MIT Lincoln Laboratory at TREC 2025 Retrieval-Augmented Generation Track

Daniel Gwon
daniel.gwon@ll.mit.edu
MIT Lincoln Laboratory
Lexington, MA, USA

Kynnedy Smith*
kynnedys@andrew.cmu.edu
Carnegie Mellon University
Pittsburgh, PA, USA

Nour Jedidi†
njedidi@uwaterloo.ca
University of Waterloo
Waterloo, ON, Canada

Abstract

This paper describes MIT Lincoln Laboratory's participation in the TREC 2025 RAG track. We made submissions to each of the Retrieval (R) and Retrieval-Augmented Generation (RAG) tasks. We focused primarily on various steps in the retrieval pipeline and used a strong, proprietary LLM for the generation step. We describe a multistage retrieval pipeline that combines query decomposition, learned sparse retrieval, and pointwise reranking to retrieve the highest-quality ranked list prior to generation. The LLM uses the ranked list to generate a response with citations using a standard prompt.

1 Introduction

This report discusses MIT Lincoln Laboratory's submission to the TREC Retrieval-Augmented Generation (RAG) track. We participated in both the Retrieval (R) and Augmented Generation (AG) tasks.

Our proposed pipeline is displayed in fig. 4 and runs as follows. Given the longer and more complex multi-hop queries in TREC RAG 2025, we first leveraged an LLM to break down the user-issued query into a set of smaller, more focused sub-queries. Each of these sub-queries were then independently processed in a standard retrieve-and-rerank process using SPLADEv3 [2] for initial retrieval and an LLM-based pointwise reranker for reranking. The ranked lists for each of the sub-queries were then fused via Reciprocal Rank Fusion (RRF) [1] and fed into SETR [3] – a method that prompts an LLM to identify the optimal set of passages from a final ranked list (in our case the RRF outputs) that contains all the information needed to answer the user query. This final set of SETR selected passages are subsequently passed into GPT-5 for answer generation.

Our submission seeks to investigate various steps in a multi-stage retrieval pipeline with the aim of isolating the impact of each component in the overall RAG pipeline. We report all the different retrieval-based approaches we consider in Table 1.

2 Retrieval

We made five submissions to the retrieval-only task, each outlined in figs. 1 to 4 (fig. 4 details the pipeline for two submissions). Specific models and techniques are highlighted in 1.

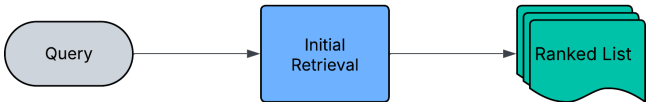

Figure 1: Initial retrieval only.

Figure 2: Initial retrieval with reranking.

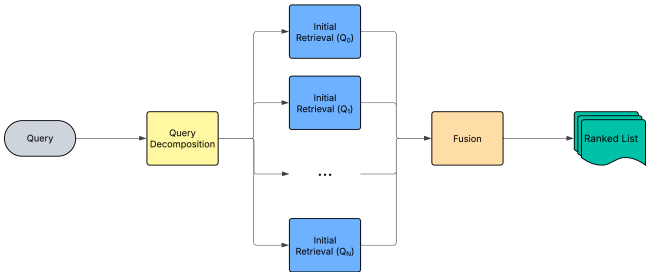

Figure 3: Query decomposition with initial retrieval and fusion.

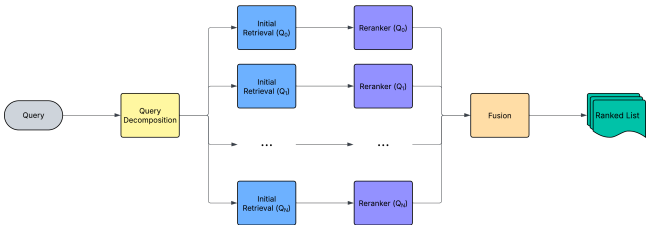

Figure 4: Query decomposition, initial retrieval, reranking, and fusion.

Table 1: Specific models/techniques used for retrieval-only task submissions.

