RMIT–ADM+S at the SIGIR 2025 LiveRAG Challenge

GRAG: Generation-Retrieval-Augmented Generation

Kun Ran
RMIT University
Melbourne, Australia
kun.ran@student.rmit.edu.au

Shuoqi Sun
RMIT University
Melbourne, Australia
shuoqi.sun@student.rmit.edu.au

Khoi Nguyen Dinh Anh
RMIT University
Melbourne, Australia
s3695517@rmit.edu.vn

Damiano Spina
RMIT University
Melbourne, Australia
damiano.spina@rmit.edu.au

Oleg Zendel
RMIT University
Melbourne, Australia
oleg.zendel@rmit.edu.au

Abstract

This paper presents the RMIT–ADM+S participation in the SIGIR 2025 LiveRAG Challenge. Our Generation-Retrieval-Augmented Generation (GRAG) approach relies on generating a hypothetical answer that is used in the retrieval phase, alongside the original question. GRAG also incorporates a pointwise large language model (LLM)-based re-ranking step prior to final answer generation. We describe the system architecture and the rationale behind our design choices. In particular, a systematic evaluation using the Grid of Points (GoP) framework and *N*-way ANOVA enabled comparison across multiple configurations, including query variant generation, question decomposition, rank fusion strategies, and prompting techniques for answer generation. Our system achieved a Relevance score of 1.199 and a Faithfulness score of 0.477 on the private leaderboard, placing among the top four finalists in the LiveRAG 2025 Challenge.

Keywords

Retrieval-Augmented Generation, RAG, n-way ANOVA, LLM evaluation

1 Introduction

Evaluation campaigns such as the SIGIR 2025 LiveRAG Challenge provide a structured and standardized setting for researchers and practitioners to develop and evaluate different Retrieval-Augmented Generation (RAG) approaches on a common dataset using shared metrics. These campaigns enable fair comparisons between systems in a controlled environment. LiveRAG 2025, the first edition of the challenge, required participants to develop RAG systems using the `Falcon3-10B-Instruct` model [19] for final answer generation, thereby standardizing the generation component across all submissions. This constraint allows for a more focused evaluation of the retrieval components and prompts, as the generation model is fixed and does not introduce variability in the results.[1] In this context, the RMIT–ADM+S team submitted *GRAG* (Generation-Retrieval-Augmented Generation), a system selected through an internal evaluation process using a Grid of Points (GoP) approach and *N*-way ANOVA.[2]

Our submission builds on previous work and integrates several components: (1) hypothetical answer generation prior to retrieval; (2) large language model (LLM)-based query variant generation [2, 14]; (3) LLM-based re-ranking [17]; and (4) answerability estimation in RAG systems [13]. We also developed supporting infrastructure for dynamic cloud-based resource allocation and LLM deployment in the AWS environment, enabling more efficient resource usage.

The remainder of this paper describes the system architecture and the design choices made to optimize our RAG system for the LiveRAG Challenge: Section 2 presents the system architecture; Section 3 describes our GoP-ANOVA-based run selection method; and Section 4 concludes the work.

The source code for both the system and the evaluation framework are publicly available at https://github.com/rmit-ir/GRAG-LiveRAG.

2 GRAG System Architecture

Figure 1 illustrates the pipeline of our proposed approach and the system used in our submitted run. In addition to the retrieval and answer generation stages of a RAG system, GRAG includes two additional stages: question augmentation and re-ranking. For all components involving LLMs, we used the same fixed open-weight model: `Falcon3-10B-Instruct`. The prompts used throughout the system are provided in Appendix A.

2.1 Question Augmentation

Intuitively, the retrieval phase may benefit from generating queries that offer complementary ways of retrieving relevant information from the corpus. We experimented with three query augmentation approaches: query variants, question decomposition, and hypothetical answer generation.

Query Variants. This approach generates multiple query variations from the original question to be used as search queries. We adapted prompts from prior work [2, 14] to better suit the `Falcon3-10B-Instruct` model.

Question Decomposition. During manual examination and evaluation of the dataset – particularly when identifying *tricky questions*

[1]Other open-weight LLMs, such as the Llama models, were allowed in other components of the system, but only up to their 10B versions.

[2]The team includes members from the RMIT Centre for Human-AI Interaction (CHAI) and the ARC Centre of Excellence for Automated Decision-Making and Society (ADM+S).

This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.

