UTokyo-HitU at TREC 2025 RAG Track: HyDE-Enhanced Sparse-Dense Retrieval Fusion with LLM Reranking

Sho Fukada
fukada@hal.t.u-tokyo.ac.jp
The University of Tokyo
Tokyo, Japan

Atsushi Keyaki
a.keyaki@r.hit-u.ac.jp
Hitotsubashi University
Tokyo, Japan

Yusuke Matsui
matsui@hal.t.u-tokyo.ac.jp
The University of Tokyo
Tokyo, Japan

Abstract

This paper describes our submission to TREC RAG 2025 from the University of Tokyo and Hitotsubashi University. Our approach integrates hybrid retrieval combining sparse retrievers (BM25 and SPLADE) with dense retrievers (BGE-small and Qwen3-Embedding-0.6B), Hypothetical Document Embeddings (HyDE) for query augmentation, and LLM-based reranking. A key contribution is our HyDE Vector Mix strategy, which creates a weighted combination of original query and hypothetical answer embeddings with a mixing ratio α. Our four-method hybrid retrieval system achieved first place in the Retrieval task among 12 participating teams (46 runs), with nDCG@30 of 0.693, nDCG@100 of 0.613, and recall@100 of 0.257. We participated in all three tasks: Retrieval (R), Augmented Generation (AG), and RAG.

Keywords

retrieval-augmented generation, dense retrieval, hypothetical document embeddings, hybrid search, reranking

1 Introduction

Retrieval-Augmented Generation (RAG) [5] has emerged as a powerful paradigm for knowledge-intensive natural language processing tasks. By combining retrieval mechanisms with generative models, RAG systems can leverage external knowledge sources to produce more accurate and grounded responses. The TREC RAG 2025 track provides a comprehensive evaluation framework for such systems, challenging participants to retrieve relevant passages from the MS MARCO v2.1 corpus and generate well-cited answers to user queries.

This paper presents the system developed by The University of Tokyo and Hitotsubashi University (UTokyo-HitU) for the TREC RAG 2025 track. In designing our system, we prioritized maximizing retrieval accuracy without initial constraints on computational cost, adopting techniques that improve performance even at the expense of increased computational requirements. This design philosophy led our system to achieve first place in the Retrieval task among 12 participating teams (46 runs).

Our approach builds upon three key technical contributions. First, we propose HyDE Vector Mix, a novel method for incorporating hypothetical document embeddings that outperforms both the original HyDE [4] approach and text concatenation strategies. Second, we demonstrate the effectiveness of four-method hybrid retrieval, combining sparse methods (BM25 [8] with keyword expansion and SPLADE [3]) with dense methods (BGE-small [11] and Qwen3-Embedding-0.6B [13]) through Reciprocal Rank Fusion. Third, we show that LLM-based reranking with document URL context significantly improves ranking quality.

Table 1: System components overview

Component	Method	Model/Tool
Query Preprocessing	HyDE [4], Keyword Exp.	GPT-4.1
Sparse Retrieval	BM25 [8], SPLADE [3]	Anserini [12]
Dense Retrieval	Embedding	BGE-small [11], Qwen3 [13]
Fusion	RRF [1]	$k = 60$
Reranking	Sliding Window [9]	GPT-4.1-mini
Generation	Answer + Citation	GPT-4.1, Ragnarok [7]

2 Methods

2.1 System Architecture

Our system follows a multi-stage pipeline architecture as illustrated in Figure 1. The pipeline consists of query preprocessing, four-method hybrid retrieval, RRF fusion, LLM reranking, and optional augmented generation. Table 1 summarizes the components and models used in each stage of our system.

2.2 Query Preprocessing

The query preprocessing phase prepares augmented representations for both sparse and dense retrieval.

HyDE Generation. HyDE [4] generates a hypothetical answer to the query using an LLM, then uses this answer for dense retrieval. Importantly, this generation relies solely on the LLM's internal knowledge and does not access any documents from the target corpus. We use GPT-4.1 to generate approximately 150-word hypothetical answers. The prompt instructs the model to produce natural passages containing information relevant to the query without requiring factual correctness. For example, given the query "what is vicarious trauma and how can it be coped with?", the model generates a passage describing vicarious trauma as the emotional impact experienced by individuals indirectly exposed to others' traumatic experiences, along with coping strategies such as self-care and setting professional boundaries.