Run	Query Decomposition	Initial Retrieval	Reranking	Fusion
1	-	SPLADEv3	-	-
2	-	SPLADEv3	Qwen3-Reranker-8B	-
3	gemma-3-27b-it	SPLADEv3	-	RRF
4	gemma-3-27b-it	SPLADEv3	Qwen3-Reranker-8B	RRF
5	gemma-3-27b-it	SPLADEv3	gemma-3-27b-it	RRF

*Work done while at MIT Lincoln Laboratory
†Work done while at MIT Lincoln Laboratory

[0]DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited. This material is based upon work supported by the Department of the Air Force under Air Force Contract No. FA8702-15-D-0001 or FA8702-25-D-B002. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Department of the Air Force.

2.1 Initial Retrieval

For initial retrieval in all submissions, we relied on the pre-built SPLADEv3[2] index provided by Anserini [6]. It would have been interesting to evaluate the performance of hybrid retrieval using a combination of sparse and dense retrieval models, but the size of the data set prevented us from indexing the corpus with a neural-network-based dense retrieval model.

2.2 Reranking

We used two LLM-based rerankers in our submissions depicted in fig. 4:

(1) Qwen/Qwen3-Reranker-8B [7]
(2) google/gemma-3-27b-it [5]

Both models were used as point-wise rerankers to improve the quality of the ranked lists produced by the initial retrieval step, either from the single starting query or from each individual decomposed query (discussed in 2.3).

For the Qwen reranker, we used the system message and chat format detailed in Zhang et. al [7] with the following instruction:

```
Given a web search query, retrieve relevant passages that answer the query.
```

We did not make any modifications to the gemma model beyond the prompt. We used the following prompt:

```
Determine if the following passage is relevant to the query.
Answer only with 'true' or 'false'
Query: {query}
Passage: {passage_content}
```

2.3 Query Decomposition

We noticed that the queries for the TREC 2025 RAG track were significantly different from previous years. Queries from 2024, for example, were simple and may have been answerable with self-contained pieces of evidence. For example, one of the queries from 2024 asked

```
what is vicarious trauma and how can it be coped with?
```

This question may have required retrieval of a small handful of documents to answer successfully. By contrast, queries from 2025 were long and complex, requiring many hops with synthesis across many documents to answer successfully. For example, one of the queries for 2025 asked

```
I want a thorough understanding of what makes up a community, including its definitions in various contexts like science and what it means to be a 'civilized community.' I'm also interested in related terms like 'grassroots organizations,' how communities set boundaries and priorities, and their roles in important areas such as preparedness and nation-building.
```

To answer such a complex multifaceted question, we used google/gemma-3-27b-it [5] to decompose queries into constituent subqueries. We selected the gemma model as our query decomposer because it is open-source, it has a large context window, and it offers a good balance between performance and size. We used the following system prompt and few-shot examples:

```
System: Given the following user query, break it down into a minimal spanning set of fully-independent subquestions that would need to be answered to comprehensively address the user's information need. Each subquestion should be necessary and collectively should over the query, without redundancy or unnecessary detail. Output your subquestions as a Python list of strings, like this:

[
    "First subquestion.",
    "Second subquestion.",
    "Third subquestion.",
    ...
]

User 1: What are the key factors driving the decline in traditional retail sales, how has e-commerce growth compensated for this decline, and what strategies should brick-and-mortar retailers adopt to remain competitive in the next 5 years?

Assistant 1: ["What factors are causing traditional retail sales to decline?", "What is the growth rate of e-commerce over the past 5 years?", "How much has e-commerce growth offset traditional retail decline?", "What successful digital transformation strategies have retailers implemented?", "What consumer behavior trends will shape retail in the next 5 years?", "What are the most effective omnichannel strategies for brick-and-mortar stores?"]

User 2: How does machine learning model performance vary across different data preprocessing techniques, what are the computational costs of each approach, and which combination would be optimal for a real-time recommendation system handling 100k users?

Assistant 2: ["What are the main data preprocessing techniques for ML models?", "How does each preprocessing technique affect model accuracy?", "What are the computational requirements for each preprocessing method?", "What are the specific requirements for real-time recommendation systems?", "How do preprocessing choices impact inference speed?", "What preprocessing combinations work best for 100k+ user systems?", "What are the memory and storage implications of each approach?"]

User 3: Analyze the competitive landscape for electric vehicle charging infrastructure in North America, identify market gaps, assess the regulatory environment, and recommend entry strategies for a new player with $50M funding.

Assistant 3: ["Who are the major players in North American EV charging infrastructure?", "What is the current market size and growth projections?", "Where are the geographic gaps in charging coverage?", "What types of charging solutions are underserved?", "What federal and state regulations affect EV charging deployment?", "What are typical infrastructure deployment costs?", "What market entry strategies have been successful in this space?", "How far would $50M funding go for different deployment strategies?"]
```

