RAG4Reports 2026

The First Workshop on Multilingual Report Generation via Retrieval Augmented Generation (RAG4Reports)

Proceedings of the Workshop

July 4, 2026

Order copies of this and other ACL proceedings from:

Association for Computational Linguistics (ACL)
317 Sidney Baker St. S
Suite 400 - 134
Kerrville, TX 78028
USA
Tel: +1-855-225-1962
acl@aclweb.org

ISBN 979-8-89176-417-0

Preface

Welcome to the First Workshop on Multilingual Report Generation via Retrieval Augmented Generation (RAG4Reports), a half-day workshop co-located with ACL 2026 in San Diego, CA, USA.

> *A workshop on multilingual long-form report generation focusing on evaluating faithfulness and information coverage, tying together retrieval, generation, and evaluation research.*

Incorporating external source information with parametric knowledge from large language models to provide a comprehensive and grounded response to the users has become a central component in modern AI applications. *Report generation* is a long-form retrieval-augmented generation (RAG) task with strict attestation requirements that makes it well-suited to explore questions of RAG evaluation and generation: a long-form report summarizing the relevant information in a corpus is produced in response to a report request, and the generated report should provide proper attribution to source documents to establish trust. RAG4Reports 2026 focuses on two urgent problems in report generation that require a community effort to tackle: *evaluation* and *multilinguality*. While numerous research papers on RAG have been published in recent years, including proposed evaluation approaches, the community has yet to reach a consensus on benchmarks and evaluation measures for long-form, citation-grounded outputs. At the same time, as generation models are increasingly multilingual and capable of incorporating source information in different languages, report generation systems should be fair to different languages and produce reports that the user can consume from relevant sources regardless of languages. The aim of this workshop is to bring together researchers from information retrieval, natural language processing, and applied domains to establish common ground for developing and evaluating multilingual report generation systems.

The workshop solicited contributions through two tracks. The *research track* received 11 submissions through OpenReview, of which 9 were accepted; of these, one was subsequently withdrawn and two were accepted as non-archival submissions. The *shared task track* attracted 21 registered teams, from which we received 8 system description papers. The proceedings therefore contain 14 papers (6 research and 8 shared task), and 16 accepted papers are presented at the workshop as posters, with a subset selected for oral presentation.

A central component of this workshop is the RAG4Reports shared task, which builds on the 2024 NeuCLIR[1] and 2025 RAGTIME[2] shared tasks at the Text Retrieval Conference (TREC) and extends them to the natural language understanding community, with an explicit focus on non-English languages. The shared task was hosted on TIRA[3], which enables reproducible system evaluation through containerized submissions.

The shared task consists of two sub-tasks. *Task A: Automatic Report Evaluation* asks participants to rank system-generated reports from the 2025 TREC RAGTIME submissions against human annotations, accepting both fully automatic and semi-automatic submissions (the latter using organizer-provided essential facts). *Task B: Multilingual Report Generation* asks participants to generate long-form, citation-grounded reports from a corpus of four million English, Chinese, Russian, and Arabic documents sampled from Common Crawl News (2021–2024), where each generated sentence must be attributable to a cited source and the report must match the language of the request. Submissions to Task B were scored automatically with the ARGUE framework and its automatic implementation, Auto-ARGUE[4], which combines nugget coverage (whether information units expected of a good report are present) with sentence support (whether each generated sentence is faithful to its cited evidence).

We are delighted to host a keynote talk by Professor Chris Callison-Burch (University of Pennsylvania) on rubric-based evaluation of LLM-generated text, which speaks to the central theme of the RAG4Reports Workshop. Prof Callison-Burch has more than 200 publications, which have been cited

[1] https://neuclir.github.io/
[2] https://trec-ragtime.github.io/
[3] https://www.tira.io/
[4] https://github.com/hltcoe/auto-argue

over 36,000 times. He is a Sloan Research Fellow, and he has received faculty research awards from Google, Microsoft, Amazon, Facebook, and Roblox, in addition to funding from DARPA, IARPA, and the NSF.

This workshop would not have been possible without the contributions of many people. We thank the program committee and external reviewers for their careful and timely feedback; the shared task participants for engaging with the benchmark and pushing its limits; and the TIRA team, particularly Maik Fröbe, for hosting and supporting our evaluation infrastructure. We thank the ACL 2026 workshop chairs and publication chairs for their guidance throughout the organization process. Finally, we thank all of the authors and shared task participants who submitted their work.

We hope you enjoy the workshop and the discussions it sparks.

The RAG4Reports 2026 Workshop Organizers
Eugene Yang, Dawn Lawrie, Sean MacAvaney, James Mayfield,
Luca Soldaini, and Andrew Yates

Organizing Committee

Workshop Organizers

Eugene Yang, Human Language Technology Center of Excellence, Johns Hopkins University
Dawn Lawrie, Human Language Technology Center of Excellence, Johns Hopkins University
Sean MacAvaney, University of Glasgow
James Mayfield, Human Language Technology Center of Excellence, Johns Hopkins University, and Johns Hopkins University Applied Physics Laboratory
Luca Soldaini, Microsoft AI
Andrew Yates, Human Language Technology Center of Excellence, Johns Hopkins University

Program Committee

Program Chairs

Eugene Yang, Human Language Technology Center of Excellence, Johns Hopkins University
Dawn Lawrie, Human Language Technology Center of Excellence, Johns Hopkins University
Sean MacAvaney, University of Glasgow
James Mayfield, Human Language Technology Center of Excellence, Johns Hopkins University, and Johns Hopkins University Applied Physics Laboratory
Luca Soldaini, Microsoft AI
Andrew Yates, Human Language Technology Center of Excellence, Johns Hopkins University

Reviewers

Andreas Chari, University of Glasgow
Susmita Das, University of Glasgow
Maxime Dassen, University of Amsterdam
Pooja Jhunjhunwala, Google
Kangheng Liang, University of Glasgow
Emmanouil Georgios Lionis, University of Glasgow
Andrew Parry, University of Glasgow
Saron Samuel, Johns Hopkins University
Siddharth AK Singh, University of Amsterdam

Keynote Talk

Autorubric: A Unified Framework for Rubric-Based LLM Evaluation

Chris Callison-Burch
University of Pennsylvania

Abstract: LLM-as-a-judge has become the default for evaluating open-ended generation, but the approach is riddled with silent failure modes, including position bias, verbosity bias, criterion conflation, sycophancy, and run-to-run inconsistency, that corrupt judgments without any visible signal. Mitigations exist, scattered across the LM-as-judge literature and decades of work in psychometrics and educational measurement, but every research group ends up paying a "Reinvention Tax," reimplementing option shuffling, ensemble voting, calibration, and reliability metrics from scratch.
I will present Autorubric, an open-source framework that consolidates these best practices into a single library with opinionated defaults: analytic per-criterion decomposition, mixed criterion types, ensemble judging, length penalties, and a full suite of psychometric reliability metrics. Beyond measurement, Autorubric's mandatory per-criterion explanations function as "textual gradients" for two downstream applications: rubric-guided prompt induction and RL with rubric rewards. Autorubric is available at https://autorubric.org.

Bio: Chris Callison-Burch is the Raj and Neera Singh Professor of Artificial Intelligence at the University of Pennsylvania, where he directs the online Master's in AI and teaches Penn Engineering's flagship AI course to more than 500 students each fall. In 2026 he received the Lindback Award for Distinguished Teaching, Penn's highest teaching honor. He chairs the advisory board for the Human Language Technology Center of Excellence at Johns Hopkins University. He testified before Congress in 2023 on generative AI and copyright law, and in 2026 participated in the Isaac Asimov Memorial Debate at the American Museum of Natural History, moderated by Neil deGrasse Tyson. He has authored more than 200 publications with over 36,000 citations, and is a Sloan Research Fellow with research support from DARPA, IARPA, NSF, and industry partners including Google, Microsoft, and Amazon.

Table of Contents

Program

Saturday, July 4, 2026

09:00 - 09:15 *Opening Remarks and Shared Task Introduction*

09:15 - 10:00 *Keynote: Chris Callison-Burch (University of Pennsylvania)*

10:00 - 10:30 *Research Paper Orals*

Decompose, Retrieve, Cite: A RAG Pipeline for Structured Report Generation from Technical Documentation
Himanshu Dhurve, Sreedath Panat, Rajat Dandekar and Raj Dandekar

REFSafE: A RAG-Enabled Framework for Predictive Risk Analysis and Automated Safety Report Generation in Mission-Critical Environments
Sanjay Das, Ran Elgedawy, Ethan Seefried, Ryan A. Burchfield, Gavin Wiggins, Dana Hewit, Sudarshan Srinivasan, Prasanna Balaprakash, Robert M. Patton, Todd Thomas and Tirthankar Ghosal

10:30 - 11:00 *Coffee Break*

11:00 - 11:30 *Shared Task Orals*

GenAIus at RAG4Reports 2026: Citation-Aware Compression for Multilingual Report Generation
Reyyan Yeniterzi and Suveyda Yeniterzi

UNH @ Rag4Reports: A Broad Exploration of LLM-Judges for RAG
Minna Tran, Ryan McCarthy, Aiden Parsons, Jaren Unzen and Laura Dietz

Crucible @ Rag4Reports: Generating Nuggets for Report Generation and Evaluation
Laura Dietz and Eugene Yang

11:40 - 12:30 *Poster Session*

StructSurvey: Structured Agentic Retrieval for Automated Survey Paper Generation
Paolo Pedinotti and Enrico Santus

DEO: Training-Free Direct Embedding Optimization for Negation-Aware Retrieval
Taegyeong Lee, Jiwon Park, Seunghyun Hwang and JooYoung Jang

Saturday, July 4, 2026 (continued)

EFSG: Evidence-First Structured Generation for Multilingual RAG Report Generation
Shaurya Gupta and Jatin Bedi

Adapting AutoARGUE for Automatic Report Evaluation under Missing Citation Annotations
Divrose Kaur, Jatin Bedi and Jasmeet Singh

JU-NLP-PG at RAG4Reports 2026: Memory-Efficient Multilingual Report Generation with 4-bit Quantized LLMs
Swayam Chatterjee and Dipankar Das

A Tale of Trust and Accuracy: Base vs. Instruct LLMs in RAG Systems

Florin Cuconasu[1], Giovanni Trappolini[1,2,3], Nicola Tonellotto[4], Fabrizio Silvestri[1]
[1]Sapienza University of Rome, [2]Universitas Mercatorum, [3]ISTI-CNR, [4]University of Pisa
{cuconasu, fsilvestri}@diag.uniroma1.it; giovanni.trappolini@uninercatorum.it; nicola.tonellotto@unipi.it

Abstract

Retrieval-Augmented Generation (RAG) represents a significant advancement in artificial intelligence combining a retrieval phase with a generative phase, with the latter typically being powered by Large Language Models (LLMs). Common wisdom and practices in RAG involve using "instructed" LLMs, which are fine-tuned with supervised training to enhance their ability to follow instructions and are aligned with human preferences using state-of-the-art techniques. However, contrary to this popular belief, our study demonstrates that base models outperform their instructed counterparts in RAG tasks by 20% on average under our experimental settings. This finding challenges the prevailing assumptions about the superiority of instructed LLMs in RAG applications. Further investigations reveal a more complex situation, questioning fundamental aspects of RAG and suggesting the need for broader discussions on the topic; or, as Fromm would have it, "Seldom is a glance at the statistics enough to understand the meaning of the figures".[1] [2]

1 Introduction

Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) enhances Large Language Models (LLMs) by retrieving relevant information from vast corpora before generating responses. This approach improves accuracy and addresses limitations of standalone generative models like hallucinations (Huang et al., 2025) and context drift (Wang et al., 2022). As the demand for more sophisticated and context-aware AI systems grows, the ability to generate accurate and contextually relevant information becomes crucial (Gao et al., 2024). RAG achieves this by leveraging the vast amount of information available, ensuring that the outputs of the models are informed by up-to-date and contextually appropriate data. This has profound implications for various applications, including conversational AI, information retrieval, and automated content generation (Shuster et al., 2021; Wang et al., 2024).

LLMs, the key component in RAG systems, are initially pre-trained on the next token prediction task (Radford et al., 2018), where they acquire a broad understanding of language, syntax, semantics, and general knowledge. We call this the "base" version. These models typically undergo two refinement stages: supervised instruction fine-tuning (SFT) to improve instruction-following abilities (Taori et al., 2023); and alignment with human preferences through techniques like RLHF (Ouyang et al., 2024) or similar methods (Rafailov et al., 2024; Hong et al., 2024; Rahman and Xue, 2022). The resulting "instruct" versions are the go-to models for RAG tasks (Liu et al., 2024; LangChain, 2023; DSPy, 2023). Moreover, these instruct models often come with a "template": specific prompt formatting patterns with special tokens that structure the input to mark system and user instructions.

In this paper, we conduct a principled evaluation comparing instruct models and their accompanying templates against their base versions in RAG settings. Surprisingly, our results reveal that base models, without the instruction-specific fine-tuning, outperform instruct models in our experimental setting. This finding challenges the prevailing assumption that instruct models are inherently superior for these tasks. Further investigation shows that the situation is more complex, with various factors contributing to this unexpected effectiveness difference.

In summary, our contributions are: **(a)** We conduct a principled evaluation comparing instruct models against base models in RAG, revealing base models' superior performance; **(b)** Through detailed analysis, we uncover the complexities and

[1]Translated from I cosiddetti Sani.

[2]The code and data are available at github.com/florin-git/Base-vs-Instruct-LLMs-in-RAG-Systems

Proceedings of the First Workshop on Multilingual Report Generation via Retrieval Augmented Generation (RAG4Reports), pages 1–23

features that influence the effectiveness of RAG systems; **(c)** Our findings challenge existing assumptions and stimulate further discussion on RAG's state of the art, helping the development of more effective and reliable systems.

2 Preliminaries

LLM Training LLM training consists of three key steps: pre-training, instruction fine-tuning, and preference alignment.

Pre-training involves unsupervised learning on vast text corpora, where the model learns to predict the next token given a sequence of tokens $w_{1:i-1}$. The probability of generating a sequence y is $p(y) = \prod_{i=1}^{n} p_\theta(w_i|w_{1:i-1})$. This process produces a "base" model with linguistic patterns and contextual understanding.

Supervised fine-tuning (SFT) enhances the model's ability to follow instructions through training on curated datasets of instruction-response pairs (Taori et al., 2023). This transforms general language capabilities into directive-following abilities using traditional supervised learning approaches.

The final step **aligns** LLMs with human preferences, typically through Reinforcement Learning from Human Feedback (RLHF) (Ouyang et al., 2024), which optimizes the following formula:

$$J_r(p_\theta) = \mathbb{E}_{x,y}\left[r(x,y) - \beta \log \frac{p_\theta(y \mid x)}{p_{ref}(y \mid x)}\right]$$

Here, $r(x, y)$ represents human preferences for the input prompt x and response y; β controls regularization between the LLM p_θ and reference model p_{ref}.

RAG Retrieval-Augmented Generation (Lewis et al., 2020) combines information retrieval with generative models. Given a query q, a retriever identifies relevant documents $\{d_1, \ldots, d_k\}$ from corpus D using similarity scoring: $sim(q, d_i) \propto \overrightarrow{q} \cdot \overrightarrow{d_i}$. The LLM then generates a response based on both the query and retrieved documents:

$$p(y) = \prod_{i=1}^{n} p_\theta(w_i \mid q, d_1, \ldots, d_k, w_{1:i-1})$$

3 Experimental Methodology

In this paper, we investigate the effectiveness of base models compared to their instruction-tuned versions in the context of RAG. Our research extends to examine the underlying factors that influence RAG models' performance and the impact of additional training techniques, i.e., SFT and Alignment, on these systems. To address these research objectives, we conduct a series of rigorous experiments, providing a comprehensive analysis of the advantages and limitations of both model versions in RAG tasks.

3.1 Datasets and Models

In our experiments, we utilize three widely used open-domain question-answering datasets that are commonly employed in RAG research (Lewis et al., 2020; Gao et al., 2024). Specifically, we use the open version of Natural Questions (NQ-open) (Kwiatkowski et al., 2019; Lee et al., 2019) and TriviaQA (Joshi et al., 2017) for single-hop QA, and HotpotQA (Yang et al., 2018) for multi-hop QA. These datasets offer a diverse range of question types and complexities, allowing for a comprehensive evaluation of RAG systems. Additional information about the datasets is available in Appendix A.1. For each query, we use Contriever (Izacard et al., 2021) to retrieve the most relevant documents from the English Wikipedia corpus. The retriever's effectiveness is discussed in detail in Appendix A.2.

For the generation phase, we utilize several LLMs in both their base and instruct/chat versions: Llama 2 7B, Llama 2 13B, Llama 3 8B, Llama 3 70B, Mistral 7B, and Falcon 7B. Models are quantized to 4-bit precision due to our limited computational resources. All models utilize greedy decoding to ensure reproducibility of results. The maximum response length is tailored to the requirements of each dataset and task instruction. For more details on dataset-specific configurations, please refer to Appendix A.1.

3.2 Task Instructions

We perform our experiments with two distinct task instructions to evaluate the effectiveness of the models. Task Instruction I (Figure 1) requires the LLM to answer based on provided documents. This instruction also includes a critical component: the models are required to respond with *NO-RES* if the answer is not present in the retrieved documents. This tests the models' *negative rejection* capabilities (Chen et al., 2024), assessing their ability to recognize they lack sufficient information to provide a reliable answer. Task Instruction II extends

Llama 3 70B	Llama 3 70B-Instruct + Template
	<|*start_header_id*|>*system*<|*end_header_id*|>
You are given a question and you MUST respond based on the provided documents. If none of the documents contain the answer, respond with NO-RES.	You are given a question and you MUST respond based on the provided documents. If none of the documents contain the answer, respond with NO-RES.<|*eot_id*|><|*start_header_id*|>*user* <|*end_header_id*|>
Documents: **Document [1]**(Title: West Sand Lake, New York) West Sand Lake, New York West Sand Lake is a hamlet and census-designated place in Rensselaer County, New York, United States. The population was 2,845 people in 2015. The community is located in the northwest corner of the town of Sand Lake... **Document [2]**(Title: Averill Park, New York) Averill Park, New York Averill Park is a census-designated place in Rensselaer County, New York, United States. The population was 1,693 at the 2010 census. The Sand Lake Baptist Church was listed on the National Register of Historic Places in 2004... **Document [3]**(Title: Sand Lake, New York) Work upon the church edifice was begun in the fall of 1869, and the church was incorporated January 4, 1870... **Question:** Sand Lake Baptist Church is located in a hamlet with a population of what at the 2010 census?	
Answer: 1,693	**Answer:**<|*eot_id*|><|*start_header_id*|>*assistant* <|*end_header_id*|> According to Document [1], the Sand Lake Baptist Church is located in West Sand Lake, which had a population of 2,845 people in 2015 (2010 census data not available).

Figure 1: Base vs. Instruct + Template under Task Instruction I on HotpotQA. The figure presents a comparison between the responses generated by two versions of the Llama 3 70B model: the base version and the instruct + template version. Both are tasked with answering the same question based on the provided documents. The base model correctly identifies the answer as "1,693", while the instruct + template version hallucinates. Notably, the instruct version erroneously states that the "2010 census data [is] not available", despite this information being present in Document [2]. *Italic* text denotes the template.

this by requiring models to also provide supporting evidence from the context, which we call a *Proof* (Figure 5 in Appendix).

For both tasks, prompts consist of the instruction, retrieved documents, and query. Documents are ordered by ascending similarity score, positioning high-similarity documents nearest to the query (Liu et al., 2024).

3.3 Instruct Templates

When fine-tuning LLMs to create their instruct versions, specific prompt templates are used during training. These templates are designed to clearly distinguish between model responses, task instructions, and user inputs, often using special tokens. For instance, Llama 3 utilizes <|*begin_of_text*|>, and Mistral uses *[INST]* to denote the beginning of instructions. While these templates are integral to the training process of instruct LLMs, their impact on RAG tasks outside conversational settings remains largely unexplored. Our study addresses this knowledge gap by evaluating instruct models with and without their standard chat templates. We aim to determine if these templates, designed for conversational AI, enhance or potentially hinder effectiveness in information retrieval tasks.

3.4 Evaluation

Accuracy is the main metric adopted to evaluate the models' responses. In particular, it is checked whether one of the ground truth answers of the dataset is contained in the generated response after applying a normalization process. This normalization involves lowercasing and the removal of punctuation and articles to ensure that the answer is not unfairly penalized by minor discrepancies in formatting.

Negative Rejection. As shown in Figure 1, LLMs are tasked to respond with *NO-RES* when documents lack the necessary knowledge to answer the query. This tests their ability to follow instructions and correctly refuse to respond when the information is not present, thereby reducing the occurrence of hallucinations (Zhang et al., 2023). We measure the *negative rejection* using the *rejection rate* (Chen et al., 2024)–the proportion of instances where the model correctly responds with *NO-RES* when answers are absent from the documents. Higher rates indicate a better ability to avoid generating incorrect or misleading answers.

4 Results

In this section, we present the results for three types of models–base, instruct, and instruct + template–evaluated under two task instructions across different datasets, as detailed in the previous section.

4.1 Evaluation on Task Instruction I

In the initial set of experiments, we evaluate the models using Task Instruction I, which is illustrated in Figure 1. Table 1 presents the accuracy scores for each model/version combination across varying numbers of retrieved documents for NQ, TriviaQA, and HotpotQA datasets. Contrary to expectations, we observe that base models consistently outperform their instruct counterparts across all three datasets, with only one partial exception. Llama 2's base model surpasses its instruct version (without template) by an average of 9.23 (+48%), 17.88 (+42%), and 4.02 (+20%) accuracy points on NQ, TriviaQA, and HotpotQA, respectively. Similarly, Falcon's base model shows improvements of 1.94 (+10%), 7.48 (+20%), and 0.89 (+5%) accuracy points on the same datasets.

Llama 3 demonstrates the most substantial gains, particularly on NQ and TriviaQA, with increases of 10.92 (+59%) and 37.5 (+186%) points. On HotpotQA, the improvement is 5.28 (+28%). This dramatic difference is partly attributable to Llama 3's reliance on its template, which we explore further in Section 4.3.

Mistral presents the only "half" exception. While its base model underperforms the instruct version by 2.49 (-8%) accuracy points on NQ, it still outperforms by 5.83 (+10%) and 1.91 (+8%) points on TriviaQA and HotpotQA.

In most cases, including more retrieved documents in the prompt leads to better effectiveness. This improvement can be attributed to the higher likelihood of incorporating relevant information, as evidenced by the increased top-k accuracy of the retriever (see Table 3 in the Appendix).