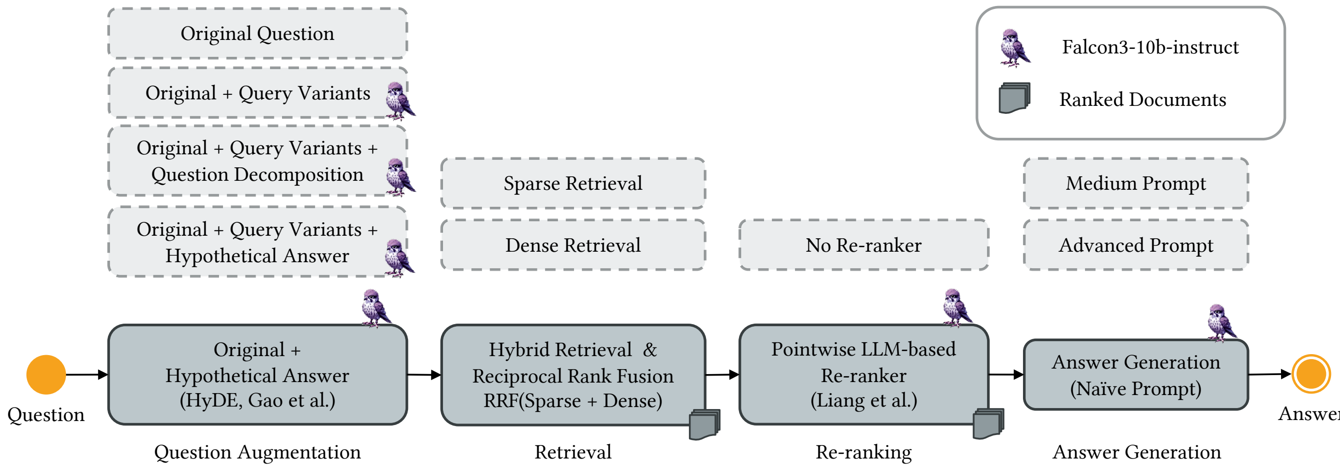


Figure 1: Pipeline of our GRAG approach. Components used in the final selected run are shown with solid borders; components analyzed during in-house evaluation are shown with dashed borders.

(see Section 3.1.1) – we observed cases where query variants were insufficient for handling complex questions that involve comparisons or require multiple aspects. To address this, we designed a process involving multiple calls to the LLM to decompose such questions into sub-queries. We begin by identifying the essential components of the question using shorthand entity annotation. These components are then used to rephrase the original question into a more detailed and human-readable form. Finally, we developed a lightweight classifier based on the rephrased question to decide whether to apply query variants or to reformulate sub-questions as independent queries. Since the model is relatively small, each step is efficient; the classifier runs in approximately 100ms on an NVIDIA L40S GPU.

Hypothetical Answer Generation. Inspired by the Hypothetical Document Embeddings (HyDE) approach proposed by Gao et al. [9], we use the Falcon LLM to generate a "hypothetical answer." Since the initial prompt did not yield an answer for certain questions, it was modified to encourage the model to be more flexible and produce a response that "could be true." The hypothetical answer is then treated as a search query: it is added to the list of queries and used in the subsequent retrieval stage.

2.2 Retrieval

We used the retrieval services provided by the organizers. Documents were split into sentence-based chunks with a maximum length of 512 tokens using the LlamaIndex sentence splitter. For each query, we applied both sparse and dense retrieval methods. The results were then merged into a single ranked list using Reciprocal Rank Fusion (RRF) [3] with $k = 60$ (hereafter referred to as hybrid retrieval). Documents retrieved across all queries were combined into a unified list for the subsequent re-ranking stage. For sparse retrieval, we sent each search query directly to the provided OpenSearch service, which applies Okapi BM25 [15] over a prebuilt inverted index. For dense retrieval, we used the provided Pinecone service to search over sentence chunks embedded as 512-dimensional vectors using E5-base embeddings [20]. The search query was encoded using the same embedding model (`intfloat/e5-base-v2`) and sent to the Pinecone service to retrieve documents ranked by cosine similarity.

2.3 Re-ranking

We employ a Pointwise LLM-based re-ranker, leveraging the *likelihood* of the model to generate "Yes" – extracted from the associated token logits – as an indicator of document relevance to the query, following the method proposed by Liang et al. [10, p. 21]. For each document, we prompt the LLM to determine whether it contains the necessary information to answer the question. Documents are then ranked by the resulting *likelihood* scores, and those scoring below a threshold of 0.5 are discarded.

2.4 Answer Generation

To generate the final answer, we describe the task in system prompt and include the retrieved documents along with the original question in the user prompt. We created three sets of system and user prompts and selected the best-performing one based on the *champion* configuration (see Section 3.2).

Naïve Prompt. We used the similar system prompt used by ChatGPT "`You are a helpful assistant`," combined with a simple task instruction "`Answer the question based on the provided documents`."

Medium Prompt. In addition to minor refinements, this version explicitly instructed the model to respond with "I don't know" when the answer was not present in the context, following suggestions from prior work [8, 13]. This instruction aligns with the LiveRAG Relevance evaluation metric, where a response of "I don't know" receives a score of 0, while incorrect answers receive a score of -1. It may also contribute to improved faithfulness by discouraging unsupported generations.[3]

Advanced Prompt. This version incorporated prompting techniques from Tamber et al. [18] to further discourage hallucinated

[3] Evaluation details are available at https://liverag.tii.ae/challenge-details.php.

