RMIT–ADM+S at the MMU-RAG NeurIPS 2025 Competition

Kun Ran
RMIT University
Melbourne, Australia

Marwah Alaofi
RMIT University
Melbourne, Australia

Danula Hettiachchi
RMIT University
Melbourne, Australia

Chenglong Ma
RMIT University
Melbourne, Australia

Khoi Nguyen Dinh Anh
RMIT University
Melbourne, Australia

Khoi Vo Nguyen
RMIT University
Melbourne, Australia

Sachin Pathiyan Cherumanal
RMIT University
Melbourne, Australia

Lida Rashidi
RMIT University
Melbourne, Australia

Falk Scholer
RMIT University
Melbourne, Australia

Damiano Spina
RMIT University
Melbourne, Australia

Shuoqi Sun
RMIT University
Melbourne, Australia

Oleg Zendel
RMIT University
Melbourne, Australia

Abstract

This paper presents the award-winning RMIT-ADM+S system for the Text-to-Text track of the NeurIPS 2025 MMU-RAG Competition. We introduce Routing-to-RAG (R2RAG), a research-focused retrieval-augmented generation (RAG) architecture composed of lightweight components that dynamically adapt the retrieval strategy based on inferred query complexity and evidence sufficiency. The system uses smaller LLMs, enabling operation on a single consumer-grade GPU while supporting complex research tasks. It builds on the G-RAG system, winner of the ACM SIGIR 2025 LiveRAG Challenge, and extends it with modules informed by qualitative review of outputs. R2RAG won the Best Dynamic Evaluation award in the Open Source category, demonstrating high effectiveness with careful design and efficient use of resources.

1 Introduction

Retrieval-Augmented Generation (RAG) [7] has become a standard approach for improving the grounding and reliability of Large Language Models (LLMs). Evaluation campaigns such as LiveRAG [2] provide standardized settings for deploying and assessing RAG systems. MMU-RAG, a NeurIPS 2025 competition with a focus on user-centric evaluation, complements these efforts. Final rankings were determined across four categories: *Static* and *Dynamic* evaluation, and *Open Source* and *Closed Source* system types. The evaluation framework combined a robustness-aware aggregation of normalized automatic metrics and human judgments based on ordinal Likert-scale ratings, spanning both static and real-time dynamic (RAG-Arena) evaluation modes.[1] Our system, R2RAG, was a top-performing submission in the Text-to-Text track, excelling in the open-source dynamic evaluation mode.

[1] See MMU-RAG: `https://agi-lti.github.io/MMU-RAGent/`

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: MMU-RAG NeurIPS 2025 Competition.

Recent research on RAG has increasingly emphasized the importance of user-centric perspective by making RAG systems more adaptive, enabling them to plan searches, inspect retrieved evidence, and adjust their actions dynamically throughout a task. Motivated by the real-time evaluation setting of MMU-RAG, our system is designed to support dynamic decision-making through specialized agents that manage planning, search, and reasoning. The goal is to construct a clear and efficient pipeline that can run on low-cost GPUs and consumer-grade hardware while addressing complex information needs. To facilitate reproducibility, we release our code repository.[2]

This report presents our system for the MMU-RAG Text-to-Text track. We describe the architecture, justify the design choices behind each component, and explain how insights from a qualitative evaluation study informed system refinement.

2 Qualitative Evaluation

To study the strengths and limitations of different retrieval and generation strategies, we conducted a qualitative focus-group evaluation. The session examined differences in retrieval quality, synthesis coherence, response relevance, verbosity, and overall perceived usefulness in (semi-)realistic deep-research scenarios. Consistent with Section 1, the goal was to assess user-perceived trade-offs that may not appear in aggregate metrics.

The initial system was adapted from our champion submission to the SIGIR 2025 LiveRAG [2] challenge, which relied on synthetic LLM-generated queries and LLM-as-a-Judge evaluation [13]. For MMU-RAG, we focused on performance under the RAG Arena framework. While synthetic evaluation enables rapid iteration, prior work highlights limitations, including distributional mismatch and evaluation biases [1, 4, 11, 17]. To capture realistic user interactions and failure modes, we emphasized qualitative assessment.