Larger Models. To further validate our findings and address potential concerns about model size influencing these results, we extended our experiments to include larger models. Table 2 presents results for Llama 2 13B and Llama 3 70B across the same datasets and retrieval settings. Notably, the trends observed with smaller models persist in these larger counterparts. Llama 2 13B's base version consistently outperforms its instruct variants across all datasets and retrieval levels. The accuracy gap is particularly evident in TriviaQA, where the base model achieves up to a 42.48 points improvement over its instruct counterpart with a template when 10 retrieved documents are included in the prompt. Llama 3 70B exhibits a similar behavior; for instance, the base model surpasses the instruct versions by margins of up to 27.39 accuracy points on TriviaQA.

These results from larger models not only corroborate our findings but also suggest that the observed phenomenon–base models outperforming instruct models–is not limited to smaller model sizes.

4.2 Evaluation on Task Instruction II

Intrigued by our initial experiments, we proceeded to examine a new task instruction designed to test the models' ability to ground their answers. In this setting, models are required to provide a *Proof*: a piece of evidence substantiating their answers based on information present in the context documents. Examples of this setup are illustrated in the appendix in Figures 5 and 6 for TriviaQA and NQ, respectively.

The results, presented in the bottom part of Table 1, reveal several insights. First, we observe a general upward shift in accuracy across all models and settings. For instance, Llama 2 base shows an average increase of 3.56 points (+12%) compared to Task Instruction I. This improvement suggests that requiring a proof acts as a form of prompt engineering, potentially encouraging more processing of the context.

Notably, the accuracy gap between base and instruct models persists: the base models continue to outperform their instruct counterparts, with the difference becoming even bigger. For example, Mistral's base model, which slightly underperformed its instruct version on NQ in Task Instruction I, now achieves higher accuracy by 3.41 points (+10%).

4.3 Instruct Models with Template

Our results reveal significant challenges faced by instruct models when using their recommended templates. The performance metrics in Table 1 clearly illustrate this phenomenon. For example, using Task Instruction I on the NQ dataset, Llama 2's templated version barely achieves a 3% accuracy rate. Even under Task Instruction II, it only surpasses 10% accuracy when the context includes more than 8 documents.

This behavior is particularly evident in the NQ dataset, which requires short answers. Despite the

Table 1: Task Instruction I and II Accuracy across document levels (*#Docs*) and LLM configurations: B (*Base*), I (*Instruct*), I + T (*Instruct with Template*). Results show base models generally outperform instruct counterparts by a considerable margin, with Mistral on Task Instruction I being a partial exception. Values *not* marked with an asterisk (*) indicate statistically significant differences between base and instruct models (Wilcoxon test, p-value < 0.01).

Task Instruction I

Dataset	#Docs	Llama 2 7B			Llama 3 8B			Mistral 7B			Falcon 7B	
		B	I	I + T	B	I	I + T	B	I	I + T	B	I
NQ	1	**23.88**	16.06	3.36	**27.03**	8.52	14.40	**24.26**	20.04	18.17	**17.13**	15.68
	2	**24.71**	18.48	1.21	**30.22**	10.25	17.34	**25.30***	24.99	23.54	**18.97**	17.72
	3	**27.83**	18.62	0.69	**30.53**	15.40	19.04	25.72	**26.69***	19.14	**21.15**	17.96
	4	**29.53**	18.59	0.48	**31.08**	15.85	15.30	26.17	**30.56**	27.90	**21.08**	19.21
	5	**30.22**	19.21	0.45	**29.08**	22.57	18.10	27.66	**31.67**	27.45	**21.95**	20.08
	8	**31.01**	21.98	0.73	**29.49***	28.80	20.35	26.65	**33.33**	27.07	**22.64**	20.04
	10	**31.46**	21.08	1.52	**28.70***	28.25	19.35	28.87	**34.82**	26.79	**22.33**	20.98
TriviaQA	1	**55.85**	32.59	23.35	**44.64**	4.13	16.45	**58.67**	48.85	45.88	**41.64**	33.19
	2	**57.15**	34.63	21.63	**53.48**	2.62	27.94	**59.15**	52.31	50.22	**43.11**	36.52
	3	**59.28**	41.09	21.13	**56.78**	3.47	31.32	**58.87**	54.72	52.42	**43.69**	36.46
	4	**60.40**	42.95	17.70	**59.59**	4.44	30.80	**60.49**	55.90	53.73	**44.22**	37.71
	5	**61.24**	46.14	18.05	**58.97**	15.73	34.18	**62.02**	56.97	54.37	**45.60**	38.42
	8	**62.93**	49.19	22.30	**64.93**	51.90	44.04	**64.16**	59.06	57.56	**46.35**	38.83
	10	**63.89**	48.96	25.68	**65.05**	58.61	50.12	**65.92**	60.60	58.31	**48.24**	39.33
HotpotQA	1	**21.89**	16.16	15.75	16.12	15.36	**18.36**	**23.73**	20.64	16.27	**16.43**	14.88
	2	**23.04**	17.96	13.39	**21.29**	15.93	14.77	**24.84**	22.61	16.43	**17.27**	15.66
	3	**23.52**	19.52	14.66	**23.18**	15.77	14.32	**24.39**	22.71	16.57	**17.00***	16.39
	4	**24.71**	20.57	16.27	**25.45**	16.52	16.66	**25.45**	23.89	17.48	**17.43**	16.27
	5	**24.45**	20.43	16.77	**26.30**	18.89	17.57	**26.00**	24.43	19.36	**18.05***	17.07
	8	**25.20**	22.30	17.82	**28.66**	25.07	20.05	**27.41**	25.62	21.02	**18.50***	18.06
	10	**26.09**	23.79	18.91	**28.16**	24.66	19.11	**27.98**	26.48	20.79	18.81	**18.90***

Task Instruction II

Dataset	#Docs	Llama 2 7B			Llama 3 8B			Mistral 7B			Falcon 7B	
		B	I	I + T	B	I	I + T	B	I	I + T	B	I
NQ	1	**24.82**	18.41	1.59	**29.39**	18.83	11.70	**24.82**	21.53	16.86	**17.58**	16.10
	2	**28.70**	24.96	2.42	**31.53**	23.57	16.44	**30.74**	27.66	18.10	**20.04**	18.31
	3	**31.71**	28.76	3.88	**34.72**	25.65	6.40	**33.75**	29.91	20.87	**22.43**	18.93
	4	**32.85**	30.22	5.75	**37.07**	27.97	6.61	**35.17**	32.85	22.43	**23.75**	19.87
	5	**34.09**	32.16	8.45	**36.59**	30.84	11.60	**36.83**	33.58	24.09	**23.85**	20.91
	8	**35.62**	33.16	12.81	**39.15**	34.13	5.64	**40.15**	36.45	24.44	**26.12**	21.61
	10	**35.79**	32.05	12.91	**40.22**	37.97	0.69	**40.15**	35.76	26.96	**26.72**	21.30
TriviaQA	1	**54.94**	41.86	4.22	**61.57**	42.67	30.52	**57.98**	48.51	33.73	**40.32**	33.54
	2	**56.94**	47.18	6.55	**63.32**	44.44	36.89	**60.49**	52.60	33.21	**42.76**	35.69
	3	**58.88**	49.53	8.75	**64.56**	47.11	31.11	**62.18**	54.79	35.47	**44.92**	37.50
	4	**60.07**	51.19	10.15	**65.93**	48.10	33.17	**63.70**	56.05	36.57	**45.45**	38.09
	5	**60.87**	52.59	14.90	**66.52**	48.71	38.08	**65.23**	56.91	34.98	**46.47**	39.02
	8	**62.75**	55.73	20.69	**67.67**	52.59	33.69	**67.04**	58.80	42.39	**47.23**	41.08
	10	**63.73**	56.00	27.48	**68.04**	56.48	23.42	**67.44**	59.54	48.45	**48.57**	42.95

task instruction for NQ specifying brevity, templated models often generate verbose outputs. This tendency may be linked to their fine-tuning and alignment for conversational purposes, where verbosity can be advantageous to assist users. These observations suggest that templates, designed to enhance conversational abilities, may not be optimally suited for all RAG tasks, especially those requiring concise responses.

An additional notable aspect is observed with Llama 3's instruct on Task Instruction I. With fewer than 4 retrieved documents, the non-templated ver-

Table 2: Task Instruction I Accuracy for larger LLMs across different retrieved document levels (*#Docs*) and configurations: B (*Base*), I (*Instruct*), I + T (*Instruct with Template*). Results show base models generally outperform their instruct counterparts, reinforcing findings from smaller models. Values *not* marked with an asterisk (*) denote statistically significant differences between base and instruct models (Wilcoxon test, p-value < 0.01).

Dataset	# Docs	Llama 2 13B			Llama 3 70B		
		B	I	I + T	B	I	I + T
NQ	1	**27.83**	21.74	0.21	**29.46**	19.04	19.00
	2	**29.08**	26.51	0.35	**33.16**	24.78	24.40
	3	**30.70**	29.08	0.35	**35.41**	29.15	28.59
	4	**32.40**	30.63	0.45	**37.35**	32.16	31.46
	5	**34.13**	29.66	0.66	**38.53**	33.82	32.68
	8	**35.13**	30.60	0.69	**40.57**	37.24	35.69
	10	**35.48**	33.44	0.97	**41.05**	39.60	37.24
TriviaQA	1	**61.45**	46.36	12.03	**64.64**	37.25	26.22
	2	**62.51**	51.43	23.03	**63.86**	42.71	32.39
	3	**64.36**	53.95	22.42	**63.48**	47.14	31.01
	4	**64.86**	54.95	24.22	**65.03**	50.51	32.94
	5	**66.30**	56.41	21.87	**66.44**	53.77	37.71
	8	**68.01**	59.91	21.27	**69.23**	60.24	46.50
	10	**68.61**	62.74	26.13	**69.93**	62.90	52.68
HotpotQA	1	**23.52**	20.73	15.25	**21.89**	14.11	12.04
	2	23.25	**23.95***	12.95	**22.84**	15.05	13.04
	3	25.15	**25.38***	13.51	**22.89**	15.84	14.11
	4	**26.57***	26.43	14.54	**25.21**	17.36	14.55
	5	**27.05***	26.93	14.29	**25.96**	19.07	15.52
	8	**28.48***	28.20	17.25	**28.88**	24.25	18.48
	10	**29.43***	28.19	18.93	**29.82**	26.93	20.29

sion struggles to interpret the prompt correctly, often producing random text. However, as the number of documents increases, its accuracy improves dramatically, eventually outperforming its templated counterpart. This observation indicates that larger input lengths might play a role in overriding learned behaviors, allowing the model to better adapt to the task at hand.

5 Is Accuracy Sufficient?

Section 4 clearly indicates that base models outperform instruct models on RAG. But is that really the case? *Are base models truly better than the instruct counterpart on RAG-like prompts?* To answer this question, in this section, we go more in-depth in analyzing and comparing their behavior. First, we test the ability of these models to adhere to the task instructions. In particular, we examine whether they appropriately respond with *NO-RES* when no relevant answer is present in the retrieved documents, hence analyzing their negative rejection capabilities (Chen et al., 2024).

5.1 Negative Rejection

Figure 2 illustrates the rejection rates for various models and their configurations–base, instruct, and instruct + template–on the TriviaQA dataset. Analogous analyses can be performed on the rates for NQ (Figure 9) and HotpotQA (Figure 10).

Across all models, we observe a general failure to consistently respond with *NO-RES* when the answer is absent from the retrieved documents. This is especially true for base models where their rejection rates are often close to 0.

Among the instruct models, Llama 2 and Mistral exhibit similar trends. For both, the highest rejection rates are observed when there is only one document in the context. In this scenario, Llama 2 responds with *NO-RES* only 30.23% of the time, while Mistral does so only 20.12% of the time when using the template. As the number of documents in the context increases, both models show a decreasing tendency to answer with *NO-RES*. This suggests that a higher number of documents might introduce more distracting information (Shi et al., 2023), leading the LLM to respond but erroneously, and that a larger input length might override task instructions.

Llama 3 exhibits a distinct trend, where its instruct versions appear more consistent in their rejection rates, maintaining a mean rate of 34.0 with the template, and 35.57 without. However, when the model is not using the template, the rejection rates for Llama 3 decline with an increase in document count, similar to Llama 2 and Mistral. Indeed, it shows a significant reduction of 14.86 (-50%) passing from 29.44 to 14.58 when the number of retrieved documents increases from 5 to 8.

Falcon models show the least tendency to respond with *NO-RES*, where rejection rates are consistently low or even zero in some configurations. This behavior indicates a propensity to generate answers even when the information is not present, potentially leading to higher rates of hallucination.

5.2 Recall From Parametric Memory

Next, we consider cases where the correct answer is not present in the provided documents, yet the model still responds accurately. As illustrated in the left part of Figure 3, base models frequently manage to provide the correct answer even when it is not in the retrieved documents, suggesting that they "know" the answer from prior training. We call this "recall from parametric memory" (by parametric memory, we mean knowledge learned during training and stored in the parameters of the model, as opposed to non-parametric memory provided in the context through retrieved documents).

Recall from parametric memory is not inherently

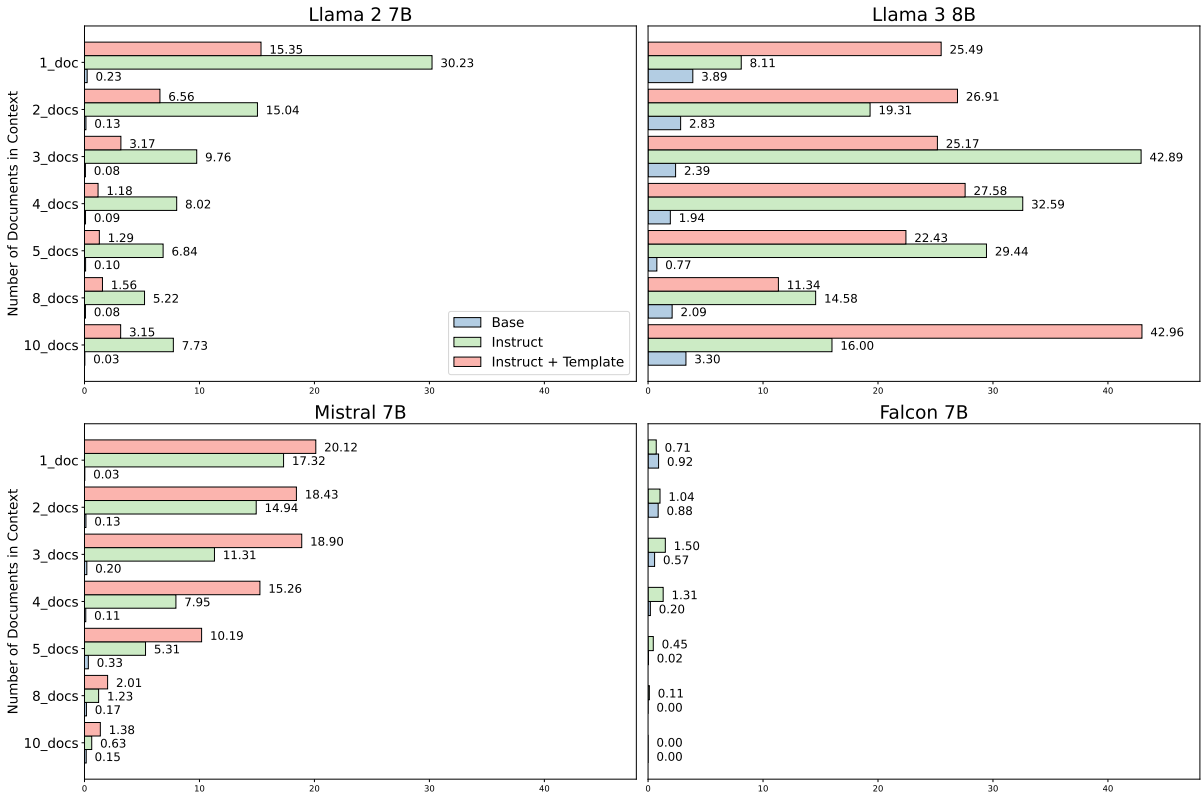

Figure 2: Reported is the **rejection rate** on TriviaQA, defined as the number of times the model responds with *NO-RES* when the correct answer is not in the context, divided by the number of times the answer is indeed missing. Instruct models are much more effective at detecting such cases and following the instructions provided.

Figure 3: **Recall from parametric memory rate** for Llama 2 7B on TriviaQA, defined as the proportion of correct answers when information is absent from retrieved documents. *Left*: Task Instruction I (Figure 1); *Right*: No Rejection setting without *NO-RES* instruction (Figure 7). In the latter case, parametric memory recall increases for both base and instruct models.

problematic. A user might choose to both fine-tune on proprietary data and use RAG to achieve the highest possible accuracy. However, the specific instructions for this study emphasize that models should opt not to answer when the correct response is not evident in the documents. Not following

this guideline raises important questions about the models' reliance on internal knowledge versus contextual information, particularly in settings where accurate rejection of unanswerable questions is crucial.

5.3 Evaluation with No Rejection

To investigate whether base models perform better simply due to non-adherence to instructions, we conduct further experiments by removing the requirement to respond with *NO-RES* when the answer is not present in the provided documents.

As shown in the right part of Figure 3, the removal of this requirement leads to notable changes in model behavior. Both base and instruct models demonstrate a higher tendency to rely on their parametric memory when the rejection constraint is removed, evidenced by the increased rates across all document counts for both model versions.

Interestingly, the instruct version with template of Llama 2 7B shows a more dramatic increase in recall from parametric memory when the rejection constraint is removed. For instance, with 4 documents, the rate nearly triples from 3.88 to 11.27. This trend is also consistently observed across other models (Llama 3, Mistral, and Falcon) and datasets NQ (Figure 11), TriviaQA (Figure 12), and HotpotQA (Figure 13). The enhanced ability to recall information translates into improved accuracy scores for instruct models, as shown in Table 5 (Appendix). However, base models still generally outperform their instruct counterparts.

These findings reveal a tradeoff in the RAG paradigm between model trustworthiness and effectiveness. The processes of supervised fine-tuning and alignment, while enhancing the model's reliability and adherence to desired behaviors, appear to detrimentally impact its capabilities in RAG tasks.

6 Related Works

Recent studies have highlighted challenges and potential improvements in language models' use of non-parametric versus parametric memory in question-answering tasks. Several papers (Krishna et al., 2021; Shi et al., 2024; Carlini et al., 2019; Kandpal et al., 2023; Mallen et al., 2023) demonstrate that LMs often rely on memorized answers, capable of responding correctly even when presented with irrelevant documents. Similarly, other studies (Longpre et al., 2021; Xie et al., 2024) observe that LMs continue to leverage their parametric knowledge despite prompt modifications with contrasting entities. Wu et al. (2024) describes this phenomenon as a balance between the model's inherent knowledge and its adherence to newly retrieved information, underscoring the ongoing challenge of enhancing model responsiveness to dynamic inputs.

On enhancing reliance on provided content, Zhang et al. (2024) introduced a training strategy that emphasizes evidence-based responses, similar to the *Proof* mechanism in our QA tasks. This method has shown potential in improving model effectiveness by grounding responses in factual evidence, even though hallucination issues still remain an open problem (Zuccon et al., 2023; Gao et al., 2023). Cuconasu et al. (2024) found instruct models to be slightly more effective, but theirs was a controlled setting in which the ground truth was always provided. These insights collectively underline the intricate balance between leveraging learned knowledge and external data in improving QA systems, suggesting directions for future research in training strategies and model design.

7 Conclusion

In this paper, we aimed to systematically investigate the differences between LLM's base and instruct versions when used in RAG systems. Our findings reveal an unexpected outcome: base models exhibit superior effectiveness on RAG tasks compared to their instructed and aligned counterparts. However, further analysis reveals a more complex situation, with a tradeoff between the base models' higher accuracy and the instruct versions' higher fidelity. This tradeoff calls for novel evaluation methodologies for RAG pipelines and suggests the necessity for mechanisms that afford users greater control in managing this tradeoff in a more direct and explicit manner.

8 Limitations

Our analysis could benefit from incorporating a broader range of datasets, particularly those that do not rely on Wikipedia. Moreoever, all datasets used in our experiments are in English, and it remains an open question whether our findings extend to other languages. Finally, our study does not cover the latest reasoning-oriented models, such as DeepSeek-R1 (Guo et al., 2025), which may exhibit different behavior in RAG settings due to their extended reasoning capabilities.

Acknowledgments

This work was carried out while Florin Cuconasu was enrolled in the Italian National Doctorate on Artificial Intelligence run by the Sapienza University of Rome. This project was also supported by PNRR MUR project PE0000013-FAIR and partially by the FoReLab and CrossLab projects (Departments of Excellence), and the NVIDIA Academic Grants Program 2026.

References

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. 2023. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901, Singapore. Association for Computational Linguistics.

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, Daniele Mazzotta, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. The falcon series of open language models. *Preprint*, arXiv:2311.16867.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*.

Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. 2019. The secret sharer: evaluating and testing unintended memorization in neural networks. In *Proceedings of the 28th USENIX Conference on Security Symposium*, SEC'19, page 267–284, USA. USENIX Association.

Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. 2024. Benchmarking large language models in retrieval-augmented generation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17754–17762.

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 *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 719–729, New York, NY, USA. Association for Computing Machinery.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library. *Preprint*, arXiv:2401.08281.

DSPy. 2023. Dspy: Dynamic structured programming for python.

Abhimanyu Dubey and Abhinav Jauhri et al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Tianyu Gao, Howard Yen, Jiatong Yu, and Danqi Chen. 2023. Enabling large language models to generate text with citations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6465–6488, Singapore. Association for Computational Linguistics.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-augmented generation for large language models: A survey. *Preprint*, arXiv:2312.10997.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

Jiwoo Hong, Noah Lee, and James Thorne. 2024. ORPO: Monolithic preference optimization without reference model. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11170–11189, Miami, Florida, USA. Association for Computational Linguistics.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Trans. Inf. Syst.*, 43(2).

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2023. Large language models struggle to learn long-tail knowledge. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

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

Kalpesh Krishna, Aurko Roy, and Mohit Iyyer. 2021. Hurdles to progress in long-form question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4940–4957, Online. Association for Computational Linguistics.

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. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

LangChain. 2023. Langchain: Building applications with llms through composability.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 6086–6096, Florence. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

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*, 12:157–173.

Shayne Longpre, Kartik Perisetla, Anthony Chen, Nikhil Ramesh, Chris DuBois, and Sameer Singh. 2021. Entity-based knowledge conflicts in question answering. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7052–7063, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822, Toronto, Canada. Association for Computational Linguistics.

