UiS-IAI@LiveRAG: Retrieval-Augmented Information Nugget-Based Generation of Responses

Weronika Łajewska
University of Stavanger
Stavanger, Norway
weronika.lajewska@uis.no

Ivica Kostric
University of Stavanger
Stavanger, Norway
ivica.kostric@uis.no

Gabriel Iturra-Bocaz
University of Stavanger
Stavanger, Norway
gabriel.e.iturrabocaz@uis.no

Mariam Arustashvili
University of Stavanger
Stavanger, Norway
mariam.arustashvili@uis.no

Krisztian Balog
University of Stavanger
Stavanger, Norway
krisztian.balog@uis.no

Abstract

Retrieval-augmented generation (RAG) faces challenges related to factual correctness, source attribution, and response completeness. The LiveRAG Challenge hosted at SIGIR'25 aims to advance RAG research using a fixed corpus and a shared, open-source LLM. We propose a modular pipeline that operates on information nuggets—minimal, atomic units of relevant information extracted from retrieved documents. This multistage pipeline encompasses query rewriting, passage retrieval and reranking, nugget detection and clustering, cluster ranking and summarization, and response fluency enhancement. This design inherently promotes grounding in specific facts, facilitates source attribution, and ensures maximum information inclusion within length constraints. In this challenge, we extend our focus to also address the retrieval component of RAG, building upon our prior work on multi-faceted query rewriting. Furthermore, for augmented generation, we concentrate on improving context curation capabilities, maximizing the breadth of information covered in the response while ensuring pipeline efficiency. Our results show that combining original queries with a few sub-query rewrites boosts recall, while increasing the number of documents used for reranking and generation beyond a certain point reduces effectiveness, without improving response quality.

CCS Concepts

• **Computing methodologies** → **Natural language generation**;
• **Information systems** → **Information extraction**.

Keywords

Retrieval-augmented generation; Query Rewriting; Grounding

1 Introduction

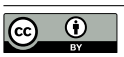
The increasing reliance on conversational assistants such as ChatGPT for complex open-ended queries [2, 9, 42] presents challenges in factual correctness [16, 18, 35], source attribution [32], information verifiability [23], consistency, and coverage [11]. Although retrieval-augmented generation models aim to build responses based on retrieved sources [11, 14, 22], they often struggle with transparency and source attribution. Current generative search engines frequently produce unsupported claims and inaccurate citations [23], underscoring the need for more reliable grounding. Although injecting evidence into prompts can mitigate hallucinations, long and redundant contexts can lead to the "lost in the middle" problem, where relevant information becomes inaccessible [24]. A post-retrieval refinement step is recommended to retain only essential details while preserving key information [10].

To address these limitations, we use a modular system for retrieval-augmented nugget-based response generation. It combines a strong retrieval pipeline with query rewriting, sparse and dense retrieval, and reranking with **G**rounded **I**nformation **N**ugget-Based **GE**neration of **R**esponses (`GINGER`) [21] (see Figure 1). Unlike traditional RAG approaches, our method operates on atomic units of relevant information, called information nuggets [26]. Response generation involves identifying and clustering nuggets detected in retrieved passages, ranking clusters by relevance, summarizing them to eliminate redundancy, and then refining these summaries into a final, cohesive response. This process ensures comprehensive yet concise answers, maintains strong source attribution, and, as demonstrated in the TREC RAG'24 augmented generation task, significantly outperforms strong baselines. The core strength of GINGER lies in the granular, nugget-based processing of highly relevant information.

When developing our pipeline for the LiveRAG Challenge,[1], we conducted experiments on the TREC RAG'24 dataset as well as a small test dataset generated with DataMorgana [7]. Our results show that naive answer-based or single sub-question query rewriting can harm retrieval effectiveness, while combining the original query with a few diverse rewrites improves recall. Furthermore, optimizing reranking and generation parameters reveals that response quality improves only up to a point, beyond which sacrificing time efficiency yields limited gains.