We first involved the co-authors as representative users and later extended the session to a broader group from RMIT University's Centre for Human-AI Information Environments (CHAI).[3] The study included ($N = 20$) participants across academic levels, from master's students to professors, with expertise in information retrieval, machine learning, human-computer interaction, natural language processing, and data science.[4]

Participants interacted with the system by submitting queries and providing binary feedback (👍/👎), open-ended comments, and verbal reflections. The two-hour session allowed free exploration, during which participants submitted an average of 17 queries.

We analyzed feedback for each query using two complementary approaches: (i) Preference Ratio, computed as the ratio of 👍 to 👎 responses, and (ii) manual inspection of open-ended comments to identify qualitative themes and contextual nuances. The diversity of feedback proved informative in two respects. First, it enabled us to identify concrete system limitations, including failure cases in retrieval grounding and synthesis. Second, it surfaced dimensions that mattered differently to participants, such as perceived system biases, preferences for response structure and level of detail, and sensitivity to tone and framing. These aspects were not directly captured by quantitative metrics but influenced overall user judgments. Based on these findings, we refined several prompts and tuned selected system components to address recurring issues and improve alignment with user expectations.

Although the Preference Ratio suggested similar overall performance between the two variants, qualitative analysis revealed complementary strengths: Vanilla Agent handled complex queries more effectively but was verbose for simple ones, whereas Vanilla RAG was concise for simple queries but weaker on multi-faceted tasks.

These observations align with the MMU-RAG organizers' key insight:

> "**Live evaluation matters:** Arena preferences revealed qualitative distinctions not captured by static metrics."

[2] See `https://github.com/rmit-ir/NeurIPS-MMU-RAG`

[3] See `https://www.rmit.edu.au/chai`

[4] No personal or identifiable data were collected or stored; feedback was used solely to improve the system and was not shared externally.

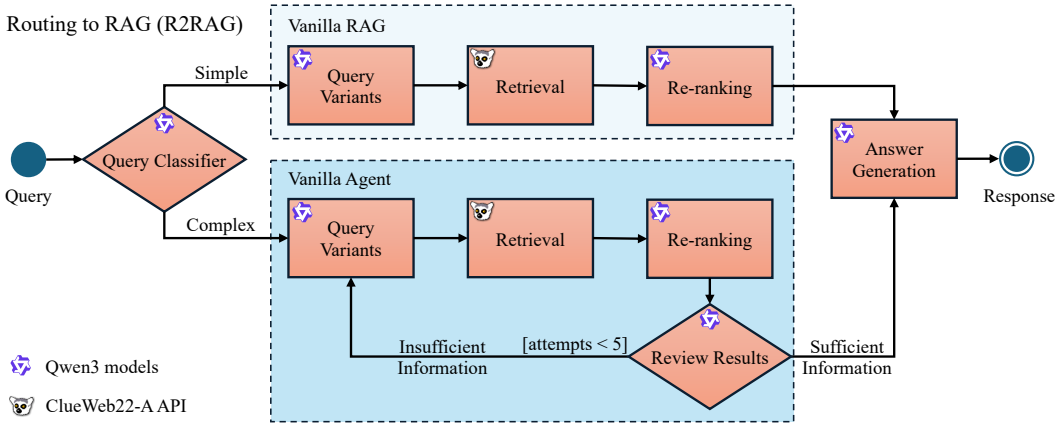

Figure 1: Overview of the system. The workflow includes a query classifier that routes each query to either a Vanilla RAG path for simple queries or a Vanilla Agent for complex queries that iteratively issues revised search query variants until sufficient evidence is gathered, followed by an answer generation module.

Our findings likewise indicate that preference-based and qualitative feedback can reveal system trade-offs that are not apparent from aggregate metrics alone.

3 System Architecture

A central design principle of R2RAG is the explicit distinction between simple and complex queries, as illustrated in Figure 1. The system first applies a query classifier that assigns each incoming query to one of these two categories. Based on this decision, the query is routed to either a *Vanilla RAG* pipeline for simple queries or a *Vanilla Agent* pipeline for complex queries. Simple queries are processed using a single retrieval step followed by answer generation. In contrast, complex queries are handled through an iterative process in which the agent reformulates search queries and performs multiple retrieval rounds to gather complementary and sufficient evidence. After the relevant documents are collected, an LLM-based answer generator produces the final response.