Table 1: Configuration used to create the test collection using `DataMorgana`.

	Category Name	Value	Probability
User Categories	Expertise	Expert	0.5
		Novice	0.5
Question Categories	Factuality	Factoid	0.5
		Open-ended	0.5
	Premise	Direct	0.5
		With-premise	0.5
	Phrasing	concise-and-natural	0.25
		verbose-and-natural	0.25
		short-search-query	0.25
		long-search-query	0.25
	Linguistic variation	similar-to-document	0.5
		distant-from-document	0.5
	multi-doc	single-doc	0.4
		comparison	0.2
		multi-aspect	0.3
		three-doc	0.1

content and emphasize faithful generation.

2.5 Submitted Run Configuration

The submitted run used hypothetical answer generation for query augmentation (G), hybrid retrieval, LLM-based re-ranking (R), and the naïve prompt for answer generation (AG). This configuration corresponds to the main horizontal pipeline shown in Figure 1.

3 In-House Evaluation

This section presents our systematic in-house evaluation. We describe the setup, including the test dataset, the RAG system interface, and the LLM-based evaluation method, as well as the methodologies used for system optimization and live challenge submission (i.e., Grid of Points and ANOVA).

3.1 Experimental Setup

3.1.1 Dataset Preparation. We utilized `DataMorgana` to construct our dataset of question–answer pairs. Building on the setup presented in the original paper [8], we created a configuration file (Table 1) comprising one user category and five distinct question categories. Our configuration remains largely consistent with that detailed by Filice et al. [8], with a minor adjustment to the *multi-doc* field. This field is intended to regulate the number of documents used for question generation, particularly relevant when exploring whether questions can be formulated using information drawn from multiple documents.

Multiple datasets were generated at varying sizes: 2, 5, 50, 100, 200, 500, and 1,000 questions. The smaller datasets were used during the initial development and debugging phases of the RAG system, while the larger datasets supported a more extensive evaluation of the system and its individual components. The largest dataset (1,000 questions) was specifically used for stress testing and benchmarking runtime efficiency to ensure the system could operate within the time constraints of the live event.

To support the development and evaluation of our RAG system, we curated several smaller datasets with varying levels of complexity. We first constructed a baseline RAG system that used the original question as the query, combined with sparse retrieval and an initial answer generation prompt. This system was applied to a set of 500 questions to obtain initial evaluation results. We then analyzed the outputs to identify *tricky questions*—those that the baseline system answered incorrectly or incompletely. Questions receiving the lowest Relevance and Faithfulness scores were further marked as *challenging questions*. In total, we identified 179 *tricky questions* and 15 *challenging questions*. To support the GoP and ANOVA analyses described in Sections 3.2 and 3.3, we randomly selected 15 questions from the *tricky questions* and combined them with 85 questions drawn from a separate dataset to construct the main test set.

These datasets were employed iteratively throughout the development process to examine the behavior of the RAG pipeline and its individual components. After each evaluation cycle, we reviewed the results to detect recurring failure patterns and refine the system accordingly. For example, in a question "total funding amount digital health startups ryse," the term "ryse" refers to a startup. However, the query augmentation module frequently miscorrected it to "rise," misinterpreting it as a typographical error. This illustrates a negative optimization effect caused by query rewriting. Another example is the question "How does the artwork 'For Proctor Silex' create an interesting visual illusion for viewers as they approach it?" Here, the named entity "For Proctor Silex" is rare and difficult to retrieve from the corpus. As a result, the relevant document was ranked low, leading to an inaccurate answer. These cases highlight the need for retrieval components to return a sufficiently diverse set of documents, especially for queries containing rare or ambiguous named entities.

3.1.2 Combined RAG System. After testing individual components across multiple RAG system variants, we developed a modular, configurable RAG system that supports the specification of different components and settings for each stage of the pipeline. This combined system enables controlled experimentation by allowing individual components to be modified independently, while holding others constant. As such, it serves as a foundational tool for systematic evaluation and parameter exploration.

The configurable aspects of the combined RAG system include:

- Answer Generation Prompts: Introduced in Section 2.4.
- Question Augmentation: As detailed in Section 2.1.
- Query Variants Generation Prompts: Similar in structure to the answer generation prompts; these are used to generate query variants within the *Question Augmentation* module.
- Number of Query Variants: Controls how many variants are generated when employing query augmentation.
- Retrieval: As described in Section 2.2.
- Re-ranker: Optionally applies a pointwise LLM-based re-ranker. Pointwise re-ranking has shown competitive performance [17].
- Number of Documents Retrieved: Specifies how many documents are retrieved for each query prior to re-ranking.
- Context Words Length Limits: Retrieved documents are concatenated and truncated to a specified number of words before

being passed to the LLM for answer generation.

