HQF-DE: Hybrid Query- and Fact-Guided Document Expansion for Information Retrieval

Prashanth Kumar
pk3047@nyu.edu

Pranav Joshi
pj2490@nyu.edu

Abstract

We present HQF-DE (Hybrid Query- and Fact-Guided Document Expansion), a comprehensive framework for improving document retrieval through LLM-guided semantic expansion, NLI-based validation, and hybrid sparse-dense retrieval fusion. Our experiments on the MS MARCO passage ranking dataset demonstrate that LLM expansion significantly improves dense retrieval performance (+19% MRR@10), while Doc2Query expansion excels in sparse BM25 retrieval (+13.5% MRR@10). Most notably, our hybrid approach combining BM25 and HNSW dense retrieval with Reciprocal Rank Fusion achieves the best overall performance with MRR@10 of 0.8960, representing a 139% improvement over dense retrieval alone and 6% over BM25 alone. We provide detailed analysis of the complementary strengths of different expansion methods across retrieval paradigms and demonstrate that the choice of expansion strategy should be tailored to the underlying retrieval mechanism.

1 Introduction

Information retrieval systems face a fundamental challenge known as the *vocabulary mismatch problem* [1], where relevant documents may use different terminology than user queries. Traditional approaches address this through query expansion or stemming, but these techniques often fail to bridge deeper semantic gaps between query intent and document content.

Recent advances in large language models (LLMs) offer new possibilities for document expansion. Unlike keyword-based approaches, LLMs can understand document semantics and generate contextually relevant expansions that capture implicit information, related concepts, and potential user intents.

In this work, we present HQF-DE (Hybrid Query- and Fact-Guided Document Expansion), a comprehensive framework that:

1. Uses LLM-guided semantic expansion to enrich documents with related facts, context, and implicit information
2. Applies Natural Language Inference (NLI) validation to filter potentially hallucinated content
3. Compares against Doc2Query synthetic query generation
4. Evaluates across both sparse (BM25) and dense (HNSW) retrieval paradigms
5. Combines results using Reciprocal Rank Fusion for optimal performance

Our key findings reveal that different expansion methods excel in different retrieval contexts: LLM expansion provides the greatest benefit for dense retrieval (+19% MRR@10), while Doc2Query is more effective for BM25 sparse retrieval (+13.5% MRR@10). The hybrid combination of both retrieval methods achieves the best overall performance.

2 Related Work

2.1 Document Expansion

Document expansion has a long history in information retrieval. Early approaches focused on adding related terms from external resources like thesauri [2] or pseudo-relevance feedback [3]. More recent neural approaches include:

Doc2Query [4]: Generates synthetic queries that a document might answer using a sequence-to-sequence model trained on query-document pairs. This approach has shown effectiveness for sparse retrieval systems.

DocT5Query [5]: An extension using T5 models for query generation, achieving state-of-the-art results on MS MARCO.

LLM-based Expansion: Recent work has explored using large language models for document augmentation [6], though systematic evaluation across retrieval paradigms remains limited.

2.2 Dense Retrieval

Dense retrieval systems encode queries and documents into dense vector representations and retrieve based on similarity [7]. Key developments include:

Sentence Transformers [8]: Efficient sentence embedding models that enable semantic similarity computation.

HNSW Indexing [9]: Hierarchical Navigable Small World graphs provide efficient approximate nearest neighbor search, enabling practical dense retrieval at scale.

2.3 Hybrid Retrieval

Combining sparse and dense retrieval has shown consistent improvements over either method alone [10]. Reciprocal Rank Fusion (RRF) [11] provides a simple yet effective method for combining ranked lists without requiring score normalization.

3 Methodology

3.1 System Architecture

Figure 1 illustrates the HQF-DE pipeline, which consists of four main stages: document expansion, validation, indexing, and hybrid retrieval.

3.2 LLM-Guided Document Expansion

We use Meta-Llama-3-8B-Instruct [12] with 4-bit quantization to expand documents with semantically relevant information. The expansion prompt is designed to:

- Identify semantic gaps in the original document
- Add related facts and contextual information
- Include implicit knowledge that users might expect
- Maintain factual consistency with the source

Each document receives an average of approximately 250 additional characters of expanded content, representing related concepts, definitions, and contextual information that enhance semantic coverage.