2 Related Work

Unlike traditional search engines that return a ranked list of documents, RAG systems provide a single, comprehensive response by synthesizing varied perspectives from multiple sources, blending the language fluency and world knowledge of generative models with retrieved evidence [11, 25]. In retrieve-then-generate systems, generative processes are conditioned on retrieved material by



[1] https://liverag.tii.ae/

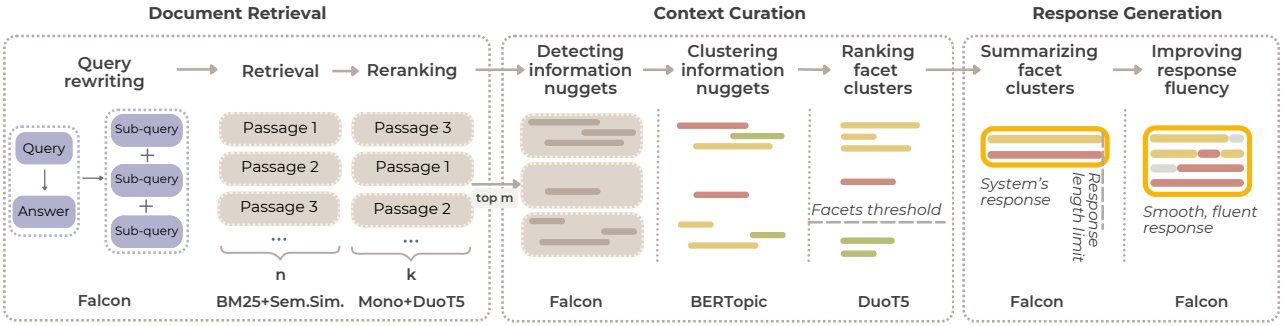


Figure 1: High-level overview of our retrieval-augmented nugget-based response generation pipeline (GINGER).

adding evidence to the prompt [15, 31, 33] or attending to sources during inference.

Systems submitted to the Retrieval-Augmented Generation track at the Text REtrieval Conference (TREC RAG'24) [29] have adopted modular architectures that improve the retrieval component by combining sparse and dense retrieval models, followed by reranking with models such as MonoT5 and DuoT5 [27], RankZephyr [28], or other LLM-based graded relevance scoring. A notable enhancement involves query decomposition using an LLM to generate sub-questions, each addressing different facets of the information need. While LLM-based rewriting is well-established [5, 39], the generation of multiple diverse reformulations per query is a more recent development that shows strong potential for boosting recall and robustness by expanding the query's semantic coverage [19, 30]. Retrieved and reranked results from these variants are typically merged using reciprocal rank fusion (RRF) [38].

For the generation stage, the most simplistic approach is to use proprietary models to generate responses in a single step based on the provided documents. However, ad hoc retrieval often returns documents with only partial relevance [26], and placing relevant content in the middle of a long prompt can degrade generation quality [24]. While generative models often produce fluent and seemingly helpful responses, they frequently suffer from hallucinations and factual errors [16, 20, 23, 36]. These limitations motivate more advanced context curation strategies, including unimportant token removal [17], content aggregation [43], and training extractors and condensers [40, 41]. Approaches at TREC RAG'24 include extracting, combining, and condensing the relevant information [8], enhanced by verifying key facts across documents, rule-based redundancy removal, and enhancing coherence [6].

3 Retrieval-Augmented Nugget-Based Response Generation

Our approach, GINGER (which stands for **G**rounded **I**nformation **N**ugget-Based **GE**neration of **R**esponses), operates on information nuggets. It explicitly models various facets of the query based on retrieved information and generates a concise response that adheres to length constraints. It generates the response in three steps by: (1) retrieving top relevant passages from the corpus, (2) curating retrieved context for response generation, and (3) synthesizing the collected information into a final response; see Figure 1. Our implementation adopts a modular architecture, with clearly separated components for each stage of the pipeline. This design allows for flexible experimentation and independent development of each component. All generation tasks, including query rewriting and context curation, are performed with the Falcon3-10B model[2] accessed via the AI71 platform API.[3]

