Trustworthy Answers, Messier Data: Bridging the Gap in Low-Resource Retrieval-Augmented Generation for Domain Expert Systems

Anonymous ACL submission

Abstract

RAG has become a key technique for enhancing LLMs by reducing hallucinations, especially in domain expert systems where LLMs may lack sufficient inherent knowledge. However, developing these systems in low-resource settings introduces several challenges: (1) handling heterogeneous data sources, (2) optimizing retrieval phase for trustworthy answers, and (3) evaluating generated answers across diverse aspects. To address these, we introduce a data generation pipeline that transforms raw multi-modal data into structured corpus and Q&A pairs, an advanced re-ranking phase improving retrieval precision, and a reference matching algorithm enhancing answer traceability. Applied to the automotive engineering domain, our system improves factual correctness (+1.94), informativeness (+1.16), and helpfulness (+1.67) over a non-RAG baseline, based on a 1-5 scale by an LLM judge. These results highlight the effectiveness of our approach across distinct aspects, with strong answer grounding and transparency.

1 Introduction

Retrieval-Augmented Generation (RAG) has shown potential in reducing hallucinations and providing up-to-date knowledge in Large Language Models (LLMs). This success has grown interest in domain expert RAG-Question Answering (QA) systems to meet specialized knowledge needs. While previous studies (Han et al. 2024; Siriwardhana et al. 2023; Mao et al. 2024) have proposed general methods for adapting RAG models to domain knowledge bases—such as syntactic QA pair generation or model fine-tuning—they face several challenges in low-resource domains.

In practical settings, available data sources in low-resource domains are often presented in heterogeneous formats and exhibit an unstructured nature, making their direct integration into RAG system development challenging (Hong et al., 2024). General models may not have enough inherent knowledge in low-resource domains (Zhao et al., 2024), making fine-tuning essential to adapt the model to specific knowledge requirements. However, the lack of structured data for training further complicates this process. In addition, data privacy and security concerns restrict the full utilization of API-based LLMs (Achiam et al. 2023; Anthropic 2024) within RAG systems, necessitating the use of open-source LLMs (Yang et al. 2024; Touvron et al. 2023).

The retrieval phase is another key aspect of domain expert RAG systems, as referencing accurate documents is essential for generating reliable answers. However, research on improving ranking in the retrieval phase or tracing the documents referenced to generate the answer remains limited. Most domain expert RAG frameworks rely on a single-stage retrieval process, with few exploring multi-stage approaches (Nogueira et al. 2019; Nogueira et al. 2020; Karpukhin et al. 2020), such as retrieval followed by re-ranking—a method widely used in Information Retrieval (IR)—which can help ensure the use of the most relevant references.

The evaluation of RAG systems is also an area that has not been fully addressed. Many studies continue to rely on gold answer similarity metrics, such as overlapping words between the generated answer and the ground truth, which inadequately capture critical dimensions like faithfulness, coherence, and contextual relevance. Recently, the LLM-as-a-judge framework (Zheng et al., 2023) has gained attention as a qualitative alternative. However, these methods often overlook diverse evaluation aspects, and a standardized framework for assessing RAG systems has yet to emerge.

In this work, we address three key challenges and present the RAG development pipeline, demonstrating its application in the automotive engineering domain, with a specific focus on QA for vehicle crash collision tests. First, we present a data generation pipeline (Section 3.1), leveraging diverse

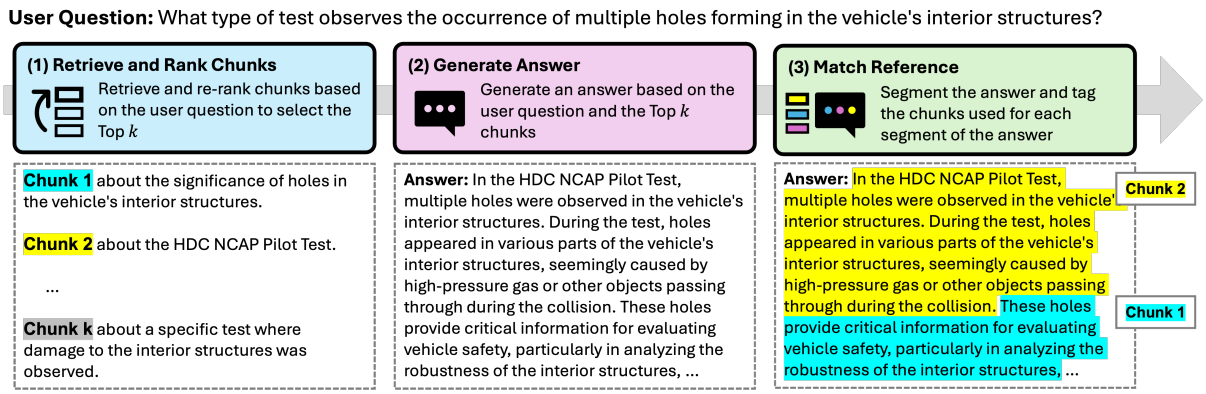

Figure 1: When a user question is given, our RAG system retrieves and re-ranks the text chunks, generates answers using the top k relevant chunks, and ensures each part of the answer is backed by clear references The example shown was originally in Korean and translated into English.

formats of internal documents from an automobile company. Second, we incorporate an advanced re-ranking phase (Section 3.2) and a reference-matching algorithm (Section 3.4), not only enhancing the accuracy of the final answers but also improving traceability by tagging sources for each segment of the answer. Third, we evaluate the answers obtained from our generation model (Section 3.3) from multiple perspectives, emphasizing their qualitative aspects. Our ultimate goal is a fully local system, ensuring data privacy and independence from external servers. The overall workflow of our RAG system is shown in Figure 1, and our key contributions are summarized as follows:

- We propose a data generation pipeline that transforms multi-modal raw data into a structured corpus and high-quality Q&A pairs.
- We integrate re-ranking and reference matching to enhance retrieval precision and answer traceability, ensuring a reliable RAG system.[1]
- We assess the final answer from diverse qualitative perspectives, evaluating each along distinct, non-overlapping dimensions.

2 Related Work

Data Processing RAG-Studio (Mao et al. 2024) employs synthetic data generation for in-domain adaptation, reducing reliance on costly human-labeled datasets. However, it assumes access to well-structured data, limiting its applicability in scenarios with unstructured raw data. To bridge this gap, Hong et al. (2024) tackle challenges with real-world formats (e.g., DOC, HWP), proposing a chunking method that converts documents to HTML, retaining structural information in low-resource settings. Meanwhile, Guan et al. (2024) address the issue of short and noisy e-commerce datasets in RAG system development by building a new dataset from a raw corpus crawled from an e-commerce website, providing a richer resource.

Retrieval in RAG Wang et al. (2024) highlight the importance of re-ranking modules in RAG systems to enhance the relevance of the retrieved documents. Similarly, Zhao et al. (2024) demonstrate that the ranking position of the gold document in the retrieval phase plays a significant role in determining the quality of the final answer. Given these insights, optimizing the retrieval phase is crucial for obtaining accurate context, particularly in specialized, low-resource domains where LLMs lack sufficient inherent knowledge (Beauchemin et al., 2024). Despite these findings, the analysis of the effectiveness and applicability of re-rankers in domain expert RAG systems remains underexplored.

RAG Evaluation The evaluation of RAG systems (Yu et al., 2024) has relied on text similarity-based metrics such as BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), and EM (Rajpurkar et al., 2016). These metrics provide a baseline but overlook valid answer diversity and qualitative aspects like factual consistency and relevance. Recent advancements are made to utilize LLM-as-a-judge (Zheng et al., 2023) to evaluate qualitative dimensions. Han et al. (2024) propose a pairwise preference comparison framework considering helpfulness and truthfulness, while Saad-Falcon et al. (2024) assess context relevance, answer faithfulness, and query relevance, addressing hallucination issues. However, a standardized framework for RAG evaluation remains a challenge.

[1]The full RAG pipeline code will be publicly available: https://github.com/anonymous

3 Approach

3.1 Data Generation & Processing

We build our dataset using two distinct sources: (1) internal reports on vehicle crash collision tests in presentation slide format from an automotive company in Korea, and (2) the textbook *Crash Safety of Passenger Vehicles* (Mizuno, 2016). Vehicle crash collision tests are exceptionally valuable, as each test incurs substantial costs and produces detailed reports that are typically confidential and difficult to access publicly. Figure 2 illustrates the process of extracting, analyzing, and converting slides and textbook PDFs into structured Markdown text and Q&A sets, leveraging Python scripts and the Claude LLM[2] (Anthropic, 2024). Table 1 summarizes the data statistics by source.

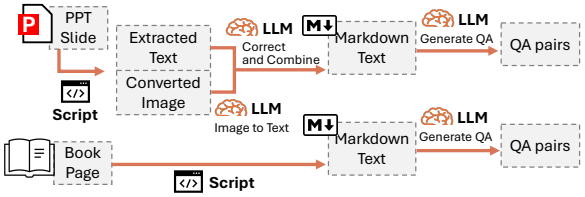

Figure 2: Overview of the data generation pipeline, converting each PPT slides and Textbook pages into text chunks and generating Q&A pairs. An example of the original PPT slide is in Figure 11 (A.1.1).