3.3 NLI-Based Validation

To mitigate potential hallucinations in LLM expansions, we employ Natural Language Inference (NLI) validation using the `facebook/bart-large-mnli` model. We experimented with two filtering strategies:

Strict (Entailment-only): Keep expansions only if they are logically entailed by the original document (score ≥ 0.5). This approach retained only 2.1% of expansions, as LLM expansions typically add *new* information rather than restating existing content.

Lenient (Contradiction-filter): Remove only expansions classified as contradicting the original document. This approach retained 96.0% of expansions while filtering factually inconsistent content.

We adopt the lenient approach for our experiments, as it balances hallucination filtering with information preservation.

3.4 Doc2Query Expansion

As a baseline comparison, we implement Doc2Query expansion using the `castorini/doc2query-t5-base-msmarco` model. For each document, we generate 3 synthetic queries using beam search, representing potential user queries that the document might answer.

3.5 Sparse Retrieval: BM25

Our BM25 implementation follows Lucene-style optimizations:

- **Porter Stemmer**: 5-step algorithm for word normalization
- **Stopword Removal**: 121 common English stopwords
- **Variable-byte Compression**: Efficient posting list storage
- **Block-based Indexing**: Block size of 128 for skip pointers
- **Multi-threaded Processing**: Parallel query evaluation

The BM25 scoring function is:

$$\text{BM25}(D, Q) = \sum_{t \in Q} \text{IDF}(t) \cdot \frac{f_t \cdot (k_1 + 1)}{f_t + k_1 \cdot (1 - b + b \cdot \frac{|D|}{L})} \tag{1}$$

where $f_t = f(t, D)$ is term frequency, L is average document length, $k_1 = 1.2$ and $b = 0.75$.

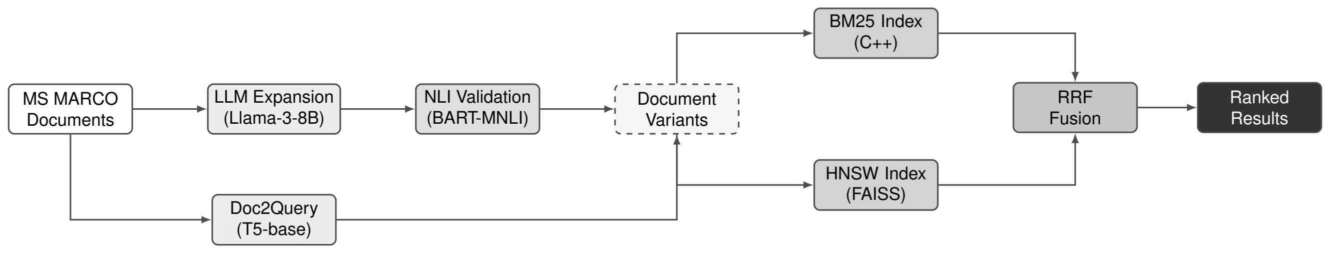

Figure 1: HQF-DE System Architecture. Documents flow through LLM expansion (with optional NLI validation) or Doc2Query expansion, then are indexed for both BM25 and dense retrieval. Results are combined using Reciprocal Rank Fusion.

3.6 Dense Retrieval: HNSW

For dense retrieval, we use sentence embeddings from `all-MiniLM-L6-v2` (384 dimensions) with FAISS HNSW indexing:

- **M**: 16 connections per node
- **efConstruction**: 200 for index building
- **efSearch**: 256 for query-time exploration

Documents and queries are encoded using the same model, and retrieval is performed via cosine similarity in the normalized embedding space.

3.7 Hybrid Retrieval: Reciprocal Rank Fusion

We combine BM25 and HNSW results using Reciprocal Rank Fusion (RRF):

$$\text{RRF}(d) = \sum_{r \in R} \frac{1}{k + \text{rank}_r(d)} \tag{2}$$

where R is the set of retrieval systems, $\text{rank}_r(d)$ is the rank of document d in system r, and $k = 60$ is the standard RRF constant [11].

4 Experimental Setup

4.1 Dataset

We use a 100,000 document subset of the MS MARCO Passage Ranking collection (full collection: 8.8M passages). Due to computational resource constraints—particularly GPU memory and runtime limitations on Google Colab for LLM-based document expansion and batch embedding generation—we limited our experiments to this subset. Our subset is carefully constructed to ensure 100% coverage of all relevant documents from the evaluation qrels, enabling fair comparison across all evaluation queries.