Sewon Min, Danqi Chen, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2019. A discrete hard EM approach for weakly supervised question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2851–2864, Hong Kong, China. Association for Computational Linguistics.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2024. Training language models to follow instructions with human feedback. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Hamza Alobeidli, Alessandro Cappelli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. 2023. The refinedweb dataset for falcon llm: Outperforming curated corpora with web data only. In *Advances in Neural Information Processing Systems*, volume 36, pages 79155–79172. Curran Associates, Inc.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. 2021. KILT: a benchmark for knowledge intensive language tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2523–2544, Online. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, and 1 others. 2018. Improving language understanding by generative pre-training.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2024. Direct preference optimization: your language model is secretly a reward model. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Md Masudur Rahman and Yexiang Xue. 2022. Robust policy optimization in deep reinforcement learning. *Preprint*, arXiv:2212.07536.

Noam Shazeer. 2019. Fast transformer decoding: One write-head is all you need. *Preprint*, arXiv:1911.02150.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Weijia Shi, Anirudh Ajith, Mengzhou Xia, Yangsibo Huang, Daogao Liu, Terra Blevins, Danqi Chen, and Luke Zettlemoyer. 2024. Detecting pretraining data from large language models. In *The Twelfth International Conference on Learning Representations*.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval augmentation reduces hallucination in conversation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3784–3803, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html*, 3(6):7.

Hugo Touvron, Louis Martin, and Kevin Stone et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Hongru Wang, Wenyu Huang, Yang Deng, Rui Wang, Zezhong Wang, Yufei Wang, Fei Mi, Jeff Z Pan, and Kam-Fai Wong. 2024. Unims-rag: A unified multi-source retrieval-augmented generation for personalized dialogue systems. *arXiv preprint arXiv:2401.13256*.

Xuezhi Wang, Haohan Wang, and Diyi Yang. 2022. Measure and improve robustness in NLP models: A survey. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4569–4586, Seattle, United States. Association for Computational Linguistics.

Kevin Wu, Eric Wu, and James Zou. 2024. Clasheval: Quantifying the tug-of-war between an llm's internal prior and external evidence. In *Advances in Neural Information Processing Systems*, volume 37, pages 33402–33422. Curran Associates, Inc.

Jian Xie, Kai Zhang, Jiangjie Chen, Renze Lou, and Yu Su. 2024. Adaptive chameleon or stubborn sloth: Revealing the behavior of large language models in knowledge conflicts. In *The Twelfth International Conference on Learning Representations*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Tianjun Zhang, Shishir G Patil, Naman Jain, Sheng Shen, Matei Zaharia, Ion Stoica, and Joseph E. Gonzalez. 2024. RAFT: Adapting language model to domain specific RAG. In *First Conference on Language Modeling*.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. 2023. Siren's song in the ai ocean: A survey on hallucination in large language models. *Preprint*, arXiv:2309.01219.

Guido Zuccon, Bevan Koopman, and Razia Shaik. 2023. Chatgpt hallucinates when attributing answers. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*, SIGIR-AP '23, page 46–51, New York, NY, USA. Association for Computing Machinery.

A More Details on Datasets and Models

A.1 Datasets

Our evaluations employ three open-domain question-answering datasets: NQ-open, TriviaQA-unfiltered, and HotpotQA. For NQ-open, we adopt the processing procedure of Cuconasu et al. (2024), resulting in 2,889 test examples. The TriviaQA-unfiltered dataset follows the validation and test split used in previous studies (Min et al., 2019; Asai et al., 2024), with 11,313 test queries for evaluation. For HotpotQA, we utilize the open-domain version from the KILT benchmark (Petroni et al., 2021), consisting of 5,600 test samples.

NQ and TriviaQA datasets use the English Wikipedia dated 20 December 2018 as a knowledge source, while HotpotQA draws from the Wikipedia dump of August 2019. Following the Dense Passage Retrieval (DPR) approach (Karpukhin et al., 2020), we split each Wikipedia article into non-overlapping passages of 100 words.

All datasets feature questions allowing multiple valid answers, ranging from synonymous terms like "New York" and "NY" to entirely distinct correct responses. TriviaQA is notable for its diversity in acceptable responses, including many questions allowing multiple valid answers. This variety significantly reduces the likelihood of correct responses being marked as incorrect due to phrasing variations, contributing to higher accuracy scores on TriviaQA compared to the other datasets, as shown in Table 1.

For the generation phase, we tailor the maximum response length of the LLMs to each dataset's requirements. Under Task Instruction I, we set the response limit to 15 tokens for the NQ dataset, which demands short responses of no more than 5 tokens (as specified in the NQ task instruction shown in Figure 6), and up to 50 tokens for TriviaQA and HotpotQA to accommodate potentially longer answers. For Task Instruction II, which requires a *Proof*, we increase the maximum response length to 200 tokens across all datasets.

A.2 Retriever

The retriever used to select the top-k documents is Contriever (Izacard et al., 2021), which is a BERT-based dense model trained using unsupervised contrastive loss. The embedding of each document and query is obtained by averaging the hidden state of the last layer of the model. For document retrieval from the corpus, we employ a FAISS index (Douze et al., 2024; Johnson et al., 2019) by using an inner product similarity metric (IndexFlatIP) with an exhaustive search.

To assess the availability of an answer within the provided documents, we compute the top-k accuracy of the retriever. This metric evaluates the number of times the ground truth answer appears within the top-k documents retrieved for a query. Scores can be seen in Table 3.

A.3 Generative Models

We utilize publicly available, open-weight LLMs accessible via Hugging Face. All models are quantized to 4-bit using the bitsandbytes library[3] to optimize computational efficiency. We perform all experiments with a single Nvidia RTXA6000 GPU.

Here is a brief description of the main characteristics of each model:

Llama 2. The Llama 2 family (Touvron et al., 2023) is pre-trained on publicly available data and optimized for a range of natural language generation tasks. This series features a context length of 4096 tokens, and the 7B[4] and 13B[5] versions employ multi-query attention (MQA) (Shazeer, 2019) to enhance processing efficiency and response quality.

Llama 3. The Llama 3 series (Dubey and et al., 2024) builds on the architecture and improvements of its predecessors, offering models with 8B[6] and 70B[7] parameters. It employs group-query attention (GQA) (Ainslie et al., 2023) and extends the context length to 8192 tokens, thus facilitating enhanced language generation across a broad range of tasks.

Mistral. Developed as a highly efficient model with 7B parameters[8], Mistral (Jiang et al., 2023) focuses on delivering high performance and accuracy in text generation. It uses GQA and sliding window attention with an 8192-token context length.

[3] https://huggingface.co/docs/bitsandbytes/main/en/index

[4] https://huggingface.co/meta-llama/Llama-2-7b-hf
https://huggingface.co/meta-llama/Llama-2-7b-chat-hf

[5] https://huggingface.co/meta-llama/Llama-2-13b-hf
https://huggingface.co/meta-llama/Llama-2-13b-chat-hf

[6] https://huggingface.co/meta-llama/Meta-Llama-3-8B
https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct

[7] https://huggingface.co/meta-llama/Meta-Llama-3-70B
https://huggingface.co/meta-llama/Meta-Llama-3-70B-Instruct

[8] hhttps://huggingface.co/mistralai/Mistral-7B-v0.1
https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.1

Table 3: Contriever top-k accuracy.

Dataset	# Retrieved Documents						
	1	2	3	4	5	8	10
NQ	25.02	35.69	42.89	47.84	51.33	57.84	60.85
TriviaQA	39.15	50.70	56.45	60.32	63.03	68.35	70.49
HotpotQA	20.95	25.48	28.14	30.48	32.27	36.14	37.91

Table 4: Closed-book Accuracy of the models. The task instruction used in this case for all datasets can be seen in Figure 8.

Dataset	Llama 2 7B			Llama 2 13B			Llama 3 8B			Llama 3 70B			Mistral 7B			Falcon 7B	
	B	I	I + T	B	I	I + T	B	I	I + T	B	I	I + T	B	I	I + T	B	I
NQ	**25.20**	16.51	13.12	**29.91**	20.84	9.55	22.01	27.17	**27.80**	35.24	**40.01**	39.67	**28.11**	18.76	15.96	**15.26**	12.74
TriviaQA	**55.57**	37.33	41.73	**59.90**	41.61	30.94	**57.37**	25.46	54.87	71.46	**75.18**	75.12	**60.68**	47.18	49.30	**41.12**	27.83
HotpotQA	**20.70**	14.61	16.98	**22.96**	13.82	16.73	21.39	11.75	**22.09**	23.27	32.16	**33.45**	**22.30**	20.88	22.05	**16.84**	14.66

Falcon. The Falcon 7B[9] is the smallest model of the Falcon series (Almazrouei et al., 2023) and was trained on the RefinedWeb dataset (Penedo et al., 2023)–a large, filtered, and deduplicated corpus. Similarly to Llama2 7B, it uses MQA, but with a smaller context length of 2048 tokens. Unlike the other models, the instruct version of Falcon 7B was not specifically trained using a fixed template, which is why no separate "instruct + template" variant is listed in any figure or table.

The closed-book accuracy of the models is detailed in Table 4. In this scenario, models are evaluated without any documents in their prompt, necessitating a modification to Task Instruction I. An example of this modified task instruction can be viewed in Figure 8.

B Further Analysis with Task Instruction II

In this section, we examine more in detail whether models can justify their answers with a *Proof*. We specifically investigate whether even the base models can adhere to instructions and provide accurate *Proof* for their responses.

Table 6 presents the "coherence" rate, defined as the percentage of instances where the generated answer, regardless of its correctness, is included in the generated *Proof*. This measure indicates how well the model's responses align with the information provided in the documents. However, it is important to note that coherence does not necessarily reflect answer correctness, as it only assesses alignment with the document evidence.

As observed in Table 6, base models are "coherent" with their answers, often outperforming their instruct counterparts. Mistral notably achieves the highest coherence score, reaching 68% with 10 retrieved documents, while Falcon exhibits the lowest, often failing to provide any *Proof* at all. Even the instruct version of Falcon typically offers only the direct answer without supporting evidence. This behavior aligns with our observations in Section 5.1, further highlighting Falcon's challenges in adhering to given instructions.

While these coherence rates provide valuable insights, they come with important caveats. The presence of an answer in the *Proof* does not guarantee its "coherence" accuracy. The *Proof* might not actually derive from the provided documents, even if it includes the answer. Additionally, even if the answer is present in the *Proof*, it may not be recognized as valid due to the inclusion of additional text in the response. For example, if a model begins its response with "The answer to the question is..." and then reports an answer that is technically part of the *Proof*, this response might still be deemed invalid because the introductory phrase does not originate from the context documents.

[9] https://huggingface.co/tiiuae/falcon-7b
https://huggingface.co/tiiuae/falcon-7b-instruct

Llama 2 7B	Llama 2 7B-Chat + Template
You are given a question and you MUST respond by EXTRACTING the answer from one of the provided documents. If none of the documents contain the answer, respond with NO-RES.	*[INST] «SYS »* You are given a question and you MUST respond by EXTRACTING the answer from one of the provided documents. If none of the documents contain the answer, respond with NO-RES. *«/SYS »*

Documents:
Document [1](Title: Batman Returns) the Penguin. We didn't really officially cast it, but for a short nasty little guy, it's a short list. I ended up writing the character for Danny DeVito. Burgess Meredith (who portrayed the Penguin in the 1960s TV series "Batman") was cast for a little cameo as Tucker Cobblepot...
Document [2](Title: Batman: Mystery of the Batwoman) This is the only time in the DC animated universe that Paul Williams did not voice the Penguin...
Document [3](Title: The Penguin's a Jinx) The Penguin goes to Wayne Manor and returns the actress. He then uses his gas-umbrella to knock out anyone inside the statues...

Question: Who played the part of 'The Penguin' in the TV series 'Batman'?

Answer: Burgess Meredith	**Answer:** *[/INST]* Based on the provided documents, the answer is Danny DeVito

Figure 4: Base vs. Instruct + Template under Task Instruction I on TriviaQA. The figure presents a comparison between the responses generated by two versions of the Llama 2 7B model: the base version and the instruct + template version. Each version is tasked with answering the same question based on the provided documents. The base model correctly identifies the answer as "Burgess Meredith", whereas the instruct + template version incorrectly attributes the answer to "Danny DeVito". *Italic* text denotes the template.

Llama 2 7B	Llama 2 7B-Chat + Template
You are given a question and you MUST respond by EXTRACTING the answer from one of the provided documents. If none of the documents contain the answer, respond with NO-RES. In addition, you must report the portion of the document (Proof) containing the answer. **START example** **Document** [209707](Title: Ancient Egyptian technology) Evidence indicates that Egyptians made use of potter's wheels in the manufacturing of pottery from as early as the 4th Dynasty. Chariots, however, are only believed to have been introduced by the invasion of the Hyksos in the Second Intermediate period; during the New Kingdom era, chariotry became central to Egypt's military. **Question:** when was the potter's wheel first used in egypt **Answer:** 4th Dynasty **Proof:** Evidence indicates that Egyptians made use of potter's wheels in the manufacturing of pottery from as early as the 4th Dynasty. **END example**	*[INST]«SYS »* You are given a question and you MUST respond by EXTRACTING the answer from one of the provided documents. If none of the documents contain the answer, respond with NO-RES. In addition, you must report the portion of the document (Proof) containing the answer. **START example** **Document** [209707](Title: Ancient Egyptian technology) Evidence indicates that Egyptians made use of potter's wheels in the manufacturing of pottery from as early as the 4th Dynasty. Chariots, however, are only believed to have been introduced by the invasion of the Hyksos in the Second Intermediate period; during the New Kingdom era, chariotry became central to Egypt's military. **Question:** when was the potter's wheel first used in egypt **Answer:** 4th Dynasty **Proof:** Evidence indicates that Egyptians made use of potter's wheels in the manufacturing of pottery from as early as the 4th Dynasty. **END example** *«/SYS »*
Documents: **Document [1]**(Title: Batman Returns) the Penguin. We didn't really officially cast it, but for a short nasty little guy, it's a short list. I ended up writing the character for Danny DeVito. Burgess Meredith (who portrayed the Penguin in the 1960s TV series "Batman") was cast for a little cameo as Tucker Cobblepot... **Document [2]**(Title: Batman: Mystery of the Batwoman) This is the only time in the DC animated universe that Paul Williams did not voice the Penguin... **Document [3]**(Title: The Penguin's a Jinx) The Penguin goes to Wayne Manor and returns the actress. He then uses his gas-umbrella to knock out anyone inside the statues... **Question:** Who played the part of 'The Penguin' in the TV series 'Batman'?	
Answer: Burgess Meredith **Proof:** Document [1](Title: Batman Returns) the Penguin. We didn't really officially cast it, but for a short nasty little guy, it's a short list. I ended up writing the character for Danny DeVito. Burgess Meredith (who portrayed the Penguin in the 1960s TV series "Batman") was cast for a little cameo as Tucker Cobblepot...	**Answer:** *[/INST]* Danny DeVito **Proof:** Document [1](Title: Batman Returns) the Penguin. We didn't really officially cast it, but for a short nasty little guy, it's a short list. I ended up writing the character for Danny DeVito.

Figure 5: Base vs. Instruct + Template under Task Instruction II on TriviaQA. This comparison of responses between the base and instruct + template versions of Llama 2 7B illustrates an example where the base model correctly identifies the answer, while the instruct + template version inaccurately attributes the answer to a different actor. Nevertheless, in both cases, the answers are "coherent" with the *Proof* since each evidence contains the generated answer. *Italic* text denotes the template.

Mistral 7B	Mistral 7B-Instruct + Template
You are given a question and you MUST respond by EXTRACTING the answer (max 5 tokens) from one of the provided documents. If none of the documents contain the answer, respond with NO-RES. In addition, you must report the portion of the document (Proof) containing the answer. **START example** **Document** [209707](Title: Ancient Egyptian technology) Evidence indicates that Egyptians made use of potter's wheels in the manufacturing of pottery from as early as the 4th Dynasty. Chariots, however, are only believed to have been introduced by the invasion of the Hyksos in the Second Intermediate period; during the New Kingdom era, chariotry became central to Egypt's military. **Question:** when was the potter's wheel first used in egypt **Answer:** 4th Dynasty **Proof:** Evidence indicates that Egyptians made use of potter's wheels in the manufacturing of pottery from as early as the 4th Dynasty. **END example**	*[INST]* You are given a question and you MUST respond by EXTRACTING the answer (max 5 tokens) from one of the provided documents. If none of the documents contain the answer, respond with NO-RES. In addition, you must report the portion of the document (Proof) containing the answer. **START example** **Document** [209707](Title: Ancient Egyptian technology) Evidence indicates that Egyptians made use of potter's wheels in the manufacturing of pottery from as early as the 4th Dynasty. Chariots, however, are only believed to have been introduced by the invasion of the Hyksos in the Second Intermediate period; during the New Kingdom era, chariotry became central to Egypt's military. **Question:** when was the potter's wheel first used in egypt **Answer:** 4th Dynasty **Proof:** Evidence indicates that Egyptians made use of potter's wheels in the manufacturing of pottery from as early as the 4th Dynasty. **END example**
Documents: **Document [12919694]**(Title: User State Migration Tool) User State Migration Tool The User State Migration Tool (USMT) is a Microsoft command line utility program intended to allow advanced users, comfortable with Scripting languages, to transfer files and settings between PCs. This task is also performed by Windows Easy Transfer, recommended for general users... **Document [12919696]**(Title: User State Migration Tool) Settings Transfer Wizard and Windows Easy Transfer. USMT is designed to perform mass migrations and allows customizations designed for corporate environments... **Document [12919695]**(Title: User State Migration Tool) there have been several attempts to provide access to its useful functionality by creating GUI wrappers for it... **Question:** what tool is available from microsoft as an alternative to the usmt for smaller migrations	
Answer: Windows Easy Transfer **Proof:** User State Migration Tool The User State Migration Tool (USMT) is a Microsoft command line utility program intended to allow advanced users, comfortable with Scripting languages, to transfer files and settings between PCs. This task is also performed by Windows Easy Transfer, recommended for general users...	**Answer:** *[/INST]* NO-RES.

Figure 6: Base vs. Instruct + Template under Task Instruction II on NQ. This comparison of responses between the base and instruct + template versions of Mistral 7B illustrates an example where the base model correctly identifies the answer, while the instruct + template version erroneously opts for a *NO-RES* response, despite the correct answer being present in the documents. *Italic* text denotes the template.

Falcon 7B	Falcon 7B-Instruct
You are given a question and you MUST respond by EXTRACTING the answer (max 5 tokens) from one of the provided documents.	You are given a question and you MUST respond by EXTRACTING the answer (max 5 tokens) from one of the provided documents.
Documents: **Document [1590238]**(Title: 1970s in film) Thanks to a stellar cast, experienced director, and a poignant story, "Network" became one of the largest critical successes of 1976. Another film, Rocky, about a clubhouse boxer (played by Sylvester Stallone) who is granted a world championship title fight won the Best Picture Academy Award that year. The film also became a major commercial success and spawned four sequels through the rest of the 1970s and 1980s... **Document [2832642]**(Title: Ned Beatty) 1975. Beatty received his first Academy Award nomination for Best Supporting Actor category for the acclaimed film "Network" (1976), portraying a television network's bombastic but shrewd chairman of the board who convinces the mad Howard Beale character (portrayed by Peter Finch) that corporation-led global dehumanization is not only inevitable, but is also a good thing... **Question:** who won the oscar for best picture in 1976	
Answer: Rocky	**Answer:** "Network"

Figure 7: Base vs. Instruct under Task Instruction I with No Rejection on NQ. This figure presents responses under a No Rejection setting, where models are not tasked with responding with *NO-RES* if the answer is not contained in the retrieved documents. It compares the base and instruct versions of Falcon 7B. In this instance, the base model accurately identifies "Rocky" as the Oscar winner for Best Picture in 1976, while the instruct version incorrectly cites "Network".

Llama 3 8B	Llama 3 8B-Instruct + Template
You are given a question and you MUST respond with a short answer based on your internal knowledge. If you do not know the answer, please respond with NO-RES.	<|*start_header_id*|>*system*<|*end_header_id*|> You are given a question and you MUST respond with a short answer based on your internal knowledge. If you do not know the answer, please respond with NO-RES.<|*eot_id*|> <|*start_header_id*|>*user*<|*end_header_id*|>
Question: In which US city did the 2004 remake of the film Alfie take place?	
Answer: New York City	**Answer:** <|*eot_id*|><|*start_header_id*|>*assistant* <|*end_header_id*|> London!

Figure 8: Base vs. Instruct + Template under Closed-Book QA on TriviaQA. This figure compares responses from the base and instruct + template versions of Llama 3 8B for a question in a closed-book setting, where no additional documents are provided. The example demonstrates how the base model accurately identifies "New York City" as the setting of the 2004 remake of the film Alfie, whereas the instruct + template version erroneously claims the location as "London". *Italic* text denotes the template.

Table 5: Task Instruction I with No Rejection Accuracy. In this setting, models are not tasked to answer *NO-RES* when the answer is absent from retrieved documents. Values *not* marked with an asterisk (*) denote statistically significant differences between base and instruct models (Wilcoxon test, p-value < 0.05). An example of the task instruction adopted in these experiments can be seen in Figure 7.

Task Instruction I with No Rejection											
Dataset	**#Docs**	**Llama 2 7B**			**Llama 3 8B**			**Mistral 7B**			**Falcon 7B**	
		B	**I**	**I + T**	**B**	**I**	**I + T**	**B**	**I**	**I + T**	**B**	**I**
NQ	1	**26.41**	19.76	4.05	**28.14**	13.36	14.88	**25.48**	21.53	19.56	**16.58***	16.03
	2	**28.11**	25.20	2.53	**30.32**	19.21	13.33	**26.48***	26.31	24.96	**19.45**	18.28
	3	**30.49**	26.00	1.56	**30.46**	23.19	12.81	26.13	**29.42**	26.76	**19.80***	18.90
	4	**31.15**	27.62	1.25	**30.81**	25.75	15.40	26.83	**31.36**	28.73	**21.46**	19.38
	5	**31.57**	27.35	1.11	28.66	**29.53***	20.32	29.39	**32.09**	27.83	**21.08**	19.42
	8	**32.02**	27.76	1.90	29.21	**31.39**	21.98	29.15	**34.06**	28.04	**21.08**	19.38
	10	**31.81**	28.31	3.63	28.94	**31.15**	21.98	29.49	**34.89**	36.55	**21.26**	19.97
TriviaQA	1	**58.09**	47.29	33.46	**49.65**	4.05	26.85	**59.52**	51.87	48.15	**41.66**	33.99
	2	**59.98**	50.61	39.70	**57.45**	3.94	37.59	**60.06**	54.42	51.99	**43.55**	36.08
	3	**61.07**	52.93	41.32	**61.07**	5.23	42.25	**60.90**	56.19	54.48	**43.40**	36.78
	4	**61.85**	54.27	41.34	**63.17**	7.01	44.53	**62.17**	57.07	55.36	**43.92**	36.98
	5	**62.22**	55.06	41.09	**64.72**	18.41	46.36	**64.65**	58.07	56.34	**45.04**	37.89
	8	**63.44**	57.11	39.01	**66.53**	53.94	53.15	**66.45**	59.93	58.86	**45.62**	38.18
	10	**64.26**	57.71	43.11	**66.80**	61.46	56.00	**67.88**	61.25	59.85	**45.54**	38.43
HotpotQA	1	**23.96**	22.30	19.00	**24.93**	23.12	17.16	**26.89**	24.64	24.50	**16.59***	16.20
	2	**25.54**	23.88	19.89	**27.07**	24.48	14.75	**27.82**	25.93	25.12	**16.79***	16.73
	3	**25.79**	23.98	21.66	**27.93**	24.68	14.95	**27.46**	26.16	25.66	**16.82***	16.23
	4	**26.41**	24.89	23.12	**28.41**	25.34	17.18	**27.52***	26.50	25.86	**18.04***	17.48
	5	**26.18***	25.36	24.14	**29.23**	27.09	18.61	**27.93**	26.88	26.00	**17.61***	16.96
	8	26.38	**28.02**	25.45	30.25	**30.34***	21.45	**28.14**	26.80	27.09	17.74	**18.34***
	10	26.48	**28.32**	25.25	29.70	**32.25**	21.79	**28.49**	27.48	27.25	**18.45***	17.75

Table 6: Task Instruction II Answer Coherence on NQ and TriviaQA. Coherence is measured as the percentage of instances where the generated answer is contained within the generated *Proof*.