3.1.3 LLM-based Evaluation. This evaluation method follows the guidelines outlined on the LiveRAG Challenge Details website.[4] To approximate the official evaluation approach, we used Claude 3.5 Sonnet,[5] accessed via Amazon Bedrock,[6] to evaluate our system outputs. While the exact prompts and procedures used by the LiveRAG organizers are not publicly disclosed, we aimed to replicate the setup as closely as possible. Our evaluation prompt instructed the LLM to compare the generated answer against the reference answer provided by `DataMorgana`. To prevent interference between metrics, we used separate prompts and independent API calls to obtain relevance and faithfulness scores.

This evaluation procedure was designed to assess both overall performance (i.e., average scores across all questions) and per-question performance. These scores were further used to compare the relative effectiveness of different RAG system variants.

3.2 Grid of Points

As shown in Figure 1, our development process involved testing various parameter combinations, revealing a large configuration space for each RAG system component. Given the time constraints of the LiveRAG challenge, identifying an effective configuration was essential. To address this, we adopted the Grid of Points (GoP) [7] and Analysis of Variance (ANOVA) [11, 16]. This section describes how we used GoP to identify a *champion* configuration—defined as the best-performing parameter combination on our development set. The ANOVA analysis is detailed separately in Section 3.3.

GoP systematically explores all combinations of parameters within a predefined configuration space. This process was facilitated by our modular RAG system described in Section 3.1.2. We evaluated a total of 96 configurations,[7] limiting the *Number of Query Variants* to eight and consistently applying a Pointwise *Re-ranker* to manage computational constraints. Each configuration was evaluated using the LLM-based evaluation method, and we ranked the configurations by their average relevance score. The configuration with the highest average relevance score was selected as the *champion*. This configuration achieved an average relevance score of 1.75 and an average faithfulness score of 0.59, making it one of the top candidates for live submission. Details of the *champion* configuration are provided in Section 3.4.

3.3 *N*-Way ANOVA Analysis

While the *champion* configuration provided a reliable option for participating in the live event, our goal remained to further improve system performance. To this end, we conducted an *N*-way ANOVA analysis, which allowed us to systematically assess the relative impact of different components and their interactions. This method guided us in identifying which components have the most influence on system performance and should therefore be prioritized for further experimentation.

The ANOVA analysis was performed on the same 96 parameter configurations used in the GoP evaluation. Since the *Number of Query Variants* and *Re-ranker* settings were held constant (fixed at eight and Pointwise, respectively), they were excluded from the ANOVA model and are not shown in Table 2. The results presented in Table 2 indicate that *Question Augmentation* is the most influential individual component affecting system performance. Furthermore, we observed a statistically significant interaction effect between *Query Variants Generation Prompts* and *Retrieval*. While neither of these components had a significant effect on their own, their interaction suggests that specific combinations of prompts and retrieval methods can jointly influence performance.

These findings suggest that further improvements beyond the current *champion* configuration may be achieved by focusing on the most impactful components. Guided by this insight, we explored an alternative *Question Augmentation* technique and compared its performance to the original *champion* configuration.

3.4 Post-ANOVA Improvement: Hypothetical Answer Generation

Although *Question Augmentation* emerged as the most impactful component in the ANOVA analysis, the configuration that omitted augmentation altogether (i.e., the "None" parameter setting) yielded the highest Relevance scores.

To further explore improvements in this component, we adopted a hypothetical answer generation technique inspired by HyDE [9]. This method introduces *GRAG*, a variant of our RAG system that employs hypothetical answer generation as the *Question Augmentation* strategy, replacing the "None" setting used in the *champion* configuration. Due to time constraints, we did not re-run the full GoP or ANOVA tests for this single parameter change. Instead, we conducted a focused comparison between the original *champion* configuration and GRAG. This ad-hoc evaluation was performed using a sample of 100 questions and measured performance in terms of both Relevance and Faithfulness.

We first compared the two systems at the aggregate level using the average Relevance score. Both the original *champion* and the GRAG variant achieved the same average score of 1.75. To gain deeper insights, we conducted a single-question analysis. As shown in Table 3, the GRAG configuration outperformed the original on 8 questions and underperformed on 7 in terms of Relevance. For Faithfulness, it scored higher on 12 questions and lower on 14. Given the time constraints and the relatively balanced performance across metrics, we prioritized Relevance as the primary evaluation criterion. Based on this, we selected the GRAG variant as an improvement over the original *champion* system.

Although HyDE was originally introduced for dense retrieval [9], our preliminary experiments on a 50-question subset indicated that applying hypothetical answers to both sparse and dense retrieval produced better results than using them with dense retrieval alone. A more comprehensive investigation of HyDE's integration within RAG pipelines is left for future work.