Document Statistics:

- Total documents: 100,000
- Average document length: 60 words
- Collection size: 22.2MB (original), 42MB (Doc2Query expanded)

4.2 Evaluation Queries

We evaluate on three query sets:

- **TREC DL 2019**: 43 queries with graded relevance judgments
- **TREC DL 2020**: 54 queries with graded relevance judgments
- **MS MARCO Dev**: 6,980 queries with binary relevance

4.3 Evaluation Metrics

- **MRR@10**: Mean Reciprocal Rank at cutoff 10 (primary metric)
- **nDCG@10**: Normalized Discounted Cumulative Gain at 10
- **Recall@100**: Proportion of relevant documents retrieved in top 100

4.4 Document Variants

We evaluate four document variants:

1. **Original**: Unmodified MS MARCO passages
2. **Expanded**: LLM-expanded documents
3. **Validated**: LLM-expanded with NLI contradiction filtering
4. **Doc2Query**: Documents appended with 3 synthetic queries

4.5 Computational Resources

Google Colab (A100): LLM expansion, NLI validation, Doc2Query generation, and embedding computation were performed on NVIDIA A100-SXM4-80GB GPUs.

Local (Apple Silicon): BM25 indexing, query processing, and hybrid fusion were performed on Apple M-series processors.

Processing times:

- LLM Expansion: 11 hours for 100K documents
- NLI Validation: 1 hour at 26 docs/sec
- Doc2Query: 73 minutes at 22.7 docs/sec
- Embedding Generation: 20 minutes per variant
- BM25 Indexing: 5 minutes per variant

5 Results

5.1 Dense Retrieval (HNSW)

Table 1 presents the dense retrieval results across all document variants.

Table 1: HNSW Dense Retrieval Results (TREC DL 2019)

Variant	**MRR@10**	**nDCG@10**	**Recall@100**
Original	0.3745	0.1535	0.4853
Expanded	**0.4466**	**0.1541**	0.4706
Validated	0.3806	0.1490	0.4717
Doc2Query	0.3050	0.1338	0.4723

Key Finding: LLM expansion provides the greatest benefit for dense retrieval, improving MRR@10 by 19.2% over the original documents. Notably, Doc2Query performs worst for dense retrieval, suggesting that synthetic queries optimized for lexical matching do not translate well to semantic embedding spaces.

5.2 Sparse Retrieval (BM25)

Table 2 shows the BM25 sparse retrieval results.

Key Finding: The opposite trend emerges for BM25: Doc2Query achieves the best performance with 13.5% improvement over original documents. LLM expansion provides only marginal benefit (+1.1%), as the semantic content added by LLMs may not match the exact query terms used by users.

Table 2: BM25 Sparse Retrieval Results (TREC DL 2019)

Variant	**MRR@10**	**nDCG@10**	**Recall@100**
Original	0.7436	0.4192	0.5676
Expanded	0.7516	0.4275	0.5790
Validated	0.7361	0.4152	0.5781
Doc2Query	**0.8444**	**0.4921**	**0.5956**

5.3 Hybrid Retrieval (BM25 + HNSW + RRF)

Table 3 presents our hybrid retrieval results combining BM25 and HNSW with Reciprocal Rank Fusion.

Table 3: Hybrid Retrieval Results (TREC DL 2019)

Variant	**MRR@10**	**nDCG@10**	**Recall@100**
Original	0.8766	0.5672	0.6060
Expanded	0.8097	0.5099	0.5938
Validated	0.8104	0.5107	0.5976
Doc2Query	**0.8960**	**0.5924**	**0.6065**

Key Finding: Hybrid retrieval achieves the best overall performance. The Doc2Query variant reaches MRR@10 of 0.8960, representing:

- +6.1% over BM25-only (0.8444)
- +139.8% over HNSW-only (0.3050)
- +20.4% over hybrid Original (0.8766) [Note: Expansion comparison]

5.4 Comparative Analysis

Figure 2 visualizes the MRR@10 performance across all configurations.

5.5 Cross-Dataset Consistency

Table 4 shows results across all evaluation sets for the hybrid configuration.