Our system relies on LLM-based inference for query classification, query reformulation, document assessment, and answer generation. Model selection was guided by model size, available GPU memory, context window capacity, and reasoning performance under resource constraints.

We used the `Qwen3-4B` [16] LLM for most components and `Qwen3-Reranker-0.6B` [18] for document reranking. The 4B model was used without quantization, as prior work shows that quantization can degrade long-context reasoning performance in LLMs [8, 9]. In preliminary experiments, 4-bit quantized larger models (e.g., `Qwen3-8B`) yielded weaker performance than the unquantized 4B variant. The compact size of `Qwen3-4B` allows it to fit within the available GPU memory alongside the reranker model, while maintaining strong performance across classification and generation tasks.

Both models were served using vLLM [6] to support efficient inference. To balance efficiency and performance, `Qwen3-4B` was configured with a reduced context window of 25,000 tokens (from the native 32,768 tokens) and run with the developer-recommended decoding parameters, *temperature* = 0.6 and *top_p* = 0.95.

3.1 Query Classifier

Queries vary in complexity and therefore require different retrieval strategies. We classify queries as *simple* or *complex*, where complex queries are defined as non-factoid questions that require long-form reasoning or multi-step synthesis [14]. We explored two classifier approaches:

Logistic Regression (LR) Classifier. We trained and evaluated the LR classifier on a dataset of 175,850 queries combining TREC Deep Learning [3], Deep-Research Questions [14], TREC RAG 2025 [10, 15], and Natural Questions [5]. Queries requiring decomposition were labeled `True`

(complex), and others `False` (simple). Each query was represented by a 147-dimensional feature vector, consisting of a 128-dimensional semantic embedding and 19 linguistic features,[5] including part-of-speech and punctuation counts.[6]

LLM-based Classifier. We used a structured prompt (Appendix A.1) to instruct the LLM to classify queries as complex (requiring more than a single Google search) or simple. This operational definition differs from standard Information Retrieval (IR) definitions but is appropriate for routing queries to the corresponding retrieval path.

The LLM-based classifier offers several advantages over LR. It can reason about query semantics, better handle out-of-distribution queries, and allows flexible adjustment of classification criteria through prompt modifications without retraining. The main trade-off is higher computational cost, as LLM inference requires more time and GPU resources. Given the 10-minute per-query limit, we prioritized these advantages over the efficiency of the LR model and adopted the LLM-based classifier, as the additional overhead remains acceptable within this constraint.

3.2 Vanilla RAG

For queries classified as simple, we use a RAG pipeline called *Vanilla RAG*, which processes the query in a single pass without iterative refinement. As shown in the top path of Figure 1, the pipeline includes four stages:

Generating Query Variants. To increase retrieval coverage, we generate three query variants using Qwen3-4B with thinking mode enabled. Following Ran et al. [12], these variants rephrase the original query with different keywords and formulations to capture a broader range of relevant documents.

Retrieval. Each variant is used to search the ClueWeb22-A index in parallel, returning up to ten documents per query.[7] After deduplication, this yields fewer than 30 documents for the next stage.

Reranking. The aggregated search results are reranked using the Qwen3-reranker-0.6b model with customized instructions (instruction provided in Appendix A.3).[8] The model serves as a Pointwise reranker [19], predicting "yes" or "no" for each query-document pair and using the probability of "yes" as the relevance score. We chose this model because its size fits the hardware budget while allowing the rest of the system to run efficiently. After reranking, we select the top-ranked documents and truncate them to a 5k-word limit to balance coverage and computational cost.

Answer Generation. The reranked documents and the original query are passed to Qwen3-4B for answer generation. The model is instructed to synthesize information from the retrieved documents, produce a coherent response suited to the query, and include inline citations in the format [ID] to reference specific source documents. For controversial topics, the model is encouraged to provide balanced perspectives. The full prompt is provided in Appendix A.5. Note that retrieval and reasoning are performed over entire documents rather than chunks.