The original slides often contain tables, graphs, and images, which are simplified into plain text descriptions during data processing. To evaluate performance on questions referencing these elements, we sample 143 multi-modal chunks and generated 1,861 targeted questions. Evaluation results are provided in Section 4.3.1.

Type	Source	File	Slide	Q&A Pairs
PPT	Test Report	1,463	4,662	59,402
	Meeting Report	249	882	7,696
		Page	**Chapter**	**Q&A Pairs**
PDF	Textbook	404	81	1,505
	Total		5,625	68,603

Table 1: Raw data statistics by source, along with the corresponding number of generated Q&A pairs. Dataset splits and token length statistics are described in A.1.1.

Additionally, we extract and prepend headers summarizing each report, including the *Test Name*, *Region*, *State*, and *Purpose*, which significantly improve the retrieval phase (See Section 4.3). Domain experts are responsible for the entire process, including prompt engineering and reviewing intermediate and final outputs. All prompts and an example of a generated Q&A pair are shown in A.2.1.

[2]We restrict the use of API-based LLMs for initial data processing; this limitation is discussed in Section 8.

3.2 Retrieval & Ranking

We employ a Dual-Encoder as our retriever, fine-tuning it on our training data. For a question q, the model retrieves the top n chunks from m chunks $\mathbf{D}$, based on the [CLS] token embedding similarity between q and each chunk $d_i \in \mathbf{D}$, as follows:

$$q_{[\text{CLS}]} \in \mathbb{R}^{1\times d};\ \mathbf{D}_{[\text{CLS}]} \in \mathbb{R}^{m\times d};$$
$$\text{Similarity}(q, \mathbf{D}) = q_{[\text{CLS}]} \cdot \mathbf{D}_{[\text{CLS}]}^{\top} \in \mathbb{R}^{1\times m}$$
$$\mathbf{D}_{\text{top}_n} = \text{Sort}\left(\mathbf{D},\ \text{key} = \text{Similarity}(q, \mathbf{D})\right)[:n]$$

Our re-ranker is a point-wise Cross-Encoder, trained on a classification task (Nogueira et al., 2019) that takes a single (q, d_i) pair as input and returns a scalar relevance score. It re-ranks D_{top_n}, obtained from the retrieval stage, to extract D_{top_k}, which will be passed to the generation model. The process is formalized as follows:

$$\mathbf{D}_{\text{top}_n} = [d_1, d_2, \ldots, d_n]$$
$$x_i = \text{"}\{q\}\ \{\text{sep\_token}\}\ \{d_i\}\text{"} \quad \forall d_i \in \mathbf{D}_{\text{top}_n}$$
$$\text{Rel}_{x_i} = \text{Ranker}(x_i) \quad \forall i \in \{1, 2, \ldots, n\}$$
$$\mathbf{D}_{\text{top}_k} = \text{Sort}(\mathbf{D}_{\text{top}_n},\ \text{key} = \text{Rel}_{x_i})[:k]$$

To optimize Rel_{x_i}, we apply Token Selection (TS) + Term Control Layer (TCL) training method from RRADistill (Choi et al., 2024). It effectively integrates the importance of the general semantic and specific query terms during the training process.

$$d_i^{\text{TS}} = \text{RRA}_{\text{TS}}(q, d_i)$$
$$x_i^{\text{TS}} = \text{"}\{q\}\ \{sep\_token\}\ \{d_i^{\text{TS}}\}\text{"}$$
$$\text{Rel}_{x_i}^{\text{TS}} = \text{Ranker}(x_i^{\text{TS}}) \quad \forall i \in \{1, 2, \ldots, n\}$$
$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}\left(\alpha \cdot \mathcal{L}(\text{Rel}_{x_i}, y_i) + \beta \cdot \mathcal{L}(\text{Rel}_{x_i}^{\text{TS}}, y_i)\right)$$

TS+TCL are used during training only, while inference uses the standard Cross-Encoder approach only with Rel_{x_i}. The target label y is binary: $y = 1$ for relevant pairs and $y = 0$ for irrelevant pairs. Negative sampling retrieves the top 10 chunks from the train data for each query q, excluding the gold chunk, and randomly selects three from the rest.

3.3 Answer Generation

Our answer generation model takes the user question q and the top k chunks ($\mathbf{D}_{\text{top}_n}$) from the retrieval and ranking phase as input to generate the

final answer a', as follow:

$$a' = \text{LLM}(q, \mathbf{D}_{\text{top}_n})$$

We fine-tune open-source LLMs on our training dataset, which consists of (q, a) pairs derived from our data generation pipeline (Section 3.1). During fine-tuning, the question q and the answer a are concatenated into a single sequence $S = [q; a]$, and the model is trained in an auto-regressive manner to predict the next token.

3.4 Reference Matching Algorithm

We propose a reference matching algorithm that segments generated answers and links them to relevant references using a Dynamic Programming (DP) approach (Bellman, 1954). Algorithm 1 outlines the detailed procedure.

Algorithm 1 Reference Matching Algorithm

Input: $a' = [s_1, \ldots, s_n]$, $\mathbf{D}_{\text{top}_k} = [d_1, \ldots, d_k]$
Output: $(s_i{:}s_j, d_t, \text{Score}_{(i,j,d_t)})$

Step 1: Compute Segment-Chunks Scores
Let $a'_{\text{segments}} = \{\text{concat}(s_i, \ldots, s_j) \mid 1 \le i < j \le n\}$ be the set of all concatenated sentence subsequences from s_i to s_j of a'.
for each segment $\mathbf{seg}_u$ in a'_{segments} **do**
  for each chunk d_t in $\mathbf{D}_{\text{top}_k}$ **do**
    Calculate $\text{Score}_{(i,j,d_t)} = \text{Re-ranker}(\mathbf{seg}_u, d_t)$
  end for
end for

Step 2: Optimal Score Selection
`Initialize` the array dp to track optimal scores.
`Initialize` the array $choice$ to store optimal choices.
for each possible ending sentence s_j **do**
  for each possible starting sentence s_i $(1 \le i < j)$ **do**
    for each chunk d_t in $\mathbf{D}_{\text{top}_k}$ **do**
      if $\text{Score}_{(i,j,d_t)} > dp[j+1]$ **then**
        Update $dp[j+1] = \text{Score}_{(i,j,d_t)}$
        Record the choice (i, j, d_t) in $choices$
      end if
    end for
  end for
end for

Step 3: Backtracking
`Initialize` $current = n$ to start from the last sentence.
while $current > 0$ **do**
  Retrieve the best (i, j, d_t) from $choices$ for $current$.
  Add the tuple $(s_i{:}s_j, d_t, \text{Score}_{(i,j,d_t)})$ to the result.
  Set $current = i$ to move to the previous segment.
end while

In Step 1, the algorithm computes the scores for all possible sentence subsequences a'_{segments} against the top-k chunks D_{top_k}, using our re-ranker (Section 3.2). In Step 2, it selects the optimal segment-chunk combinations, updating scores and recording the best choices for each ending sentence (e.g., if the answer consists of five sentences like $a = [s_1, \ldots, s_5]$, the choices might be $(s_1{:}s_1, d_1), (s_2{:}s_2, d_3), (s_2{:}s_3, d_3), (s_1{:}s_4, d_1), (s_5{:}s_5, d_2)$). Finally, in Step 3, backtracking is performed to retrieve and output the best matches from the choices (e.g., $(s_1{:}s_4, d_1), (s_5{:}s_5, d_2)$).

4 Experiment 1: Retrieval & Ranking

4.1 Models

For retriever training, we use BGE-M3 (Chen et al., 2024) as the backbone, a multilingual Encoder capable of processing Korean, and fine-tune it on our training dataset with the publicly available code[3]. For re-ranker training, we initialize the weights using the fine-tuned retriever. Further experimental details are in A.1.2.

4.2 Evaluation

We evaluate the retriever and re-ranker using Mean Average Precision (MAP@k) and Success@k (Manning et al., 2008). MAP@k measures the ranking quality by averaging precision over relevant results up to rank k, while Success@k indicates the proportion of queries with at least one relevant result in the top k. The high Success@k score indicates that relevant chunks are retrieved, while the improved MAP@k highlights better ranking of those retrieved chunks. The evaluation is conducted using test set questions, but the chunk pool for retrieval included all splits (training, validation, and test) to ensure sufficient data and avoid biases from the limited test set size.

4.3 Result

Table 2 shows that prepending a header to each chunk—adding only a small number of tokens (see Table 9)—significantly enhances retrieval performance. Fine-tuning the BGE-M3 model on `Ver.1` (BGE-FT) further improves results notably, demonstrating the importance of task-specific model adaptation to optimize performance.

Model	Train	Test	MAP@1	MAP@5	MAP@10
BGE-Vanilla	N/A	`Ver.0`	0.1978	0.2672	0.2766
		`Ver.1`	0.2991	0.3978	0.4091
BGE-FT	`Ver.1`	`Ver.1`	**0.6048**	**0.7111**	**0.7175**

Table 2: Retrieval performance comparison between models across different data configurations. `Ver.0` refers to the raw chunk without the prepended header, while `Ver.1` includes header.