In addition to performance considerations, GRAG also offered improved efficiency. The ANOVA results indicated that *Context Length Limitation* had no significant impact on evaluation metrics. However, longer contexts substantially increased LLM inference time. To reduce latency during the live event, we limited the input

[4] https://liverag.tii.ae/challenge-details.php
[5] https://www.anthropic.com/news/claude-3-5-sonnet
[6] https://aws.amazon.com/bedrock/
[7] The 96 configurations result from the Cartesian product of the options available for each component.

Table 2: *N*-way ANOVA results. This analysis utilized the LLM-based evaluation approach, specifically the average-level evaluation method described in Section 3.1.3. A factor is considered to have a statistically significant impact if its probability value, PR(>F), is less than 0.05. The factors are presented in descending order based on their corresponding F-statistic values. The header lines are as follows: Factor: The factor being analyzed, refer to Section 3.1.2. SM: Sum of Squares. DF: Degrees of Freedom. F: F-statistic. PR(>F): Probability value greater than F, i.e., p-value.

Factor	SM	DF	F	PR(>F)
Question Augmentation*	0.3193	2	125.6920	<0.0001
Query Variants Generation Prompts : Retrieval*	0.0062	1	4.9099	0.0297
Query Variants Generation Prompts	0.0030	1	2.3892	0.1263
Context Words Length Limits	0.0025	1	2.0052	0.1608
Retrieval	0.0024	1	1.8631	0.1762
Number of Documents Retrieved : Retrieval	0.0020	1	1.6100	0.2083
Answer Generation Prompts	0.0014	1	1.1061	0.2962
Context Words Length Limits : Question Augmentation	0.0008	2	0.3261	0.7227
Context Words Length Limits : Answer Generation Prompts : Question Augmentation	0.0004	2	0.1566	0.8553
Answer Generation Prompts : Question Augmentation	0.0004	2	0.1566	0.8554
Query Variants Generation Prompts : Number of Documents Retrieved : Retrieval	0.0001	1	0.1143	0.7363
Context Words Length Limits : Answer Generation Prompts	0.0001	1	0.0491	0.8253
Query Variants Generation : Number of Documents Retrieved	0.0000	1	0.0226	0.8808
Number of Documents Retrieved	0.0000	1	0.0118	0.9137
Residual	0.0978	77	–	–

Table 3: Comparison of the *champion* configuration and GRAG across 100 sampled questions. The table reports how often each system achieved higher or tied *Relevance* and *Faithfulness* scores on a per-question basis.

RAG System	Relevance Score	Faithfulness Score
GRAG (*champion* + HyDE)	**8**	12
Champion (GoP)	7	**14**
Ties (Equal Scores)	85	74

context length to 10k tokens, even though the original *champion* achieved its best performance with 15k tokens. This decision was further supported by the *runner-up* configuration, which demonstrated comparable performance with lower computational cost.

4 Conclusion and Future Work

Through a systematic evaluation of our internal runs – using `DataMorgana` to generate synthetic datasets and applying a combination of GoP and *n*-way ANOVA – we efficiently identified the most cost-effective combination of the component parameters to maximize system performance within the time constraints of the LiveRAG 2025 Challenge. This combined approach also enabled us to prioritize the most impactful components and methodically test and refine them, resulting in a well-optimized final configuration for our RAG system. Based on this evaluation, we selected GRAG as our submitted run. The system achieved a Relevance score of 1.199 and a Faithfulness score of 0.477, ranking third on the private LiveRAG 2025 leaderboard and selected as one of the top four finalists based on the final manual evaluation.

Future work will focus on a more detailed analysis of system components and specific cases, including unanswerable or out-of-knowledge-base questions [13]. We also plan to integrate post-retrieval Query Performance Prediction (QPP) to dynamically identify which questions would benefit from query variant expansion [14, 21].

5 Ethical Considerations

We acknowledge that the automated evaluation approach with LLMs used in the SIGIR LiveRAG 2025 challenge has limitations [1, 5, 6]. Higher Relevance and Faithfulness scores do not necessarily mean higher user satisfaction, and further validation with human annotations or user studies is needed.

It is also worth noting that we have not studied the unintended bias that may get amplified by the use of LLMs in the different stages of GRAG, including the generation of hypothetical answer, query variants, and re-ranking.

Acknowledgments

We thank the SIGIR 2025 LiveRAG Challenge organizers for the opportunity to participate and their support, and the reviewers for their helpful feedback. This research was conducted by the ARC Centre of Excellence for Automated Decision-Making and Society (ADM+S, CE200100005), and funded fully by the Australian Government through the Australian Research Council and was undertaken with the assistance of computing resources from RACE (RMIT AWS Cloud Supercomputing). This work was conducted on the unceded lands of the Woi wurrung and Boon wurrung language groups of the eastern Kulin Nation. We pay our respect to Ancestors and Elders, past and present, and extend that respect to all Aboriginal and Torres Strait Islander peoples today and their connections to land, sea, sky, and community.