Few-shot examples were generated with Claude's Sonnet 4[1].

On average, each query was decomposed into about 8 subqueries. For example, the query above asking about community was decomposed into the following:

```
How is 'community' defined across different academic disciplines (e.g., sociology, ecology, biology)?
What are the historical and philosophical definitions of a 'civilized community' and what values underpin them?
What are the key characteristics that define a functioning community?
```

[1] https://www-cdn.anthropic.com/6d8a8055020700718b0c49369f60816ba2a7c285.pdf

Figure 5: Retrieval-augmented generation pipeline.

```
    What are 'grassroots organizations,' how do they form, and what
role do they play within communities?
    How do communities establish and enforce boundaries (physical,
social, cultural)?
```

Since one can often only make sense of subqueries within the context of the parent query, we concatenate the original query to each subquery during initial retrieval and reranking steps.

2.4 Reciprocal Rank Fusion

We use reciprocal rank fusion (RRF) [1] to combine separate ranked lists from subqueries into a single ranked list of 100 documents. We used $k = 60$ following Cormack et. al [1].

3 Retrieval-Augmented Generation

Four of the five final ranked lists from the submissions to the retrieval-only task were used for submissions to the RAG task. In addition to each of the components used in the retrieval-only submissions, we incorporated SETR [3] prior to giving the ranked list to the augmented-generation model to produce a final synthesized answer. This decision was motivated by Lee et. al's insight that retrieved passages for complex queries in a RAG setting are relevant as a cohesive set rather than as individual passages. We used the ranked lists from submissions 1-4 to the retrieval-only task for our submissions to the RAG task. The basic pipeline is shown in fig. 5.

3.1 SETR

We used the same passage selection prompt detailed in Lee et. al [3] in a selection only setting. That is, did not perform chain-of-though reasoning or information requirement identification. We also do not use a finetuned SETR model as the authors did not release one, and instead use gemma-3-27b-it. The model occasionally produced outputs that could not be parsed; in these situations we selected the full starting ranked list to be used in the generation step.

3.2 Generation

For generation, we used OpenAI's GPT-5 model with the following prompt provided by Ragnarok [4]:

```
    Provide a concise, information-dense answer to the question.
Your response must not exceed 380 words under any circumstances.
Prioritize the most relevant and impactful information within this
strict limit. Ensure your answer directly addresses the question
and maintains coherent throughout. Cite supporting context
documents inline using IEEE format in square brackets []. Include
1-3 citations per sentence, ordered by decreasing importance.
Ensure each sentence has at least one citation. Use multiple
sources to provide a well-rounded answer when possible. If sources
contradict each other, acknowledge this and explain the
discrepancy. Express uncertainty when appropriate rather than
making unfounded claims. Prioritize factual accuracy and avoid
speculation. Focus solely on answering the question with properly
cited information. Avoid mentioning references or providing any
meta-commentary about the answering process.
```

Table 2: Run-level Results for Retrieval-only Track

Run	nDCG@30	nDCG@100	Recall@100
1	59.6	53.9	23.2
2	61.1	54.3	23.2
3	60.5	55.0	23.7
4	62.5	57.0	24.6
5	64.4	56.4	23.7
min	16.0	9.2	2.1
med	54.2	43.5	17.3
max	77.3	65.4	28.3

Table 3: Run-level Results for Full RAG Track

Run	strict_vital	sub_coverage
1	0.44	0.74
2	0.50	0.77
3	0.48	0.77
4	0.47	0.79
min	0.10	0.23
med	0.35	0.69
max	0.50	0.82

4 Results

The following subsections disclose and discuss run-level results for each run submission to the retrieval-only and RAG tracks.