[3] https://github.com/FlagOpen/FlagEmbedding

Table 3 shows that the re-ranking phase notably enhances the precision of retrieved information. The results indicate that ranking performance is optimal when the number of retrieved chunks is limited to 10. Despite the constraints of retriever failure, this improves retrieval results by approximately +4% in terms of MAP@1 on the test set. In addition, incorporating TS+TCL during training boosts ranking performance (See A.3.1).

k	MAP@1	MAP@5	MAP@10	Success@5
10	**0.6444** (± **0.014**)	**0.7416** (± **0.009**)	**0.7464** (± **0.009**)	**0.8839** (± **0.004**)
20	0.6362 (± 0.017)	0.7353 (± 0.013)	0.7413 (± 0.013)	0.8822 (± 0.008)
30	0.6328 (± 0.019)	0.7316 (± 0.015)	0.7378 (± 0.015)	0.8796 (± 0.011)

Table 3: Ranking performance based on the number of retrieved chunks k. Instances that failed during the retrieval phase were scored as 0.

4.3.1 Multi-modal Specific Questions

Table 4 summarizes the performance on questions requiring information from tables, graphs, and images. Compared to general questions (Section 4.3), the results are notably lower, with the retriever facing significant challenges. Even at k = 30, Success@k only reaches 83.61%, underscoring a major bottleneck in the retrieval phase. Despite these challenges, the re-ranker proved effective, delivering a substantial performance improvement of approximately +15% in MAP@1 at k = 20. These results underscore the need for further advancements to better handle multi-modal questions.

Phase	k	MAP@1	MAP@5	MAP@10
Retrieval w/ BGE-FT	N/A	0.3482	0.4561	0.4684
Reranking	10	0.4723	0.5587	0.5630
	20	**0.4949**	0.5923	0.5995
	30	0.4933	**0.5965**	**0.6041**

Table 4: Performance of retrieval and ranking on multi-modal specific questions. The same models and evaluation methods described in Section 4.3 were used.

4.4 Analysis

Figure 3 illustrates the trade-off between retrieval success and re-ranking performance as the number of retrieved chunks n increases. While increasing n improves the retrieval success rate (e.g., from 92% at n = 10 to 96% at n = 30), it can lead to diminishing improvements in re-ranking performance (e.g., MAP@5 decreases from 0.74 at n = 10 to 0.73 at n = 30). This result highlights the importance of carefully selecting n to balance retrieval success and re-ranking quality, as overly increasing n may not yield proportional benefits for re-ranking.

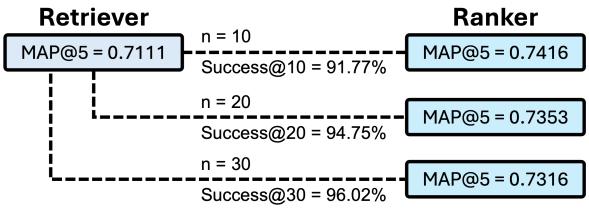

Figure 3: The impact of the number of retrieved chunks n on retrieval success rates and ranking performance.

An analysis of the failed cases during the retrieval and ranking phase identified two error types: (a) Top-10 retrieval failure (23.3%) and (b) Retrieval success but incorrect re-ranker top-1 (76.7%). Type (a) were mainly caused by the open-ended nature of the questions, which led to multiple relevant documents beyond the gold context (Figure 18 in A.3.2). In contrast, type (b) occurred with more specific and detailed questions, where there were several relevant chunks from the same vehicle crash collision test, in addition to the gold chunk, making it difficult to identify a single correct one (Figure 19 in A.3.2). Failures in both types were often due to the presence of multiple valid chunks, rather than the ranking of irrelevant chunks. Table 4 presents a human evaluation of 100 sampled cases for each error type, assessing whether the re-ranker top-1 was relevant to the given question, even though it did not match the gold chunk. A screenshot of the human evaluation interface is in Figure 21 (A.5).

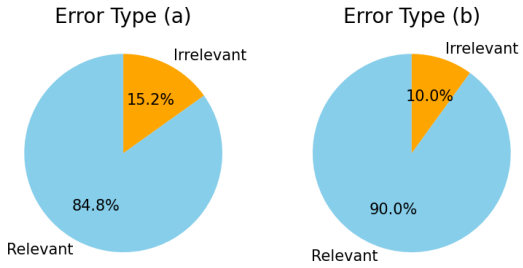

Figure 4: Human evaluation of the re-ranker top-1 for each error type, based on 100 sampled cases, assessing whether it is relevant to the given question.

5 Experiment 2: Answer Generation

5.1 Models

We use Qwen-2.5 (Yang et al., 2024) as our backbone LLMs, one of the few multilingual models that officially supports Korean. Notably, there are

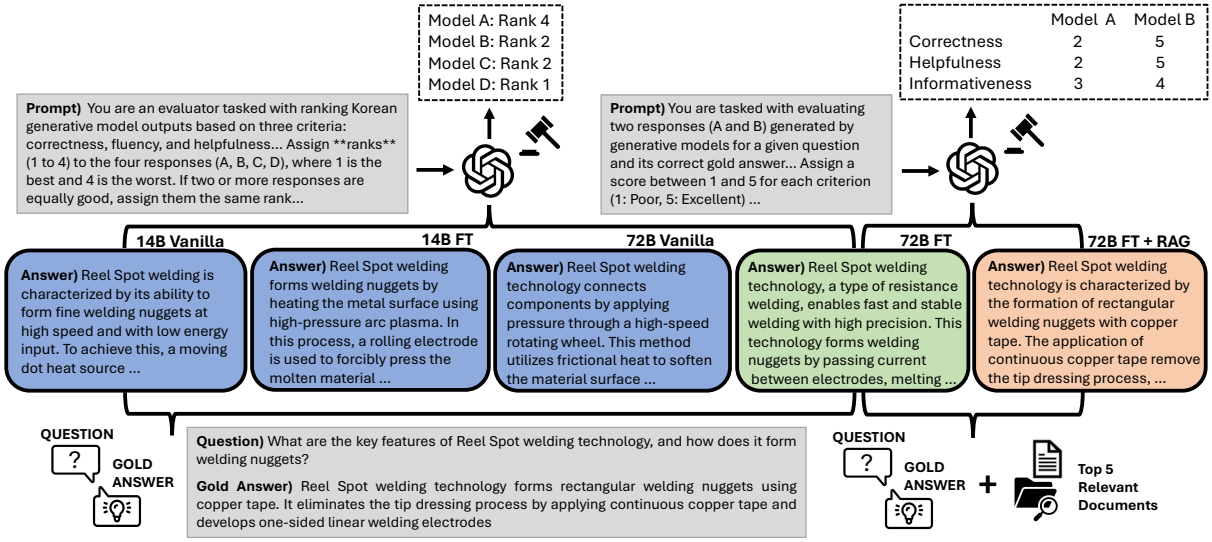

Figure 5: Final answer evaluation pipeline for the QA system, illustrating the scoring and ranking of model-generated responses using the given question, gold answer, and reference documents.

currently no Korean-centric open-source LLMs with sufficient context length to address our requirements. We fine-tune Qwen 2.5 (14B, 72B) on our training dataset, performing full fine-tuning on the 14B model and applying Low-Rank Adaptation (LoRA) (Hu et al., 2021) for the 72B model. Experimental details including hyperparameters and GPU configurations, are in A.1.2.

5.2 Evaluation

5.2.1 LLM-as-a-Judge

As illustrated in Figure 5, we employ the LLM-as-a-judge approach (Zheng et al., 2023) to evaluate the performance of both LLM-only and RAG models. For the LLM evaluation, we compare four model variations: the vanilla and fine-tuned versions of 14B and 72B models. GPT-4o[4] is used to rank anonymized models (A, B, C, and D) based on factual correctness, helpfulness, and informativeness. These metrics are chosen to evaluate distinct, non-overlapping dimensions:

- **Correctness** measures alignment with the gold answer, rewarding correct responses without hallucinations.
- **Helpfulness** assesses clarity and relevance in addressing key points, while avoiding unnecessary content.
- **Informativeness** evaluates the inclusion of relevant details or additional context that enhances completeness.

[4] https://platform.openai.com/docs/models/gpt-4o

For RAG evaluation, we use the best-performing one among four models and compare its performance with and without RAG integration. Instead of pairwise comparisons, we employ single-answer evaluations, scoring responses (1–5) on three aspects. This approach offers a more detailed assessment, emphasizing how RAG impacts specific aspects of response quality. All the evaluation prompts used are detailed in A.2.2.

5.2.2 Human Evaluation

To assess the reliability of the RAG evaluation using the LLM-as-a-judge method, we conducted a human evaluation to compare its alignment with the LLM evaluation. An expert evaluator, a native Korean automotive engineer, assessed 100 randomly selected and anonymized responses. These responses were generated by the 72B fine-tuned model, both with and without RAG integration. A screenshot of the human evaluation interface used is in shown Figure 22 (A.5).

5.3 Result

5.3.1 LLM-only Comparison

Figure 6 illustrates the rank distribution and win-lose-tie comparison across four Qwen models. Among them, the 72B fine-tuned one shows the best performance, followed by the 72B vanilla, 14B fine-tuned, and 14B vanilla models. These results emphasize the critical impact of model size and demonstrate the significant effectiveness of fine-tuning, as fine-tuned models consistently outperform their vanilla counterparts.