3.1 Document Retrieval

To reduce omissions caused by narrow queries, we apply query rewriting before retrieval. An LLM, queried without external documents, first generates a short answer to the original question. The assumption is that this intermediate answer surfaces the key aspects of the information need. We then ask the same model to generate l additional queries, each focusing on a different aspect of that provisional answer while staying semantically consistent with the initial query.[4] We combine each expanded query with the original, and then concatenate all l rewrites together to create a final search string. Formally,

$$q' = (q + q'_1) + \cdots + (q + q'_l)$$

where q is the original query, and each q'_i is a rewrite focusing on a different aspect of the intermediate answer.

For retrieval, we adopt a two-stage retrieval pipeline, consisting of an initial passage retrieval step followed by re-ranking. First-pass retrieval is a combination of rankings obtained using both sparse and dense text representations. We use BM25 for sparse retrieval with an Opensearch-based index[5] and `intfloat/e5-base-v2`[6] embeddings with a Pinecone dense index.[7] Both indices are pre-built and provided by the challenge organizers. The retrieval results are then combined using reciprocal rank fusion [4]. For re-ranking, we first apply a pointwise re-ranker (`castorini/monot5-base-msmarco`),[8] followed by a pairwise re-ranker (`castorini/duot5-base-msmarco`),[9] both fine-tuned on the MS MARCO collection [3], to refine the ranking and improve retrieval effectiveness.

[2] https://huggingface.co/tiiuae/Falcon3-10B-Instruct
[3] https://ai71.ai/
[4] Prompts used for query rewriting can be found in Appendix B.1.
[5] https://opensearch.org/
[6] https://huggingface.co/intfloat/e5-base-v2
[7] https://www.pinecone.io/
[8] https://huggingface.co/castorini/monot5-base-msmarco
[9] https://huggingface.co/castorini/duot5-base-msmarco

3.2 Context Curation

Given the retrieved passages, GINGER curates the context before the generation step to optimize response grounding and information relevance. First, we detect information nuggets within the top-m ranked passages by prompting an LLM to annotate key information without altering the original text.[10] Detected nuggets are then clustered according to different query facets to reduce redundancy and increase information density [1], leveraging the BERTopic model [13]. Next, facet clusters are ranked for relevance using DuoT5 pairwise reranking ensuring that the most crucial clusters are prioritized for response generation [10, 24]. This structured approach enables GINGER to distill key information while preserving source attribution.

3.3 Response Generation

In the last step, GINGER transforms the ranked facet clusters into a coherent response. Each top-ranked cluster is independently summarized into one sentence, following a prompt design that enforces conciseness and faithfulness to the original content [12, 34].[11] This modular summarization process ensures that the response remains factually accurate and grounded. However, since the response composed of independently summarized texts may lack fluency and coherence, we introduce a final refinement step where an LLM rephrases the response without introducing additional content. This ensures that the final output is not only factually reliable but also natural and readable, improving the overall user experience.

3.4 Batch Processing Details

To improve the efficiency of our pipeline, queries are processed in batches. We implemented multiprocessing with a concurrent queuing system, allowing each pipeline component to operate independently as long as its input queue is populated. This prevented bottlenecks and maximized hardware utilization. GPU-intensive components were distributed across 12 GPUs, with pointwise reranking and response generation using 25% of total GPU resources and pairwise reranking the remaining 75%. During the challenge day, we used 8 Tesla V100 GPUs and 4 NVIDIA A100 GPUs.

4 Experiments

In our experiments, we investigate our system's robustness with respect to the quality of the retrieved information. We also evaluate its ability to synthesize content from retrieved passages and reduce redundancy. The main goal of these experiments is to find a balance between efficiency—ensuring that responses can be generated for all test queries within a limited time window on the challenge day—and the quality of the generated responses.