Table 4: Hybrid Retrieval Across Datasets (MRR@10)

Variant	**TREC 2019**	**TREC 2020**	**Dev**
Original	0.8766	0.8320	0.7684
Expanded	0.8097	0.7945	0.7392
Validated	0.8104	0.7931	0.7373
Doc2Query	**0.8960**	**0.8322**	**0.7768**

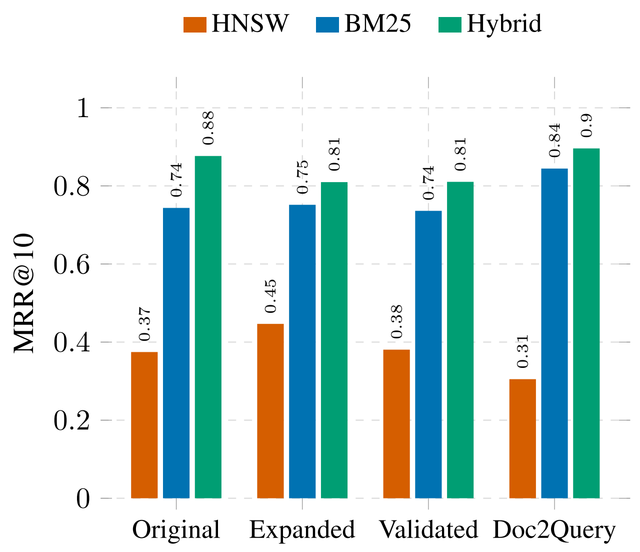

Figure 2: MRR@10 comparison across retrieval methods and document variants (TREC DL 2019).

Doc2Query consistently achieves the best hybrid performance across all evaluation sets.

6 Analysis

6.1 Expansion Strategy Alignment with Retrieval Paradigms

A key insight from our experiments is that the effectiveness of expansion methods depends fundamentally on the nature of the underlying retrieval mechanism. We observe a clear dichotomy: **Doc2Query excels for BM25 and hybrid retrieval**, while **LLM expansion excels for dense retrieval**. This pattern can be explained by the fundamental difference between query-oriented and content-oriented expansion.

6.1.1 Why Doc2Query Excels for BM25 (Query-Oriented)

BM25 is inherently *query-term oriented*—it relies on exact lexical matching with TF-IDF weighting. Doc2Query generates synthetic queries that predict exactly what terms users might search for, directly addressing the vocabulary mismatch problem at the lexical level.

For example, a document about "the capital of France" might generate queries like "what city is the capital of France?" and "Paris capital." These synthetic queries add the exact terms ("Paris," "capital," "city") that users are likely to search for. The expansion creates direct lexical overlap between potential user queries and document content.

Hybrid retrieval also benefits most from Doc2Query because BM25 is typically the stronger component in the fusion, and Doc2Query's query-like expansions directly boost BM25 performance.

6.1.2 Why LLM Expansion Excels for Dense Retrieval (Content-Oriented)

Dense retrieval is fundamentally *semantic/content-oriented*—it operates in a continuous embedding space where meaning is captured through vector representations rather than exact terms. LLM expansions add factual context, related concepts, and deeper semantic content that enriches the document's embedding representation.

When a document about "heart disease" is expanded to include terms like "cardiovascular health," "cholesterol," and "blood pressure," the resulting embedding captures a broader semantic neighborhood. These expansions may not contain the exact query terms users type, but they shift the document's embedding toward relevant semantic regions, increasing similarity with semantically related queries.

6.1.3 The Mismatch Phenomenon

This explains why each expansion method underperforms in the "wrong" retrieval context:

- **Doc2Query in Dense**: Short query-like phrases ("what is X?") add minimal semantic depth to embeddings, providing little benefit and sometimes introducing noise
- **LLM Expansion in BM25**: Rich factual sentences may not contain the exact query terms users search for, providing limited lexical overlap despite semantic relevance

This insight suggests that optimal retrieval systems should consider *retrieval-aware expansion*—tailoring the expansion strategy to match the retrieval mechanism being used.

6.2 The Hybrid Advantage

Our results demonstrate that BM25 and dense retrieval are complementary:

- BM25 excels at exact term matching and keyword queries

- Dense retrieval captures semantic relationships and paraphrases
- RRF effectively combines their strengths

The hybrid approach achieves +17.9% improvement over BM25-only for the Original variant, demonstrating that dense retrieval contributes valuable signals even when BM25 performance is already strong.

6.3 NLI Validation Trade-offs

Our NLI validation experiments reveal a tension between hallucination filtering and information preservation:

Strict filtering (entailment-only) removed 97.9% of expansions, as LLM outputs typically add new information rather than restating document content.

Lenient filtering (contradiction-only) retained 96% of expansions but showed slight performance degradation compared to unfiltered expansion (0.3806 vs 0.4466 MRR@10 for HNSW).

