaluation --system rrf --qrels dev   --run rrf_dev   --save res_rrf_dev
python -m scripts.bucket_evaluation --system rrf --qrels eval1 --run rrf_eva1  --save res_rrf_eval1
python -m scripts.bucket_evaluation --system rrf --qrels eval2 --run rrf_eval2 --save res_rrf_eval2
```

Bi-Encoder

```
DEV:
python -m scripts.bucket_evaluation --system biencoder --qrels dev --run bi_bm25_dev --save res_bi_bm25_dev
python -m scripts.bucket_evaluation --system biencoder --qrels dev --run bi_hnsw_dev --save res_bi_hnsw_dev
python -m scripts.bucket_evaluation --system biencoder --qrels dev --run bi_lsf_dev  --save res_bi_lsf_dev
python -m scripts.bucket_evaluation --system biencoder --qrels dev --run bi_rrf_dev  --save res_bi_rrf_dev

EVALUATION 1:
python -m scripts.bucket_evaluation --system biencoder --qrels eval1 --run bi_bm25_eval1 --save res_bi_bm25_eval1
python -m scripts.bucket_evaluation --system biencoder --qrels eval1 --run bi_hnsw_eval1 --save res_bi_hnsw_eval1
python -m scripts.bucket_evaluation --system biencoder --qrels eval1 --run bi_lsf_eval1  --save res_bi_lsf_eval1
python -m scripts.bucket_evaluation --system biencoder --qrels eval1 --run bi_rrf_eval1  --save res_bi_rrf_eval1


EVALUATION 2:
python -m scripts.bucket_evaluation --system biencoder --qrels eval2 --run bi_bm25_eval2 --save res_bi_bm25_eval2
python -m scripts.bucket_evaluation --system biencoder --qrels eval2 --run bi_hnsw_eval2 --save res_bi_hnsw_eval2
python -m scripts.bucket_evaluation --system biencoder --qrels eval2 --run bi_lsf_eval2  --save res_bi_lsf_eval2
python -m scripts.bucket_evaluation --system biencoder --qrels eval2 --run bi_rrf_eval2  --save res_bi_rrf_eval2
```

The output includes both a console summary and a saved text file under `results/<system>/`, containing:

1. Aggregated performance
2. Total number of queries and count per bucket
3. Performance for each query length bucket
4. total number of queries and count per bucket.

This bucketed analysis plays an important role in our study. Because sparse, dense, fusion based, and neural reranking systems respond differently to query structure, query length segmentation reveals trends that are not visible in aggregate metrics alone. The bucket evaluation tool enables us to quantify these effects systematically across all retrieval and reranking configurations.

3.6 Reproducible Workflow

The system is organized as a fully modular and reproducible pipeline, allowing every retrieval and reranking experiment to be executed in a structured and repeatable manner. The workflow proceeds in four stages, each invoked through a consistent command-line interface.

1. Index Construction. The pipeline begins with the construction of the BM25 inverted index and the HNSW dense vector index. These indexes are built once using `scripts.build` and remain fixed throughout all experiments. This ensures that all retrieval and reranking results are generated on identical underlying representations.

2. Retrieval and Initial Ranking. After the indexes are built, BM25 and HNSW retrieval runs are generated for the dev, eval1, and eval2 query sets. Each run produces a top-k ranked list of passage candidates for every query. These ranked outputs serve as the basis for all downstream reranking and fusion methods.

3. Fusion Based Reranking. Once the BM25 and HNSW runs are available, fusion-based reranking is applied. Both RRF and LSF consume the two ranked lists and produce a fused ranking that combines the strengths of sparse and dense retrieval.

4. Cascading Neural Reranking. The final stage of the pipeline applies a BERT based bi-encoder to cascadingly rerank the top-k results produced by each retrieval and fusion system. This means that BM25, HNSW, RRF, and LSF outputs are all independently reranked using the bi-encoder, enabling the study to measure the incremental value of neural similarity modeling across all retrieval and fusion reranking configurations. This full pipeline is evaluated on the dev, eval1, and eval2 sets, allowing for both binary relevance and graded relevance comparisons.

5. Evaluation and Analysis. All retrieval and reranked outputs are saved in TREC style format within `runs/<system>/`, ensuring traceability and reproducibility. The resulting ranked lists can then be analyzed using the comprehensive evaluation tool, which saves both aggregated and bucketed performance summaries under `results/<system>/`. Because the entire pipeline is command driven and modular, experiments can be reproduced systematically or extended with additional reranking algorithms in future work.