3.3 Vanilla Agent

For complex queries, we developed *Vanilla Agent*, an iterative RAG system that performs multiple rounds of retrieval to gather complementary information. As shown in the lower path of Figure 1, it shares the Query Variants, Retrieval, and Reranking components with Vanilla RAG but operates within an iterative search loop with enhanced parameters.

Iterative Search Loop. Each iteration runs the Query Variants-Retrieval-Reranking pipeline with more aggressive settings, generating up to five query variants (instead of three) and allowing a larger document limit of up to 25K tokens (compared to 5K words).[9] The system maintains state across iterations by accumulating: (1) previously used queries to avoid repetition, and (2) summaries of useful documents to guide subsequent searches.

[5] Using the `google/bert_uncased_L-2_H-128_A-2` model from Hugging Face.

[6] Using the English language model `en_core_web_sm` from spaCy.

[7] Search was performed through the API provided by the organizers: `https://www.deepresearchgym.ai/`

[8] Model card: `https://huggingface.co/Qwen/Qwen3-Reranker-0.6B`

[9] Tokens are used here to ensure the model stays within its context limits, while Vanilla RAG uses words for faster estimation.

The stopping condition S depends on the accumulated token count (T), estimated information coverage (Cov), and a fixed iteration cap (i):

$$\text{S} \iff (T > 20{,}000) \ \vee \ (Cov = 1) \ \vee \ (i \geq 5).$$

The loop terminates when any of these conditions is met. The token threshold protects against exceeding the model context window during final answer generation while indicating sufficient information density. The coverage signal is determined by the LLM based on the accumulated agent state. The iteration cap prevents unbounded search loops.

Document Review and Information Coverage. The Cov factor is updated after each reranking step by the document review component, using `Qwen3-4B` (prompt in Appendix A.4). The model evaluates: (1) whether the query requires coverage of multiple perspectives, (2) whether all sub-parts of the query are addressed, (3) balance for contested topics, (4) identification of newly useful documents, and (5) remaining knowledge gaps. If the accumulated information is insufficient, the model generates a new query targeting the identified gaps while avoiding previous unsuccessful queries. If no documents are retrieved in an iteration, the system reformulates the query and proceeds to the next iteration.

3.4 Deployment and Evaluation Infrastructure

During development, we implemented our different RAG system variants as services that can be deployed through multiple API servers, including the MMU-RAG API server required by the competition guidelines and an OpenAI-compatible API server that exposes multiple RAG variants on a single GPU server for our evaluation infrastructure. Each system configuration is assigned a unique model identifier in the `model` field. This design eliminates the need to deploy separate servers for each variant, thereby simplifying infrastructure management. For internal qualitative testing, we integrated the API with an in-house search engine that allows users to submit queries to different RAG systems and provide feedback for the responses.

4 Conclusions

This paper presented R2RAG, our dynamic RAG system for the Text-to-Text track of the NeurIPS 2025 MMU-RAG Competition. The system adapts its retrieval strategy based on query complexity, routing queries to either a single-pass Vanilla RAG pipeline or an iterative Vanilla Agent that accumulates evidence across multiple retrieval rounds. For complex queries, an LLM-based review module estimates information coverage and guides query reformulation, with termination governed by coverage signals, token budget, and iteration limits.

Our results show that small reasoning-oriented models can effectively support dynamic RAG pipelines under realistic resource constraints. Retrieval quality and controlled evidence accumulation were central to answer quality, while structured LLM-based decision components improved robustness. The entire system operates on a single consumer-grade GPU through compact model selection and careful resource tuning.

Beyond system design, our experience highlights the importance of well-aligned, multi-level evaluation. Static automatic metrics provided useful baselines but did not fully capture qualitative differences in reasoning depth, presentation, and user preference. Qualitative and dynamic evaluation settings were essential for identifying failure modes, refining prompts, and validating system behavior under realistic conditions. Strong performance in the competition was therefore not only a result of architectural choices, but also of rigorous, experimentally grounded evaluation that aligned testing conditions with real user needs.