Task Instruction II — Answer Coherence											
Dataset	**#Docs**	**Llama 2 7B**			**Llama 3 8B**			**Mistral 7B**			**Falcon 7B**	
		B	**I**	**I + T**	**B**	**I**	**I + T**	**B**	**I**	**I + T**	**B**	**I**
NQ	1	36.93	**45.07**	0	36.90	**42.02**	0.52	36.73	**50.12**	15.99	1.87	**3.39**
	2	41.16	**47.35**	0.59	41.36	**51.92**	0.03	44.03	**53.24**	10.9	2.08	**3.12**
	3	41.64	**46.97**	1.14	43.54	**50.71**	0.1	46.56	**54.38**	7.65	3.32	**3.77**
	4	41.43	**46.45**	0.07	**46.49**	43.86	0.14	45.66	**55.49**	5.75	3.01	**4.85**
	5	42.16	**45.17**	0.07	46.80	**48.36**	1.38	51.47	**57.39**	3.53	3.81	**4.26**
	8	**42.99**	40.53	0	**51.78**	48.29	0.28	54.62	**57.74**	4.12	**4.50**	1.73
	10	**41.26**	32.78	0	51.37	**53.10**	0.07	53.31	**56.07**	2.7	**1.15**	1.04
TriviaQA	1	37.22	**43.15**	0.05	**51.66**	35.45	1.10	43.23	**51.08**	16.82	3.02	**4.45**
	2	44.27	**49.61**	0.46	**57.44**	50.55	0.13	52.16	**53.52**	14.87	3.70	**5.94**
	3	44.68	**53.09**	0.75	**59.46**	50.89	0.12	**56.07**	55.43	11.42	3.47	**6.53**
	4	47.19	**53.46**	0.11	**61.30**	48.91	0.49	**58.48**	57.53	7.90	4.52	**6.08**
	5	48.00	**48.92**	0.04	**62.02**	55.02	1.59	**60.82**	58.55	5.70	4.86	**6.28**
	8	**50.35**	40.14	0	**64.12**	55.28	1.42	**66.66**	59.68	2.45	3.17	**3.34**
	10	**47.14**	28.93	0	**63.58**	56.78	0.50	**68.04**	58.58	2.16	1.61	**1.67**

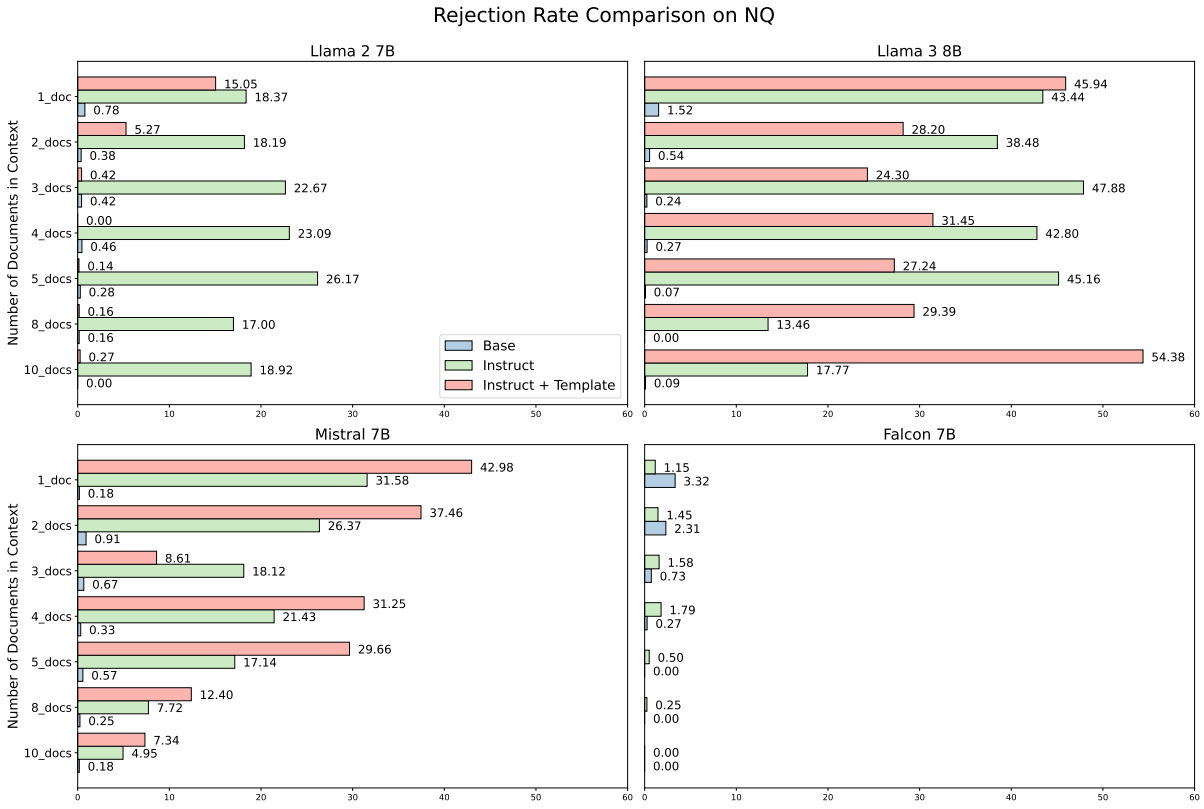

Figure 9: Reported is the rejection rate on NQ, defined as the number of times the model responds with *NO-RES* when the correct answer is not in the context, divided by the number of times the answer is indeed missing. Instruct models are much more effective at detecting such cases and following the instructions provided.

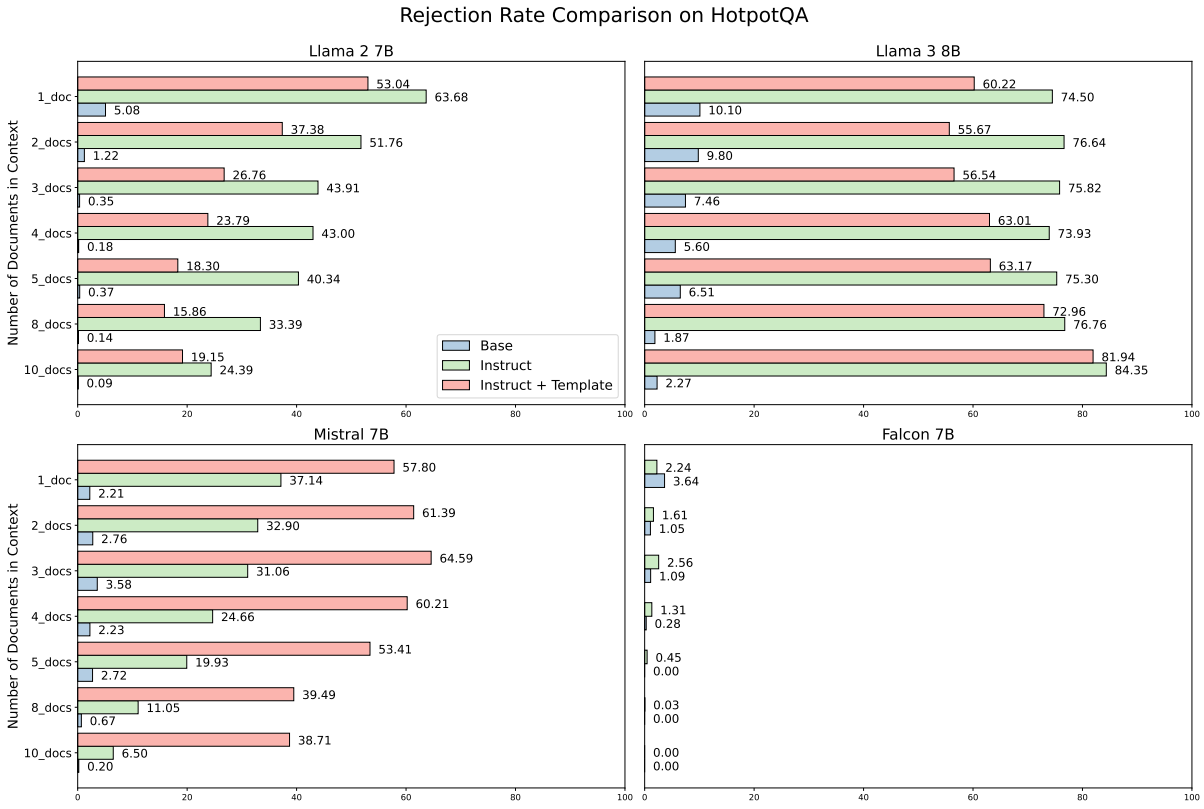

Figure 10: Reported is the rejection rate on HotpotQA, defined as the number of times the model responds with *NO-RES* when the correct answer is not in the context, divided by the number of times the answer is indeed missing. Instruct models are much more effective at detecting such cases and following the instructions provided.

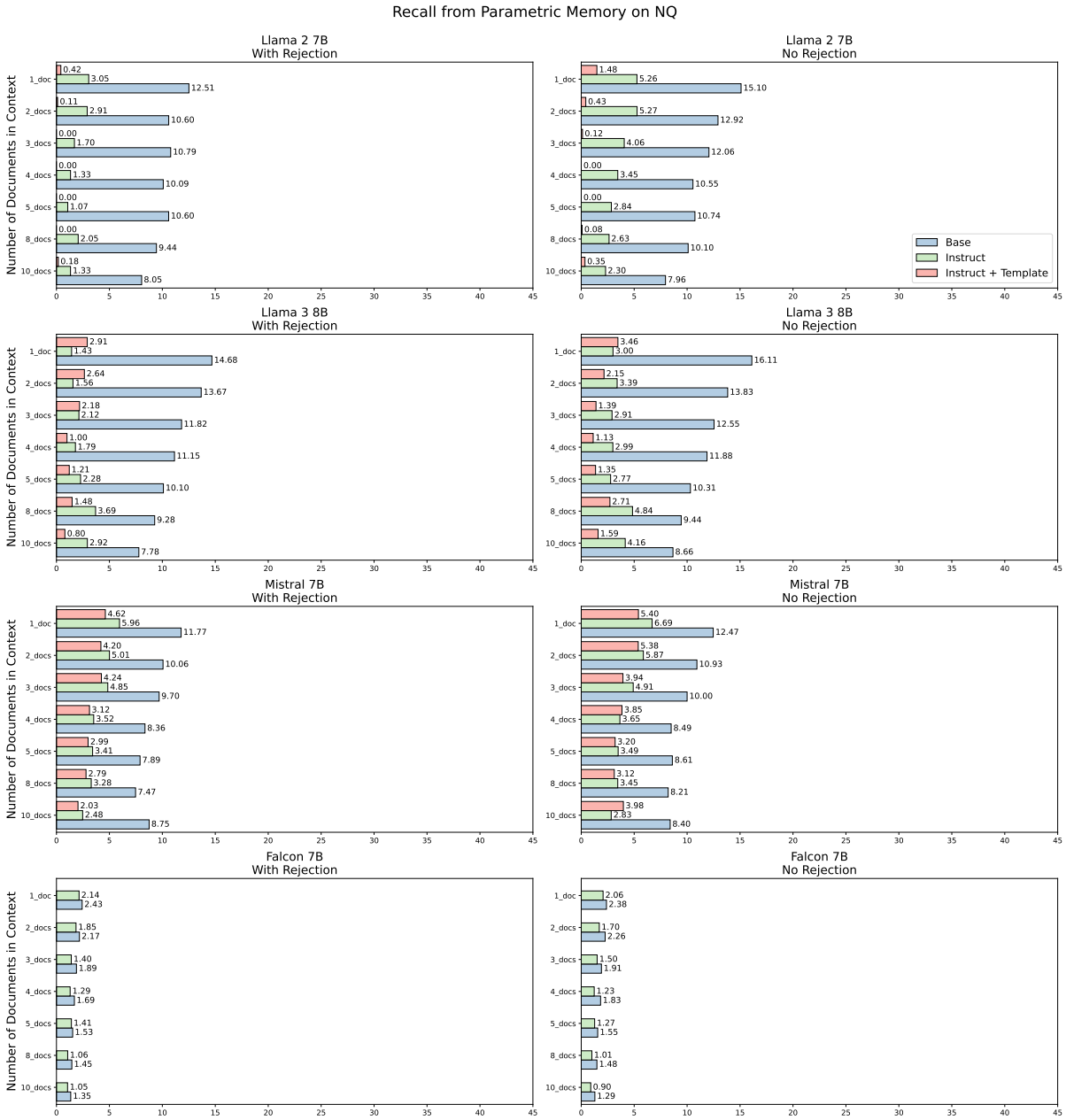

Figure 11: Comparison of Recall from Parametric Memory on NQ between the case where we specify to answer with *NO-RES* when the answer is not contained in the retrieved documents (*left*) and the No Rejection setting (*right*).

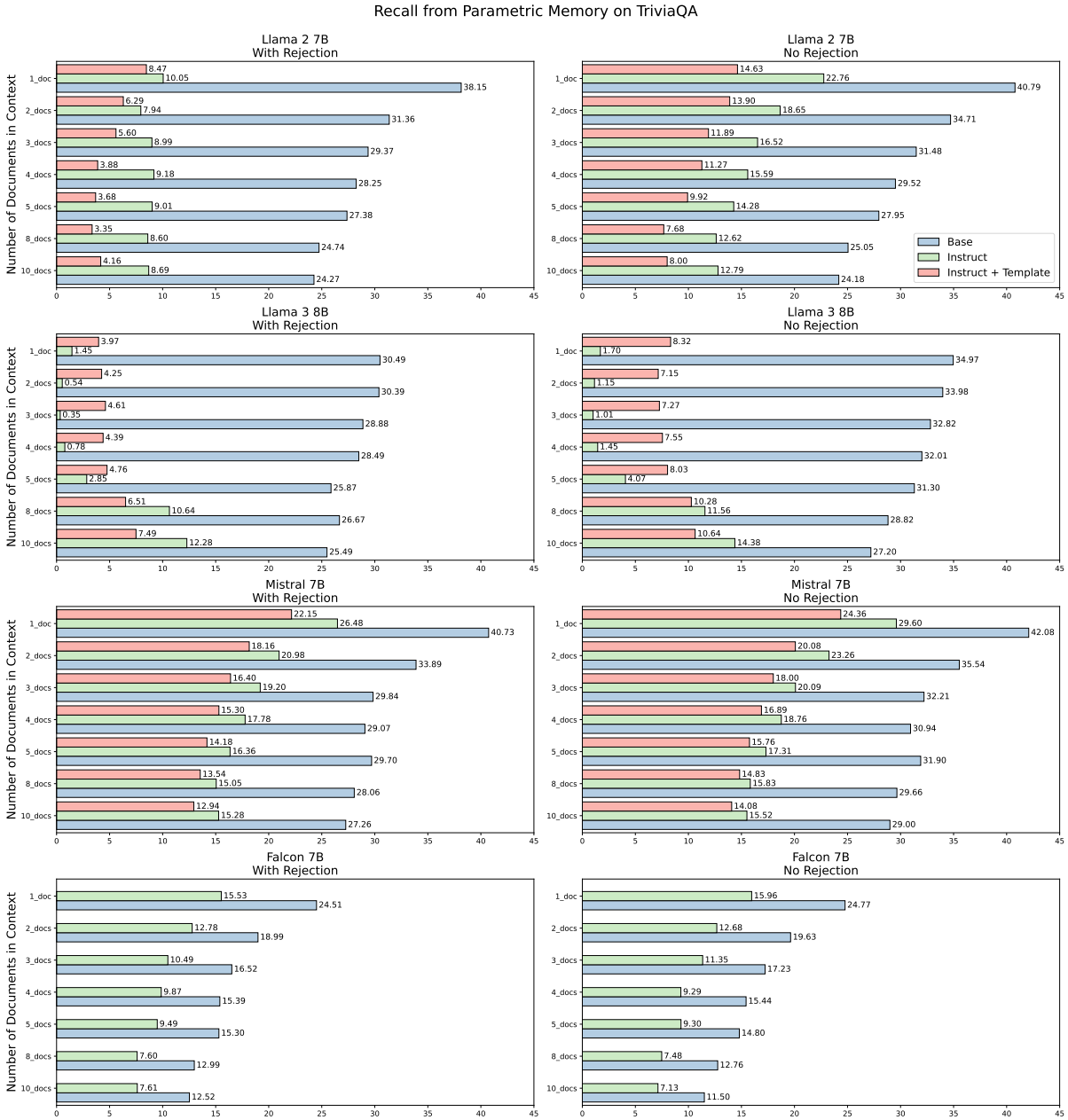

Figure 12: Comparison of Recall from Parametric Memory on TriviaQA between the case where we specify to answer with *NO-RES* when the answer is not contained in the retrieved documents (*left*) and the No Rejection setting (*right*).

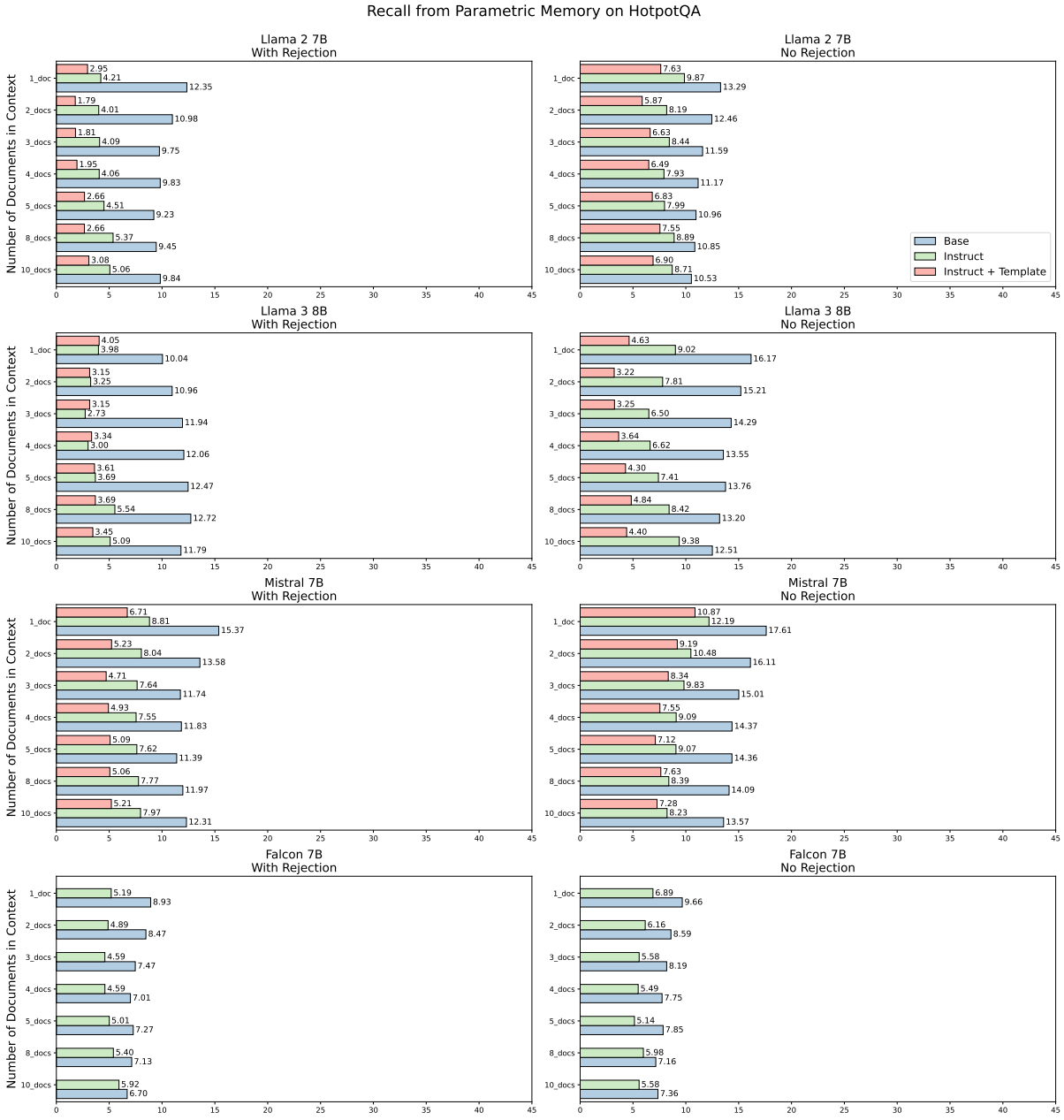

Figure 13: Comparison of Recall from Parametric Memory on HotpotQA between the case where we specify to answer with *NO-RES* when the answer is not contained in the retrieved documents (*left*) and the No Rejection setting (*right*).