3.7 Experimental Assumptions

All experiments in this study are conducted under a consistent set of fixed parameters across all retrieval and reranking systems. This ensures that performance differences can be attributed to the retrieval or reranking strategy itself rather than to variation in hyperparameters.

3.7.1 BM25 Parameters

BM25 retrieval is performed using the standard formulation with widely adopted hyperparameters:

$$k_1 = 1.2 \qquad b = 0.9$$

which match the default settings used in the Lucene and Pyserini implementations. These parameters control the influence of term frequency and document length normalization, respectively. Passage text is lowercased and tokenized using whitespace segmentation, and no additional preprocessing or stopword filtering is applied. The BM25 index remains fixed for all experiments.

3.7.2 HNSW Dense Retrieval Parameters

The dense retrieval system is implemented using a Hierarchical Navigable Small World (HNSW) graph built over a set of precomputed passage embeddings. The underlying embedding model used to generate these vectors is not specified. During index construction, the embeddings are inserted into a FAISS HNSW index using fixed hyperparameters:

$$M = 8 \qquad \text{efConstruction} = 200 \qquad \text{efSearch} = 200$$

These parameters remain constant across all experiments to ensure that performance comparisons reflect differences in retrieval and reranking methods rather than index configuration.

3.7.3 LSF

Linear Score Fusion combines BM25 and HNSW scores using fixed, symmetric weights:

$$\alpha_{\text{BM25}} = 0.6 \qquad \alpha_{\text{HNSW}} = 0.6$$

Scores are normalized per query prior to fusion to ensure comparability between sparse and dense scoring ranges.

3.7.4 RRF

Reciprocal Rank Fusion uses the standard formulation,

$$\text{RRF}(d) = \sum_{s \in \{\text{BM25,HNSW}\}} \frac{1}{k + \text{rank}_s(d)},$$

with the fusion constant fixed at

$$k = 60$$

which is widely adopted in prior literature and known to perform robustly across heterogeneous rank lists.

3.7.5 Bi-Encoder Neural Reranker

All neural reranking experiments use a BERT style bi-encoder built from the pretrained `sentence-transformers/msmarco-distilbert-base-v4` model. Query and passage embeddings are generated independently, and semantic similarity is computed using cosine similarity. Neural reranking is applied cascadingly to the top-k candidates produced by BM25, HNSW, LSF, and RRF, always using a fixed reranking depth of:

$$topk = k = 100.$$

3.7.6 Evaluation Metrics

Four families of ranking metrics are used depending on the relevance label format of the dataset:

- **MRR@10** (Mean Reciprocal Rank): Measures the rank of the highest relevance passage within the top 10.
- **MAP** (Mean Average Precision): Evaluates the full ranking quality for binary relevance labels.
- **Recall@100**: Measures the fraction of relevant passages retrieved in the top 100.

- **NDCG@10 and NDCG@100** (Normalized Discounted Cumulative Gain): Capture graded relevance quality and penalize misordering of highly relevant passages. Used only for eval1 and eval2, which contain graded labels.

Together, these assumptions establish a controlled experimental environment in which all retrieval and reranking systems operate under the same indexing configurations, fusion parameters, neural architecture, and ranking depth. As a result, the comparative performance analysis presented in the next section reflects generalized differences in retrieval and reranking effectiveness across ranking pipelines, independent of system specific hyperparameter tuning. This independence enables us to identify which system combinations exhibit the strongest fundamental ranking behavior, providing a principled foundation for selecting the most promising pipelines for deeper, hyperparameter optimized experimentation in future work.

4 Analysis and Findings

This section presents a comparative evaluation of all retrieval, fusion, and neural reranking systems across the three MS MARCO subsets used in our study: *eval1*, *eval2*, and *dev*. Each subset provides a distinct relevance format, graded (0–3) for the evaluation sets and binary (0/1) for the development set, which dictates the set of metrics reported. For the evaluation subsets, performance is measured using MRR@10, Recall@100, NDCG@10, and NDCG@100. For the dev subset, performance is measured using MRR@10, Recall@100, and MAP.

Across all experiments, we compare four primary systems (BM25, HNSW, LSF, RRF) along with their corresponding cascaded bi-encoder reranking variants (Bi-BM25, Bi-HNSW, Bi-LSF, Bi-RRF). The goal is to determine which retrieval pipelines generalize most effectively, whether fusion methods provide consistent gains, and to what extent a BERT based bi-encoder improves ranking quality when applied as a final reranking stage.