Acknowledgments and Disclosure of Funding

We thank the MMU-RAG organizers for organizing the competition and the associated awards. This work was supported by the ARC Centre of Excellence for Automated Decision-Making and Society (ADM+S, CE200100005) and was undertaken with the assistance of computing resources from RACE (RMIT AWS Cloud Supercomputing). This system was designed and developed on the unceded lands of the Woi wurrung and Boon wurrung language groups of the eastern Kulin Nation. We pay our respects to Elders past and present and extend that respect to all Aboriginal and Torres Strait Islander peoples and their ongoing connections to land, sea, sky, and community.

References

[1] Krisztian Balog, Don Metzler, and Zhen Qin. Rankers, Judges, and Assistants: Towards Understanding the Interplay of LLMs in Information Retrieval Evaluation. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '25, pages 3865–3875, 2025. doi: 10.1145/3726302.3730348.

[2] David Carmel, Simone Filice, Guy Horowitz, Yoelle Maarek, Oren Somekh, Ran Tavory, Mehdi Ghissassi, Edo Liberty, and Roy Miara. SIGIR 2025 – LiveRAG Challenge Report, 2025. URL `https://arxiv.org/abs/2507.04942`.

[3] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, Ellen M. Voorhees, and Ian Soboroff. TREC Deep Learning Track: Reusable Test Collections in the Large Data Regime. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, pages 2369–2375, 2021. doi: 10.1145/3404835.3463249.

[4] Laura Dietz, Oleg Zendel, Peter Bailey, Charles L. A. Clarke, Ellese Cotterill, Jeff Dalton, Faegheh Hasibi, Mark Sanderson, and Nick Craswell. Principles and Guidelines for the Use of LLM Judges. In *Proceedings of the 2025 International ACM SIGIR Conference on Innovative Concepts and Theories in Information Retrieval (ICTIR)*, ICTIR '25, pages 218–229, 2025. doi: 10.1145/3731120.3744588.

[5] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural Questions: A Benchmark for Question Answering Research. *Transactions of the Association for Computational Linguistics*, 7, 2019-11. ISSN 2307-387X.

[6] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023. doi: 10.1145/3600006.3613165.

[7] 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. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, 2020.

[8] Peiyu Liu, Zikang Liu, Ze-Feng Gao, Dawei Gao, Wayne Xin Zhao, Yaliang Li, Bolin Ding, and Ji-Rong Wen. Do Emergent Abilities Exist in Quantized Large Language Models: An Empirical Study. In Nicoletta Calzolari, Min-Yen Kan, Veronique Hoste, Alessandro Lenci, Sakriani Sakti, and Nianwen Xue, editors, *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 5174–5190, 2024.

[9] Anmol Mekala, Anirudh Atmakuru, Yixiao Song, Marzena Karpinska, and Mohit Iyyer. Does Quantization Affect Models' Performance on Long-context Tasks? In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng, editors, *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 9433–9481, 2025. doi: 10.18653/v1/2025.emnlp-main.479.

[10] Ronak Pradeep, Nandan Thakur, Sahel Sharifymoghaddam, Eric Zhang, Ryan Nguyen, Daniel Campos, Nick Craswell, and Jimmy Lin. Ragnarök: A Reusable RAG Framework and Baselines for TREC 2024 Retrieval-Augmented Generation Track. In *Advances in Information Retrieval: 47th European Conference on Information Retrieval, ECIR 2025, Lucca, Italy, April 6–10, 2025, Proceedings, Part I*, pages 132–148, 2025. doi: 10.1007/978-3-031-88708-6_9.

[11] Hossein A. Rahmani, Varsha Ramineni, Emine Yilmaz, Nick Craswell, and Bhaskar Mitra. Towards Understanding Bias in Synthetic Data for Evaluation. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, CIKM '25, pages 5166–5170, 2025. doi: 10.1145/3746252.3760908.

[12] Kun Ran, Marwah Alaofi, Mark Sanderson, and Damiano Spina. Two Heads Are Better Than One: Improving Search Effectiveness Through LLM-Generated Query Variants. In *Proceedings of the 2025 ACM SIGIR Conference on Human Information Interaction and Retrieval*, CHIIR '25, pages 333–341, 2025. doi: 10.1145/3698204.3716468.