Keyword Expansion. For sparse retrieval enhancement, we expand each query with related keywords using GPT-4.1, including synonyms, related terms, technical terminology, abbreviations, and common variations. This expansion addresses vocabulary-mismatch issues inherent to lexical-matching approaches.

2.3 HyDE Vector Mix

Dense retrieval systems face a fundamental challenge: user queries are typically short and abstract, while target documents are detailed and explanatory. While HyDE addresses this by using hypothetical answer embeddings for retrieval, it completely replaces the original query, potentially discarding important information such as specific named entities or domain-specific terminology.

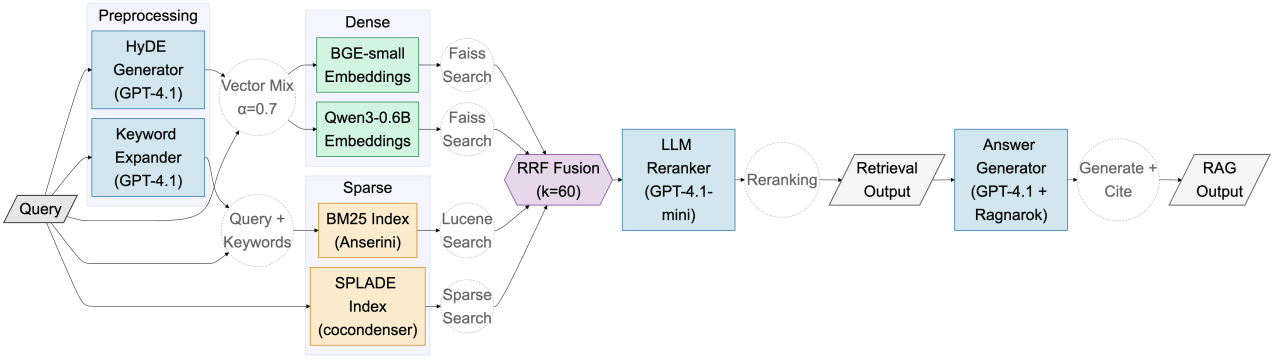

Figure 1: System architecture overview. The pipeline integrates query preprocessing with HyDE and keyword expansion, four complementary retrieval methods, RRF fusion, and LLM-based reranking.

We propose HyDE Vector Mix, which combines the embeddings of both the original query and the hypothetical answer through weighted averaging. Let q denote the original query and h denote the hypothetical answer. We define their embedding vectors as:

$$\boldsymbol{q} = \frac{f(q)}{\|f(q)\|_2}, \quad \boldsymbol{h} = \frac{f(h)}{\|f(h)\|_2}, \tag{1}$$

where f represents the embedding encoder. HyDE Vector Mix creates the final query representation $\boldsymbol{q}'$ using mixing ratio $\alpha \in [0, 1]$:

$$\boldsymbol{q}' = \frac{(1-\alpha)\boldsymbol{q} + \alpha\boldsymbol{h}}{\|(1-\alpha)\boldsymbol{q} + \alpha\boldsymbol{h}\|_2}. \tag{2}$$

This formulation generalizes the existing approaches: $\alpha = 0$ corresponds to standard dense retrieval, while $\alpha = 1$ corresponds to the original HyDE. We use $\alpha = 0.7$ based on development set optimization.

2.4 Sparse Retrieval

We employ two complementary sparse retrieval methods. BM25 is implemented through Anserini/Lucene with queries augmented by the generated keywords. SPLADE [3] uses the naver/splade-cocondenser-ensembledistil model to obtain learned sparse representations that provide both term expansion and importance weighting.

2.5 Dense Retrieval

For dense retrieval, we employ two embedding models: BGE-small-en-v1.5 [11] (33M parameters, 384 dimensions) and Qwen3-Embedding-0.6B [13] (600M parameters, 1024 dimensions). Both models apply HyDE Vector Mix with $\alpha = 0.7$. We use the mixed query vector $\boldsymbol{q}'$ from Equation (2) to search against pre-computed segment embeddings. All segment embeddings are generated offline using the same encoder f and stored in a Faiss [2] index. We use IndexFlatIP (exact inner product search without approximation) with cosine similarity as the distance metric, enabled by L2-normalizing all vectors.