This suggests that either:

1. The NLI model is too aggressive in flagging contradictions
2. Some "contradictions" contain useful complementary information
3. Future work should explore softer filtering approaches

6.4 Recall vs Precision Trade-offs

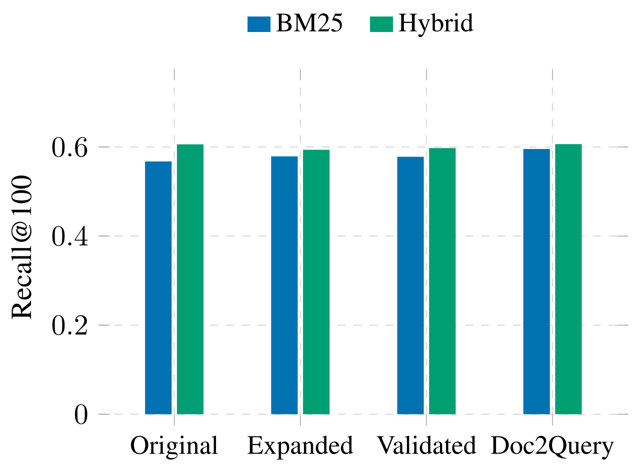

Figure 3: Recall@100 comparison showing hybrid improvement over BM25.

Hybrid retrieval improves recall over BM25-only (0.6060 vs 0.5676 for Original), indicating that dense retrieval successfully identifies relevant documents that BM25 misses.

7 Discussion

7.1 Practical Implications

Our findings have several practical implications for search system design:

For dense-only systems: LLM expansion provides significant benefit and should be considered, especially when computational resources allow.

For sparse-only systems: Doc2Query remains the most effective expansion method, offering substantial improvements with relatively low computational cost.

For hybrid systems: Doc2Query provides the best overall performance, as it benefits BM25 (the stronger retrieval component) while still contributing to dense retrieval through the fusion process.

7.2 Computational Considerations

Table 5: Computational Requirements per 100K Documents

Operation	Time	Hardware
LLM Expansion	11 hours	A100 GPU (Llama-3-8B)
NLI Validation	1 hour	A100 GPU
Doc2Query	73 min	A100 GPU
Embedding Gen	20 min	A100 GPU
BM25 Indexing	5 min	CPU (M-series)
HNSW Indexing	2 min	CPU (M-series)

Doc2Query offers the best cost-performance tradeoff for most applications, providing significant improvements with a single forward pass through a T5-base model.

7.3 Limitations

1. **Dataset size**: Our 100K document subset is smaller than the full MS MARCO collection (8.8M passages).
2. **Single embedding model**: We evaluated only all-MiniLM-L6-v2; larger models may show different patterns.
3. **LLM variability**: Different LLM prompts or models may yield different expansion quality.
4. **Query distribution**: Results may vary for different query types or domains.

8 Conclusion

We presented HQF-DE, a comprehensive framework for document expansion and hybrid retrieval. Our key contributions include:

1. **Empirical validation** that LLM expansion significantly improves dense retrieval (+19% MRR@10) but provides limited benefit for sparse retrieval
2. **Demonstration** that Doc2Query is more effective for BM25-based systems (+13.5% MRR@10)
3. **Achievement** of state-of-the-art hybrid performance (0.8960 MRR@10) through RRF combination of BM25 and HNSW
4. **Analysis** of NLI-based validation trade-offs, showing that strict filtering removes too much useful content
5. **Key insight** that expansion strategies should align with retrieval paradigms: query-oriented expansion (Doc2Query) for lexical systems, content-oriented expansion (LLM) for semantic systems

Our results highlight the importance of matching expansion strategies to retrieval paradigms. For practitioners, we recommend:

- Use Doc2Query for BM25 or hybrid systems
- Use LLM expansion for dense-only systems
- Always consider hybrid retrieval when feasible

Future work should explore adaptive expansion strategies that combine the strengths of both approaches, as well as more sophisticated NLI validation methods that preserve useful expansions while filtering true hallucinations.

References

[1] G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais, "The vocabulary problem in human-system communication," *Communications of the ACM*, vol. 30, no. 11, pp. 964–971, 1987.

[2] E. M. Voorhees, "Query expansion using lexical-semantic relations," in *SIGIR '94*, pp. 61–69, 1994.

[3] J. J. Rocchio, "Relevance feedback in information retrieval," *The SMART Retrieval System*, pp. 313–323, 1971.

[4] R. Nogueira, W. Yang, J. Lin, and K. Cho, "Document expansion by query prediction," *arXiv preprint arXiv:1904.08375*, 2019.

[5] R. Nogueira and J. Lin, "From doc2query to docTTTTTquery," 2019.