Decompose, Retrieve, Cite: A RAG Pipeline for Structured Report Generation from Technical Documentation

Himanshu Dhurve **Sreedath Panat** **Rajat Dandekar** **Raj Dandekar**
Vizuara AI Labs
himanshudhurve96@gmail.com
{sreedath, rajat, raj}@vizuara.com

Abstract

Retrieval-Augmented Generation (RAG) grounds language-model output in external knowledge, yet its application to dense technical documentation remains largely unexplored. Engineering software manuals pose compounding challenges: formulae are corrupted during PDF extraction, heterogeneous content types require different parsing treatment, and queries demand cross-document synthesis across multiple reference volumes. We present an end-to-end RAG system for OpenFOAM, an open-source computational fluid dynamics toolkit, operating in two modes. In *single-query mode*, a formula-preserving parser (Marker), adaptive header-aware chunking, two-stage dense-then-rerank retrieval, and a citation-enforcement prompt produce grounded, source-attributed answers across a 20-question benchmark. In *report mode*, a user prompt is decomposed into sub-questions via LLM planning; each sub-question undergoes independent retrieval and cross-encoder re-ranking, and the deduplicated chunk set is passed to a long-context generation call that produces a structured, multi-section report with inline citations. Evaluated on a 10-prompt golden set with a six-dimension LLM-as-a-judge framework, both pipelines achieve overall scores above 4.6/5.0 with perfect citation correctness (5.0/5.0); the decomposed pipeline demonstrates superior robustness (90% vs 70% judge success rate). Retrieval analysis using page-level ground truth reveals low absolute recall (<14%), identifying retrieval breadth as the primary bottleneck.

1 Introduction

Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) has emerged as an effective paradigm for equipping language models with up-to-date, domain-specific knowledge without retraining. By conditioning generation on retrieved passages from a curated corpus, RAG systems can substantially reduce hallucination and provide verifiable, source-attributed answers. These properties are especially valuable in high-stakes engineering domains, where an incorrect configuration or a misread formula can cause simulation failures or unsafe system designs.

Despite their potential, applying RAG systems to deeply technical documentation introduces challenges that are largely absent from general-domain benchmarks. Specifically, (i) *formula fidelity*: standard PDF extraction pipelines routinely discard or corrupt mathematical notation (Greek symbols, superscripts, inline equations), making the retrieved passages unreliable for formula-intensive queries; (ii) *structural heterogeneity*: engineering manuals interleave prose explanations, structured configuration dictionaries, code examples, and tabular parameter references, each demanding different parsing treatment; and (iii) *cross-document reasoning*: a single user question may require synthesising content from a user guide, a tutorial guide, and a programmer's reference simultaneously, placing high demands on retrieval coverage and diversity.

To address these challenges, we present a RAG system targeting OpenFOAM documentation. OpenFOAM is a widely used open-source CFD toolkit whose documentation spans four volumes: a comprehensive user manual, a condensed reference guide, a step-by-step tutorial guide, and a C++ programmer's guide, totalling approximately 13.3 MB of PDF content. The corpus covers finite volume discretisation theory, mesh topology, solver control dictionaries (`controlDict`, `fvSchemes`, `fvSolution`), mesh generation utilities, and mathematical primitives including the gradient ∇, divergence $\nabla\cdot$, Laplacian ∇^2, and material derivative D/Dt. OpenFOAM documentation thus constitutes a demanding test bed for formula-aware technical RAG.

In summary, our main contributions are:

Proceedings of the First Workshop on Multilingual Report Generation via Retrieval Augmented Generation (RAG4Reports), pages 24–35

- We compare three PDF parsers (Marker, Docling, and PyMuPDF) on mathematical preservation and propose adaptive, header-aware chunking that aligns boundaries with document section hierarchy while enforcing a token budget.

- We build a two-stage retrieval pipeline combining dense vector search (k=15) with cross-encoder re-ranking to top-5, paired with a citation-enforcement prompt requiring page-level attribution for every claim.

- We introduce a report generation pipeline that decomposes prompts into sub-questions, retrieves independently for each with deduplication, and synthesises structured long-form reports.

- We compare report mode against a controlled single-query baseline under identical generation and evaluation conditions.

- We conduct a two-tier evaluation: a 20-question single-query benchmark and a 10-prompt report evaluation with a six-dimension LLM-as-a-judge framework and page-level IR retrieval metrics.

The remainder of this paper is organised as follows. Section 2 reviews related work. Section 3 describes the system architecture, including the single-query pipeline and the report generation extension. Section 4 presents experimental results for both modes. Section 5 discusses findings and limitations, and Section 6 concludes.

2 Related Work

RAG foundations. Retrieval-Augmented Generation, formalised by Lewis et al. (2020), conditions language-model output on retrieved passages and has been shown to reduce hallucination across a range of tasks. The design space has since been surveyed along retrieval strategy (Gao et al., 2023), NLP application (Wu et al., 2024), and knowledge type (Cheng et al., 2025). Despite this breadth, nearly all existing benchmarks target open-domain corpora; Gan et al. (2025) highlight the scarcity of evaluation resources for domain-specific, formula-heavy documents, a gap our work directly addresses.

Long-context alternatives. An alternative to retrieval is to fit the full corpus into the model's context window. Li et al. (2024) compare this long-context (LC) strategy with RAG, finding LC superior when resources suffice but RAG preferable for cost; they propose Self-Route to hybridise the two. Chan et al. (2025) push the idea further with Cache-Augmented Generation (CAG), preloading entire corpora into the KV cache. Our 13.3 MB OpenFOAM corpus exceeds practical context limits, making chunked retrieval the only viable path.

Document parsing and retrieval. Converting technical PDFs into retrieval-ready text is a prerequisite for any RAG system. ML-based parsers such as Nougat (Blecher et al., 2023), Docling (Auer et al., 2024), and Marker apply vision models to recognise equations and section structure, while Faysse et al. (2025) bypass text extraction entirely by embedding page images via ColPali, at the cost of sub-page citation. Tan et al. (2024) show that preserving structural cues (HTML/markdown) during parsing improves downstream answer quality, a finding that motivates our header-aware chunking strategy. On the retrieval side, Chen et al. (2024) demonstrate the advantages of structure-aware chunking, and Li et al. (2025) and Leto et al. (2024) find that chunk size and retrieval strategy interact strongly while k-tuning has modest downstream effects. Nogueira and Cho (2019) show that BERT cross-encoders substantially improve passage re-ranking; we adopt BGE-reranker-base (Xiao et al., 2023) over MiniLM-L12-v2 bi-encoder embeddings (Reimers and Gurevych, 2019). The closest comparable system is RAG-BioQA (Panchumarthi et al., 2025), a two-stage pipeline (k=16 $\rightarrow$ top-4) for biomedical QA; ours uses k=15 $\rightarrow$ top-5 with formula-preservation and cross-encoder re-ranking over engineering documentation.

Citation, reasoning, and evaluation. Ensuring that generated answers remain grounded in retrieved evidence is an active challenge. Xia et al. (2025) propose Self-Reasoning with explicit relevance filtering and evidence extraction, while Singh et al. (2025) survey agentic RAG architectures that employ multi-step planning; our report mode can be viewed as a lightweight decomposition strategy that achieves similar goals through prompt engineering rather than a dedicated agent framework. Evaluating such systems requires moving beyond single-metric scoring: Zheng et al. (2023)

establish LLM-as-a-judge, which we extend to a six-dimension variant; Ju et al. (2025) introduce CRUX for long-form evaluation via question-based decomposition, motivating our checklist-based design; and Randl et al. (2025) find that generators ignore top-ranked documents in 47–67% of queries, motivating our controlled report-vs-baseline comparison under identical generation conditions.

3 System

3.1 Overview

Our pipeline operates in two phases, illustrated in Figure 1. During *offline indexing*, PDF documents are parsed into structured markdown, divided into metadata-enriched chunks, embedded, and stored in a persistent vector database. During *online inference*, a user query is embedded, used to retrieve and re-rank candidate passages, and passed to a generation model that produces a citation-grounded answer. In *report mode*, a prompt is first decomposed into sub-questions; each sub-question undergoes independent retrieval and re-ranking, and the deduplicated chunk set is passed to a long-context generation call that produces a structured report with section headings and inline citations.

3.2 Document Parsing

We apply three PDF parsing strategies to the four-volume OpenFOAM corpus (Table 1), motivated by the observation that no single parser handles all content types equally well. Table 1 summarises the corpus and parser trade-offs.

Marker applies a pipeline of computer vision models to detect page layout, segment equations, and render structured markdown. We configure it with `redo_inline_math=True`, which causes all detected inline mathematical regions to be re-processed through a dedicated math OCR model. This setting is critical for recovering expressions such as the Laplacian definition $\nabla^2 \equiv \partial^2/\partial x_1^2 + \partial^2/\partial x_2^2 + \partial^2/\partial x_3^2$ and the material derivative $D\phi/Dt = \partial\phi/\partial t + \mathbf{U} \cdot \nabla\phi$, which baseline extraction discards entirely. Marker inserts page boundary markers of the form `<span id="page-N">`, enabling precise page-level attribution. We select Marker as our primary parser for the main evaluation.

Docling (Auer et al., 2024) uses document AI with optional formula enrichment, but a compatibility issue prevented it from processing the largest corpus document.

Document	Size	Content
OFUserGuide-v13	8.2 MB	Solvers, mesh, utilities
UserGuide	1.9 MB	Dict. syntax, keywords
TutorialGuide	2.6 MB	Step-by-step tutorials
ProgrammersGuide	588 KB	C++ API, tensor maths
Parser comparison (13.3 MB total)		
Parser	**Formula**	**All docs**
Marker	High	Yes
Docling	High	Partial
PyMuPDF	Low	Yes

Table 1: OpenFOAM documentation corpus and parser trade-offs.

PyMuPDF4LLM performs well on multi-column layouts but offers substantially lower mathematical preservation.

3.3 Chunking Strategy

We design a two-stage adaptive chunking strategy that respects the document's own section hierarchy before enforcing a token budget.

The first stage applies LangChain's `MarkdownHeaderTextSplitter` to divide each document at section boundaries. Because each parser produces different heading conventions, we adapt the header-to-level mapping accordingly: Marker uses `#/##/####` for Section/Subsection/Subsubsection; Docling uses `#/##/###`; and PyMuPDF uses `###/#####`. This stage produces semantically coherent blocks that preserve each document's organisational structure.

In the second stage, each header-split block is further divided by LangChain's `TokenTextSplitter` using the `cl100k_base` encoding, with `max_tokens=800` and an overlap of 100 tokens between consecutive chunks. The overlap prevents critical information from being severed at split boundaries, a known failure mode when definitions span multiple sentences.

Each resulting chunk is annotated with source document, parser identifier, page number, section hierarchy path (section, subsection, subsubsection), word count, and token count. Page numbers are extracted parser-specifically using regular expressions on each parser's page marker format.

3.4 Vector Indexing and Embedding

We embed all chunks using MiniLM-L12-v2[1] (Reimers and Gurevych, 2019), a 384-dimensional bi-encoder that offers a strong

[1]HuggingFace: `all-MiniLM-L12-v2`.

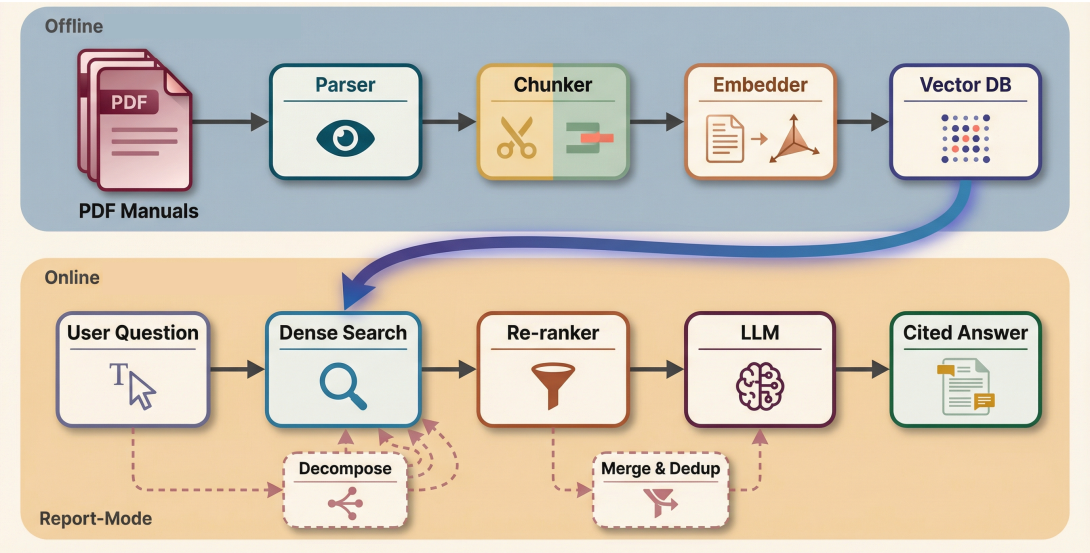

Figure 1: Two-row diagram of the RAG pipeline. Top row depicts the offline indexing path; bottom row depicts the online inference path, with a report-mode sub-track branching from the inference stage.

accuracy-speed trade-off for semantic similarity tasks. Chunks and their metadata are stored in ChromaDB with cosine similarity as the retrieval metric. We maintain a separate index per parser (`db/marker_db/`, `db/docling_db/`, `db/pymupdf_db/`), which allows direct parser ablations without rebuilding unrelated indices. Figure 2 illustrates the full offline indexing pipeline, from parsing through embedding.

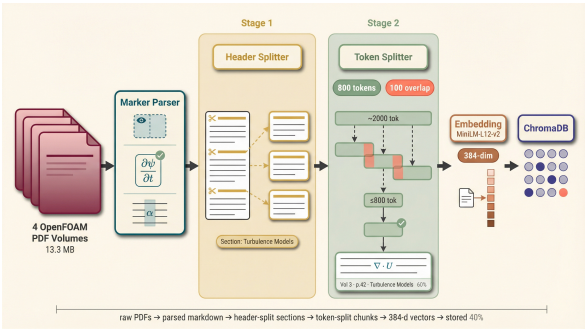

Figure 2: Offline indexing pipeline showing four sequential stages: PDF-to-markdown conversion via parser selection (Marker, Docling, or PyMuPDF), section-level splitting preserving document hierarchy, token-budget chunking with overlap, and embedding into a per-parser ChromaDB index via MiniLM-L12-v2.

3.5 Two-Stage Retrieval

We employ a two-stage retrieval strategy to balance recall and precision. In the first stage, the user query is embedded by the same bi-encoder and a cosine similarity search retrieves the top-k=15 candidate passages from ChromaDB. The broad initial pool prioritises coverage over precision, a design motivated by the observation that relevant content for technical queries is often distributed across sections with limited terminological overlap.

In the second stage, BGE-reranker-base (Xiao et al., 2023) (a cross-encoder trained for passage relevance scoring) receives each of the 15 query-passage pairs and produces a scalar relevance score. The top-5 passages by re-rank score are selected as the final context for generation. Cross-encoders attend jointly to query and passage tokens, enabling them to detect fine-grained lexical matches (e.g., a specific utility name or keyword) that bi-encoder similarity tends to underweight.

The five retained passages are formatted as a numbered context block, with each entry prefixed by its document name, section path, page number, similarity score, and re-rank score. This transparent metadata header allows the generation model to produce accurate citations without performing any additional lookup.

3.6 Answer Generation and Citation Enforcement

We use Google Gemini-2.5-flash model at temperature 0.2 with a maximum of 2048 output tokens for answer generation. Temperature 0.2 allows sufficient paraphrase for fluency while keeping generation tightly coupled to the retrieved evidence.

Our key insight is that hallucination in technical RAG can be structurally suppressed through prompt design. The system prompt casts the model

as an OpenFOAM technical expert and enforces four citation rules: (1) every factual paragraph must end with `[n]` markers corresponding to numbered context entries; (2) if information is absent from the retrieved context, the model must respond "*This information is not available in the provided documentation*" rather than drawing on parametric memory; (3) every answer must conclude with a references section listing the source document, section, and page for each cited entry; and (4) citations must correspond exactly to provided context entries; no fabricated references are permitted. By requiring page-level attribution for every claim, this design makes unsupported statements structurally visible to downstream users, though a controlled ablation without citation rules would be required to isolate the causal contribution of this design choice.

3.7 Report Generation Pipeline

The single-query pipeline described above produces concise, focused answers suited to factoid and how-to questions. For report-level prompts that require broader topical coverage, we extend the architecture with a three-stage report generation pipeline: query decomposition, multi-query retrieval with deduplication, and long-context report synthesis. To isolate the effect of query decomposition, we compare report mode against a single-query baseline under identical generation and evaluation conditions. Table 2 summarises the controlled comparison. The *only* variable is the retrieval strategy; generation prompt, token budget, and judge are held constant. Figure 3 illustrates both the single-query and report-mode inference paths.

Given a report prompt, `decompose_query()` calls the generative model to break it into 3–5 focused sub-questions. The decomposition prompt instructs the model to produce sub-questions specific enough for retrieval and to return a JSON array; a fallback returns the original prompt as a single-element list if parsing fails. For example, the prompt "Write a technical overview of discretization in OpenFOAM" decomposes into sub-questions targeting spatial discretization, temporal discretization, and gradient/divergence scheme configuration.

Each sub-question is then processed through the same two-stage retrieval pipeline (k=15 dense, top-5 after re-ranking). Because related sub-questions often retrieve overlapping passages, we deduplicate by MD5 content hash, retaining the copy

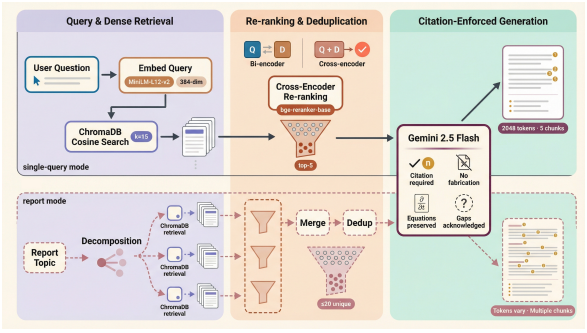

Figure 3: Online inference pipeline showing two operating modes: single-query mode, where the user query is embedded, top-15 candidates retrieved by cosine similarity, and top-5 selected by BGE-reranker-base for citation-enforced generation; and report mode, where the prompt is decomposed into 3–5 sub-questions, each undergoing independent retrieval and re-ranking, followed by content-hash deduplication, chunk merging, and structured report synthesis.

Component	Report Mode	Baseline
Query strategy	3–5 sub-questions	Single query
Dense retrieval	k=15 per sub-Q	k=35
Re-ranking	top-5 per sub-Q	top-20
Deduplication	MD5 content hash	N/A
Chunks to LLM	20 unique	20
Generation	Same prompt, `max_tokens` = 8192	
Judge	6-dim, Gemini 2.5 Pro, temp = 0.0	

Table 2: Report mode vs. baseline: controlled comparison design. Only the retrieval strategy differs; generation and evaluation are identical.

with the highest re-rank score. The merged results are sorted by re-rank score and capped at `max_unique_chunks` = 20, yielding a diverse, non-redundant context window that covers all facets of the original prompt.

The 20 deduplicated chunks are passed to a single generative model call with `max_tokens` = 8192 (four times the single-query budget). The generation prompt requires ##-level section headings, `[n]` inline citations, LaTeX math preservation, and an explicit "not covered in the provided documentation" declaration for topics absent from the retrieved context. This extends the citation-enforcement design from short-form answers to structured long-form output.

4 Experiments

4.1 Single-Query Evaluation

We first evaluate the single-query pipeline on 20 questions processed with the Marker parser, covering dictionary keyword semantics, mesh generation, utility functions, file structure, and solver out-

Metric	Value
Avg. citation coverage rate	30.00%
Avg. unique citations per answer	1.50
Avg. unique source PDFs per query	2.35
Avg. unique pages per query	4.35
Avg. unique sections per query	4.20

Table 3: Citation coverage and chunk diversity across 20 queries.

put interpretation. Each query retrieves k=15 candidates, re-ranks to top-5, and generates a citation-grounded answer via the generative Gemini 2.5 Flash model. Table 3 reports citation and diversity statistics. The system retrieves from an average of 2.35 unique source PDFs per query, demonstrating cross-document synthesis. The citation coverage rate of 30% reflects the model's tendency to anchor on its highest-confidence chunk rather than distribute citations across all relevant passages. Illustratively, the Q16 response for `mapFields` correctly distinguishes three operational modes and cites them to three distinct documents, while Q9 (residual values absent from the corpus) triggers the "not available" fallback rather than fabricating an answer. Please see Appendix A for system output examples.

4.2 Report Evaluation Setup

We evaluate the report generation pipeline on a golden set of 10 prompts (R01–R10) covering discretization, mesh generation, boundary conditions, solution algorithms, parallel execution, post-processing, tensor mathematics, case structure, multiphase solvers, and the build system. Each prompt is paired with 5–7 expected sections, 5–8 must-include facts, and 11–43 manually annotated page-level ground truth references spanning the four-volume corpus.

We employ a six-dimension LLM-as-a-judge framework (Zheng et al., 2023) using Gemini 2.5 Pro model at temperature 0.0. Table 4 defines the dimensions and their weights, which are used to compute a programmatic weighted overall score. Dimensions are scored on a 1–5 integer scale.

Separately, we evaluate retrieval quality using standard IR metrics computed against page-level ground truth: Recall@$\{5, 10, 15, 20\}$, Mean Reciprocal Rank (MRR), and NDCG@$\{10, 20\}$. This separation (generation quality via the judge, retrieval quality via IR metrics) enables diagnosis of whether failures originate in retrieval or generation.

Dimension	Weight	Focus
Groundedness	0.25	Claims supported by retrieved context
Technical Accuracy	0.20	Correctness per documentation
Coverage	0.20	Expected sections present ($\geq$2 sentences)
Citation Correctness	0.15	`[n]` markers match chunks
Factual Recall	0.10	Must-include facts present explicitly
Structure	0.10	Headings, flow, no redundancy

Table 4: Report judge dimensions and weights.

4.3 Report Evaluation Results

Table 5 presents the per-prompt judge scores for report mode. Nine of ten prompts were successfully judged (R09 produced invalid JSON from the judge); the remaining nine achieve a mean overall score of 4.64/5.0. Table 6 presents the corresponding per-prompt scores for the single-query baseline. Table 7 compares report mode against the controlled single-query baseline (Section 3.7). The baseline successfully judged 7 of 10 prompts (R05, R09, R10 failed).

ID	Grnd.	Acc.	Cit.	Cov.	Rec.	Str.	Overall
R01	4	4	5	5	3	4	4.25
R02	5	5	5	5	3	5	4.80
R03	5	5	5	3	3	5	4.40
R04	5	5	5	4	5	5	4.80
R05	5	5	5	5	5	5	5.00
R06	5	5	5	3	3	4	4.30
R07	5	5	5	5	3	5	4.80
R08	5	5	5	5	3	5	4.80
R09			*(judge failed)*				
R10	5	5	5	4	3	5	4.60
Avg	**4.89**	**4.89**	**5.00**	**4.33**	**3.44**	**4.78**	**4.64**

Table 5: Per-prompt judge scores for the decomposed multi-query pipeline (report mode, 1–5 scale). R09 judge returned invalid JSON. Avg over n=9 successful evaluations.