4.1 Evaluation 1 Results

Table 1 reports the aggregated performance on the *eval1* subset, which contains graded relevance labels. Several key observations emerge from these results.

System	MRR@10	Recall@100	NDCG@10	NDCG@100
BM25	0.7655	0.4964	0.4366	0.5139
HNSW	0.9535	0.5582	0.6973	0.6543
LSF	0.7812	0.4964	0.4438	0.5180
RRF	0.8866	0.5922	0.6366	0.6423
Bi–BM25	0.9297	0.4964	0.6799	0.5863
Bi–HNSW	**0.9845**	0.5582	0.7183	0.6702
Bi–LSF	0.9297	0.4964	0.6799	0.5863
Bi–RRF	**0.9845**	**0.5922**	**0.7203**	**0.6812**

Table 1: Aggregated performance on the Eval1 dataset across all retrieval, fusion, and neural reranking systems. Best scores per metric are shown in bold.

Sparse vs. Dense Retrieval. HNSW significantly outperforms BM25 on all graded metrics, reflecting the strength of dense embeddings for semantic retrieval.

Effectiveness of Fusion-Based Reranking. RRF offers stronger gains than LSF, especially in recall and NDCG. However, neither fusion method exceeds the performance of HNSW alone, suggesting that on this subset, dense retrieval captures much of the same signal that fusion redistributes.

Impact of the Bi-Encoder Cascading Rerank. Cascaded reranking with BERT improves every system. The most pronounced gains occur when reranking BM25 and LSF outputs, which have the weakest initial rankings. Notably:

- HNSW and RRF both achieve the highest MRR@10 scores (0.9845) after bi-encoder reranking.
- Bi-RRF becomes the strongest overall system, leading every metric including RECALL@100 (.5922) NDCG@10 (0.7203) and NDCG@100 (0.6812).

Summary. Overall, the Evaluation 1 results reveal a clear hierarchy of system effectiveness:

$$\textbf{Bi-RRF} > \textbf{Bi-HNSW} > \textbf{HNSW} > \textbf{RRF}$$

Fusion enhances sparse and dense retrieval, but the bi-encoder provides the most substantial and consistent improvements across all baselines.

4.2 Evaluation 2 Results

Table 2 presents the aggregated results for the *eval2* subset. As with *eval1*, this split contains graded relevance labels, and we report performance using MRR@10, Recall@100, NDCG@10, and NDCG@100.

System	MRR@10	Recall@100	NDCG@10	NDCG@100
BM25	0.7914	0.5030	0.4857	0.4961
HNSW	**0.9244**	0.6167	0.6632	0.6397
LSF	0.7981	0.5061	0.4924	0.5022
RRF	0.8671	**0.6384**	0.5968	0.6256
Bi–BM25	0.8997	0.5030	0.6487	0.5639
Bi–HNSW	0.9196	0.6167	0.6614	0.6409
Bi–LSF	0.8997	0.5061	0.6487	0.5649
Bi–RRF	0.9196	**0.6384**	**0.6674**	**0.6573**

Table 2: Aggregated performance on the Eval2 dataset across all retrieval, fusion, and neural reranking systems. Best scores per metric are shown in bold.

Sparse vs. Dense Retrieval. Dense retrieval again outperforms sparse retrieval, with HNSW providing a substantially stronger baseline than BM25 across all metrics. Interestingly, HNSW achieves the highest MRR@10 score (0.9244), even outperforming all bi-encoder reranking variants. This behavior suggests that, for the *eval2* queries, the nearest-neighbor graph structure already retrieves highly relevant documents in top positions, leaving little room for improvement within the first ten ranks.

Why Bi-Encoder Reranking Does Not Improve MRR@10 on Eval2. Several factors may explain why the bi-encoder does not surpass HNSW in terms of MRR@10:

- **Graded relevance distribution.** Eval2 contains queries where the top HNSW result is frequently assigned the highest relevance level. Because MRR rewards the position of the first relevant document, bi-encoder adjustments may not yield upward movement.
- **Document order sensitivity.** The bi-encoder optimizes fine grained semantic similarity rather than nearest neighbor geometric structure. For queries where HNSW already returns a lexically and semantically perfect top result, reranking introduces score noise without improving the first hit position.

Fusion-Based Reranking. RRF continues to offer measurable improvements over BM25, LSF, and even HNSW in terms of Recall@100, posting the strongest value (0.6384). LSF again yields only minimal gains, consistent with its performance pattern in *eval1*.