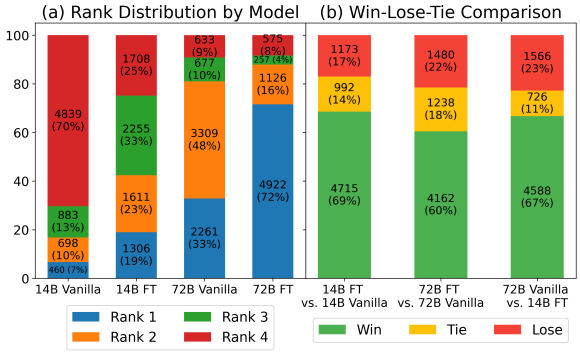

Figure 6: LLM-only performance comparison: **(a)** Proportion of responses from the four models ranked 1st, 2nd, 3rd, or 4th, **(b)** Win, tie, and loss rates for selected model pairs, where a "win" indicates the first model in the pair outperformed the second model.

5.3.2 LLM-only vs. RAG

We compare the performance of the 72B fine-tuned model in its LLM-only (non-RAG) and RAG-integrated versions. Table 5 compares the average scores for correctness, helpfulness, and informativeness between the two versions, while Figure 7 shows the win rates. The results clearly demonstrate that integrating RAG with the LLM improves factual correctness by reducing hallucinations and provides more helpful and informative answers.

Metric	non-RAG	RAG
Correctness	2.18	**4.12** (+1.94)
Helpfulness	2.52	**4.19** (+1.67)
Informativeness	2.61	**3.77** (+1.16)

Table 5: Average scores (1 to 5 scale) for the model with and without RAG across three metrics. Detailed counts and score distributions are in Table 11 (A.3.3).

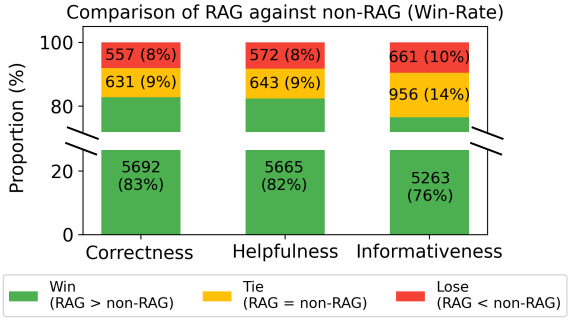

Figure 7: Win-Tie-Lose comparison of the model with and without RAG across three evaluation metrics.

5.3.3 Comparison with Human Evaluation

Table 6 shows that human evaluation aligns with GPT evaluation, both assigning higher scores to the RAG model across all three aspects. Figure 8 visualizes the differences in scoring tendencies between human and GPT evaluations. Notably, correctness and informativeness show an inverted pattern: human favors more polarized correctness scores (1 and 5), while GPT prefers mid-range (2 and 3). Additionally, human assigns higher helpfulness scores more frequently, as seen in the positive values for 4 and 5. While differences in score distributions exist between evaluators, these differences are independent of the specific models. Both evaluations consistently show that RAG outperforms non-RAG by achieving higher scores.

Metric	Model	Avg. Score	1	2	3	4	5
Correctness	A	2.54	34	19	20	13	14
	B	**4.33**	10	4	4	7	75
Helpfulness	A	2.89	2	38	36	17	7
	B	**4.22**	3	9	6	27	55
Informativeness	A	2.60	3	44	44	8	1
	B	**3.68**	2	8	31	38	21

Table 6: Human evaluation result comparing non-RAG (anonymized as A) and RAG (anonymized as B).

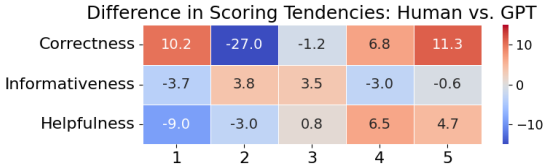

Figure 8: Heatmap illustrating the differences in score distributions between human and GPT evaluations, calculated as the percentage of scores (ranging from 1 to 5) assigned by humans, subtracted by the corresponding percentages assigned by GPT. Positive values indicate scores more frequently given by humans, while negative values indicate scores more frequently given by GPT.

6 Experiment 3: Reference Matching

6.1 Evaluation

To evaluate the reference matching task, we randomly sample 100 triplets $(q, \mathbf{D}_{\text{top}_n}, a')$ from the test set and perform human annotation to identify which chunk ($d_i \in \mathbf{D}_{\text{top}_n}$) each sentence in the generated answer (a') referenced. Each sentence is annotated with one or more chunks, allowing for multiple references. The interface of the annotation tool is in Figure 23 (A.5). We use sentence-level precision as the evaluation metric, defined as the proportion of sentences where the model prediction is included in the annotated reference set for an answer, averaged over the entire evaluation set. Since references are identified for concatenated sentences, selecting the best match for the grouped information is key. Recall would unfairly penalize valid predictions for not selecting every reference.

Question: Has there been an example where improving the vehicle structure prevented the fracture of components?			
Segment 1	**Sentence 1**	Yes, there have been cases where improving the vehicle structure prevented the fracture of components.	**Chunk 1,2,3**
Segment 2	**Sentence 2**	First, in the 'HDC EV' project, the design of the front bumper beam (FRT B/Beam) was changed to prevent fracture.	**Chunk 2**
	Sentence 3	In the T-CAR model, fracture occurred in the FRT B/Beam, but in the Proto model, it was modified to a closed cross-section shape, preventing fracture.	
Segment 3	**Sentence 4**	Additionally, in the 'HGC' project, body structure improvements were made to reduce the risk of deformation and fracture of several components.	**Chunk 1**
	Sentence 5	For example, reinforcing the upper part of the A-pillar and the COWL BAR improved bending deformation.	
	Sentence 6	Adding UPR HINGE B/H beads and other deformation-guiding methods reduced opening misalignment.	
Segment 4	**Sentence 7**	Finally, in the 'QYZ' project, a plan was proposed to improve the tendency of fracture by adding reinforcements to the S/MBR RR lower end.	**Chunk 3**
Segment 5	**Sentence 8**	All of these improvements aim to optimize the body structure and prevent component fracture during collisions.	**Chunk 1,2,3**

Figure 9: An example where multiple chunks need to be referred to in order to answer the question, in contrast to Figure 20 (A.4), which requires a single chunk.

6.2 Result

Table 7 compares the reference matching performance of our algorithm (Section 3.4) without thresholding and the fine-tuned Qwen 72B, while Figure 10 illustrates the score distribution and the increase in precision with score thresholding for our algorithm. Without thresholding, our algorithm achieves a sentence-level precision of 0.72, while the LLM reaches 0.81, showing a noticeable gap in performance. However, when thresholding is applied, the precision reaches 0.86 at a threshold of 0.5, with performance improving proportionally as the threshold is increased further. This demonstrates the reliability of our algorithm's scoring, allowing for controlled adjustment of reference matching quality through threshold selection. Our re-ranker used in the matching algorithm has only 0.5B parameters, compared to the LLM's 72B, demonstrating that our algorithm can achieve impressive performance with a smaller model and faster inference speed. The LLM prompt used for reference matching is in Figure 17 (A.2.3).

Method	**Segment**	**Sentence-level Precision**	**Inference Time**
Matching Algorithm	2.7 ($\pm$ 1.5)	0.72 ($\pm$ 0.27)	3.92
LLM	1.9 ($\pm$ 0.9)	0.81 ($\pm$ 0.26)	13.54

Table 7: Performance comparison of our reference matching algorithm (without thresholding) and the fine-tuned Qwen 72B. **Segment** is the average number of segments per answer, **Sentence-level Precision** is the average proportion of successfully matched sentences, and **Inference Time** is the average time (in seconds) to complete reference matching for one answer.

The answers are divided into 2-3 segments on average, and we identify two types of reference matching: one where all the necessary information to answer the question is contained within a single chunk, resulting in a single-segment answer, and another where information from multiple chunks is needed to answer the question, leading to a multi-segment answer. The first type often occurs in factual questions related to specific vehicle crash collision test, while the second type is more common in open-ended questions that require referencing multiple tests (Figure 9).

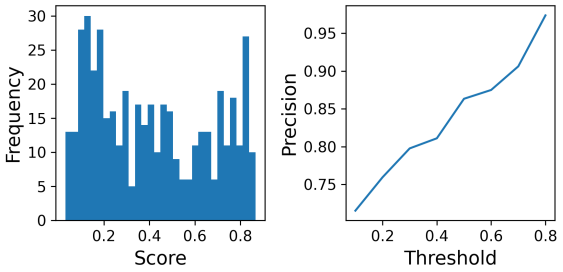

Figure 10: Distribution of matching scores from our reference matching algorithm (left) and the relationship between precision and the score threshold (right). Increasing the threshold leads to a proportional improvement in matching precision.

7 Conclusion

In this work, we presented a low-resource approach to RAG for domain expert QA, utilizing heterogeneous data sources. We showed that our data generation pipeline, guided by expert-driven prompting, facilitates an effective RAG system by converting multi-modal raw data. By enhancing the retrieval phase with re-ranker, we demonstrated that the final answers are both accurate and traceable to relevant sources. These findings highlight the potential of RAG in data-scarce domains and its applicability to specialized knowledge-intensive tasks. As future work, we are exploring open-source Vision models to better handle multi-modal questions, addressing current limitations discussed in Section 8.