Both pipelines score above 4.6/5.0 overall, with perfect citation correctness (5.0) across all successfully judged prompts. This validates the citation-enforcement prompt design: even in long-form report generation with up to 8192 output tokens, the model reliably attributes claims to retrieved chunks via `[n]` markers.

Despite strong overall scores, factual recall (3.44 report, 3.83 baseline) is the weakest dimension for both modes, indicating that must-include facts are often absent from retrieved chunks rather than ignored during generation. Coverage scores are

ID	Grnd.	Acc.	Cit.	Cov.	Rec.	Str.	Overall
R01	4	4	5	5	4	5	4.45
R02	5	5	5	5	3	5	4.80
R03	5	5	5	4	3	5	4.60
R04	5	5	5	5	5	5	5.00
R05			*(judge failed)*				
R06	5	5	5	5	5	5	5.00
R07	5	5	5	3	—	—	4.50*
R08	5	5	5	5	3	5	4.80
R09			*(judge failed)*				
R10			*(judge failed)*				
Avg	**4.86**	**4.86**	**5.00**	**4.57**	**3.83**	**5.00**	**4.74**

Table 6: Per-prompt judge scores for the single-query baseline pipeline (1–5 scale). R05, R09, R10 judge failures. *R07 partial parse (factual recall and structure unavailable). Avg over n=7 successful evaluations.

Dimension (weight)	**Report** (n=7)	**Baseline** (n=7)	Δ
Groundedness (0.25)	4.86	4.86	0.00
Tech. Accuracy (0.20)	4.86	4.86	0.00
Citation Corr. (0.15)	5.00	5.00	0.00
Coverage (0.20)	4.29	4.57	−0.28
Factual Recall (0.10)	3.29	3.83	−0.54
Structure (0.10)	4.71	5.00	−0.29
Overall (weighted)	4.59	4.74	−0.14

Table 7: Aggregate report-mode vs. single-query baseline under controlled evaluation. Generation settings and judge are identical; only retrieval strategy differs. Both achieve near-ceiling generation quality; report mode shows higher judge robustness. Report (n=7) restricts report-mode averages to the seven prompts both modes successfully judged (R01–R04, R06–R08), enabling a fair like-for-like comparison. Δ (n=7) is the difference on this shared subset.

moderate (4.33 vs. 4.57), suggesting that decomposition does not uniformly improve topical breadth; sub-question overlap may cause merged chunks to converge to the same documentation region.

A notable practical difference is robustness: report mode achieves a 90% judge success rate vs. 70% for the baseline. The three baseline failures (R05, R09, R10) suggest that single-query retrieval produces context that is harder for the judge to evaluate, possibly due to less structured or less coherent report output. In a production setting, reliability is a practical advantage.

This comparison carries a caveat: the baseline averages are computed over 7 prompts vs. report mode's 9. If the three failed baseline prompts would have scored lower, the true baseline mean could be below the reported 4.74; conversely, if they represent easier prompts, the gap could widen. To address this asymmetry directly, Table 7 also reports report-mode scores restricted to the seven prompts on which both modes were successfully judged (R01–R04, R06–R08). On this shared subset, report mode achieves a weighted overall of 4.59 versus the baseline's 4.74, a delta of −0.14. This controlled comparison confirms the gap is modest and does not alter the qualitative interpretation: both pipelines perform near ceiling, with report mode's primary practical advantage being robustness (7/7 judge success on shared prompts) rather than raw generation quality. Coverage (4.29 vs. 4.57, $\Delta = -0.28$) and factual recall (3.29 vs. 3.83, $\Delta = -0.54$) show the largest per-dimension gaps, consistent with sub-question overlap causing merged chunks to converge on the same documentation region rather than broadening topical coverage.

4.4 Retrieval Quality Analysis

Table 8 presents IR metrics computed against page-level ground truth for both modes (n=10 prompts each). Both modes achieve Recall@20 of only 0.135, reflecting an inherent granularity mismatch: 20 retrieved chunks cover approximately 20–40 pages of content, while the ground truth spans 11–43 relevant pages per prompt. This identifies retrieval breadth (not generation quality) as the primary bottleneck for long-form technical report generation.

Metric	**Report**	**Baseline**
Recall@5	0.068	0.075
Recall@10	0.107	0.111
Recall@15	0.135	0.119
Recall@20	0.135	0.135
MRR	0.379	0.423
NDCG@10	0.263	0.291
NDCG@20	0.224	0.245

Table 8: IR retrieval metrics (Report vs. Baseline, n=10). Both modes exhibit low absolute recall against page-level ground truth.

Beyond this shared bottleneck, the two retrieval strategies perform similarly overall. The baseline achieves a modest MRR advantage (0.423 vs. 0.379), indicating it finds the first relevant page slightly earlier, consistent with its broader initial retrieval (k=35). However, at higher k, recall converges: both modes reach 0.135 at Recall@20, and the decomposed pipeline achieves slightly higher Recall@15 (0.135 vs. 0.119), suggesting that multi-query retrieval captures diversity at intermediate ranks. These modest differences align with the finding of Leto et al. (2024) that retrieval parame-

ter tuning has limited downstream impact on RAG performance.

5 Discussion

In single-query mode, citation coverage is 30% with 1.50 unique citations per answer, reflecting the model's tendency to anchor on its highest-confidence chunk. In report mode, citation correctness is perfect (5.0/5.0) across all successfully judged prompts in both modes. The longer-form generation format (up to 8192 tokens) appears to encourage more thorough attribution, as the model distributes citations across the report's multiple sections.

The central finding from the report evaluation is that query decomposition improves robustness (90% vs. 70% judge success) but does not uniformly improve retrieval precision or generation quality. When sub-questions target related concepts (as is common for technical documentation prompts), the merged chunk set converges to the same documentation region as a single broad query. The baseline's higher k (k=35) captures marginally more diverse pages (MRR 0.423 vs. 0.379), though recall converges at k=20. This suggests a hybrid routing strategy: decompose only multi-topic prompts, and route focused prompts through a single broad retrieval.

Taken together, our findings suggest four design priorities for report-level RAG over technical documentation: (1) parser selection directly determines formula fidelity and retrieval quality; (2) re-ranking is disproportionately valuable for keyword-sensitive queries; (3) citation enforcement via prompt design is effective across both short-form and long-form generation; and (4) query decomposition provides a robustness advantage but not uniform retrieval gains, aligning with the finding of Leto et al. (2024) that retrieval parameter tuning has modest downstream effects.

6 Conclusion

We presented a formula-aware RAG system for OpenFOAM that operates in two modes. In single-query mode, a math-preserving parser, adaptive header-aligned chunking, two-stage cross-encoder re-ranking, and citation-enforcement prompt design achieve cross-document synthesis (2.35 distinct source documents per query). In report mode, LLM-driven query decomposition, multi-query retrieval with content-hash deduplication, and long-context synthesis produce structured, multi-section reports. Evaluated on a 10-prompt golden set with a six-dimension judge, both pipelines achieve overall scores above 4.5/5.0 with perfect citation correctness (5.0/5.0), demonstrating that citation enforcement scales from short-form answers to structured long-form reports. The decomposed pipeline achieves 90% judge success vs. 70% for the single-query baseline, providing a practical robustness advantage. On the seven prompts successfully judged by both modes, the fair like-for-like comparison yields a weighted overall of 4.59 (report) vs. 4.74 (baseline), a modest delta of 0.14 that does not change the qualitative interpretation.

Our analysis also reveals honest limitations. Query decomposition trades modest precision for robustness: the baseline's broader retrieval (k=35) achieves slightly higher MRR (0.42 vs. 0.38), though recall converges at k=20. More fundamentally, both modes are limited to $<$14% page recall, identifying retrieval breadth as the primary bottleneck for long-form technical report generation. Future work should explore iterative retrieval with generation-informed re-querying, hybrid sparse-dense retrieval to improve coverage over keyword-rich technical content, structure-aware chunking that preserves key-value dictionary entries as atomic units, and hybrid query routing that applies decomposition selectively to multi-topic prompts. Evaluating with an independent judge (e.g., GPT-4o) would quantify the Gemini self-preference effect; a citation-enforcement ablation comparing outputs with and without citation rules would establish whether hallucination suppression is causally attributable to prompt design rather than model behaviour.

Limitations

- **Low retrieval recall.** Both pipelines achieve below 14% page recall (Recall@20 $= 0.135$), reflecting a granularity mismatch between 20 retrieved chunks and ground truth spanning up to 43 relevant pages per prompt. This identifies retrieval breadth as the primary bottleneck for long-form technical report generation.

- **Single embedding model.** We evaluate only one bi-encoder (MiniLM-L12-v2, 384 dim.). Larger or domain-adapted embedding models may improve retrieval quality, but we do not explore this axis.

- **Corpus scope.** Evaluation is limited to four OpenFOAM manuals (13.3 MB). Generalisability to other engineering domains (e.g., ANSYS, COMSOL) or broader technical corpora remains untested.

- **LLM-as-a-judge reliability.** The judge failed to produce valid JSON on 10–30% of prompts (1/10 for report mode, 3/10 for baseline), introducing selection bias in aggregate scores. Judge agreement with independent human raters is not measured.

- **No human judge validation.** Generation quality scores are LLM-generated; retrieval metrics use manually annotated page-level ground truth, but the six-dimension judge scores have not been validated against independent human raters.

- **Gemini self-preference in LLM-as-a-judge.** The generation model (Gemini 2.5 Flash) and the judge (Gemini 2.5 Pro) belong to the same model family, introducing a potential self-preference bias well-documented in LLM-as-a-judge settings (Zheng et al., 2023). We do not evaluate with an independent judge (e.g., GPT-4o); the extent to which same-family preference inflates scores is unknown.

- **Citation enforcement is not ablated.** We assert that the citation-enforcement prompt design structurally suppresses hallucination, but we include no condition without citation rules to quantify this effect. The perfect citation correctness score (5.0/5.0) measures model compliance with an explicit prompt constraint and does not constitute a causal claim about hallucination reduction; a no-citation ablation baseline would be required to establish causality.

References

Christoph Auer, Maksym Lysak, Ahmed Nassar, Michele Dolfi, Nikolaos Schwarz, Daniele Baracchi, and Peter Staar. 2024. Docling technical report. *arXiv preprint arXiv:2408.09869*.

Lukas Blecher, Guillem Cucurull, Thomas Scialom, and Robert Stojnic. 2023. Nougat: Neural optical understanding for academic documents. *arXiv preprint arXiv:2308.13418*.

Brian J. Chan, Chao-Ting Chen, Jui-Hung Cheng, and Hen-Hsen Huang. 2025. Don't do RAG: When cache-augmented generation is all you need for knowledge tasks. *arXiv preprint arXiv:2412.15605*.

Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. 2024. Benchmarking large language models in retrieval-augmented generation. In *Proceedings of AAAI*, volume 38, pages 17754–17762.

Miao Cheng, Yuluo Luo, Jia Ouyang, Qiang Liu, Hao Liu, Li Li, Shengyue Yu, Bozhao Zhang, Jiacheng Cao, Junming Ma, Dian Wang, and Enming Chen. 2025. A survey on knowledge-oriented retrieval-augmented generation. *arXiv preprint arXiv:2502.08188*.

Manuel Faysse, Hugues Sibille, Tony Wu, Bilel Omrani, Gautier Viaud, Céline Hudelot, and Pierre Colombo. 2025. ColPali: Efficient document retrieval with vision language models. In *Proceedings of ICLR*.

Anding Gan, Hao Yu, Kai Zhang, Qiang Liu, Wenyan Yan, Zhanhui Huang, Song Tong, Enming Chen, and Guanghui Hu. 2025. Retrieval augmented generation evaluation in the era of large language models: A comprehensive survey. *Frontiers of Computer Science*.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, and Hongyang Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Jen-Hung Ju, Suzan Verberne, Maarten de Rijke, and Andrew Yates. 2025. Controlled retrieval-augmented context evaluation for long-form RAG. *arXiv preprint arXiv:2501.14674*.

Arize Leto, Carolina Aguerrebere, Ishwar Bhati, Ted Willke, Mariano Tepper, and Viet Anh Vo. 2024. Toward optimal search and retrieval for RAG. In *NeurIPS 2024 Workshop on Adaptive Foundation Models*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474.

Siran Li, Linus Stenzel, Carsten Eickhoff, and Seyed Ali Bahrainian. 2025. Enhancing retrieval-augmented generation: A study of best practices. *arXiv preprint arXiv:2501.07391*.

Zhuowan Li, Cheng Li, Mingyang Zhang, Qiaozhu Mei, and Michael Bendersky. 2024. Retrieval augmented generation or long-context LLMs? A comprehensive study and hybrid approach. In *Proceedings of EMNLP*, pages 15157–15173.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with BERT. *arXiv preprint arXiv:1901.04085*.

Lekha Yadav Panchumarthi, Sreenivasa Phani Gudari, Ankita Negi, Parsha Reddy Budime, and Harini Upadhya. 2025. RAG-BioQA: Retrieval-augmented generation for long-form biomedical question answering. *arXiv preprint arXiv:2501.12345*.

Konstantin Randl, Giulia Rocchietti, Aron Henriksson, Ziawasch Abedjan, Tony Lindgren, and John Pavlopoulos. 2025. RAG-E: Quantifying retriever-generator alignment and failure modes. *arXiv preprint arXiv:2502.05899*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of EMNLP*, pages 3982–3992.

Aditi Singh, Abul Ehtesham, Saket Kumar, and Tala Talaei Khoei. 2025. Agentic retrieval-augmented generation: A survey on agentic RAG. *arXiv preprint arXiv:2501.09136*.

Jiejun Tan, Zhicheng Dou, Wen Wang, Mang Wang, Weipeng Chen, and Ji-Rong Wen. 2024. HtmlRAG: HTML is better than plain text for modeling retrieved knowledge in RAG systems. In *Proceedings of The Web Conference*.

Shangyu Wu, Ying Xiong, Yufei Cui, Haolun Wu, Can Chen, Ye Yuan, Lianming Huang, Xue Liu, Tei-Wei Kuo, Nan Guan, and Chun Jason Xue. 2024. Retrieval-augmented generation for natural language processing: A survey. *arXiv preprint arXiv:2407.13193*.

Yixuan Xia, Jian Zhou, Zhongwei Shi, Junhui Chen, and Haoran Huang. 2025. Improving retrieval augmented language model with self-reasoning. In *Proceedings of AAAI*, volume 39.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-pack: Packaged resources to advance general Chinese embedding. *arXiv preprint arXiv:2309.07597*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*.

A System Output Examples

The following boxes show verbatim (lightly truncated) outputs from the two pipeline modes described in this paper, illustrating citation grounding, cross-document synthesis, graceful degradation on out-of-scope queries, clean refusal on absent keywords, and structured report generation.

Example A — Single-Query Mode: Concise Answer

Q: *What is the standard file format for OpenFOAM data (e.g., ASCII or binary)?*

```
OpenFOAM data files begin with a
FoamFile dictionary that includes
a format keyword entry to specify
the data format [1].  The possible
entries for the format keyword are
ascii or binary [1].
```

References

[1] UserGuide.pdf | Section: 2.2.3 The data file header | Page: 16

Example B — Single-Query Mode: Cross-Document Synthesis

Q: *How does the mapFields utility handle the transfer of data between different geometries?*

```
The mapFields utility in OpenFOAM
is designed to transfer one or
more fields from a given source
geometry to corresponding fields
for a target geometry [1][2][3].
Data is mapped from the source
to the target.  The utility reads
field data from the time directory
specified by startFrom/startTime in
the controlDict of the target case
[1][2][3].
```

Mapping Consistent Fields

```
Fields are deemed consistent if
the geometry and boundary types
of both source and target fields
are identical [1][2][3].  Mapping
is performed using the -consistent
command-line option [2][3][4].
```

Mapping Inconsistent Fields

```
When fields are not consistent,
mapFields requires a mapFieldsDict
dictionary in the system directory
[2].  The dictionary contains
patchMap (pairs of coincident
patches) and cuttingPatches (target
patches cutting through source
internal field) [2].
```

Mapping Parallel Cases

```
Additional options -parallelSource
and -parallelTarget are provided for
decomposed cases [2].
```

References

[1] UserGuide.pdf | Section: 4.6 Mapping fields between different geometries | Page: 58

[2] OpenFOAMUserGuide-v13.pdf | Section: 5.8 Mapping fields between different geometries | Page: 166

[3] TutorialGuide.pdf | Section: 2.1.5.3 Mapping the coarse mesh results onto the fine mesh | Page: 27

[4] UserGuide.pdf | Section: 4.6.1 Mapping consistent fields | Page: 58

Example C — Single-Query Mode: Graceful Degradation

Q: *What information is provided by the Ux_0 and UxFinalRes values in the solver output?*

```
While the specific terms
Ux_0 and UxFinalRes are not
explicitly defined in the provided
documentation, the context describes
the general behavior of linear
solvers and solution tolerances,
from which their meaning can be
inferred [5].

In OpenFOAM, linear solvers are
iterative and aim to reduce the
equation residual over successive
solutions [5].  The residual is a
measure of the error in the solution,
evaluated by substituting the
current solution into the equation
and taking the magnitude of the
difference between the left and
right-hand sides, normalized to be
independent of the problem's scale
[5].

Based on this, Ux_0 would represent
the initial residual for the velocity
field U (x-component) before the
linear solver begins its iterative
process [5].  UxFinalRes would
represent the final residual
after the solver has completed its
iterations [5].  [...]
```

References

[5] OpenFOAMUserGuide-v13.pdf | Section: 4.6.1 Linear solver control | Page: 112

Note: 4 of 5 retrieved chunks were index pages (highest rerank score 0.04); the generator correctly identified the documentation gap and hedged its response rather than hallucinating.

Example D — Single-Query Mode: Clean Refusal

Q: *What is the purpose of the runTimeModifiable keyword in the controlDict file?*

```
This information is not available in
the provided documentation.
```

Note: All 5 retrieved chunks scored highly on re-ranking (top score 0.97) and covered controlDict time/IO control, but none mentioned the `runTimeModifiable` *keyword. The generator correctly refused rather than extrapolating from related content.*

Example E — Report Mode: Structured Technical Report (Excerpt)

Q: *Write a technical overview of the discretization framework in OpenFOAM, covering spatial and temporal schemes, the fvSchemes dictionary structure, and how users configure gradient, divergence, Laplacian, interpolation, and surface normal gradient schemes.*

Sub-questions generated by decomposition:

1. What are the fundamental concepts of spatial and temporal discretization in OpenFOAM, and what role does the fvSchemes dictionary play?
2. How are gradient, divergence, and Laplacian schemes configured within OpenFOAM's fvSchemes dictionary?
3. How are interpolation and surface normal gradient schemes configured in OpenFOAM's fvSchemes dictionary, and what is their purpose?

Discretization Principles in OpenFOAM

```
Discretization is the process
of approximating a continuous
problem into discrete quantities,
a fundamental step in numerical
simulations [7].  In OpenFOAM,
this involves three key aspects:
spatial discretization, temporal
discretization, and equation
discretization [7].  Spatial
discretization defines the solution
domain using a set of points that
fill and bound a region of space [7].
OpenFOAM's mesh supports arbitrary
polyhedral cells in 3D, bounded by
arbitrary polygonal faces [9].  The
Finite Volume (FV) method bases
spatial derivatives on Gaussian
integration, which sums values
on cell faces [1][2].  Temporal
discretization divides the time
domain into finite intervals [7].
```

The fvSchemes Dictionary: Structure and Purpose

```
The fvSchemes dictionary,
located in the system directory
of an OpenFOAM case, is the
central configuration file for
specifying numerical schemes
[1][2].  Its sub-dictionaries
include:  timeScheme (∂Ψ/∂t),
gradSchemes (∇Ψ), divSchemes
(∇ · Ψ), laplacianSchemes (∇ · Γ∇Ψ),
interpolationSchemes, snGradSchemes,
and wallDist [1][2].

Within each sub-dictionary, schemes
can be specified globally via a
default entry or individually for
specific terms [1][2][5].  Setting
default none obliges the user to
specify all terms individually [5].
```

[... report continues with 5 additional sections covering ddtSchemes, gradSchemes, divSchemes, laplacianSchemes, interpolationSchemes, snGradSchemes, and scheme selection guidance ...]

References (10 unique chunks from 4 source PDFs)

[1] OpenFOAMUserGuide-v13.pdf | 4.5 Numerical schemes | p. 103

[2] UserGuide.pdf | 6.2 Numerical schemes | p. 75

[5] OpenFOAMUserGuide-v13.pdf | 4.5 Numerical schemes | p. 103

[7] ProgrammersGuide.pdf | 3.2 Overview of discretisation | p. 29

[9] OpenFOAMUserGuide-v13.pdf | 5.1 Mesh description | p. 128

EncouRAGe: Evaluating RAG Local, Reliable, and Efficient

Jan Strich, Martin Semmann, Chris Biemann
Hub of Computing and Data Science (HCDS)
University of Hamburg, Germany

Correspondence: {first_name}.{last_name}@uni-hamburg.de

Abstract

We introduce **EncouRAGe**, a comprehensive Python library designed to streamline the development and evaluation of Retrieval-Augmented Generation (RAG) systems using Large Language Models (LLMs) and Embedding Models. EncouRAGe comprises five modular and extensible components: *Type Manifest*, *RAG Factory*, *Inference*, *Vector Store*, and *Metrics*, facilitating flexible experimentation and extensible development. Each component helps to make the development RAG evaluation and emphasizes **scientific reproducibility**, **diverse evaluation metrics**, and **local deployment**, enabling researchers to efficiently assess datasets within RAG workflows. This paper presents implementation details and an extensive evaluation across multiple benchmark datasets, including *25k QA pairs* and *over 51k documents*. Our results show that RAG still underperforms compared to the *Oracle Context*, while *Hybrid BM25* consistently achieves the best results across all four datasets.
Code: github.com/uhh-hcds/encourage

1 Introduction

Retrieval-Augmented Generation (RAG) (Lewis et al., 2020; Huang and Huang, 2024; Nikishina et al., 2025) has emerged as a dominant approach for enhancing large language models (LLMs) with external knowledge sources. By combining retrieval and generation, RAG systems can provide more factual, contextually grounded, and up-to-date responses (Shuster et al., 2021; Sahoo et al., 2024). However, the rapid development of RAG architectures (Huang and Huang, 2024; Sarthi et al., 2024), methods (Gao et al., 2021, 2023), and LLM-as-a-Judge metrics (Es et al., 2024a) has created a need for an implementation-focused RAG evaluation library to create a reliable way to compare performance on domain-specific datasets.