4.1 Retrieval

Table 2 contains run-level results for each run submitted to the retrieval-only track. Minimum, median, and maximum values are shown as averages of topic-level results. That is, with median as an example, the reported value is the average of all topic-level median scores for each metric. Table 1 contains a breakdown of each run's components.

Despite using what we would characterize as a relatively low-cost, basic retrieval pipeline, all of our submissions exceed the median values at the three aggregated metrics: nDCG@30, nDCG@100, and Recall@100. As one would expect, including query decomposition and a reranker achieve the strongest performance across all three metrics. Surprisingly, using an LLM as a point-wise reranker performs better at nDCG@30 than using a specialized Qwen reranker. Depending on the available compute, a user can forego a specialized reranker and use the same model for query decomposition and reranking for the best performance. Since the gemma model can also be used for augmented generation, this can significantly reduce the compute requirements for a full retrieval pipeline.

4.2 Retrieval-Augmented Generation

Table 3 contains run-level results for each run submitted to the full RAG track. We report NIST post-edited scores. Minimum, median, and maximum values come directly from the TREC evaluators, and appear to be averages of topic-level results just as we calculated for retrieval-only results. Since the primary point of differentiation

between our RAG runs are the retrieval pipeline, we again refer readers to table 1 to identify the retrieval components associated with each run.

Each of our RAG submissions beat median scores for the two reported metrics, strict vital and sub-coverage. Run 1, which only uses SPLADEv3 for retrieval, with no additional components, performs the worst across both metrics as one would expect. Surprisingly, the quality of the retrieval pipeline does not appear to play a significant role in RAG scores, as our retrieval pipelines 2, 3, and 4 all perform comparably across both RAG metrics. This would seem to indicate that a strong LLM, OpenAI's GPT-5 in our case, can make up for minor weaknesses in a retrieval pipeline.

5 Conclusion

In this report we overview our submission to the TREC RAG 2026 track. Based on our results, we find that our more traditional retrieval pipeline is still performant despite the emergence of more costly agentic retrieval approaches. We also find that a stronger LLM can compensate for minor weaknesses in a retrieval pipeline.

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* (Boston, MA, USA) *(SIGIR '09)*. Association for Computing Machinery, New York, NY, USA, 758–759. doi:10.1145/1571941.1572114

[2] Carlos Lassance, Hervé Déjean, Thibault Formal, and Stéphane Clinchant. 2024. SPLADE-v3: New baselines for SPLADE. arXiv:2403.06789 [cs.IR] https://arxiv.org/abs/2403.06789

[3] Dahyun Lee, Yongrae Jo, Haeju Park, and Moontae Lee. 2025. Shifting from Ranking to Set Selection for Retrieval Augmented Generation. arXiv:2507.06838 [cs.CL] https://arxiv.org/abs/2507.06838

[4] Ronak Pradeep, Nandan Thakur, Sahel Sharifymoghaddam, Eric Zhang, Ryan Nguyen, Daniel Campos, Nick Craswell, and Jimmy Lin. 2024. Ragnarök: A Reusable RAG Framework and Baselines for TREC 2024 Retrieval-Augmented Generation Track. arXiv:2406.16828 [cs.IR] https://arxiv.org/abs/2406.16828

[5] Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. 2025. Gemma 3 Technical Report. *arXiv preprint arXiv:2503.19786* (2025).

[6] 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* (Shinjuku, Tokyo, Japan) *(SIGIR '17)*. Association for Computing Machinery, New York, NY, USA, 1253–1256. doi:10.1145/3077136.3080721

[7] 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