References

[1] Marwah Alaofi, Negar Arabzadeh, Charles L. A. Clarke, and Mark Sanderson. 2025. *Generative Information Retrieval Evaluation.* Springer Nature Switzerland, Cham, 135–159. doi:10.1007/978-3-031-73147-1_6

[2] Marwah Alaofi, Luke Gallagher, Mark Sanderson, Falk Scholer, and Paul Thomas. 2023. Can Generative LLMs Create Query Variants for Test Collections? An Exploratory Study. In *Proc. SIGIR.* 1869–1873. doi:10.1145/3539618.3591960

[3] Gordon V. Cormack, Charles L A Clarke, and Stefan Buettcher. 2009. Reciprocal Rank Fusion Outperforms Condorcet and Individual Rank Learning Methods. In *Proc. SIGIR.* 758–759. doi:10.1145/1571941.1572114

[4] Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonellotto, and Fabrizio Silvestri. 2024. The Power of Noise: Redefining Retrieval for RAG Systems. In *Proc. SIGIR.* 719–729. doi:10.1145/3626772.3657834

[5] Laura Dietz, Oleg Zendel, Peter Bailey, Charles Clarke, Ellese Cotterill, Jeff Dalton, Faegheh Hasibi, Mark Sanderson, and Nick Craswell. 2025. Principles and Guidelines for the Use of LLM Judges. In *Proc. ICTIR.*

[6] Guglielmo Faggioli, Oleg Zendel, J Shane Culpepper, Nicola Ferro, and Falk Scholer. 2022. sMARE: A New Paradigm to Evaluate and Understand Query Performance Prediction Methods. *Information Retrieval Journal* 25, 2 (2022), 94–122. doi:10.1007/s10791-022-09407-w

[7] Nicola Ferro and Gianmaria Silvello. 2016. A General Linear Mixed Models Approach to Study System Component Effects. In *Proc. SIGIR.* 25–34. doi:10.1145/2911451.2911530

[8] Simone Filice, Guy Horowitz, David Carmel, Zohar Karnin, Liane Lewin-Eytan, and Yoelle Maarek. 2025. Generating Diverse Q&A Benchmarks for RAG Evaluation with DataMorgana. arXiv:2501.12789 [cs.CL]

[9] Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023. Precise Zero-Shot Dense Retrieval without Relevance Labels. In *Proc. ACL.* 1762–1777. doi:10.18653/v1/2023.acl-long.99

[10] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D Manning, Christopher Re, Diana Acosta-Navas, Drew A. Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue WANG, Keshav Santhanam, Laurel Orr, Lucia Zheng, Mert Yuksekgonul, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri S. Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Andrew Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. 2023. Holistic Evaluation of Language Models. *Transactions on Machine Learning Research* (2023), 162 pages. https://openreview.net/forum?id=iO4LZibEqW

[11] Scott E Maxwell, Harold D Delaney, and Ken Kelley. 2017. *Designing Experiments and Analyzing Data: A Model Comparison Perspective.* Routledge. doi:10.4324/9781315642956

[12] Erxue Min, Hsiu-Yuan Huang, Min Yang, Xihong Yang, Xin Jia, Yunfang Wu, Hengyi Cai, Junfeng Wang, Shuaiqiang Wang, and Dawei Yin. 2025. From Prompting to Alignment: A Generative Framework for Query Recommendation. arXiv:2504.10208 [cs.IR]

[13] Sachin Pathiyan Cherumanal, Lin Tian, Futoon M. Abushaqra, Angel Felipe Magnossão de Paula, Kaixin Ji, Halil Ali, Danula Hettiachchi, Johanne R. Trippas, Falk Scholer, and Damiano Spina. 2024. Walert: Putting Conversational Information Seeking Knowledge into Action by Building and Evaluating a Large Language Model-Powered Chatbot. In *Proc. CHIIR.* 401–405. doi:10.1145/3627508.3638309

[14] Kun Ran, Marwah Alaofi, Mark Sanderson, and Damiano Spina. 2025. Two Heads Are Better Than One: Improving Search Effectiveness Through LLM-Generated Query Variants. In *Proc. CHIIR.* 333–341. doi:10.1145/3698204.3716468

[15] SE Robertson, S Walker, S Jones, M Hancock-Beaulieu, and M Gatford. 1994. Okapi at TREC-3. In *Proc. of the Third Text REtrieval Conference (TREC 1994).* 18 pages. https://trec.nist.gov/pubs/trec3/papers/city.ps.gz

[16] Andrew Rutherford. 2011. *ANOVA and ANCOVA: A GLM Approach* (1 ed.). Wiley. doi:10.1002/9781118491683

[17] Shuoqi Sun, Shengyao Zhuang, Shuai Wang, and Guido Zuccon. 2025. An Investigation of Prompt Variations for Zero-shot LLM-based Rankers. In *ECIR.* 185–201. doi:10.1007/978-3-031-88711-6_12

[18] Manveer Singh Tamber, Forrest Sheng Bao, Chenyu Xu, Ge Luo, Suleman Kazi, Minseok Bae, Miaoran Li, Ofer Mendelevitch, Renyi Qu, and Jimmy Lin. 2025. Benchmarking LLM Faithfulness in RAG with Evolving Leaderboards. arXiv:2505.04847 [cs.CL]

[19] TII Team. 2024. The Falcon 3 family of Open Models.