Assessing a RAG system for a specific dataset requires analyzing the performance of both the retriever and generator, as well as their interaction. It is particularly challenging because LLM outputs can be lengthy, complex, and ambiguous, making it difficult to define clear correctness or objectively measure factual accuracy (Ru et al., 2024). Recent RAG evaluation frameworks, such as RAGAS (Saad-Falcon et al., 2024), RAGChecker (Ru et al., 2024), and TruLens (Snowflake Inc., 2024), have begun addressing these challenges by providing libraries to jointly analyze retrieval and generation quality, reflecting the interdependence between the two components. Instead of relying solely on traditional metrics like BLEU (Papineni et al., 2002) or ROUGE (Lin, 2004), they use LLM-based or embedding-based measures to directly evaluate factual correctness, relevance, and consistency of the generated text (Es et al., 2024a). Additionally, they often provide enhanced UI and monitoring features for RAG in production environments.

Despite recent advances, a lack of the following features in state-of-the-art (SOTA) frameworks remains, hindering the effective comparison of RAG methods for domain-specific datasets:
(1) Lack of comparability of scientific methods. Existing libraries in this field focus solely on implementing metrics, with no standardized implementation of SOTA RAG methods. This heterogeneity hinders meaningful and reproducible comparisons across different RAG approaches.
(2) Limited flexibility of LLM-as-a-judge metrics. LLM-as-a-judge metrics are sensitive to domain and dataset, and must be flexible to be effective. Therefore, custom prompts for individually designing evaluations are crucial and are not currently available in existing frameworks.
(3) Primarily cloud-oriented, with limited flexibility for integrating new metrics or methods. Many existing frameworks are designed with a narrow focus on specific local setups and lack modu-

Proceedings of the First Workshop on Multilingual Report Generation via Retrieval Augmented Generation (RAG4Reports), pages 36–46

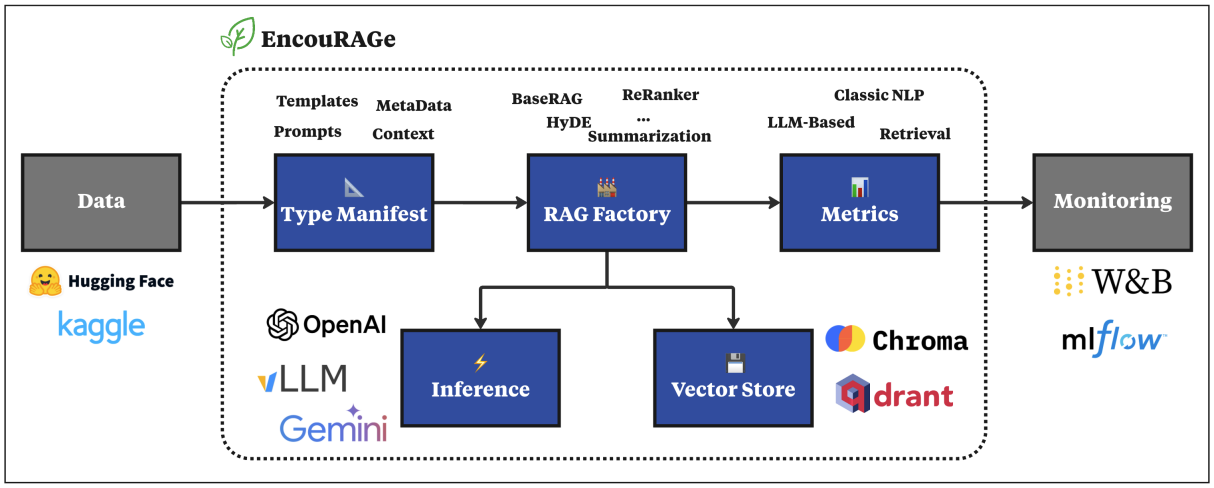

Figure 1: System overview of the **EncouRAGe** Python library. Input data in any format must be transformed to fit the **Type Manifest**. The **RAG Factory** provides various RAG methods, while the **Metrics** component implements all evaluation metrics. **Inference** can be executed locally or via cloud providers using the OpenAI SDK, and supported **Vector Store** includes Chroma and Qdrant. The output fits popular monitoring systems.

larity for broader integration. A flexible, plug-and-play architecture is essential to support new RAG methods and additional new metrics.

In order to solve these challenges, we present **EncouRAGe**, an open-source Python library for comprehensive, reproducible, and extensible RAG evaluation (Fig. 1). It can be integrated with any dataset into a parsable, object-oriented structure for effective and traceable workflows. EncouRAGe includes 9 RAG methods, manages LLM and embedding inference, and provides metrics in three categories:
1. **Generator Metrics**: Evaluating the performance end-to-end with classic NLP metrics (ROUGE, BLEU, etc.).
2. **Retrieval Metrics**: Evaluating the effectiveness of the retriever in finding relevant information from the knowledge base.
3. **LLM-as-a-Judge**: Evaluating the effectiveness of the RAG pipeline enhanced by custom prompts.

To address the rapid pace and volume of new RAG and LLM research, reproducing and comparing methods has become increasingly time-consuming. With EncouRAGe, evaluating new approaches or verifying others' claims becomes straightforward and extensible. The library enables quick benchmarking across diverse methods, models, and datasets with minimal manual effort, requiring mainly GPU time rather than complex reimplementation. We demonstrate this through experiments comparing RAG methods on four QA datasets.

The main contributions of this paper are as follows:

- We present **EncouRAGe**, a novel RAG evaluation library that consists of **9** SOTA RAG methods, an object-oriented `Type Manifest`, and over 20 `Metrics` to get transparent and reliable results.
- We conduct a case study across four datasets with **25k QA pairs** and **50k documents**, demonstrating the workflow, effectiveness, and usefulness of the framework.

2 Related Work

Retrieval-Augmented Generation. LLMs excel at text generation but face challenges such as outdated knowledge and hallucinations (Shuster et al., 2021; Huang and Huang, 2024). RAG addresses these issues by incorporating external knowledge, improving accuracy and factuality (Lewis et al., 2020; Sarthi et al., 2024). This approach is critical in domains such as law (Cui et al., 2024), medicine (Xiong et al., 2024; Chen et al., 2025b), and finance (Chen et al., 2025b), where precision is crucial. RAG systems have achieved strong results in tasks such as open-domain question answering (Kim and Lee, 2024), code generation (Wang et al., 2025), and dialogue (Chen et al., 2025a). They have also been adopted in real-world applications, including LlamaIndex (Liu, 2022), where they help integrate external data sources into large language model workflows for more context-aware responses.

Evaluation of RAG. (1) Lightweight and Dataset-Based Evaluation Tools. RAGAS (Es et al., 2024a) and RAGChecker (Ru et al., 2024) provide systematic metrics for assessing RAG pipelines. RAGAS was the first library to provide LLM-as-a-Judge metrics and offers additional classic NLP evaluation metrics. In contrast, RAGChecker employs a modular approach that decomposes RAG performance into retrieval and generation components, allowing detailed error attribution and interpretability. But both lack implementation of RAG methods and are cloud-focused.

(2) Monitoring Frameworks for Production RAG Applications. TruLens (Snowflake Inc., 2024) and Opik (Comet ML, Inc., 2025) extend evaluation capabilities to deployed RAG and LLM-based applications. TruLens enables real-time tracking, custom evaluation functions, and integration with human feedback loops, making it suited for agent-oriented systems in which RAG serves as an auxiliary mechanism. Opik provides end-to-end monitoring and analytics for production environments, supporting model observability for improvement workflows. Both systems are application-focused and not designed for research.

(3) General LLM Evaluation Frameworks. DeepEval (AI, 2024) and OpenAI evals (OpenAI, 2025a) offer broader evaluation capabilities that extend beyond RAG-specific scenarios. DeepEval provides standardized testing interfaces and model-agnostic benchmarking tools for assessing reasoning, factual accuracy, and robustness.

3 EncouRAGe

As shown in Fig. 1, EncouRAGe is an end-to-end RAG evaluation library in Python consisting of five main components. To showcase our library, we define the formal idea in Subsec. 3.1. It supports any dataset containing queries, answers, and golden context triples. The data is standardized via the *Type Manifest*, as explained in Subsec. 3.2, ensuring type safety of all elements during processing. Users can select from a wide range of RAG methods defined in the *RAG Factory* (Subsec. 3.3), which utilize the *Inference Handler* and *Vector Store* described in Subsec. 3.4. For evaluation, EncouRAGe uses over 20 *Metrics* to determine whether the performance loss originates from the retriever or the generator. EncouRAGe is a fully tested codebase and supports all LLMs and embedding models available on HuggingFace.

3.1 Task Formulation

We formalize the framework underlying our library as follows. **EncouRAGe** provides a collection of RAG methods $\mathcal{R}$ and evaluation metrics $\mathcal{M}$, which can be applied to any dataset $\mathcal{D}$ that is defined as $\mathcal{D} = \{(q_i, a_i, c_i)\}_{i=1}^{N}$, where q_i denotes a question, a_i its corresponding answer, and c_i the associated gold context, which may also include additional elements not paired with a q_i and a_i.

All contexts c_i are embedded and stored in a vector store $\mathcal{V} = \text{Embed}(\{c_i\}_{i=1}^{N})$, which serves as the retrieval space for each RAG method $\mathcal{R}$. A RAG method is denoted as $\mathcal{R} = (R, G)$, where R represents the retriever and G the generator. Given a query q_i, the retriever R accesses the vector store $\mathcal{V}$ to return a set of top-k relevant contexts

$$C_k = [c_1, c_2, \ldots, c_k] = R(q_i; \mathcal{V}), \tag{1}$$

where c_1 is the most relevant context and c_k the least among the top k.

C_k is then passed to the generator G to produce a predicted answer

$$\hat{a}_i \sim P(a \mid q_i, C_k). \tag{2}$$

The quality of the predicted answer $\hat{a}_i$ is evaluated against the ground-truth a_i using a set of metrics $\mathcal{M} = \{m_1, m_2, \ldots, m_J\}$, producing scores

$$s_{i,j} = m_j(\hat{a}_i, a_i), \quad \forall i \in [1, N],\ j \in [1, J]. \tag{3}$$

Each score $s_{i,j}$ reflects the performance of the retriever, the generator, and their interaction.

3.2 Type Manifest

The *Type Manifest* is a collection of object-oriented Python structures designed to streamline evaluation, improve reliability, and enhance maintainability, as illustrated in Fig. 2. Each data point is represented as a `Document` object containing a unique identifier, textual content, and an associated `MetaData` object. The `MetaData` object is a dictionary for meta information not used in processing, enabling flexible organization and analysis. A `Context` may include multiple `Documents`, reflecting the capability of RAG methods to retrieve several documents. Each `Context` is injected into a `Jinja2` template, allowing users to define prompt structures and incorporate document content via template variables. The `Prompt` object contains an identifier, conversation history (sys and user messages), a `Context`, and a `MetaData` object.

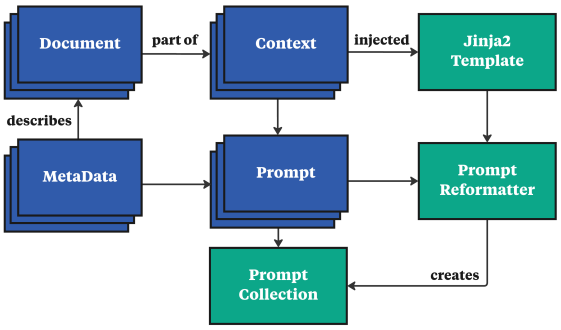

Figure 2: *Type Manifest*. Gold **Documents** link to the **Context**, combined via **Jinja2 templates** with the prompt to form the **Prompt Collection. Metadata** ensures traceability for documents and prompts.

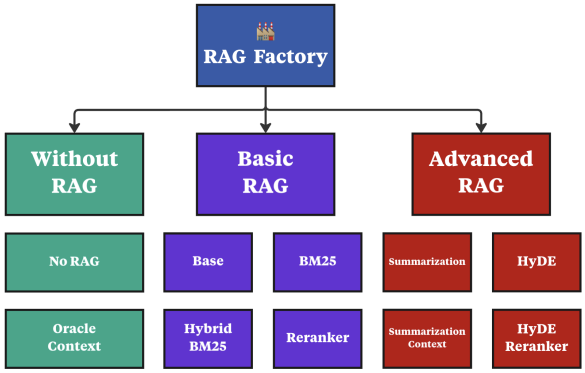

Figure 3: Overview of *RAG Factory*. RAG Factory is organized into three categories: Without RAG, Basic RAG, and Advanced. In total, EncouRAGe supports **10 methods**, with more to be added in the future.

When constructing a `PromptCollection`, the `PromptReformatter` renders the `Jinja2` template, and the related template variables. This ensures consistent usage across the evaluation pipeline, allowing seamless integration with the *RAG Factory* and reproducible metric computation.

3.3 RAG Factory

The *RAG Factory* is the core of our Python library, defining all available methods for dataset evaluation (Fig. 3). It currently provides ten methods across three categories: **Without RAG**, **Basic RAG**, and **Advanced RAG**, with a standardized interface for easy extension and a *Type Manifest* for consistent integration.

Without RAG. In the *Pretrained-Only* setup, no retriever is employed, and models must answer questions solely based on their pretraining knowledge. Conversely, the *Oracle Context* setting assumes that the relevant context is directly passed to the generator.

Basic RAG Methods. The *Base RAG* implementation follows the original RAG approach (Lewis et al., 2020), where only the question is embedded to retrieve the top-k documents, which are then passed unchanged to the generator. In addition, a *Standard BM25* (Robertson and Zaragoza, 2009) retriever is provided as a purely lexical baseline, relying on term-frequency and inverse document-frequency weighting to rank documents based on keyword overlap. *Hybrid BM25* (Gao et al., 2021) combines sparse lexical retrieval using BM25 with dense vector retrieval, leveraging both methods to improve recall and relevance. Finally, the *Reranker* method (Glass et al., 2022) can apply any cross-encoder model after initial retrieval to reorder documents in a shared embedding space.

Advanced RAG Methods. This category consists of methods that modify the query, transform retrieved contexts, or employ iterative retrieval strategies. The *HyDE* method (Gao et al., 2023) generates hypothetical answers for each question, using them as refined queries to retrieve more relevant documents. *Summarization* reduces noise by summarizing each retrieved context using an LLM, focusing on essential information. *SumContext* applies a similar summarization step, retaining the original full documents for generation, aiming to reduce distractions while preserving the content.

3.4 Inference and Vector Store

For *Inference*, we focus on local deployment, primarily leveraging `vLLM` (Kwon et al., 2023), which is one of the fastest LLM engines available. Communication with the LLM is handled through the OpenAI Python SDK (OpenAI, 2025b), allowing seamless integration of HuggingFace, OpenAI, and Gemini. Also structured output is possible through providing a pydantic model to the runner. Other integrations are planned.

For the *Vector Store*, we utilize a serverless version of Chroma (Chroma Team, 2025), which creates an in-memory `SQLite3` database stored locally. As an alternative, we also support Qdrant (Qdrant Team, 2025), which can handle large-scale document collections. Both implementations share a unified interface, simplifying the integration of additional vector database backends in the future.

3.5 Metrics

Evaluating different RAG methods is a core feature of our library, implemented through the *Metrics* module. To provide a transparent and comprehensive assessment, we include metrics for all components of RAG across multiple datasets, grouped into three categories: generator metrics, retrieval metrics, and LLM-based metrics. To align with existing research, we primarily use the `evaluate` (HuggingFace, 2025) library for most generator metrics. For retrieval metrics, we used the `ir_measures` (MacAvaney et al., 2022) library, which is specialized for information retrieval. LLM-based metrics are fully implemented and easily customizable with prompts and examples, enabling flexible use across datasets. Appendix B gives an overview of the metrics of the library.

The *Type Manifest* typing system integrates with all metrics, enabling automatic computation once a RAG method is selected. Outputs are stored for easy integration with `mlflow` (Zaharia et al., 2018) or `wandb` (Biewald, 2020).

4 Case Study

To demonstrate our library, we conduct a case study analyzing four popular datasets using each RAG method implemented in EncouRAGe. The datasets used are summarized in Table 1 and Appendix A. Each QA pair has at least one golden document, enabling comparisons of RAG methods across generation and retrieval metrics.

4.1 Implementation

Type Manifest. For each dataset, we followed the following implementation steps. Each sample was converted to a list of `user_prompts`, a collection of typed `Document` objects, where each document corresponds to a single context and has a unique identifier used for retrieval metrics, and an associated collection of `MetaData` objects.

RAG Factory. For the initialization of each method, it is required to provide the `Document` objects, the embedding function identifier from Hugging Face, an `InferenceRunner` instance, and the related prompt template to render the content. Each method also has optional parameters, such as vector DB filter queries and batch sizes for inserting or querying documents, to be flexible across various data sizes. Running each RAG method requires passing `user_prompts`, a system prompt, and a set of `retrieval_prompts` for querying the vector db.

Subset	#QA	Q Tok.	#Docs	Doc Tok.
BioSQA	4,012	13.5	34,634	3,222.8
FinQA	6,237	45.1	2,110	1,080.4
FeTaQA	7,324	20.4	6,929	520.0
HotPotQA	7,395	21.4	7,395	1,421.2
Total	**24,968**	25.7	**51,068**	2,506.7

Table 1: Comparison of QA pairs, documents, and average token lengths across subsets. HotPotQA (Yang et al., 2018) and FeTaQA (Nan et al., 2022) are based on Wikipedia articles, while FinQA (Chen et al., 2021) and BioSQA (Krithara et al., 2023) use PDF documents. Avg. tokens are computed with Gemma-3 tokenizer.

It is also possible to just get the retrieval results for debugging. This modular design allows swapping prompts, methods, and runners easily, without modifying the surrounding evaluation pipeline.

Inference and Vector Store. We initialized a `BatchInferenceRunner` with standard `SamplingParams` (e.g., temperature = 0) and chose Gemma3 27B (Kamath et al., 2025) as the generator. For retrieval, we selected E5-Large Instruct (Wang et al., 2024), which has an embedding space of 1024 tokens. We used a pydantic model for providing structured output, having three fields: reasoning steps (`list[str]`), list of supporting facts (`list[str]`) and final answer (`str`) to make evaluation more transparent. All experiments were conducted on NVIDIA H100 GPUs.

Metrics. For retrieval evaluation, metrics such as *MRR@10*, *MAP*, and *Recall@10* are computed by comparing ranked document lists against the annotated gold documents. The metrics objects can be initialized with parameter such as k for MRR or Recall, making them flexible for any use case. *MRR@10* is used for tasks with a single relevant document, while *MAP* is used when multiple documents are required to answer a query. *Recall@10* complements these metrics by measuring whether relevant documents appear in the retrieved set, independent of rank.

For generation evaluation, task-specific metrics are applied depending on the answer format. Short textual answers are evaluated using macro-averaged *F1*, while numerical answers are evaluated using *NM* (Strich et al., 2025). Since all predictions are returned as responses, metrics can be applied consistently across datasets and RAG methods.

RAG Method	HotPotQA			FeTaQA			FinQA			BioSQA		
	F1	MRR	R@10	F1	MRR	R@10	NM	MRR	R@10	F1	MAP	R@10
+ Pretrained-Only	36.7	–	–	29.3	–	–	9.6	–	–	39.3	–	–
+ Oracle Context	43.4	100	100	49.4	100	100	72.9	100	100	54.5	100	100
+ Vanilla RAG	37.1	68.8	82.5	49.8	87.5	**92.7**	47.8	45.6	71.9	47.2	42.1	50.1
+ BM25	40.5	29.1	**98.4**	39.0	18.6	68.5	50.8	45.5	77.2	44.3	26.7	44.3
+ Hybrid BM25	**40.8**	70.4	96.9	**50.1**	**87.6**	92.4	**51.8**	46.7	**82.1**	**49.9**	43.6	**55.7**
+ Reranker	36.1	63.4	78.4	49.6	86.9	92.2	50.2	45.5	71.7	44.5	42.1	50.1
+ HyDE	30.2	40.6	61.0	48.4	82.8	89.5	41.7	37.3	61.4	49.2	**45.9**	54.7
+ Summarization	31.7	**71.0**	83.3	40.0	83.9	90.5	45.9	**48.5**	74.6	45.0	43.2	51.0
+ SumContext	37.7	**70.9**	83.3	48.6	84.0	90.5	48.6	48.1	74.6	47.8	43.0	50.4

Table 2: Overall Performance (F1, MRR/MAP, R@10) for Gemma3 with E5-Large on all four datasets. Cells in **Bold** indicate the highest value over all RAG methods, and underlined across RAG method categories.

4.2 Main Results

Without RAG Runs. The *Pretrained-Only* results for HotPotQA, FeTaQA, and BioSQA are comparatively high, indicating that these datasets are likely part of the LLM's pretraining corpus (e.g., Wikipedia-based sources). In contrast, FinQA shows a substantially lower score, suggesting that its domain-specific financial questions are less represented in the model's training data. However, when provided with *Oracle Context*, all datasets achieve higher performance than in the *Pretrained-Only*, confirming that the model can effectively reason when supplied with relevant context. Notably, the *Oracle Context* results for HotPotQA (Yang et al., 2018) and FeTaQA (Nan et al., 2022) are slightly below the original papers, as our setup relies solely on a zero-shot setting.

Base RAG. When incorporating retrieval, *Base-RAG* performs close to the *Oracle Context* on HotPotQA, FeTaQA, and BioSQA, but shows a larger performance gap on FinQA. Interestingly, the traditional *BM25* retriever performs comparably to the dense retriever in terms of F1, but exhibits substantially lower MRR on HotPotQA and BioSQA. Combining both retrieval methods (*Hybrid BM25*) yields the most consistent improvements, achieving the highest F1 and Recall@10 across all datasets. Adding a reranker does not yield measurable gains, as the MRR remains similar to base RAG when the top-10 retrieved documents are reordered.

Advanced RAG. Among the advanced retrieval enhancements, *HyDE* offers modest improvements only for BioSQA, which are insufficient to justify its additional computational cost. *Summarization*-based retrieval increases MRR and Recall@10 for all datasets except FeTaQA, but leads to lower F1 scores, likely due to information loss during summarization. The combined approach, *SumContext*, which integrates both summarized and original documents, achieves a balanced trade-off, yielding results similar to *Base-RAG* with slightly better retrieval performance. However, given its additional retrieval and generation overhead, its practical benefit remains limited.

Overall, the results demonstrate that hybrid retrieval remains the most effective RAG method across diverse QA domains. In contrast, advanced generation-based retrieval strategies offer only small improvements at a higher computational cost.