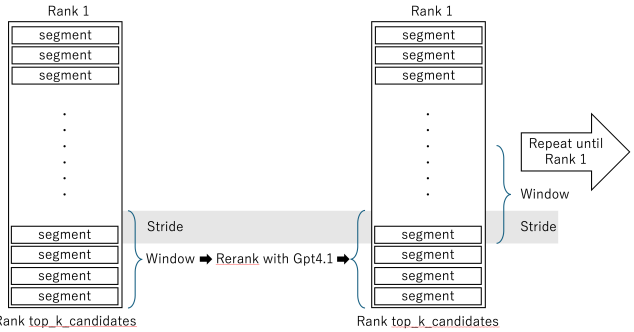

Figure 2: Sliding window reranking process. Documents are reranked in windows, moving from lower to higher ranks across multiple passes.

2.6 Result Fusion

We combine results from all four retrieval methods using Reciprocal Rank Fusion (RRF) [1]:

$$\text{RRF}(d) = \sum_{r \in R} \frac{1}{k + \text{rank}_r(d)}, \tag{3}$$

where d is a candidate document, R is the set of retrieval methods, $r \in R$ denotes an individual method, and $\text{rank}_r(d)$ is the rank of document d in the result list returned by method r. We use $k = 60$ as the smoothing parameter. In our system, R consists of four methods: BM25 with keyword expansion, SPLADE, BGE-small with HyDE Vector Mix, and Qwen3 with HyDE Vector Mix. Each method contributes its top 1,000 candidates, and the fused results retain the top 1,000 documents for subsequent reranking.

2.7 LLM Reranking

We employ LLM-based reranking using the sliding window approach [9] with the RankLLM framework [6] and GPT-4.1-mini. Figure 2 illustrates the reranking process. Starting from lower ranks, we extract a window of documents, rerank them with the LLM, then move up by a stride and repeat until reaching the top. We repeat this process for multiple passes.

Table 2: Reranking parameters

Parameter	Value
Model	GPT-4.1-mini
window_size	10
stride	5
num_passes	3
top_k_candidates	200
Context fields	Title + URL + Segment text

Table 3: Comparison of sparse retrieval methods

Method	P@100	nDCG@5	nDCG@10	nDCG@20
BM25	0.333	0.362	0.348	0.342
BM25 + Keywords	0.338	0.452	0.428	0.402
SPLADE	0.435	0.481	0.467	0.453

Table 2 summarizes our reranking parameters. A notable aspect is the inclusion of document URL information as context, which helps the LLM understand document credibility and domain relevance.

2.8 Augmented Generation

For the AG and RAG tasks, we use GPT-4.1 with the Ragnarok framework [7]. The pipeline builds a prompt from the query and the top-20 retrieved segments, generates an answer, performs sentence segmentation with SpaCy, and extracts citations.

3 Experiments

3.1 Experimental Setup

We developed and validated our methods using TREC RAG 2024 queries (100 queries) with UMBRELA relevance judgments [10] on the MS MARCO v2.1 Segment corpus.

3.2 Sparse Retrieval Comparison

Table 3 compares the sparse retrieval methods. Keyword expansion improves BM25's nDCG@5 by 24.9%. While SPLADE achieves higher standalone performance, BM25 retrieves different relevant documents, making both methods valuable for fusion.

3.3 HyDE Strategy Comparison

Table 4 compares different strategies for incorporating hypothetical document embeddings. HyDE Vector Mix with $\alpha = 0.7$ consistently achieves the highest performance across all metrics and embedding models. For BGE-small, Vector Mix improves nDCG@10 from 0.493 (no HyDE) to 0.564, a 14.4% relative improvement. Notably, using HyDE embeddings alone decreases P@100 compared to no HyDE (0.401 vs. 0.444), while Vector Mix improves it to 0.473.

3.4 Hybrid Retrieval Ablation

Table 5 presents an ablation study of our hybrid retrieval approach. The full four-method configuration achieves P@100 of 0.610, a 40.2% relative improvement over the BM25+BGE baseline. Each additional method contributes unique relevant documents.

Table 4: Comparison of HyDE strategies on development data. Vector Mix uses $\alpha = 0.7$.