4.1 Datasets

We generated a test set of 100 instances using the DataMorgana API, a synthetic benchmark generator platform used in the LiveRAG challenge [7]. DataMorgana enables RAG developers to create synthetic questions and answers from a given corpus based on configurable instructions. Half of the questions in our test set have answers grounded in a single document, while the other half are based on two documents. We experimented with several question categorizations proposed in the original paper, including factuality, premise, phrasing, and linguistic variation (see Table 3 in Appendix A). Additionally, we incorporated the user expertise categorization and introduced two new categories for multi-document questions: comparisons between two entities and questions covering two aspects of the same topic. The documents provided by DataMorgana for each question are treated as ground-truth passages, and the generated answers serve as references to evaluate our system's responses.

We additionally employed the TREC RAG'24 dataset [29], derived from the MS MARCO v2.1 collection and containing 301 information-seeking queries with graded relevance judgments. Unlike DataMorgana, which offers at most two judged passages per query, TREC RAG provides relevance labels for many candidate documents, giving a more reliable signal for retrieval evaluation. We used these judgments to benchmark the query rewriting component.

4.2 Evaluation

We evaluate the effectiveness of query rewriting primarily using the TREC RAG'24 dataset. The main metric is *Recall@500*, computed using the trec_eval tool.[12] This cutoff corresponds to the number of top-ranked documents passed onto the pointwise reranker. We use the original query without any rewriting as the baseline.

For response generation, we use the AutoNuggetizer framework proposed for RAG evaluation and validated at TREC RAG'24 [29]. AutoNuggetizer comprises two steps: nugget creation and nugget assignment. In nugget creation, nuggets are formulated based on relevant documents and classified as either "vital" or "okay" [37]. The second step, nugget assignment, involves assessing whether a system response contains specific nuggets from the answer key. The score V_{strict} for the system's response is defined as:

$$V_{strict} = \frac{\sum_i ss_i^v}{|n^v|},$$

where n^v represents the subset of the vital nuggets, and ss_i^v is 1 if the response supports the i-th nugget and is 0 otherwise. The score of a system is the mean of the scores across all queries.

4.3 Results

Results in Table 1 show that using a single rewrite alone underperforms even the original query, suggesting that naive rewriting can hurt retrieval effectiveness. While combining the original query with multiple rewrites improves recall, the gains saturate quickly. Adding more than three rewrites yields only marginal improvements, indicating diminishing returns beyond a small number of diverse reformulations. Notably, the recall achieved by using multiple rewrites alone is consistently lower than the recall obtained when those rewrites are concatenated with the original query, underscoring the importance of preserving the original formulation.[13]

[10] Prompts used for context curation can be found in Appendix B.2.
[11] Prompts used for response generation can be found in Appendix B.3.
[12] https://github.com/usnistgov/trec_eval
[13] These experiments use TREC RAG data with a different retrieval collection, so the comparison to our pipeline is not direct. However, since we evaluate only the query rewriting component with retrieval frozen, the findings are expected to generalize to similar retrieval setups.

Table 1: Recall@500 for different query rewriting strategies on the TREC RAG'24 dataset. The best-performing configuration is shown in bold. Teal background indicates the configuration used in the final submission.

Rewriting Strategy	R@500
Original Query	0.320
Single Rewrite	0.217
Multi Rewrite (3)	0.325
Multi Rewrite (10)	0.357
Original Query + Single Rewrite	0.343
Original Query + Multi Rewrite (3)	0.397
Original Query + Multi Rewrite (5)	**0.400**
Original Query + Multi Rewrite (10)	0.398

Table 2 presents the evaluation of responses generated using different GINGER configurations, assessed with the AutoNuggetizer framework. We varied two key parameters: the number of documents used for pairwise reranking (k) and the number of documents used for response generation (m). These parameters directly impact both the quality of the generated responses and the system's efficiency. The reranking step with DuoT5 scales exponentially with k, while the number of Falcon API calls—dependent on m—is the main bottleneck in information nugget detection.