8 Limitations

Constraints in Backbone Model Selection The selection of a backbone model was constrained by the need for an open-source model supporting Korean with sufficient context length, limiting the available options across retrieval and LLM components. BGE-M3 for retriever & re-ranker, along with Qwen2.5 for the LLM, were the options available to us. However, it is worth noting that the newly released Qwen2.5-1M (Yang et al., 2025) offers a longer context length, and our preliminary experiments demonstrated its potential for better handling long inputs. As the availability of open-source LLMs continues to grow, we anticipate that this limitation will gradually be resolved over time.

Restricted Use of API-based LLM for Initial Data Processing While our ultimate goal is to develop a fully local system, we initially relied on the API-based LLM, specifically the Claude API, for limited data processing. Among the available open-source LLMs, none met the requirements due to context limits and certain performance considerations, making them unsuitable for our needs. We determined that leveraging an API-based LLM with expert-driven prompting would be more effective in terms of both quality and cost compared to full human annotation. However, this reliance was limited to the initial data required for fine-tuning in the open-source model, and any future updates to the document pool will completely eliminate the need for external dependencies.

Weakness in Multi-modal Questions Our RAG system exhibited relatively lower performance when retrieving contexts for multi-modal specific questions that require information from tables, graphs, and images to answer the given question (Section 4.3.1). We hypothesize that this is due to the loss of rich content when multi-modal elements are converted into plain text during the data processing stage. To address this, we are experimenting with Llama 3.2 Vision[5] to improve the conversion of multi-modal elements into text, aiming for better representation of these elements.

Experiments Limited by Computational Constraints The number of experiments we could conduct on the generation model was constrained by computational limitations. Fine-tuning the model on $(q, a, D_{\text{top}k})$ instead of (q, a) (Section 3.3) proved impractical due to the extensive context length, which exponentially increased GPU memory requirements, surpassing our available computational resources. Although we observed that Qwen models inherently possess summarization capabilities for the given D_{top_k}, and fine-tuning on (q, a) yielded effective results, we were unable to explore a variety of learning strategies for the generation model within the RAG framework.

[5] https://huggingface.co/meta-llama/Llama-3.2-11B-Vision

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Anthropic. 2024. Claude: An anthropic large language model.

David Beauchemin, Richard Khoury, and Zachary Gagnon. 2024. Quebec automobile insurance question-answering with retrieval-augmented generation. In *Proceedings of the Natural Legal Language Processing Workshop 2024*, pages 48–60, Miami, FL, USA. Association for Computational Linguistics.

Richard Bellman. 1954. The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60(6):503–515.

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

Nayoung Choi, Youngjune Lee, Gyu-Hwung Cho, Haeyu Jeong, Jungmin Kong, Saehun Kim, Keunchan Park, Sarah Cho, Inchang Jeong, Gyohee Nam, Sunghoon Han, Wonil Yang, and Jaeho Choi. 2024. RRADistill: Distilling LLMs' passage ranking ability for long-tail queries document re-ranking on a search engine. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 627–641, Miami, Florida, US. Association for Computational Linguistics.

Kaisi Guan, Qian Cao, Yuchong Sun, Xiting Wang, and Ruihua Song. 2024. BSharedRAG: Backbone shared retrieval-augmented generation for the E-commerce domain. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1137–1158, Miami, Florida, USA. Association for Computational Linguistics.

Rujun Han, Yuhao Zhang, Peng Qi, Yumo Xu, Jenyuan Wang, Lan Liu, William Yang Wang, Bonan Min, and Vittorio Castelli. 2024. RAG-QA arena: Evaluating domain robustness for long-form retrieval augmented question answering. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 4354–4374, Miami, Florida, USA. Association for Computational Linguistics.

Seongtae Hong, Joong Min Shin, Jaehyung Seo, Taemin Lee, Jeongbae Park, Cho Man Young, Byeongho Choi, and Heuiseok Lim. 2024. Intelligent predictive maintenance RAG framework for power plants: Enhancing QA with StyleDFS and domain specific instruction tuning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 805–820, Miami, Florida, US. Association for Computational Linguistics.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81.

Ilya Loshchilov and Frank Hutter. 2017. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, USA.

Kelong Mao, Zheng Liu, Hongjin Qian, Fengran Mo, Chenlong Deng, and Zhicheng Dou. 2024. RAG-studio: Towards in-domain adaptation of retrieval augmented generation through self-alignment. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 725–735, Miami, Florida, USA. Association for Computational Linguistics.

Koji Mizuno. 2016. *Crash Safety of Passenger Vehicles*. Golden Bell. Translated by Kyungwon Song; Edited by Inhwan Han, Jongjin Park, Sungjin Kim, Jongchan Park, and Namkyu Park. Original work published as 自動車の衝突安全.

Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document ranking with a pre-trained sequence-to-sequence model. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 708–718, Online. Association for Computational Linguistics.

Rodrigo Frassetto Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-stage document ranking with BERT. *CoRR*, abs/1910.14424.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–318.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2383–2392.

Jon Saad-Falcon, Omar Khattab, Christopher Potts, and Matei Zaharia. 2024. ARES: An automated evaluation framework for retrieval-augmented generation systems. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 338–354, Mexico City, Mexico. Association for Computational Linguistics.

Shamane Siriwardhana, Rivindu Weerasekera, Elliott Wen, Tharindu Kaluarachchi, Rajib Rana, and Suranga Nanayakkara. 2023. Improving the domain adaptation of retrieval augmented generation (RAG) models for open domain question answering. *Transactions of the Association for Computational Linguistics*, 11:1–17.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Xiaohua Wang, Zhenghua Wang, Xuan Gao, Feiran Zhang, Yixin Wu, Zhibo Xu, Tianyuan Shi, Zhengyuan Wang, Shizheng Li, Qi Qian, Ruicheng Yin, Changze Lv, Xiaoqing Zheng, and Xuanjing Huang. 2024. Searching for best practices in retrieval-augmented generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17716–17736, Miami, Florida, USA. Association for Computational Linguistics.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

An Yang, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoyan Huang, Jiandong Jiang, Jianhong Tu, Jianwei Zhang, Jingren Zhou, Junyang Lin, Kai Dang, Kexin Yang, Le Yu, Mei Li, Minmin Sun, Qin Zhu, Rui Men, Tao He, Weijia Xu, Wenbiao Yin, Wenyuan Yu, Xiafei Qiu, Xingzhang Ren, Xinlong Yang, Yong Li, Zhiying Xu, and Zipeng Zhang. 2025. Qwen2.5-1m technical report. *Preprint*, arXiv:2501.15383.

Hao Yu, Aoran Gan, Kai Zhang, Shiwei Tong, Qi Liu, and Zhaofeng Liu. 2024. Evaluation of retrieval-augmented generation: A survey. *Preprint*, arXiv:2405.07437.

Yiyun Zhao, Prateek Singh, Hanoz Bhathena, Bernardo Ramos, Aviral Joshi, Swaroop Gadiyaram, and Saket Sharma. 2024. Optimizing llm based retrieval augmented generation pipelines in the financial domain. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 6: Industry Track)*, pages 279–294.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Preprint*, arXiv:2306.05685.

A Appendix

A.1 Experimental Details

A.1.1 Dataset

We split the generated dataset at the chunk level, slide level for slides, and chapter level for the textbook, using an 8:1:1 ratio for training, validation, and testing. Table 8 summarizes the dataset splits, while token length statistics for the training data are presented in Table 9.

Source	Data	Train	Val	Test	Total
Test Report	Chunk	3,729	466	467	4,662
	Q&A Pair	47,660	5,823	5,919	59,402
Meeting Report	Chunk	705	88	89	882
	Q&A Pair	6,144	752	800	7,696
Textbook	Chunk	64	8	9	81
	Q&A Pair	1,182	162	161	1,505

Table 8: Dataset split statistics by source, detailing the distribution of chunks and Q&A pairs.

	Min	Mean	Max
Question	9	35	82
Answer	6	56	271
Chunk (`Ver.0`)	66	843	9,528
Chunk (`Ver.1`)	106	884	9,528

Table 9: Token length statistics in the training data, processed with the BGE M3 tokenizer. `Ver.0` refers to the raw chunk without the pre-pended header, while `Ver.1` includes header.

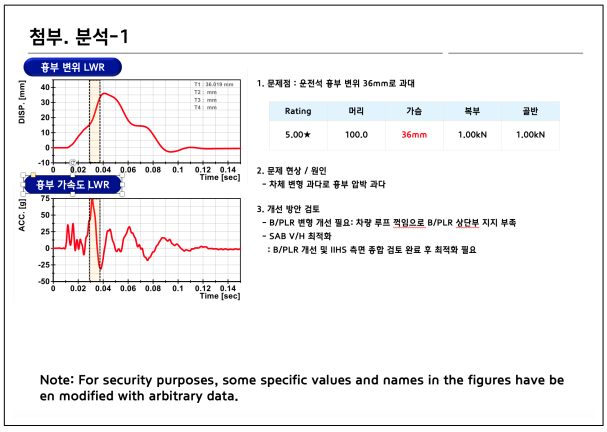

Figure 11: A screenshot of an original PPT slide from an internal report on vehicle crash collision tests by an automotive company in Korea.