Model	Strategy	P@100	nDCG@5	nDCG@10	nDCG@20
BGE-small	No HyDE	0.444	0.510	0.493	0.468
BGE-small	HyDE Only	0.401	0.533	0.497	0.475
BGE-small	Text Concat	0.466	0.566	0.539	0.520
BGE-small	Vector Mix	**0.473**	**0.578**	**0.564**	**0.535**
Qwen3 0.6B	No HyDE	0.332	0.390	0.386	0.372
Qwen3 0.6B	HyDE Only	0.456	0.542	0.520	0.510
Qwen3 0.6B	Vector Mix	**0.496**	**0.570**	**0.554**	**0.538**

Table 5: Ablation study of hybrid retrieval methods.

Configuration	P@100	nDCG@5	nDCG@10	nDCG@20
BM25 + BGE	0.435	0.486	0.468	0.449
Qwen-HyDE + SPLADE	0.557	0.602	0.593	0.580
+ BGE-HyDE	0.565	0.615	0.606	0.591
+ BM25-KW (Full)	**0.610**	**0.629**	**0.626**	**0.618**

Table 6: Effect of reranking configurations on BM25+BGE baseline.

Configuration	P@100	nDCG@5	nDCG@10	nDCG@20
No reranking	0.435	0.486	0.468	0.449
Basic reranking	0.454	0.673	0.657	0.604
+ URL context	0.454	0.691	0.662	0.606
+ 3 passes	0.454	0.697	0.669	0.640
+ Top-200	0.511	0.730	0.701	0.635
Full config	**0.661**	**0.778**	**0.769**	**0.753**

3.5 Reranking Analysis

Table 6 analyzes the impact of different reranking configurations. LLM reranking dramatically improves ranking quality, with nDCG@5 increasing from 0.486 to 0.778. Key findings include: URL information improves reranking by providing domain context; increasing top_k_candidates from 100 to 200 significantly improves P@100; and smaller windows (10) with more passes (3) outperform larger windows with fewer passes.

3.6 Submitted Runs

We submitted runs for all three tasks: two for Retrieval, one for AG, and two for RAG. Our primary Retrieval run (4method_merge) uses the full four-method hybrid pipeline with GPT-4.1-mini reranking, while the secondary run (qwen_splade) uses only Qwen-HyDE and SPLADE. For AG and RAG tasks, we used GPT-4.1 with the Ragnarok framework.

3.7 Official Results

Table 7 presents our official results. Our primary submission achieved first place among 12 teams (46 runs) in the Retrieval task. Table 8 summarizes our AG and RAG task performance.

Table 7: Official Retrieval task results (1st place among 12 teams, 46 runs).

Run	nDCG@30	nDCG@100	recall@100
4method_merge	**0.693**	**0.613**	**0.257**
qwen_splade	0.652	0.529	0.217

Table 8: Official AG and RAG task results.

Task	Run	vital	sub_cov	w_prec	w_rec
AG	gpt41	0.54	0.86	0.572	0.545
RAG	r_4method	0.53	0.79	0.499	0.480
RAG	r_2method	0.56	0.84	0.450	0.421

4 Discussion

The success of HyDE Vector Mix stems from its ability to balance two complementary signals. The original query embedding (weighted at $1 - \alpha = 0.3$) preserves user-specified terminology, named entities, and the precise information need expressed by the user. Meanwhile, the hypothetical answer embedding (weighted at $\alpha = 0.7$) provides richer semantic context that better matches the typical writing style of relevant documents. This asymmetric weighting reflects our empirical finding that semantic expansion from hypothetical answers is more valuable than exact query matching, while some original query signal remains essential to prevent semantic drift.

The substantial gains from hybrid retrieval stem from the complementary nature of different retrieval paradigms. BM25 with keyword expansion excels at exact term matching and handles rare or domain-specific terms effectively. SPLADE provides learned sparse representations with automatic term expansion and importance weighting. Dense models capture semantic similarity at the embedding level, finding relevant documents even without lexical overlap. RRF allows each method to contribute its unique strengths without requiring learned fusion weights, making it robust across different query types.

Several approaches did not yield improvements during our development experiments. Query paraphrasing, where we generated alternative phrasings of the original query, consistently decreased retrieval accuracy, likely due to semantic drift from the user's intended meaning. Using GPT-5-mini for reranking showed no improvement over GPT-4.1-mini. More complex AG pipelines incorporating MMR-based segment re-ranking, LLM-based relevance filtering, and multi-step citation generation showed no improvement over direct generation with Ragnarok. We also experimented with e5-small as an alternative embedding model, but it performed worse than BGE-small and was not adopted.