Given the two-hour time limit for processing 500 queries during the challenge (with three parallel processes), we aimed for a setup capable of handling at least 100 queries per hour. Although the setup with $k = 50$ and $m = 20$ produced the best responses, it exceeded our time constraints. Configurations with $k = 40$, $m = 10$ and $k = 20$, $m = 10$ yielded similar scores with much more efficient runtimes. Despite $k = 20$ scoring slightly higher, we selected $k = 40$ for our final submission to increase topic coverage and response diversity.

This choice is further supported by the limitations of AutoNuggetizer, which evaluates responses using nuggets extracted from only two documents. As a result, it may overlook relevant content captured by a broader reranking scope. In our manual analysis, we observed low scores for responses that were clearly grounded in relevant retrieved passages but where the available ground-truth nuggets were sparse. Conversely, high scores occurred mainly when our responses aligned exactly with the nuggets identified by AutoNuggetizer. This suggests that the framework's effectiveness is constrained by its limited access to reference passages, which in turn restricts the evaluation of information quality.

4.4 Lessons Learned

Participating in the LiveRAG challenge underscored the need to balance time efficiency with handling diverse query types. The time limit and the diversity of questions generated with DataMorgana posed unexpected challenges, requiring careful pipeline tuning and manual analysis.

Our initial query rewriting strategy, designed to sharpen the focus of the question using potential answer clues, worked well for factoid questions but underperformed for open-ended queries, where broader context is needed. This led us to revise our approach:

Table 2: Evaluation with AutoNuggetizer of responses generated with GINGER using different setups. All variants use the top $n = 500$ retrieved documents for pointwise reranking. Teal background indicates the configuration used in the final submission.

Pairwise reranking	Response generation	V_strict	Time estimate
$k = 50$	$m = 20$	**0.406**	70 min
$k = 40$	$m = 10$	0.397	41 min
$k = 20$	$m = 10$	0.404	42 min
$k = 20$	$m = 5$	0.350	26 min

using rewritten queries only for retrieval to ensure a diverse document pool, while letting reranking and generation rely on the original query to maintain relevance.

To meet the strict time window on challenge day, we had to rigorously optimize our system for efficiency. This involved extensive use of multiprocessing, batching, and distributing processes across multiple GPUs. The most resource-intensive component was the pairwise reranking stage, and the heavy reliance on the Falcon model across modules strained API rate limits. These constraints forced us to reduce the number of documents processed at each stage, carefully balancing efficiency against the quality of generated responses.

Finally, evaluating the responses with AutoNuggetizer surfaced key limitations of the framework. Its effectiveness depends on having a rich set of ground-truth nuggets derived from a broad set of relevant passages. In practice, especially for open-ended queries, this was often not the case, leading to unfairly low scores for responses that were, in fact, well grounded. This experience underlines the need for more robust response evaluation strategies, particularly when testing with limited access to ground-truth sources.

5 Conclusions

This paper has presented our participation in the LiveRAG Challenge at SIGIR'25, proposing a modular system for retrieval-augmented, nugget-based response generation. Our approach integrates query rewriting, sparse and dense retrieval, and reranking within the Grounded Information Nugget-Based Generation of Responses (GINGER) framework. Evaluation on the TREC RAG'24 dataset and QA test samples from DataMorgana using the AutoNuggetizer framework demonstrates that our system effectively balances time efficiency and response quality.

References

[1] Griffin Adams, Alex Fabbri, Faisal Ladhak, Eric Lehman, and Noémie Elhadad. 2023. From Sparse to Dense: GPT-4 Summarization with Chain of Density Prompting. *Proceedings of the 4th New Frontiers in Summarization Workshop* (2023), 68–74.

[2] Valeriia Bolotova-Baranova, Vladislav Blinov, Sofya Filippova, Falk Scholer, and Mark Sanderson. 2023. WikiHowQA: A Comprehensive Benchmark for Multi-Document Non-Factoid Question Answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (ACL '23)*. 5291–5314.