[13] Kun Ran, Shuoqi Sun, Khoi Nguyen Dinh Anh, Damiano Spina, and Oleg Zendel. RMIT-ADM+S at the SIGIR 2025 LiveRAG Challenge. *arXiv preprint arXiv:2506.14516*, 2025. doi: 10.48550/arXiv.2506.14516.

[14] Corbin Rosset, Ho-Lam Chung, Guanghui Qin, Ethan Chau, Zhuo Feng, Ahmed Awadallah, Jennifer Neville, and Nikhil Rao. Researchy Questions: A Dataset of Multi-Perspective, Decompositional Questions for Deep Research. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '25, pages 3712–3722, 2025. doi: 10.1145/3726302.3730275.

[15] Nandan Thakur, Ronak Pradeep, Shivani Upadhyay, Daniel Campos, Nick Craswell, Ian Soboroff, Hoa Trang Dang, and Jimmy Lin. Assessing Support for the TREC 2024 RAG Track: A Large-Scale Comparative Study of LLM and Human Evaluations. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '25, pages 2759–2763, 2025. doi: 10.1145/3726302.3730165.

[16] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025.

[17] Oleg Zendel, Sara Fahad Dawood Al Lawati, Lida Rashidi, Falk Scholer, and Mark Sanderson. A Comparative Analysis of Linguistic and Retrieval Diversity in LLM-Generated Search Queries. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, CIKM '25, pages 4014–4023, 2025. doi: 10.1145/3746252.3761382.

[18] 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. Qwen3 Embedding: Advancing Text Embedding and Reranking Through Foundation Models. *arXiv preprint arXiv:2506.05176*, 2025.

[19] Honglei Zhuang, Zhen Qin, Kai Hui, Junru Wu, Le Yan, Xuanhui Wang, and Michael Bendersky. Beyond Yes and No: Improving Zero-Shot Pointwise LLM Rankers via Scoring Fine-Grained Relevance Labels. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2024.

A Prompts

A.1 Query Classifier

System Prompt:

```
Judge if the user question is a complex question.  Note that
the answer can only be "yes" or "no".
Given the question below, if you are doing the research, do
you think the question is very easy and you can find the
answer easily with a single search on Google?
If so, it's not a complex question, respond with "no",
otherwise, it's a complex question, respond with "yes".
Generally, for straightforward, single question, it's a
simple question.  If the question is ambiguous, multifaceted,
contains multiple parts, has 2 or more sub-questions or
requires multiple steps to answer, it's a complex question.
Give the final answer based on your last reasoning, yes
indicates it's a complex question or no indicating it's not
a complex question.
```

User Prompt:

```
{question}
```

A.2 Query Variants Generator

System Prompt:

```
You will receive a question from a user and you need interpret
what the question is actually asking about and come up with 2
to 5 Google search queries to answer that question.
Try express the same question in different ways, use different
techniques, query expansion, query relaxation, query
segmentation, use different synonyms, use reasonable guess
and different keywords to reach different aspects.
Try to provide a balanced view for controversial topics.
To comply with the format, put your query variants enclosed in
queries xml markup:
<queries>
query variant 1
query variant 2
...
</queries>
Put each query in a line, do not add any prefix on each query,
only provide the query themselves.
```

User Prompt:

```
User question:  {query}
```

A.3 Re-ranker

Prompt:

```
<|im_start|>system
Judge whether the Document meets the requirements based on the
Query and the Instruct provided.  Note that the answer can
only be "yes" or "no".<|im_end|>
<|im_start|>user
<Instruct>:  Given the web search query, is the retrieved
document
(1) from a high quality and relevant website based on the URL,
(2) published recently, and
(3) contains key information that helps answering the query?
<Query>:  {query}
<Document>:  Web URL: {url}
Content:  {document_text}
<|im_end|>
<|im_start|>assistant
<think>
</think>
```

A.4 Review Results

System Prompt:

You are an expert in answering user question "{question}". We are doing research on user's question and currently working on aspect "{next_query}"

Go through each document in the search results, judge if they are sufficient for answering the user question. Please consider:

1. Does the user want a simple answer or a comprehensive explanation? For comprehensive explanation, we may need searching with different aspects to cover a wider range of perspectives.
2. Does the search results fully addresses the user's query and any sub-components?
3. For controversial, convergent, divergent, evaluative, complex systemic, ethical-moral, problem-solving, recommendation questions that will benefit from multiple aspects, try to search from different aspects to form a balanced, comprehensive view.
4. Don't mention irrelevant, off-topic documents.
5. When you answer 'yes' in <is-sufficient>, we will proceed to generate the final answer based on these results. If you answer 'no', we will continue the next turn of using your new query to search, and let you review again.
6. If information is missing or uncertain, always return 'no' in <is-sufficient> xml tags for clarification, and generate a new query enclosed in xml markup <new-query>your query</new-query> indicating the clarification needed. If the search results are too off, try clarifying sub-components of the question first, or make reasonable guess. If you think the search results are sufficient, return 'yes' in <is-sufficient> xml tags.
7. If current search results have very limited information, try use different techniques, query expansion, query relaxation, query segmentation, use different synonyms, use reasonable guess and different keywords to get relevant results, and put the new query in <new-query>your query</new-query> xml tags. If there are previous search queries, do not repeat them in the new query, we know they don't work.
8. Identify unique, new documents that are important for answering the question but not included in previous documents, and list their IDs (# in ID=[#]) in a comma-separated format within <useful-docs> xml tags. If multiple documents are similar, choose the one with better quality. Do not provide duplicated documents that have been included in previous turns. If no new documents are useful, leave <useful-docs></useful-docs> empty.
9. If useful-docs is not empty, provide a brief summary of what these documents discuss within <useful-docs-summary> xml tags, in 1-2 sentences. Start your summary with "These documents discuss...". Do not mention specific document IDs in the summary.
10. Your purpose is to judge documents relevance against the question, not to provide the final answer yet, do not answer the question.

Response format:

- <is-sufficient>yes or no</is-sufficient> (For all of the documents we have collected, including previous documents, do we have enough information to answer the user question?

```
If yes, then provide <useful-docs> and <useful-docs-summary>
tags; if 'no', then provide <new-query> tag)
- <new-query>your new query</new-query> (only if is-sufficient
is 'no')
- <useful-docs>1,2,3</useful-docs> (list of document IDs that
are useful for answering the question, separated by commas)
- <useful-docs-summary></useful-docs-summary> (short summary
of what these useful documents are talking about, just the
summary, only if useful-docs is not empty)

{prev_questions}
{prev_docs_summaries}
Here is the current question:  "{query}"
Here is the search results for current question:
<search-results>
Webpage ID=[1] URL=[{url1}] Date=[{date1}]:
{document_text}
Webpage ID=[2] URL=[{url2}] Date=[{date2}]:
{document_text}
</search-results>
```

A.5 Answer Generator

System Prompt:

```
You are a knowledgeable AI search assistant built by the RMIT
IR team.
Your search engine has returned a list of relevant webpages
based on the user's question, listed below in <search-results>
tags.  These webpages are your knowledge.
The next user message is the full user question, and you
need to explain and answer the question based on the search
results.  Do not make up answers that are not supported by
the search results.  If the search results do not have the
necessary information for you to answer the search question,
say you don't have enough information for the question.
Try to provide a balanced view for controversial topics.
Tailor the complexity of your response to the user question,
use simpler bullet points for simple questions, and sections
for more detailed explanations for complex topics or rich
content.
Do not answer to greetings or chat with the user, always reply
in English.
You should refer to the search results in your final response
as much as possible, append [ID] after each sentence to
point to the specific search result.  e.g., "This sentence
is referring to information in search result 1 [1].".
Current time at UTC+00:00 timezone:  {datetime.now(timezone.utc)}
Search results knowledge cutoff:  December 2024
<search-results>
Webpage ID=[1] URL=[{url1}] Date=[{date1}]:
{document_text}
Webpage ID=[2] URL=[{url2}] Date=[{date2}]:
{document_text}
</search-results>
```

User Prompt:

```
{question}
```