A.1.2 Model Training

The retriever was trained for 10 epochs, determined to be optimal, using 4 × 48GB RTX A6000 GPUs. The hyperparameters were set to the default configuration provided in the open-source implementation of BGE-M3. For the ranker, early stopping was employed by validating the model every 1,000 steps. Training was stopped if performance did not improve after 3 consecutive validations, and the best-performing model was selected. This training used 1 × 48GB RTX A6000 GPU with a learning rate of 1e-5, the AdamW optimizer (Loshchilov and Hutter, 2017), and a batch size of 4. For both components, validation was based on the MAP@10 metric evaluated on the validation set. To train our answer generation model, full fine-tuning was conducted for the 14B model with a batch size of 2, gradient accumulation steps of 64, a learning rate of 2e-5, and 3 training epochs using 3 × 80GB H100 GPUs. For the 72B model, LoRA was applied with a batch size of 1, gradient accumulation steps of 64, a learning rate of 2e-5, and 3 training epochs using 2 × 141GB H200 GPUs.

A.2 Prompt Engineering

A.2.1 Prompts for Data Generation

The prompt used to convert slides into Markdown text is shown in Figure 12, and the prompt for Q&A generation is depicted in Figure 13. An example of a generated Q&A pair from a chunk is shown in Figure 14. An average of twelve Q&A pairs were generated from a single chunk.

A.2.2 Prompts for Final Answer Evaluation

Our prompts for evaluating final answers from LLM-only models and comparing LLM-only and RAG models are in Figure 15 and 16, respectively.

A.2.3 Prompts for Reference Matching

The prompt used for LLM reference matching is shown in Figure 17.

A.3 Further Analysis

A.3.1 TS+TCL Effect on Ranker Training

Table 10 shows the re-ranking performance without TS+TCL training. When compared to the performance with TS+TCL training (Table 3), there is a noticeable decline across all metrics.

A.3.2 Analysis of Retrieval & Ranking Phase

Figure 18 and 19 present real-world examples of two error types: (a) cases where the top-retrieval process failed, and (b) cases where retrieval succeeded but the ranker's top-1 result did not match the gold context.

Prompt

1. Examine the provided image of a PowerPoint slide from a car collision safety report.
2. Convert all content into markdown format, preserving the original structure and information.
3. Use a single hashtag # for section headings only. Do not use any variation of hashtags for anything else. Only use one hashtag for each section heading. All heading MUST have section content underneath it. There should never be two headings in a row without content in between.
4. Each graphic element type should have a section heading. This means tables, graphs, and images should each have their own section heading. Text should be included in the appropriate section or if different enough, have its own section.
5. Structure your output as follows:
   < markdown_conversion >
   # Heading appropriate for section content underneath
   [section content goes here]
   < markdown_conversion >
6. For each element type:
   Text:
   - Convert to markdown, maintaining structure, bullet points, and numbering.
   - Include all text: titles, subtitles, and body.
   Tables:
   - Convert to markdown table format.
   - Maintain original content and structure.
   - Provide a detailed interpretation after each table.
   - For images in cells, describe them within the appropriate cell in markdown.
   Graphs and Images:
   - Describe in detail: type, labels, data representation, trends.
   - Explain significance in the context of car collision safety under it.
   Remember:
   - Do not use unnecessary affirmations or filler phrases.
   - Do not include personal opinions or anecdotes.
   - Use markdown for code snippets if applicable.

Figure 12: Prompt used to convert a presentation slide into markdown text.

k	MAP@1	MAP@5	MAP@10	Success@5
10	0.6349 (± 0.013)	0.7341 (± 0.009)	0.7394 (± 0.009)	0.8801 (± 0.002)
20	0.6271 (± 0.017)	0.7276 (± 0.013)	0.7344 (± 0.012)	0.8769 (± 0.006)
30	0.6237 (± 0.019)	0.7243 (± 0.014)	0.7314 (± 0.014)	0.8748 (± 0.008)

Table 10: Re-ranker performance with TS+TCL training disabled.

A.3.3 LLM-only vs. RAG

Table 11 presents detailed counts and score distributions from GPT-4o, comparing RAG and non-RAG.

A.4 Analysis of Reference Matching

An example where only a single chunk needs to be referred to answer the question is in Figure 20.

A.5 Human Evaluation Interface

The human evaluation interfaces for assessing the retrieval and re-ranking phase, as well as the reliability of the RAG evaluation with LLM, are shown in Figures 21 and 22, respectively. The human evaluation interface used for annotating the evaluation dataset to aseess the reference matching task is shown in Figure 23.

Metric	Score	Non-RAG	RAG
Correctness	1	1,640	388
	2	3,166	579
	3	1,457	705
	4	430	1,359
	5	187	3,849
	Avg. Score	2.1799	**4.1195**
Helpfulness	1	758	154
	2	2,819	522
	3	2,422	868
	4	722	1,649
	5	159	3,687
	Avg. Score	2.5211	**4.1908**
Informativeness	1	461	142
	2	2,764	564
	3	2,789	1,609
	4	756	2,974
	5	110	1,591
	Avg. Score	2.6061	**3.7715**

Table 11: Comparison of the Non-RAG and RAG models, reporting the distribution of scores (1–5) assigned to each model's outputs.

Prompt

You are an AI assistant tasked with creating Q&A sets based on a car crash safety test report. Your goal is to generate question-answer pairs that can be used for training a language model. Follow these instructions carefully:
1. Read the following crash test report: <crash_test_report> {crash_test_report} </crash_test_report>
2. Extract the test information from the head of the report. It should contain the following details: Test ID, PRJ ID, Region, Test Name, Stage, Purpose
3. Create Q&A pairs based on the information provided in the report and the test information. The number of pairs should be sufficient to cover the report's content adequately. Follow these guidelines:
a. Focus on creating questions that a human would naturally ask about the crash test results, vehicle performance, or safety implications.
b. Create a diverse set of questions covering different aspects of the crash test, such as test conditions, vehicle performance, safety features, and results.
c. Ensure that all questions can be answered solely based on the information provided in the report and test information.
d. Include a mix of factual questions and more analytical questions.
e. Avoid creating questions that require information not present in the report or test information.
f. Create meaningful and insightful questions that provide valuable information about the crash test and its results.
g. Write all questions and answers in Korean.
h. Incorporate the test information (Test ID, PRJ ID, Region, Test Name, Stage, Purpose) into your questions where relevant, but don't create questions that simply ask for this information directly.
j. When referring to the test in questions, use the following format: "[PRJ ID] [Region] [Stage] [Test Name]" (e.g., "HDC Domestic Proto 64kph Offset 40% (LH) Test"). Do not use phrases like "The 64kph Offset 40% (LH) test of the HDC prototype conducted in Korea."
k. When referring to the region in questions, use "Domestic" instead of "Korea" and "North America (USA)" instead of "USA."
4. Format your output as follows:
<qa_pairN>
<question> Write your N th question here</question>
<answer>Write the corresponding answer here</answer>
</qa_pairN>
5. Additional tips for creating meaningful questions:
a. Focus on questions that relate to vehicle safety, performance, and test outcomes rather than technical details of the test setup.
b. Consider creating questions about specific safety features mentioned in the report and their effectiveness.
c. Include questions about the overall safety rating or performance of the vehicle, if such information is provided.
d. Ask about any notable findings or unusual results mentioned in the report.
e. Create questions that compare the results to safety standards or expectations, if such information is available.
6. Create questions that can be definitively answered based solely on the information provided in the crash test report and test information. Do not include any information or assumptions that are not explicitly stated. Ensure that your questions and answers are meaningful, insightful, and provide valuable information about the crash test and its results.
7. Before finalizing your Q&A pairs, review them to ensure they are diverse, meaningful, and directly relevant to the crash test report and test information provided. Make sure that each question includes the Region and Test Name along with the PRJ ID and Stage where appropriate, using the format specified in guideline 3j.
8. Use the following explanations for the test information components:
- Test ID: A unique test number used to distinguish tests (this number is rarely included directly in questions).
- PRJ ID: The name of the vehicle model (project name) under development.
- Region: The country where the crash test is conducted.
- Test Name: The type of crash test, such as frontal test or side test.
- Stage: The development stage.
- Purpose: The purpose of conducting the test.
9. The number of Q&A pairs you create should be flexible based on the content of the report. If the report contains a lot of information, you may create up to 30 pairs. If the report has limited content, create fewer pairs, but ensure that all relevant information from the report is covered in the Q&A sets. The goal is to adequately represent the report's content without creating redundant or overly similar questions.
10. In addition to the standard question-answer format, create some Q&A pairs using the following alternative format:
a. Ask about specific phenomena, results, or improvements mentioned in the report.
b. In the answer, specify which test produced these results or phenomena.
c. Use the following structure for these alternative Q&A pairs:
<question> Ask about a specific phenomenon, result, or improvement.</question>
<answer>[PRJ ID] [Region] [Stage] [Test Name] The phenomenon/result/improvement was observed during the test. (Additional explanation) </answer>
Example:
<question> Has there been a case where moving the weld point location increased the survival space?</question>
<answer>In the HDC North America (USA) P2 60kph Side 90 (LH) IIHS test, a measure was implemented to increase survival space by moving the weld point location. As a result, the survival space rating is expected to improve from 'A' to 'G'.</answer>
Generate your Q&A pairs now, following all the guidelines and instructions provided above. Include a mix of standard and alternative format Q&A pairs to provide a comprehensive representation of the crash test report's content.