[3] Daniel Fernando Campos, Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, Li Deng, and Bhaskar Mitra. 2016.

MS MARCO: A Human Generated MAchine Reading Comprehension Dataset. arXiv:1611.09268 [cs.CL]

[4] Gordon V. Cormack, Charles L. A. Clarke, and Stefan Büttcher. 2009. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval (SIGIR '09)*.

[5] Kaustubh D. Dhole and Eugene Agichtein. 2024. GenQREnsemble: Zero-Shot LLM Ensemble Prompting for Generative Query Reformulation. In *Advances in Information Retrieval: 46th European Conference on Information Retrieval (ECIR '24)*. 326––335.

[6] Naghmeh Farzi and Laura Dietz. 2024. TREMA-UNH at TREC: RAG Systems and RUBRIC-style Evaluation. In *The Thirty-Third Text REtrieval Conference Proceedings (TREC '24)*.

[7] 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]

[8] Maik Fröbe, Lukas Gienapp, Harrisen Scells, Eric Oliver Schmidt, Matti Wiegmann, Martin Potthast, and Matthias Hagen. 2024. Webis at TREC 2024: Biomedical Generative Retrieval, Retrieval-Augmented Generation, and Tip-of-the-Tongue Tracks. In *The Thirty-Third Text REtrieval Conference Proceedings (TREC '24)*.

[9] Matteo Gabburo, Nicolaas Paul Jedema, Siddhant Garg, Leonardo F. R. Ribeiro, and Alessandro Moschitti. 2024. Measuring Retrieval Complexity in Question Answering Systems. In *Findings of the Association for Computational Linguistics: ACL 2024 (ACL '24')*. 14636–14650.

[10] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2023. Retrieval-Augmented Generation for Large Language Models: A Survey. arXiv:2312.10997 [cs.CL]

[11] Lukas Gienapp, Harrisen Scells, Niklas Deckers, Janek Bevendorff, Shuai Wang, Johannes Kiesel, Shahbaz Syed, Maik Fröbe, Guido Zuccon, Benno Stein, Matthias Hagen, and Martin Potthast. 2024. Evaluating Generative Ad Hoc Information Retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '24)*. 1916–1929.

[12] Tanya Goyal, Junyi Jessy Li, and Greg Durrett. 2023. News Summarization and Evaluation in the Era of GPT-3. arXiv:2209.12356 [cs.CL]

[13] Maarten Grootendorst. 2022. BERTopic: Neural topic modeling with a class-based TF-IDF procedure. (2022). arXiv:2203.05794 [cs.CL]

[14] Yizheng Huang and Jimmy Huang. 2024. A Survey on Retrieval-Augmented Text Generation for Large Language Models. arXiv:2203.05794 [cs.CL]

[15] Gautier Izacard and Edouard Grave. 2021. Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume (EACL '21)*. 874–880.

[16] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of Hallucination in Natural Language Generation. *Comput. Surveys* 55, 12 (2023), 1–38.

[17] Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2024. LongLLMLingua: Accelerating and Enhancing LLMs in Long Context Scenarios via Prompt Compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (ALC '24)*. 1658–1677.

[18] Bevan Koopman and Guido Zuccon. 2023. Dr ChatGPT tell me what I want to hear: How different prompts impact health answer correctness. In *Findings of the Association for Computational Linguistics: EMNLP 2023 (EMNLP '23)*. 15012–15022.

[19] Ivica Kostric and Krisztian Balog. 2024. A Surprisingly Simple yet Effective Multi-Query Rewriting Method for Conversational Passage Retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '24)*. 2271–2275.

[20] Faisal Ladhak, Esin Durmus, He He, Claire Cardie, and Kathleen McKeown. 2022. Faithful or Extractive? On Mitigating the Faithfulness-Abstractiveness Trade-off in Abstractive Summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (ACL '22)*. 1410–1421.