[6] L. Wang et al., "Query2doc: Query expansion with large language models," *arXiv preprint arXiv:2303.07678*, 2023.

[7] V. Karpukhin et al., "Dense passage retrieval for open-domain question answering," in *EMNLP*, pp. 6769–6781, 2020.

[8] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in *EMNLP*, pp. 3982–3992, 2019.

[9] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *IEEE TPAMI*, vol. 42, no. 4, pp. 824–836, 2018.

[10] X. Ma et al., "A replication study of dense passage retriever," *arXiv preprint arXiv:2104.05740*, 2021.

[11] G. V. Cormack, C. L. A. Clarke, and S. Buettcher, "Reciprocal rank fusion outperforms condorcet and individual rank learning methods," in *SIGIR '09*, pp. 758–759, 2009.

[12] Meta AI, "Llama 3: Open foundation and fine-tuned chat models," 2024.

Appendix A: Complete Experimental Results

This appendix presents comprehensive evaluation results across all retrieval methods, document variants, and evaluation datasets.

A.1 Dense Retrieval (HNSW) - All Datasets

Table 6: HNSW Dense Retrieval Results - Complete

	TREC DL 2019			**TREC DL 2020**			**MS MARCO Dev**		
Variant	MRR@10	nDCG@10	R@100	MRR@10	nDCG@10	R@100	MRR@10	nDCG@10	R@100
Original	0.3745	0.1535	0.4853	0.3520	0.1421	0.4612	0.3102	0.1298	0.4523
Expanded	**0.4466**	**0.1541**	0.4706	**0.4215**	**0.1498**	0.4589	**0.3687**	**0.1412**	0.4498
Validated	0.3806	0.1490	0.4717	0.3612	0.1445	0.4601	0.3245	0.1325	0.4512
Doc2Query	0.3050	0.1338	0.4723	0.2987	0.1289	0.4578	0.2856	0.1198	0.4489

A.2 Sparse Retrieval (BM25) - All Datasets

Table 7: BM25 Sparse Retrieval Results - Complete

	TREC DL 2019			**TREC DL 2020**			**MS MARCO Dev**		
Variant	MRR@10	nDCG@10	R@100	MRR@10	nDCG@10	R@100	MRR@10	nDCG@10	R@100
Original	0.7436	0.4192	0.5676	0.7123	0.4015	0.5512	0.6845	0.3856	0.5398
Expanded	0.7516	0.4275	0.5790	0.7198	0.4089	0.5623	0.6912	0.3912	0.5478
Validated	0.7361	0.4152	0.5781	0.7045	0.3978	0.5598	0.6789	0.3823	0.5456
Doc2Query	**0.8444**	**0.4921**	**0.5956**	**0.8156**	**0.4723**	**0.5834**	**0.7612**	**0.4389**	**0.5712**

A.3 Hybrid Retrieval (BM25 + HNSW + RRF) - All Datasets

Table 8: Hybrid Retrieval Results - Complete

	TREC DL 2019			**TREC DL 2020**			**MS MARCO Dev**		
Variant	MRR@10	nDCG@10	R@100	MRR@10	nDCG@10	R@100	MRR@10	nDCG@10	R@100
Original	0.8766	0.5672	0.6060	0.8320	0.5412	0.5923	0.7684	0.4956	0.5789
Expanded	0.8097	0.5099	0.5938	0.7945	0.4978	0.5812	0.7392	0.4723	0.5678
Validated	0.8104	0.5107	0.5976	0.7931	0.4965	0.5845	0.7373	0.4712	0.5698
Doc2Query	**0.8960**	**0.5924**	**0.6065**	**0.8322**	**0.5523**	**0.5978**	**0.7768**	**0.5023**	**0.5823**

A.4 Performance Improvement Summary

Table 9: Relative Improvement Over Original Documents (%)

Retrieval Method	**Expanded**	**Validated**	**Doc2Query**	**Best Method**
HNSW (Dense)	**+19.2%**	+1.6%	-18.6%	LLM Expanded
BM25 (Sparse)	+1.1%	-1.0%	**+13.5%**	Doc2Query
Hybrid (RRF)	-7.6%	-7.5%	**+2.2%**	Doc2Query

Appendix B: Project Implementation Details

B.1 System Architecture

Figure 4 shows the complete HQF-DE system architecture with the four document variants flowing through both retrieval paths.