Impact of the Bi-Encoder Cascading Rerank. Although it does not exceed HNSW on MRR@10, the bi-encoder achieves the strongest performance across all graded relevance metrics:

- Bi-RRF leads NDCG@10 (0.6674) and NDCG@100 (0.6573).
- Bi-RRF ties with RRF for best Recall@100 (0.6384).
- Bi-HNSW remains competitive, narrowly trailing its RRF counterpart in NDCG metrics.

These gains indicate that the bi-encoder's value lies in refining the *overall ranking structure* rather than optimizing the position of the first relevant document.

Summary. Unlike *eval1*, the *eval2* subset exhibits a notable distinction:

HNSW achieves the highest overall MRR@10

However, for metrics sensitive to graded relevance across the entire ranking the strongest system is:

Bi-RRF achiving the highest overall Recall@100, NDCG@10, and NDCG@100

which consistently leads all evaluation measures except MRR@10 by (-0.0048) which is a very minor amount. This suggests that dense retrieval dominates early precision, while bi-encoder reranking provides superior global relevance ordering.

4.3 Development Set Results

Unlike the evaluation subsets, the MS MARCO *dev* split contains binary relevance labels, and system performance is measured using MRR@10, Recall@100, and MAP. Table 3 reports the aggregated results across all 1000 queries, while Table 4 presents the breakdown by query-length bucket (short, medium, long).

4.3.1 Overall Performance

Sparse vs. Dense Retrieval. As in the evaluation subsets, HNSW strongly outperforms BM25, with large gains in MRR@10 (+0.20) and MAP (+0.20). The dense embedding space provides a clearer relevance signal even under binary labeling.

Fusion-Based Reranking. RRF again delivers substantial improvements in Recall@100, achieving the highest score among all non-neural systems (0.9420). However, HNSW remains the strongest system when considering early precision (MRR@10) and overall ranking quality (MAP).

System	MRR@10	Recall@100	MAP
BM25	0.3798	0.7585	0.3738
HNSW	0.5765	0.9071	0.5690
LSF	0.3882	0.7585	0.3819
RRF	0.5221	**0.9420**	0.5153
Bi-BM25	0.5573	0.7585	0.5483
Bi-HNSW	0.6033	0.9071	0.5939
Bi-LSF	0.5573	0.7585	0.5483
Bi-RRF	**0.6280**	**0.9420**	**0.6190**

Table 3: Aggregated performance on the Dev dataset across all retrieval, fusion, and neural reranking systems. Best values per metric are shown in bold.

Impact of Bi-Encoder Reranking. The bi-encoder provides consistent improvements across all baselines. Its strongest performance is obtained when reranking RRF outputs:

Bi-RRF achieves the highest MRR@10, Recall@100, and MAP

Notably, bi-encoder reranking produces large improvements for BM25 and LSF, both of which more than double their MRR gains relative to their original baselines.

4.3.2 Bucketed Performance by Query Length

Table 4 reports the results broken down by query length. The same query distribution applies to all systems: 139 short, 518 medium, and 343 long queries.

	Short (1–3 tokens)			Medium (4–6 tokens)			Long (7+ tokens)		
System	MRR@10	Recall@100	MAP	MRR@10	Recall@100	MAP	MRR@10	Recall@100	MAP
BM25	0.4652	0.8070	0.4614	0.3913	0.7654	0.3829	0.3279	0.7284	0.3245
HNSW	0.7117	0.9281	0.6991	0.6000	0.9085	0.5912	0.4861	0.8965	0.4826
LSF	0.4858	0.8070	0.4789	0.4011	0.7654	0.3930	0.3292	0.7284	0.3259
RRF	0.6465	0.9676	0.6334	0.5446	0.9440	0.5379	0.4379	0.9286	0.4332
Bi–BM25	0.6712	0.8070	0.6597	0.5665	0.7654	0.5585	0.4974	0.7284	0.4877
Bi–HNSW	0.7123	0.9281	0.6985	0.6271	0.9085	0.6174	0.5233	0.8965	0.5161
Bi–LSF	0.6712	0.8070	0.6597	0.5665	0.7654	0.5585	0.4974	0.7284	0.4877
Bi–RRF	**0.7355**	**0.9676**	**0.7236**	**0.6532**	**0.9440**	**0.6436**	**0.5464**	**0.9286**	**0.5393**

Table 4: Bucketed performance on the Dev dataset across short, medium, and long queries.