[21] Weronika Łajewska and Krisztian Balog. 2025. GINGER: Grounded Information Nugget-Based Generation of Responses. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '25)*.

[22] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS '20)*. 9459–9474.

[23] Nelson Liu, Tianyi Zhang, and Percy Liang. 2023. Evaluating Verifiability in Generative Search Engines. In *Findings of the Association for Computational Linguistics: EMNLP 2023 (EMNLP '23)*. 7001–7025.

[24] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the Middle: How Language Models Use Long Contexts. *Transactions of the Association for Computational Linguistics* 12 (2024), 157–173.

[25] Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ram Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, Edouard Grave, Yann LeCun, and Thomas Scialom. 2023. Augmented Language Models: a Survey. arXiv:2302.07842 [cs.CL]

[26] Virgil Pavlu, Shahzad Rajput, Peter B. Golbus, and Javed A. Aslam. 2012. IR system evaluation using nugget-based test collections. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining (WSDM '12)*. 393–402.

[27] Ronak Pradeep, Rodrigo Nogueira, and Jimmy Lin. 2021. The Expando-Mono-Duo Design Pattern for Text Ranking with Pretrained Sequence-to-Sequence Models. arXiv:2101.05667 [cs.IR]

[28] Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023. RankZephyr: Effective and Robust Zero-Shot Listwise Reranking is a Breeze! arXiv:2312.02724 [cs.IR]

[29] Ronak Pradeep, Nandan Thakur, Shivani Upadhyay, Daniel Campos, Nick Craswell, and Jimmy Lin. 2024. Initial Nugget Evaluation Results for the TREC 2024 RAG Track with the AutoNuggetizer Framework. arXiv:2411.09607 [cs.IR]

[30] Zackary Rackauckas. 2024. RAG-Fusion: a New Take on Retrieval-Augmented Generation. *International Journal on Natural Language Computing* 13, 1 (2024), 37–47.

[31] Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-Context Retrieval-Augmented Language Models. *Transactions of the Association for Computational Linguistics* 11 (2023), 1316–1331.

[32] Hannah Rashkin, Vitaly Nikolaev, Matthew Lamm, Lora Aroyo, and Michael Collins. 2021. Measuring Attribution in Natural Language Generation Models. *Computational Linguistics* 49, 4 (2021), 777–840.

[33] Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2023. REPLUG: Retrieval-Augmented Black-Box Language Models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers) (NAACL-HLT '24)*. 8371–8384.

[34] Melanie Subbiah, Sean Zhang, Lydia B. Chilton, and Kathleen McKeown. 2024. Reading Subtext: Evaluating Large Language Models on Short Story Summarization with Writers. arXiv:2403.01061 [cs.CL]

[35] Liyan Tang, Tanya Goyal, Alex Fabbri, Philippe Laban, Jiacheng Xu, Semih Yavuz, Wojciech Kryscinski, Justin Rousseau, and Greg Durrett. 2023. Understanding Factual Errors in Summarization: Errors, Summarizers, Datasets, Error Detectors. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (ACL '23)*. 11626–11644.

[36] Xiangru Tang, Alexander Fabbri, Haoran Li, Ziming Mao, Griffin Adams, Borui Wang, Asli Celikyilmaz, Yashar Mehdad, and Dragomir Radev. 2022. Investigating Crowdsourcing Protocols for Evaluating the Factual Consistency of Summaries. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT '22)*. 5680–5692.

[37] Ellen M. Voorhees. 2003. Overview of the TREC 2003 Question Answering Track. In *The Twelfth Text REtrieval Conference Proceedings (TREC '03)*.

[38] Yue Wang, John M. Conroy, Neil Molino, Julia Yang, and Mike Green. 2024. Laboratory for Analytic Sciences in TREC 2024 Retrieval Augmented Generation Track. In *The Thirty-Third Text REtrieval Conference Proceedings (TREC '24)*.