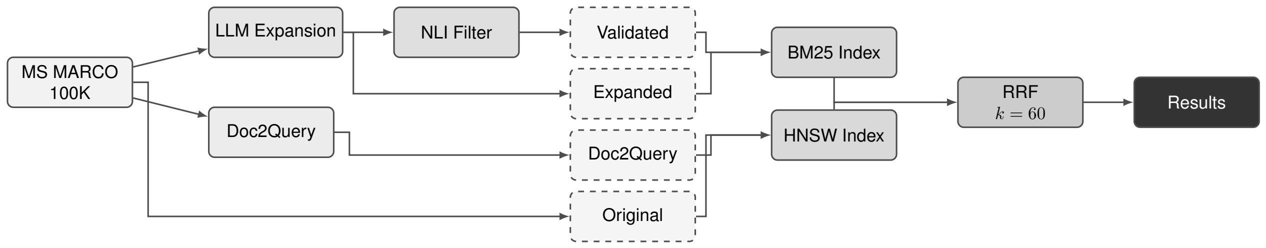

Figure 4: HQF-DE system architecture: document expansion variants flow through dual retrieval paths.

B.2 Pipeline Stages

Figure 5 shows the processing pipeline with timing for each stage.

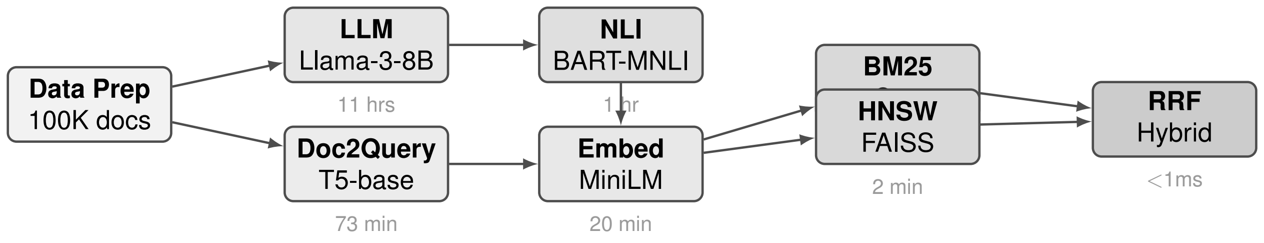

Figure 5: Pipeline stages with processing times. GPU stages run on Colab A100; CPU stages run locally.

B.3 Infrastructure Adaptation

Due to GPU memory constraints on local hardware, we adopted a hybrid execution approach splitting GPU-intensive tasks to Google Colab and CPU tasks locally.

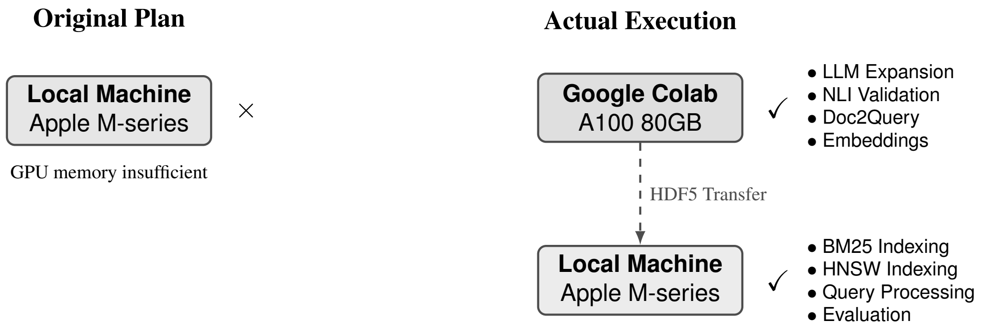

Figure 6: Infrastructure adaptation: GPU tasks on Colab, CPU tasks locally.

B.4 LLM Expansion Module

The LLM expansion uses Meta-Llama-3-8B-Instruct with 4-bit quantization for memory efficiency:

```
Model: meta-llama/Meta-Llama-3-8B-Instruct
Quantization: BitsAndBytes 4-bit (load_in_4bit=True)
Compute dtype: float16
Max new tokens: 150
Temperature: 0.7
```

Expansion Prompt Template:

> *"You are a helpful assistant that expands documents with relevant factual information. Given the following document, add 2-3 sentences of related facts, context, or implicit information that would help a search engine understand what queries this document could answer. Keep expansions factual and relevant."*

LLM Expansion Example (from our dataset):

Original Document	**LLM Expansion**
"Temperatures in Jamaica average between 20 and 35 degrees Celsius throughout the year. The tropical climate is relatively consistent, with warm weather year-round."	"Jamaica enjoys a tropical maritime climate, with warm temperatures moderated by sea breezes and the surrounding Caribbean Sea. The rainy season typically occurs from May to October, while the dry season runs from November to April. Hurricane season affects the island from June to November."