[20] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2024. Text Embeddings by Weakly-Supervised Contrastive Pre-training. arXiv:2212.03533 [cs.CL]

[21] Oleg Zendel, J. Shane Culpepper, and Falk Scholer. 2021. Is Query Performance Prediction With Multiple Query Variations Harder Than Topic Performance Prediction?. In *Proc. SIGIR.* 1713–1717. doi:10.1145/3404835.3463039

A Prompts

A.1 Query Generation

A.1.1 Naïve Prompt.

System Prompt:

```
Generate a list of {k_queries} search query
variants based on the user's question, give
me one query variant per line. There are no
spelling mistakes in the original question.
Do not include any other text.
```

User Prompt:

```
{question}
```

A.1.2 Medium Prompt. Inspired by Min et al. [12], although the task in this paper is not totally aligned with query variants generation, we adapted it into a query generation task.

System Prompt:

```
You are an expert in query generation, you
will be given a question, please generate
{k_queries} relevant queries based on the question.
Make sure every query generated can yield
new information when I use them to search.
NEVER repeat similar search queries.
```

User Prompt:

```
Original question: {question}
```

A.1.3 Advanced Prompt.

System Prompt:

```
Generate {k_queries} diverse search query
variations for the given question. Follow
these guidelines:
1. Each query should focus on different aspects
or interpretations of the original question
2. Use synonyms and related terms where appropriate
3. Include both broad and specific variations
4. Maintain the core meaning while varying
the expression
5. Write each query on a new line
6. Do not include any additional text or formatting
The original question is correctly spelled.
```

User Prompt:

```
Question to analyze: {question}
Please generate diverse query variations that
capture different aspects of this question:
```

A.1.4 Hypothetical Answer Generation.

System Prompt:

```
Given the question, write a short hypothetical
answer that could be true. Be brief and concise.
```

User Prompt:

```
{question}
```

A.1.5 Question Decomposition.

System Prompt:

```
You are an experienced Google search user,
help the user breaking down a search question
into key components with shorthand entity
annotation in numbered list style
```

User Prompt:

```
Question: {question}
```

A.1.6 Query-to-Question Rephrasing.

System Prompt:

```
You are an experienced Google search user,
help the search engine to find the results
user wanted. Given the main question and its
components analysis, rephrase into a longer
question. What does the user really want?
```

User Prompt:

```
Question: {question}
{components_str}
```

A.1.7 Question Classifier.

System Prompt:

```
You are an experienced Google search user,
help the user determine if the search question
is a simple question or a composite question
that consists of multiple sub-questions. If
it's a simple question, you should respond:
SIMPLE, if it's a composite question, you
should respond: COMPOSITE.
```

User Prompt:

```
Question: {question}
```

A.1.8 Sub-question-to-Query Rephrasing.

System Prompt:

```
You are an experienced Google search user,
help the user to answer the question. Given
the main question, for each sub-question, create
a search query, row by row. Your generated
query must start with: query:
```

User Prompt:

```
Question: {question}
```

A.2 Documents Re-ranking

A.2.1 Pointwise LLM-based Re-ranker.

System Prompt:

```
You are a helpful assistant that determines
if a document contains information that helps
answer a given question. Answer only with
'Yes' or 'No'.
```

User Prompt:

```
Document: {doc_text}
Question: {question}
Does this document contain information that
helps answer this question (only answer `Yes'
or `No')?
```

A.3 Answer Generation

A.3.1 Naïve Prompt.

System Prompt:

```
You are a helpful assistant. Answer the question
based on the provided documents.
```

User Prompt:

```
Documents: {context}
Question: {question}
Answer:
```

A.3.2 Medium Prompt. We adapted Cuconasu et al. [4] to support unanswerable questions in the LiveRAG challenge, e.g., we replaced "No-RES" with "I don't know". The reason is that LiveRAG challenge suggests that if the system does not know the answer, it is better to generate "I don't know", instead of the wrong answer.

System Prompt:

```
You are given a question and you MUST respond
by EXTRACTING the answer from provided documents.
If none of the documents contain the answer,
respond with *`I don't know'*.
```

User Prompt:

```
Documents: {context}
Question: {question}
Answer:
```

A.3.3 Advanced Prompt. We adapted Tamber et al.'s prompt Tamber et al. [18], used for their LLM-as-a-judge approach – FaithJudge.[8]

System Prompt:

```
You must respond based strictly on the information
in provided passages. Do not incorporate any
external knowledge or infer any details beyond
what is given in the passages.
```

User Prompt:

```
Provide a concise answer to the following
question based on the information in the provided
documents. Documents:
{context}
*Question: {question}*
Answer:
```

A.4 LLM-based Evaluation

We are evaluating the relevance and faithfulness scores of an answer and the corresponding document rankings separately, in different LLM requests with different sets of prompts.