[39] Orion Weller, Kyle Lo, David Wadden, Dawn Lawrie, Benjamin Van Durme, Arman Cohan, and Luca Soldaini. 2024. When do Generative Query and Document Expansions Fail? A Comprehensive Study Across Methods, Retrievers, and Datasets. In *Findings of the Association for Computational Linguistics: EACL 2024 (ACL '24)*. 1987–2003.

[40] Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2023. RECOMP: Improving Retrieval-Augmented LMs with Compression and Selective Augmentation. arXiv:2310.04408 [cs.CL]

[41] Haoyan Yang, Zhitao Li, Yong Zhang, Jianzong Wang, Ning Cheng, Ming Li, and Jing Xiao. 2023. PRCA: Fitting Black-Box Large Language Models for Retrieval Question Answering via Pluggable Reward-Driven Contextual Adapter. In *Findings of the Association for Computational Linguistics: EMNLP 2023 (EMNLP '23)*. 5364–5375.

[42] Hamed Zamani, Johanne R. Trippas, Jeff Dalton, and Filip Radlinski. 2023. Conversational Information Seeking. *Foundations and Trends in Information Retrieval* 17, 3-4 (2023), 244–456.

[43] Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan Ö Arik. 2024. Chain of Agents: Large Language Models Collaborating on Long-Context Tasks. arXiv:2406.02818 [cs.CL]

Table 3: Categorizations used in DataMorgana to generate our test samples.

	Category	Description
Factuality	Factoid	Seeks a specific fact (e.g., date, number)
	Open-ended	Invites elaborative or exploratory answers
Premise	Direct	No premise or context about the user
	With Premise	Includes short user-relevant background info
Phrasing	Concise and Natural	Natural, direct questions (<10 words)
	Verbose and Natural	Natural questions with more than 9 words
	Short Search Query	Keyword-style, <7 words, no punctuation
	Long Search Query	Keyword-style, >6 words, no punctuation
Linguistic Variation	Similar to Document	Uses terms and phrasing from the source documents
	Distant from Document	Uses different wording than the source documents
User Expertise	Expert	Asks complex, domain-specific questions
	Common Person	Asks basic, general-interest questions
Answer Type	Multi-Aspect	Covers two aspects of the same topic; needs info from two documents
	Comparison	Compares two entities; each described in separate documents

Appendix

A Datasets

Categorizations used in DataMorgana to generate our test samples are presented in Table 3.

B Prompts

This section presents all the prompts used by our system for query rewriting, context curation and final response generation.

B.1 Query Rewriting

Prompt for generating a concise answer to the query using the Falcon model:

```
System: You are a knowledgeable question answering AI that can answer a wide range of queries either in question form or keywords.
User: {query}.
```

Prompt for rewriting query into a richer natural language variant:

```
System: You are a query rewriter that understands all necessary components of a good search query and helps users improve their queries.
User: Rewrite and return 3 query rewrites, each of which should cover a different aspect of the answer. The query rewrites should still be relevant to the original query. Return only the queries, one in each line. Do not add context, or any other information, or text.

original query: {query}
answer: {answer}
```

B.2 Context Curation

Prompt for detecting information nuggets in a passage given a query:

```
System: You are given a query and a relevant passage. Your task is to pinpoint and annotate the succinct excerpts within the passage that directly respond to the query. Ensure these excerpts are brief yet complete. Once identified, copy the entire passage and encapsulate the relevant snippets using <START> and </END> tags without changing any part of the original text. This includes avoiding modifications to words, punctuation, or formatting, as well as not adding any extra characters, symbols, or spaces.
User: Question: {query} Passage: {passage}
```

B.3 Response Generation

Prompt for summarizing an information cluster into a one-sentence-long text:

```
System: Summarize the provided information into one sentence (approximately 35 words). Generate one-sentence long summary that is short, concise and only contains the information provided.
User: {information_cluster}.
```

Prompt for improving the fluency of the generated response:

```
System: Rephrase the response given a query to improve its fluency. Do not change the information included in the response. Do not add information not mentioned in the original response.
User: Question: {query} Response: {response}
```