Figure 13: Prompt used to generate Q&A pairs from a given chunk

Chunk

Attachment 4. Kinematic Energy

1. HDC1 Energy Analysis: Comparison of Front & A/PLR absorption rates based on the timing of TIRE contact with S/SILL (using ACU X & Y data), and setting the Final Time considering the vehicle's acceleration.

Kinematic Energy Data Table

Category	Initial Kinetic	Absorbed Energy		Final Kinetic		Remarks
		Front	A/PLR	Final X	Final Y	
HDC 1	428.8	19.4%	63.1%	9.0%	8.6%	45ms, 150ms
GDC 2	532.4	21.2%	72.3%	2.5%.	4.0%	45ms, 150ms
QYZ base	261.3	37.8%	46.8%	6.4%	7.1%	
LFA	261.3	21.7%	52.0%	1.4%	24.5%	
GPC	321.6	34.5%	39.7%	19.1%	6.6%	

This table illustrates the distribution of kinematic energy across various vehicle models. Initial Kinetic refers to the initial kinetic energy, absorbed Energy represents the proportion of energy absorbed by the Front and A/PLR, and Final Kinetic indicates the final energy distribution on the X-axis and Y-axis. For the HDC 1 and GDC 2 models, ...

Question Which model exhibited the lowest Front Energy Absorption Rate during the HDC 1 64kph small overlap 25% (RH) test?

Answer The model with the lowest Front Energy Absorption Rate during the QT 64kph small overlap 25% (RH) test is the HDC 1 model, with a rate of 19.4%.

Figure 14: An example of a generated QA pair from the chunk, originally in Korean and translated into English.

Prompt

You are an evaluator tasked with ranking Korean generative model outputs based on three criteria: accuracy, fluency, and helpfulness.

1. **Accuracy**: Evaluate how closely each response aligns with the given gold answer. Responses that are factually correct and contain relevant information rank higher.
2. **Fluency**: Evaluate the quality of the language. Responses written in natural and grammatically correct Korean rank higher. Responses with awkward phrasing, grammar mistakes, or written in other languages rank lower.
3. **Helpfulness**: Evaluate how effectively the response answers the question. Clear and informative answers rank higher, while incomplete or overly verbose responses rank lower.

Instructions:

- Assign **ranks** (1 to 4) to the four responses (A, B, C, D), where 1 is the best and 4 is the worst.
- If two or more responses are equally good, assign them the same rank.
- Provide ranks in the following format:
  A: [rank]
  B: [rank]
  C: [rank]
  D: [rank]

Examples:

1. If all responses are equally good:
   A: 1
   B: 1
   C: 1
   D: 1
2. If A is the best, B and C are tied for second place, and D is the worst:
   A: 1
   B: 2
   C: 2
   D: 4
3. If all responses have distinct ranks:
   A: 1
   B: 2
   C: 3
   D: 4

Input:

Question: {question}
Gold Answer: {correct_answer}
A: {response_a}
B: {response_b}
C: {response_c}
D: {response_d}

Figure 15: Prompt used to compare answers from LLM-only models

Prompt

You are tasked with evaluating two responses (A and B) generated by generative models for a given question and its correct gold answer. Evaluate each response based on the following three criteria, and assign a score between 1 and 5 for each criterion:

1. **Accuracy**: How well does the response align with the gold answer without hallucination? Factual correctness is essential.
2. **Helpfulness**: How helpful is the response in answering the question? A helpful response addresses the key points clearly and avoids unnecessary content.
3. **Informativeness**: How informative is the response in addressing the question? An informative response includes relevant details or additional context derived from the provided sources, enhancing the overall completeness and clarity of the answer.

Instructions:

- Assign a score between 1 and 5 for each criterion (1: Poor, 5: Excellent).
- Provide only the scores for each aspect without any explanations.
- Use the following exact format to provide scores:
  A_Accuracy: [ ]
  B_Accuracy: [ ]
  A_Helpfulness: [ ]
  B_Helpfulness: [ ]
  A_Informativeness: [ ]
  B_Informativeness: [ ]

Input:

Question: {question}
Gold Answer: {correct_answer}
Response A: {response_a}
Response B: {response_b}

Figure 16: Prompt used to compare answers from LLM-only and RAG models

Prompt

You are an expert tasked with analyzing user questions, an answer split into sentences, and a list of 5 related chunks. Your job is to identify the single most relevant chunk for each sentence based on sentence indices provided. Map the segments to relevant chunks as follows:

Example:
  sentence_idx [0,1,2]: chunk 0
  sentence_idx [3,4]: chunk 3
  sentence_idx [5]: chunk 4

Follow the format above and do not provide additional explanations.
  Question: {question}
  Answer: {Answer with sentence_idxs}
  Documents: {chunks}

Figure 17: Prompt used for LLM reference matching

Question: Were there cases where the body rating significantly decreased?	
Model Top-1 Chunk	**Gold Chunk**
— Region: USA Test Name: 64kph Small Overlap 25% (RH) Stage: T-Car Purpose: Evaluation of HDC Small Overlap Improvements — **Test Conditions** The table below compares the test conditions for the two trials of HDC. In the second trial, the speed slightly increased, the weight decreased, and the impact location shifted. ... **Evaluation Results** The evaluation results indicate a decline in ratings between the first and second tests. Target: Overall rating - Good (G), Body rating - Good (G) 1st Test: Overall rating - G+0, Body rating - Acceptable (A) 2nd Test: Overall rating - A+1, Body rating - Marginal (M) All other ratings, including restraint/dummy behavior and passenger injuries, remained at "Good" (G) for both tests. ...	— Region: USA Test Name: 64kph Small Overlap 25% (RH) Stage: Proto Purpose: Development of GTC Proto Performance. — **Body Deformation Analysis** The following table shows the deformation measurements at various vehicle sections during a crash test. The red line represents the evaluation model, while the blue line represents the baseline model (P2).Measurements are shown in centimeters and are categorized into 'GOOD', 'ACCEPTABLE', 'MARGINAL', and 'POOR' ratings. ... **Body Deformation Table** {Table} The table provides a detailed breakdown of deformation measurements for each section of the vehicle body. The baseline model (P2) achieved an overall rating of 'G' (Good), while the evaluation model received an overall rating of 'A' (Acceptable). ...

Figure 18: Comparison of Retriever Top-1 Result and Gold Chunk for the Question *"Were there cases where the body rating significantly decreased?"*. The blue text highlights the parts relevant to the question, demonstrating that the top-1 retrieved chunk is as acceptable as the gold chunk.

Question: What differences are observed in the head acceleration graphs for the driver and passenger seats during the HDC P9 60kph side 90° (LH) AEMDB test?	
Model Top-1 Chunk	**Gold Chunk**
— Region: Europe Test Name: 60kph Side 90° (LH) AEMDB Stage: P9 Purpose: Performance evaluation of HDC P9 — **Driver Seat Head Acceleration Graph** {Table} The graph illustrates the head acceleration of the driver seat over time. The maximum acceleration reaches approximately 40G at 0.05 seconds, followed by a sharp decline and subsequent fluctuations. ... **Passenger Seat Head Acceleration Graph** {Table} The passenger seat head acceleration graph shows a delayed response compared to the driver seat. The maximum acceleration is lower, at approximately 30G. While the overall impact duration is similar, the acceleration pattern differs slightly. ...	— Region: Europe Test Name: 60kph Side 90° (LH) AEMDB Stage: P9 Purpose: Performance evaluation of HDC P9 — **Driver Seat Head Acceleration Graph** {Table} The graph illustrates the head acceleration of the driver seat over time. The maximum acceleration reaches approximately 40G at 0.05 seconds, followed by a sharp decrease and subsequent fluctuations. ... **Passenger Seat Head Acceleration Graph** {Table} The passenger seat head acceleration graph shows a delayed response compared to the driver seat. The maximum acceleration is lower, at approximately 30G. While the overall impact duration is similar, the acceleration patterns differ slightly. ...

Figure 19: Comparison of Re-ranker Top-1 Result and Gold Chunk for the Question *"What differences are observed in the head acceleration graphs for the driver and passenger seats during the HDC P9 60kph side 90° (LH) AEMDB test?"*. The blue text highlights the parts relevant to the question, demonstrating that the top-1 ranked chunk is as acceptable as the gold chunk.

Question: What was the material of the back beam used in the HDC P5 40kph slope 30 (RH) test?				
Answer	**Segment 1**	**Sentence 1**	The material of the back beam used in the HDC P5 40kph slope 30 (RH) test was aluminum.	**Chunk 1**
		Sentence 2	This information can be found in the test conditions section.	

Figure 20: An example where the question can be answered by referring to only a single chunk.