[8] https://github.com/vectara/FaithJudge

A.4.1 Relevance Score.

System Prompt:

```
You are an expert evaluator for Retrieval-
Augmented Generation (RAG) systems.
Your task is to assess the quality of responses
generated by a RAG system based on the relevance
(correctness) criteria:
Relevance - Measures the correctness and relevance
of the answer to the question on a four-point
scale:
2: The response correctly answers the user
question and contains no irrelevant content
1: The response provides a useful answer to
the user question, but may contain irrelevant
content that do not harm the usefulness of
the answer
0: No answer is provided in the response (e.g.,
"I don't know")
-1: The response does not answer the question
whatsoever
You will be provided with:
- A question
- The response generated by the RAG system
- The retrieved documents used as context
- A gold reference answer (if available)
When a gold reference answer is provided, use
it as an additional reference point for evaluating
the correctness and completeness of the RAG
system's response. The gold reference represents
an ideal answer to the question.
Provide your evaluation in a structured JSON
format with the following fields:
- evaluation_notes: Brief explanation of your
reasoning for each score
- relevance_score: The relevance score (-1,
0, 1, or 2)
Be objective and thorough in your assessment.
Focus on whether the response correctly answers
the question.
```

User Prompt:

````
Please evaluate the following RAG system response:
QUESTION:
{question}
RESPONSE:
{answer}
GOLD REFERENCE ANSWER:
{reference_answer}
RETRIEVED DOCUMENTS:
{documents}
Based on the above, please evaluate the response
on relevance (2, 1, 0, or -1).
Provide your evaluation in the following JSON
format:
```json
{{
"evaluation_notes": "[your reasoning in a
single paragraph]",
"relevance_score": [score]
}}
```
````

A.4.2 Faithfulness Score.

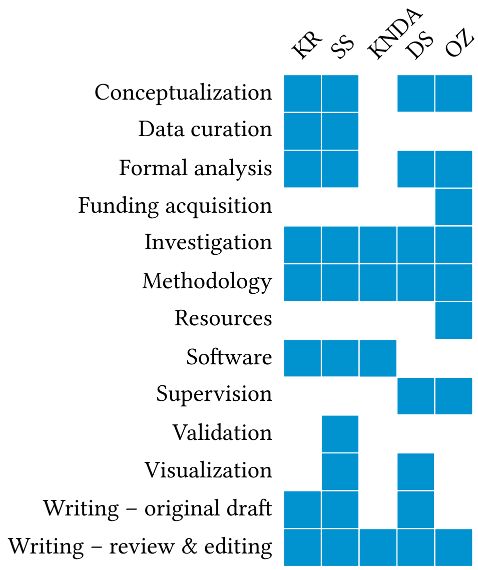
System Prompt:

```
You are an expert evaluator for Retrieval-
Augmented Generation (RAG) systems.
Your task is to assess the quality of responses
generated by a RAG system based on the faithfulness
(support) criteria:
Assess whether the response is grounded in
the retrieved passages on a three-point scale:
1: Full support, all answer parts are grounded
0: Partial support, not all answer parts are
grounded
-1: No support, all answer parts are not grounded
You will be provided with:
- A question
- The response generated by the RAG system
- The retrieved documents used as context
Provide your evaluation in a structured JSON
format with the following fields:
- evaluation_notes: Brief explanation of your
reasoning for each score
- faithfulness_score: The faithfulness score
(-1, 0, or 1)
Be objective and thorough in your assessment.
Focus on whether the response correctly answers
the question and is supported by the retrieved
documents.
```

User Prompt:

````
Please evaluate the following RAG system response:
QUESTION:
{question}
RESPONSE:
{answer}
GOLD REFERENCE ANSWER:
{reference_answer}
RETRIEVED DOCUMENTS:
{documents}
Based on the above, please evaluate the response
on faithfulness (1, 0, or -1).
Provide your evaluation in the following JSON
format:
```json
{{
"evaluation_notes": "[your reasoning in a
single paragraph]",
"faithfulness_score": [score]
}}
```
````

B CRediT Author Statements



Conceptualization: KR, SS, DS, OZ
Data curation: KR, SS
Formal analysis: KR, SS, DS, OZ
Funding acquisition: OZ
Investigation: KR, SS, KNDA, DS, OZ
Methodology: KR, SS, KNDA, DS, OZ
Resources: OZ
Software: KR, SS, KNDA
Supervision: DS, OZ
Validation: SS
Visualization: SS, DS
Writing – original draft: KR, SS, DS
Writing – review & editing: KR, SS, KNDA, DS, OZ.