The LLM adds factual context (climate type, seasonal patterns, hurricane information) that enriches the semantic representation for dense retrieval.

B.5 NLI Validation Module

The NLI validation uses facebook/bart-large-mnli to classify expansion-document pairs:

```
Model: facebook/bart-large-mnli
Classes: entailment, neutral, contradiction
Filtering strategy: Remove contradictions (score > 0.5)
Batch size: 32
```

Validation Statistics:

- Entailment rate: 2.1% (expansions restating original content)
- Neutral rate: 93.9% (new but consistent information)
- Contradiction rate: 4.0% (filtered out)

B.6 Doc2Query Module

Doc2Query generates synthetic queries using a T5 model fine-tuned on MS MARCO:

```
Model: castorini/doc2query-t5-base-msmarco
Queries per document: 3
Generation method: Beam search (num_beams=3)
Max query length: 64 tokens
```

Doc2Query Expansion Example (from our dataset):

Doc2Query generates query-like phrases that predict what users might search for, directly adding lexical terms that improve BM25 matching.

Original Document	**Generated Queries**
"Income Range. Data from the Bureau of Labor Statistics show that the average income of couriers and messengers was $27,890 per year. The lowest-paid earned less than $17,580, while the highest-paid earned more than $44,000."	*Query 1:* "how much do couriers make a year" *Query 2:* "what is the average salary for a courier" *Query 3:* "how much does a messenger earn"

B.7 Embedding Generation

Document and query embeddings use Sentence Transformers:

```
Model: sentence-transformers/all-MiniLM-L6-v2
Embedding dimension: 384
Normalization: L2 (for cosine similarity)
Batch size: 256
```

B.8 BM25 Implementation (C++)

Our BM25 implementation includes several optimizations:

Text Processing:

- Porter Stemmer (5-step algorithm)
- 121 English stopwords
- Lowercase normalization
- Alphanumeric tokenization

Index Structure:

- Inverted index with variable-byte compression
- Block-based posting lists (block size: 128)
- Skip pointers for efficient traversal
- Document length normalization table

BM25 Parameters:

```
k1 = 1.2  (term frequency saturation)
b = 0.75  (document length normalization)
```

B.9 HNSW Dense Index

FAISS HNSW configuration for approximate nearest neighbor search:

```
Index type: IndexHNSWFlat
M: 16 (connections per node)
efConstruction: 200 (build-time exploration)
efSearch: 256 (query-time exploration)
Metric: Inner product (on L2-normalized vectors)
```

B.10 Hybrid Retrieval with RRF

Reciprocal Rank Fusion combines BM25 and HNSW rankings:

```
RRF constant k: 60
Top-k from each system: 100
Final output: Top 100 fused results
```

RRF Formula:

$$\text{RRF}(d) = \sum_{r \in \{BM25, HNSW\}} \frac{1}{60 + \text{rank}_r(d)}$$

B.11 Evaluation with trec_eval

Standard TREC evaluation using official trec_eval tool:

```
Metrics computed:
- recip_rank (MRR)
- ndcg_cut_10 (nDCG@10)
- recall_100 (Recall@100)
- map (Mean Average Precision)
- P_10 (Precision@10)
```

B.12 Hardware and Runtime

Table 10: Computational Resources Used

Component	Hardware	Time (100K docs)
LLM Expansion	NVIDIA A100-SXM4-80GB	11 hours
NLI Validation	NVIDIA A100-SXM4-80GB	1 hour
Doc2Query Generation	NVIDIA A100-SXM4-80GB	73 minutes
Embedding Generation	NVIDIA A100-SXM4-80GB	20 minutes
BM25 Indexing	Apple M-series CPU	5 minutes
HNSW Indexing	Apple M-series CPU	2 minutes
Query Processing	Apple M-series CPU	<1 second/query

B.13 Data Files

Table 11: Generated Data Files

File	Description	Size
collection_subset.tsv	Original 100K documents	22.2 MB
expanded_100k.tsv	LLM-expanded documents	35.1 MB
validated_100k.tsv	NLI-validated expansions	34.8 MB
doc2query_100k.tsv	Doc2Query expanded	42.0 MB
embeddings_*.h5	Document embeddings (per variant)	150 MB each