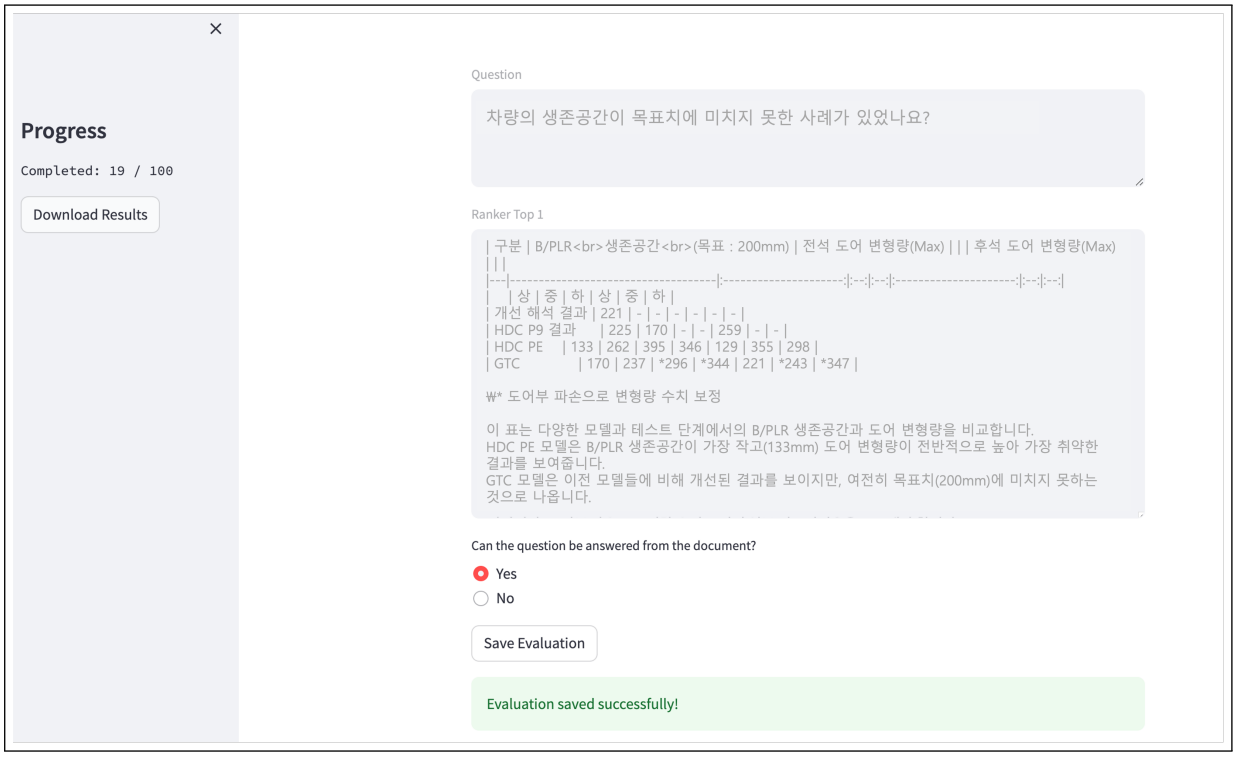

Figure 21: A screenshot of the user interface used for human evaluation to assess the retrieval & re-ranking phase.

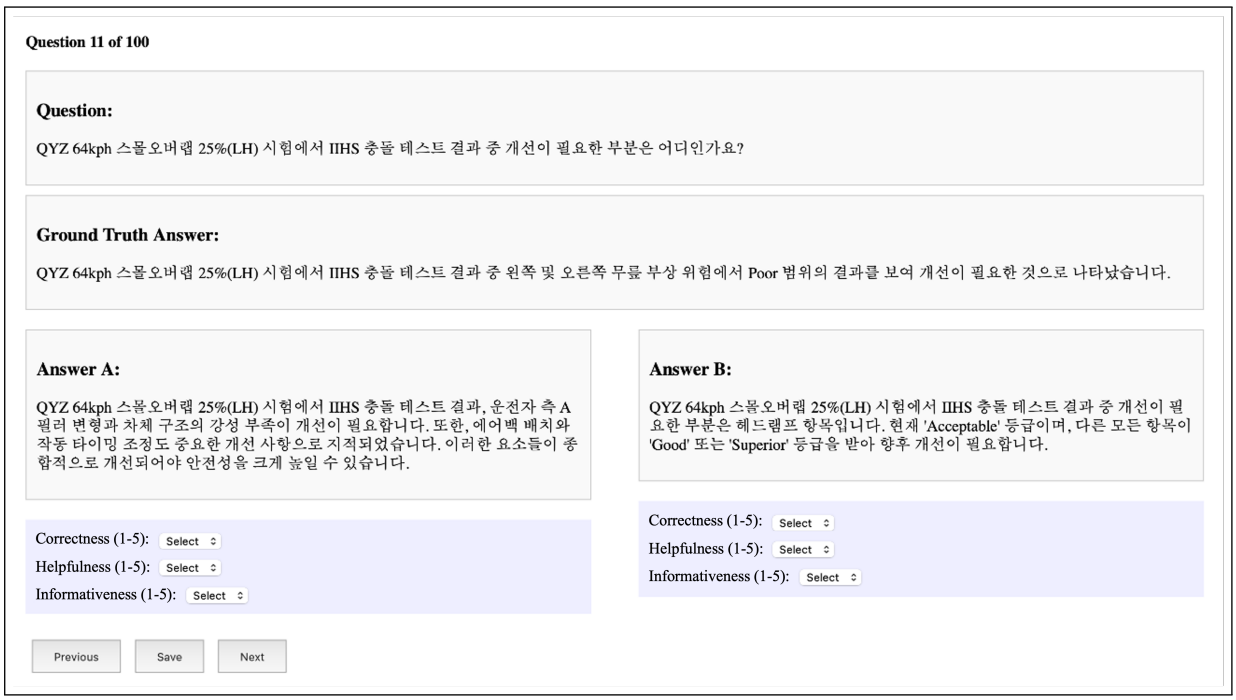

Figure 22: A screenshot of the user interface used for human evaluation to assess the reliability of the RAG evaluation with LLM.

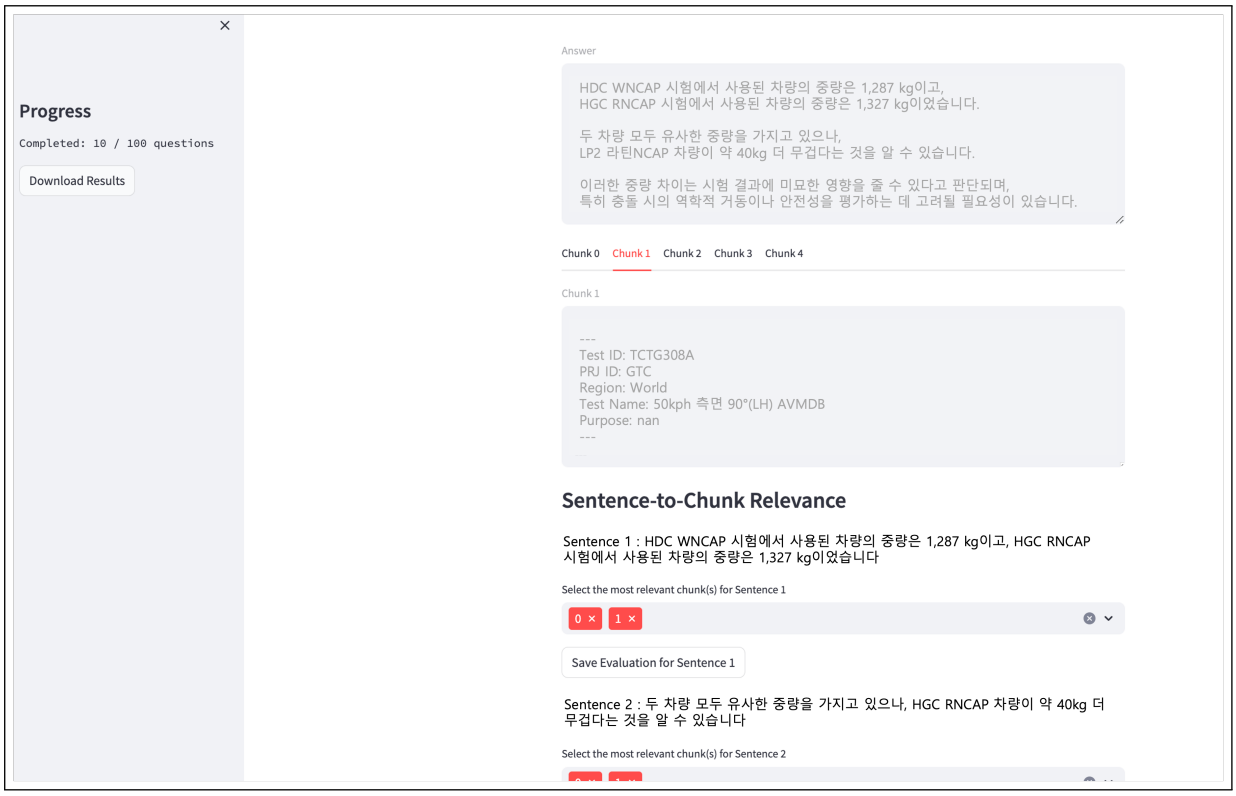

Figure 23: A screenshot of the user interface used for annotating the data to evaluate the reference matching task.