5 Conclusion

This paper presented the UTokyo-HitU system for TREC RAG 2025, achieving first place in the Retrieval task among 12 participating teams. Our primary contributions include: (1) HyDE Vector Mix, a simple yet effective strategy for combining query and hypothetical document embeddings with a tunable mixing ratio; (2) four-method hybrid retrieval combining sparse and dense approaches through RRF fusion; and (3) optimized LLM reranking with URL context information.

Future work includes developing methods to optimize the mixing ratio α on a per-query basis, as our analysis suggests that different query types may benefit from different balances between original query and hypothetical answer signals. We also plan to investigate larger embedding models such as BGE-large and Qwen3-Embedding-4B to further improve retrieval accuracy.

References

[1] 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)*. Association for Computing Machinery, 758–759. doi:10.1145/1571941.1572114

[2] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The Faiss Library. *IEEE Transactions on Big Data* (2024). doi:10.1109/TBDATA.2024.3411053

[3] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2022. From Distillation to Hard Negative Sampling: Making Sparse Neural IR Models More Effective. arXiv:2205.04733 [cs.IR] https://arxiv.org/abs/2205.04733

[4] Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023. Precise Zero-Shot Dense Retrieval without Relevance Labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 1762–1777. doi:10.18653/v1/2023.acl-long.99

[5] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems*, Vol. 33. Curran Associates, Inc., 9459–9474.

[6] Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023. RankLLM: Listwise Reranking with Large Language Models. https://github.com/castorini/rank_llm

[7] Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2025. Ragnarök: A Reusable RAG Framework and Baselines for TREC 2024 Retrieval-Augmented Generation Track. arXiv:2501.00353 [cs.IR] https://arxiv.org/abs/2501.00353

[8] Stephen Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval* 3, 4 (2009), 333–389.

[9] 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*. https://openreview.net/forum?id=3Q6LON8y2I

[10] Shivani Upadhyay, Ronak Pradeep, Nandan Thakur, Nick Craswell, and Jimmy Lin. 2024. UMBRELA: UMbrela is the (Open-Source Reproduction of the) Bing RELevance Assessor. arXiv:2406.06519 [cs.IR]

[11] Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-Pack: Packaged Resources To Advance General Chinese Embedding. arXiv:2309.07597 [cs.CL]

[12] Peilin Yang, Hui Fang, and Jimmy Lin. 2017. Anserini: Enabling the Use of Lucene for Information Retrieval Research. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 1253–1256.

[13] 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:2506.05176 [cs.CL] https://arxiv.org/abs/2506.05176

A Prompts

This appendix provides the prompts used for query preprocessing in our system.

A.1 HyDE Generation Prompt

The following prompt is used with GPT-4.1 to generate hypothetical answers for dense retrieval:

```
Generate a hypothetical answer of approximately
150 words for the following query to improve
dense retrieval accuracy. The answer doesn't
need to be factually correct, but should be a
natural passage containing information relevant
to the query.
```

Example Input: "what is vicarious trauma and how can it be coped with?"

Example Output: "Vicarious trauma refers to the emotional and psychological impact experienced by individuals who are indirectly exposed to others' traumatic experiences, often seen in professionals like therapists, social workers, and first responders. This secondary trauma can lead to symptoms similar to post-traumatic stress disorder, including anxiety, emotional numbness, and difficulty concentrating. Coping with vicarious trauma involves several strategies such as practicing regular self-care, including mindfulness and relaxation techniques, setting clear professional boundaries to avoid emotional overload, and seeking peer support or supervision to process difficult feelings."

A.2 Keyword Expansion Prompt

The following prompt is used with GPT-4.1 to generate keywords for BM25 query expansion:

```
Generate approximately 10 related keywords for
the following query to improve BM25 retrieval
accuracy. Include synonyms, related terms, technical
terminology, abbreviations, and common variations.
```

Example Input: "what is vicarious trauma and how can it be coped with?"

Example Output: "vicarious trauma, secondary traumatic stress, compassion fatigue, empathic strain, indirect trauma, trauma exposure response, psychological impact of trauma, coping strategies for vicarious trauma, self-care for trauma workers, stress management in helping professions."