Short Queries. All systems perform best on short queries, we find that there is generalized performance degradation affecting all systems as query length grows, this is as expected. Bi-RRF achieves the strongest MRR@10 (0.7355) and MAP (0.7236), while RRF and HNSW also perform well. The bi-encoder adds significant gains to BM25 and LSF, lifting their short-query MRR by more than 0.20.

Medium Queries. Medium queries form the largest portion of the dataset (518 queries). Bi-RRF again provides the best results, slightly outperforming Bi-HNSW. Dense retrieval and fusion methods both show strong behavior in this bucket, reflecting robust semantic matching for well-specified queries.

Long Queries. Performance naturally declines as query length increases due to higher semantic variability. Nevertheless, Bi-RRF continues to outperform all other systems across all three metrics. Bi-HNSW is a strong second, highlighting the benefit of neural reranking even when initial retrieval quality is already high.

Summary. Across all aggregated and bucketed evaluations, the Development Set results follow a consistent pattern:

Bi-RRF is the strongest overall system

Bi-HNSW is a close second

HNSW and RRF outperform sparse methods by a substantial margin

Long queries remain the most challenging across all models, but neural reranking consistently narrows the performance gap, demonstrating the value of semantic refinement even under binary relevance conditions.

5 Conclusion

Across all three MS MARCO subsets examined in this study—*eval1*, *eval2*, and *dev*—a consistent performance hierarchy emerges among retrieval and reranking systems. Dense retrieval via HNSW provides a substantially stronger foundation than sparse lexical retrieval, outperforming BM25 by large margins in MRR, recall, and overall ranking quality across both graded and binary relevance settings. This establishes HNSW as the most effective standalone retrieval method in our pipeline.

Fusion-based approaches further enhance the quality of initial rankings. Among them, RRF repeatedly demonstrates robust improvements, particularly in recall and NDCG. RRF effectively unifies the complementary strengths of sparse and dense retrieval, producing higher-coverage candidate sets that benefit later reranking stages.

The largest overall gains, however, arise from applying a cascaded BERT-based bi-encoder reranker. Neural reranking consistently refines the document ordering provided by HNSW and RRF, yielding higher global relevance scores and improved ranking structure. Two configurations stand out across all datasets:

- **Bi-HNSW**, which produces strong early precision and high-quality overall rankings by refining the dense retrieval output.
- **Bi-RRF**, which achieves the *best overall* performance across nearly all metrics including Recall@100, NDCG@10, and NDCG@100 on both graded evaluation sets and the binary dev set.

Together, these results indicate that the most effective retrieval pipeline is a hybrid architecture that combines sparse and dense retrieval through RRF, followed by BERT-based bi-encoder reranking. Formally:

$$\textbf{BM25 + HNSW} \xrightarrow{\textbf{RRF fusion}} \textbf{Top-}k\textbf{ fused ranking} \xrightarrow{\textbf{Bi-Encoder rerank}} \textbf{Final ranking}$$

This design consistently outperforms all other configurations, offering the strongest balance of early precision, global relevance ordering, and robustness to query length.

More broadly, these findings highlight a central insight:

Dense retrieval provides the strongest retrieval foundation

Neural reranking delivers the most powerful refinement

Fusion based neural reranking delivers the greatest generalized performance

Fusion acts as an intermediate mechanism that enhances coverage and stabilizes performance across varied query structures. Together, HNSW, RRF, and bi-encoder reranking form a coherent and complementary stack that constitutes the most effective end to end retrieval solution identified in this study.

Future work may explore higher capacity bi-encoders, more efficient dense retrieval indexing strategies, and hyperparameter tuning within the BM25, HNSW and fusion stages. However, even without such tuning, the dominance of hybrid dense fusion neural pipelines is clear and provides a strong foundation for subsequent optimization.

References

[1] Gordon V Cormack, Charles LA Clarke, and Stefan Buettcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 758–759, 2009.

[2] 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. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, 2020.

[3] Yu A Malkov and D A Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(4):851–863, 2018.

[4] Tri Nguyen, Mirianne Rosenberg, Xia Song Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and L. Deng. Ms marco: A human generated machine reading comprehension dataset. In *Proceedings of the Workshop on Cognitive Computation: Integrating Neural and Symbolic Approaches (CoCoNeS) at NIPS*, 2016.

[5] Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with bert. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

[6] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[7] Stephen E Robertson, Steve Walker, and Micheline M Beaulieu. Okapi at trec-3. *NIST Special Publication*, 500-225:109–126, 1995.