5 Conclusion

In this paper, we introduce **EncouRAGe**, a Python library for evaluating RAG methods locally, reliably, and efficiently. EncouRAGe enables researchers to systematically investigate retrieval-augmented generation techniques on their own datasets, providing insights into which configurations yield the best results for specific domains. The library includes over 10 RAG methods, more than 20 metrics, and supports models from Hugging Face, OpenAI, and Gemini. Furthermore, users can flexibly choose between Chroma and Qdrant as the vector store backend.

To demonstrate the capabilities of EncouRAGe, we conducted experiments on four widely used datasets, revealing that the optimal RAG method varies across domains. Overall, our results indicate that the *Hybrid BM25* approach delivers the best performance among the tested configurations.

With EncouRAGe, we aim to contribute to the RAG research community by providing a flexible, extensible evaluation library that enables deeper analysis and supports research and product development for RAG.

Ethical Statement

We do not foresee ethical risks in our work. The dedicated use of this library is to evaluate the performance of RAG methods on multiple metrics. We believe that the library can help identify errors in datasets and uncover biases or common mistakes when applying RAG to a dataset.

Acknowledgement

This work is supported by the Genial4KMU project at the Universität Hamburg, funded by the BMFTR (grant no. 16IS24044B).

References

Confident AI. 2024. DeepEval: The Open-Source LLM Evaluation Framework. Https://deepeval.com.

Lukas Biewald. 2020. Experiment Tracking with Weights and Biases. Https://www.wandb.com/.

Yifu Chen, Shengpeng Ji, Haoxiao Wang, Ziqing Wang, Siyu Chen, Jinzheng He, Jin Xu, and Zhou Zhao. 2025a. WavRAG: Audio-Integrated Retrieval Augmented Generation for Spoken Dialogue Models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12505–12523, Vienna, Austria. Association for Computational Linguistics.

Zhe Chen, Yusheng Liao, Shuyang Jiang, Pingjie Wang, YiQiu Guo, Yanfeng Wang, and Yu Wang. 2025b. Towards Omni-RAG: Comprehensive Retrieval-Augmented Generation for Large Language Models in Medical Applications. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15285–15309, Vienna, Austria. Association for Computational Linguistics.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. 2021. FinQA: A Dataset of Numerical Reasoning over Financial Data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3697–3711, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Chroma Team. 2025. Chroma: Open-source search and retrieval database for AI applications. Https://github.com/chroma-core/chroma.

Comet ML, Inc. 2025. Comet — The AI Developer Platform. Https://www.comet.com/site/.

Jiaxi Cui, Munan Ning, Zongjian Li, Bohua Chen, Yang Yan, Hao Li, Bin Ling, Yonghong Tian, and Li Yuan. 2024. Chatlaw: A Multi-Agent Collaborative Legal Assistant with Knowledge Graph Enhanced Mixture-of-Experts Large Language Model. *arXiv preprint*. ArXiv:2306.16092.

Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. 2024a. RAGAs: Automated Evaluation of Retrieval Augmented Generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 150–158, St. Julians, Malta. Association for Computational Linguistics.

Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. 2024b. RAGAs: Automated evaluation of retrieval augmented generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 150–158, St. Julians, Malta. Association for Computational Linguistics.

Luyu Gao, Zhuyun Dai, Tongfei Chen, Zhen Fan, Benjamin Van Durme, and Jamie Callan. 2021. Complementing Lexical Retrieval with Semantic Residual Embedding. *arXiv preprint*. ArXiv:2004.13969.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023. Precise Zero-Shot Dense Retrieval without Relevance Labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1762–1777, Toronto, Canada. Association for Computational Linguistics.

Michael Glass, Gaetano Rossiello, Md Faisal Mahbub Chowdhury, Ankita Naik, Pengshan Cai, and Alfio Gliozzo. 2022. Re2G: Retrieve, rerank, generate. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2701–2715, Seattle, Washington, United States. Association for Computational Linguistics.

Yizheng Huang and Jimmy Huang. 2024. A Survey on Retrieval-Augmented Text Generation for Large Language Models. *arXiv preprint*. ArXiv:2404.10981.

HuggingFace. 2025. Evaluate: A library for easily evaluating machine learning models and datasets. Https://github.com/huggingface/evaluate.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446.

Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, and et al. Mesnard. 2025. Gemma 3 Technical Report. *arXiv preprint*. ArXiv:2503.19786.

Kiseung Kim and Jay-Yoon Lee. 2024. RE-RAG: Improving Open-Domain QA Performance and Interpretability with Relevance Estimator in Retrieval-Augmented Generation. In *Proceedings of the*

2024 Conference on Empirical Methods in Natural Language Processing, pages 22149–22161, Miami, Florida, USA. Association for Computational Linguistics.

Anastasia Krithara, James G. Mork, Anastasios Nentidis, and Georgios Paliouras. 2023. The Road From Manual To Automatic Semantic Indexing Of Biomedical Literature: a 10 Years Journey. *Frontiers in Research Metrics and Analytics*, 8.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, pages 611–626, Koblenz, Germany. Association for Computing Machinery.

Patrick S. H. 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 *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020*, virtual.

Chao Li, Zhiyuan Liu, Mengmeng Wu, Yuchi Xu, Huan Zhao, Pipei Huang, Guoliang Kang, Qiwei Chen, Wei Li, and Dik Lun Lee. 2019. Multi-Interest Network with Dynamic Routing for Recommendation at Tmall. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, pages 2615–2623, Beijing, China. Association for Computing Machinery.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Jerry Liu. 2022. LlamaIndex. Https://github.com/jerryjliu/llama_index.

Sean MacAvaney, Craig Macdonald, and Iadh Ounis. 2022. Streamlining Evaluation with ir-measures. In *Advances in Information Retrieval: 44th European Conference on IR Research, ECIR 2022, Stavanger, Norway, April 10–14, 2022, Proceedings, Part II*, page 305–310, Berlin, Heidelberg. Springer-Verlag.

Andrew Mutton, Mark Dras, Stephen Wan, and Robert Dale. 2007. GLEU: Automatic Evaluation of Sentence-Level Fluency. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 344–351, Prague, Czech Republic. Association for Computational Linguistics.

Linyong Nan, Chiachun Hsieh, Ziming Mao, Xi Victoria Lin, Neha Verma, Rui Zhang, Wojciech Kryściński, Hailey Schoelkopf, Riley Kong, Xiangru Tang, Mutethia Mutuma, Ben Rosand, Isabel Trindade, Renusree Bandaru, Jacob Cunningham, Caiming Xiong, Dragomir Radev, and Dragomir Radev. 2022. FeTaQA: Free-form Table Question Answering. *Transactions of the Association for Computational Linguistics*, 10:35–49.

Irina Nikishina, Özge Sevgili, Mahei Manhai Li, Chris Biemann, and Martin Semmann. 2025. Creating a Taxonomy for Retrieval Augmented Generation Applications. *arXiv preprint*. ArXiv:2408.02854.

OpenAI. 2025a. OpenAI Evals — A framework for evaluating LLMs. Https://github.com/openai/evals.

OpenAI. 2025b. openai-python: Official Python library for the OpenAI API. Https://github.com/openai/openai-python.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Qdrant Team. 2025. Qdrant: High-performance, massive-scale Vector Database and Vector Search Engine for the next generation of AI. Https://github.com/qdrant/qdrant.

Stephen Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389.

Dongyu Ru, Lin Qiu, Xiangkun Hu, Tianhang Zhang, Peng Shi, Shuaichen Chang, Cheng Jiayang, Cunxiang Wang, Shichao Sun, Huanyu Li, Zizhao Zhang, Binjie Wang, Jiarong Jiang, Tong He, Zhiguo Wang, Pengfei Liu, Yue Zhang, and Zheng Zhang. 2024. RAGChecker: A Fine-grained Framework for Diagnosing Retrieval-Augmented Generation. In *Advances in Neural Information Processing Systems*, volume 37, pages 21999–22027. Curran Associates, Inc.

Jon Saad-Falcon, Omar Khattab, Christopher Potts, and Matei Zaharia. 2024. ARES: An Automated Evaluation Framework for Retrieval-Augmented Generation Systems. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 338–354, Mexico City, Mexico. Association for Computational Linguistics.

Pranab Sahoo, Prabhash Meharia, Akash Ghosh, Sriparna Saha, Vinija Jain, and Aman Chadha. 2024. A Comprehensive Survey of Hallucination in Large Language, Image, Video and Audio Foundation Models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 11709–11724, Miami, Florida, USA. Association for Computational Linguistics.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. 2024. RAPTOR: Recursive Abstractive Processing for Tree-Organized Retrieval. In *The Twelfth International Conference on Learning Representations*, Vienna, Austria. The Association for Computational Linguistics.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval Augmentation Reduces Hallucination in Conversation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3784–3803, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Snowflake Inc. 2024. TruLens Eval. Https://www.trulens.org.

Marina Sokolova and Guy Lapalme. 2009. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437.

Jan Strich, Enes Kutay Isgorur, Maximilian Trescher, Chris Biemann, and Martin Semmann. 2025. T^2-RAGBench: Text-and-Table Benchmark for Evaluating Retrieval-Augmented Generation. *Preprint*, arXiv:2506.12071.

Ellen M. Voorhees. 1993. Using WordNet to disambiguate word senses for text retrieval. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '93, pages 171–180, Pittsburgh, Pennsylvania, USA. Association for Computing Machinery.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Multilingual E5 Text Embeddings: A Technical Report. *arXiv preprint*. ArXiv:2402.05672.

Yuchen Wang, Shangxin Guo, and Chee Wei Tan. 2025. From Code Generation to Software Testing: AI Copilot With Context-Based Retrieval-Augmented Generation. *IEEE Software*, 42(4):34–42.

Guangzhi Xiong, Qiao Jin, Zhiyong Lu, and Aidong Zhang. 2024. Benchmarking Retrieval-Augmented Generation for Medicine. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 6233–6251, Bangkok, Thailand. Association for Computational Linguistics.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Matei A. Zaharia, Andrew Chen, Aaron Davidson, Ali Ghodsi, Sue Ann Hong, Andy Konwinski, Siddharth Murching, Tomas Nykodym, Paul Ogilvie, Mani Parkhe, Fen Xie, and Corey Zumar. 2018. Accelerating the Machine Learning Lifecycle with MLflow. *IEEE Data Eng. Bull.*, 41:39–45.

A Dataset Overview

For evaluation, we selected four datasets from different domains to demonstrate EncouRAGe's flexibility in handling diverse datasets.

HotPotQA. The dataset (Yang et al., 2018) contains 7,395 general knowledge question–answer pairs sourced from Wikipedia, emphasizing multi-hop reasoning. Each question is linked to a single gold document, resulting in 7,395 papers in total, with an average document length of 1,421.2 tokens. Questions average 21.4 tokens, reflecting concise query formulation. HotPotQA challenges models to combine information across multiple articles to answer a single question accurately. For evaluation, we employ *MRR@10* for retrieval and *F1-Score* for answer generation, consistent with multi-hop reasoning tasks.

FetaQA. The dataset (Nan et al., 2022) includes over 7,300 QA pairs that require complex reasoning over tabular data, extracted from Wikipedia articles. The dataset emphasizes semantic understanding and reasoning across multiple table cells. Each question is accompanied by a textual explanation and a gold-standard answer derived from structured tables. We use the training split and all table documents related to these QA pairs, resulting in a total of 6,929 documents to process. Evaluation employs *MRR@10*, *Recall@10*, and *F1-Score* to measure both retrieval effectiveness and answer quality.

FinQA. We use a variant from FinQA that is focused on retrieval from T^2-RAGBench (Strich et al., 2025). The dataset comprises approximately 6,200 finance-related question–answer pairs derived from company reports and other financial documents. Each question is associated with a single financial document containing a combination of text and tables. On average, there are roughly three QA pairs per document, resulting in a total of 2,110 documents for retrieval. We used a modified version of the training split, reformulating questions to make them more suitable for RAG. For reasons of anonymity, the original paper is not cited in this submission. For evaluation, we employ *MRR@10*, *Recall@10*, and *Number Match (NM)*, a metric specifically designed to compare numerical values, like Exact Match, up to the second decimal place.

BioASQ. The dataset (Krithara et al., 2023) contains approximately 4,000 biomedical question–answer pairs derived from scientific literature in the life sciences. It is designed to evaluate factual biomedical question answering in combination with semantic retrieval. The dataset is updated annually, and the version used in our evaluation is BioASQ11. Each QA pair includes multiple reference documents, over 34k in total, making it particularly challenging for the retriever to locate the relevant information. Questions are grouped into four types: list, yes/no, factoid, and summary, each requiring different response formats. For evaluation, we use *MAP* and *Recall@10* for retrieval, and *F1-Score* for generation.

B Metrics Overview

Metric	Description
Generator Metrics	
Exact Match	Checks whether the generated text exactly matches the reference answer without any differences in wording or formatting.
Number Match (Anonym)	Verifies whether numeric values in the generated text are approximately equal to the reference values within a small tolerance.
BLEU (Papineni et al., 2002)	Measures the overlap of n-grams between generated and reference text, emphasizing precision of matching word sequences.
ROUGE (Lin, 2004)	Measures overlap of n-grams and longer sequences, often used to evaluate recall of important content.
GLEU (Mutton et al., 2007)	Variant of BLEU that balances recall and precision, providing a more symmetric similarity measure.
Precision (Sokolova and Lapalme, 2009)	Fraction of generated tokens that are correct compared to the reference text.
Recall (Sokolova and Lapalme, 2009)	Fraction of reference tokens that are successfully generated by the model.
F1 (Sokolova and Lapalme, 2009)	Harmonic mean of precision and recall, balancing correctness and completeness.
Retrieval Metrics	
Mean Reciprocal Rank (MRR) (Voorhees, 1993)	Evaluates how highly the first relevant document is ranked by averaging reciprocal ranks over multiple queries.
Mean Average Precision (MAP) (Voorhees, 1993)	Computes the mean of average precision values across queries, reflecting ranking quality over all relevant documents.
nDCG (Järvelin and Kekäläinen, 2002)	Measures ranking quality by assigning higher importance to relevant documents appearing at top positions.
Recall@k (Robertson and Zaragoza, 2009)	Measures the proportion of relevant documents retrieved within the top-k ranked results.
HitRate@k (Li et al., 2019)	Indicates whether at least one relevant document appears within the top-k retrieved results.
LLM-Based Metrics	
Answer Relevance (Es et al., 2024b)	Evaluates whether the generated answer is relevant and directly addresses the given query.
Answer Faithfulness (Es et al., 2024b)	Measures whether the generated answer is factually consistent with the provided context.
Non-Answer Critic (Es et al., 2024b)	Detects responses that fail to answer the query or contain only irrelevant information.
Answer Similarity (Es et al., 2024b)	Measures semantic similarity between the generated answer and a reference answer or alternative outputs.
Context Recall (Es et al., 2024b)	Evaluates whether the retrieved context contains the information required to produce the answer.
Context Precision (Es et al., 2024b)	Measures how much of the retrieved context is actually relevant for answering the query.

Table 3: Overview of the most used RAG evaluation metrics that we implemented in *EncouRAGe*.

REFSafE: A RAG-Enabled Framework for Predictive Risk Analysis and Automated Safety Report Generation in Mission-Critical Environments

Sanjay Das[*]
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
dass3@ornl.gov

Ran Elgedawy[*]
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
elgedawyr@ornl.gov

Ethan Seefried
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
seefriedej@ornl.gov

Ryan Burchfield
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
burchfieldra@ornl.gov

Gavin Wiggins
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
wigginsg@ornl.gov

Dana Hewit
Savannah River National Laboratory
Aiken, South Carolina, USA
dana.hewit@srnl.gov

Sudarshan Srinivasan
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
srinivasans@ornl.gov

Prasanna Balaprakash
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
pbalapra@ornl.gov

Robert Patton
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
pattonrm@ornl.gov

Todd Thomas
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
thomastm@ornl.gov

Tirthankar Ghosal
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
ghosalt@ornl.gov

Abstract

Operational safety in mission-critical environments requires AI systems that are accurate, interpretable, and resistant to hallucination. We present an agentic Retrieval-Augmented Generation (RAG) framework, REFSafe, for grounded hazard analysis and automated safety report generation. The system integrates Large Language Models (LLMs) with structured operational data, historical incident repositories, policy documents, and external authoritative sources. Through iterative agentic reasoning, the framework retrieves, verifies, and synthesizes evidence prior to generation, enforcing citation-backed outputs with explicit source attribution (documents, links, and prior events) to ensure traceability and trust.

To mitigate hallucinations and unsupported claims, all risk assessments and forecasts are constrained to retrieved evidence, with confidence signals derived from retrieval relevance and source consistency. A transparent pipeline enables subject matter experts (SMEs) to validate predictions, and provide structured feedback, forming a continuous performance calibration loop. Preliminary deployment demonstrates improved reliability in hazard detection and safety/vulnerability report generation. This work advances trustworthy, evidence-grounded AI for predictive safety intelligence in mission-critical operations.

[*]Both authors contributed equally to this research.

1 Introduction

Operational safety at mission-critical and high-risk facilities demands proactive, rigorous, and continuously adaptive hazard identification. Even minor gaps in assessment can lead to catastrophic consequences, including loss of critical infrastructure, environmental damage, and human casualties. Environments such as nuclear facilities, high-voltage power laboratories, and chemical research sites inherently operate at the edge of technical and operational risk due to hazardous materials, extreme energy densities, and tightly coupled system dependencies. Despite established safety programs and regulatory controls, incidents persist—often attributable to incomplete hazard assessments, fragmented safety information distributed across heterogeneous systems, and limited awareness of historical failure patterns.

Traditional safety analysis workflows—such as hazard identification, risk indemnification assessment, safety control evaluation, and root cause analysis—remain largely manual, expert-driven, and policy-search intensive. These processes are time-consuming, often requiring days or weeks, and are susceptible to cognitive bias and oversight of emerging or non-obvious risks. While Natural Language Processing (NLP) techniques have been applied to safety documentation, including HAZOP reports (Feng et al., 2021), chemical accident databases (Single et al., 2020), and incident knowledge graphs (Wang et al., 2022), these approaches typically address isolated subtasks rather than providing integrated, end-to-end safety decision support. Even recent benchmarking efforts such as Lab-Safety Bench (Zhou et al., 2025) demonstrate that state-of-the-art

Proceedings of the First Workshop on Multilingual Report Generation via Retrieval Augmented Generation (RAG4Reports), pages 47–56

LLMs fail to exceed 70% accuracy in laboratory hazard identification, underscoring the limitations of standalone LLM reasoning in safety-critical contexts.

Recent advances in LLMs have demonstrated strong capabilities in extracting insights from unstructured safety narratives. For example, Smetana et al. (Smetana et al., 2024) applied GPT-3.5 to OSHA highway construction accident reports by clustering incident narratives using SBERT embeddings and prompting the model to summarize root causes, successfully identifying major accident categories and contributing factors. This approach revealed major accident categories (e.g. heat-related, struck-by) and their contributing factors, demonstrating that LLMs can augment traditional statistics with insights useful for prevention. Similarly, Baek et al. (Baek et al., 2025) propose a RAG-based framework where an LLM generates safety guidance by retrieving and referencing similar accident cases from a database. Their "automated safety risk guidance" model uses retrieval to ground the LLM's advice in actual past incidents. Other studies compare LLMs to human analysts in safety-risk tasks. For instance, Esposito and Palagiano (Esposito et al., 2024) evaluated fine-tuned and RAG-augmented LLMs on mission-critical security risk analysis. They found that LLMs were faster and more "actionable" than experts, and that RAG-assisted models had the lowest hallucination rates and uncovered hidden risks most comprehensively. In essence, fine-tuned LLMs gave more accurate answers while RAG models covered a broader scope of potential hazards. Some work examines LLMs on domain-specific safety narratives. Mumtarin et al. (Mumtarin et al., 2023) compared ChatGPT, Bard, and GPT-4 on U.S. crash reports: on simple yes/no questions (e.g. "Work-zone involved?"), the models agreed >85% of the time, but on complex queries (e.g. collision type) agreement fell to 35%. This suggests LLMs can extract clear facts (work-zone involvement: 96% consensus) but struggle with nuanced inferences (collision manner: 35% consensus). Taken together, these studies indicate that LLM/NLP tools can enhance text-based safety analyses (e.g. clustering narratives, extracting causal factors) and that retrieval-augmented techniques in particular help ground LLM outputs in actual safety data. However, these efforts remain fragmented and do not constitute a unified, grounded framework for predictive hazard intelligence for operational decision support.

The paradigm of RAG has fundamentally changed how LLMs interact with external knowledge, moving beyond their pre-trained parameters to incorporate real-time, domain-specific information (Lewis et al., 2020). While RAG has seen broad application in improving factual accuracy and reducing hallucinations in general knowledge-intensive NLP tasks, its specialized application in safety-critical domains represents a crucial area of ongoing research and development. Traditional RAG systems, by their nature, enhance an LLM's ability to provide more accurate and relevant responses by retrieving pertinent documents from a knowledge base. While there is a growing body of work on using AI for incident analysis (as discussed in the previous section) and general safety recommendations, the integration of RAG specifically to synthesize actionable safety insights from vast, unstructured historical incident reports is a significant advancement. This involves not just retrieving information, but semantically understanding complex safety narratives, identifying subtle precursors, and generating precise, context-aware recommendations that align with established safety protocols.

In this work, we introduce a unified, RAG-based framework, REFSafE, for predictive hazard identification in mission-critical environments. Our system integrates advanced embedding models, retrieval re-ranking, structured operational data, historical incident repositories, regulatory policies, and external sources within an iterative reasoning architecture. This fusion of advanced information retrieval and LLM-driven reasoning shifts safety management from a reactive, retrospective process toward a predictive and anticipatory paradigm. By operationalizing lessons learned at scale and transforming fragmented historical data into actionable foresight, the proposed approach enables earlier hazard detection, informed work planning, and systematic prevention of high-consequence incidents. In doing so, it advances the development of transparent, evidence-grounded AI systems capable of meeting the reliability demands of mission-critical safety operations.

Our contributions are as follows:

- We present the REFSafE system, a novel, multi-agent RAG pipeline for proactive hazard forecasting in high-risk operational environments.
- We propose a smart RAG system for intelligent retrieval of relevant incidents data from a vector database.
- We integrate Standards-Based Management System (SBMS) and external control policies for providing appropriate mitigation approaches for identified critical hazards.
- REFSafE generates comprehensive vulnerability reports combining all the relevant past events, critical missing hazards and controls towards presenting an overall risk profile of the corresponding work plan.

2 Historical Incident Repository

Past incidents are not merely records of failure—they are encoded lessons in system vulnerability, human factors, and control breakdowns. Yet in many mission-critical environments, this knowledge remains fragmented across reports, logs, and policy archives, limiting its operational value. A centralized, structured repository transforms dispersed safety narratives into actionable intelligence, enabling grounded risk reasoning, pattern discovery, and hazard forecasting.

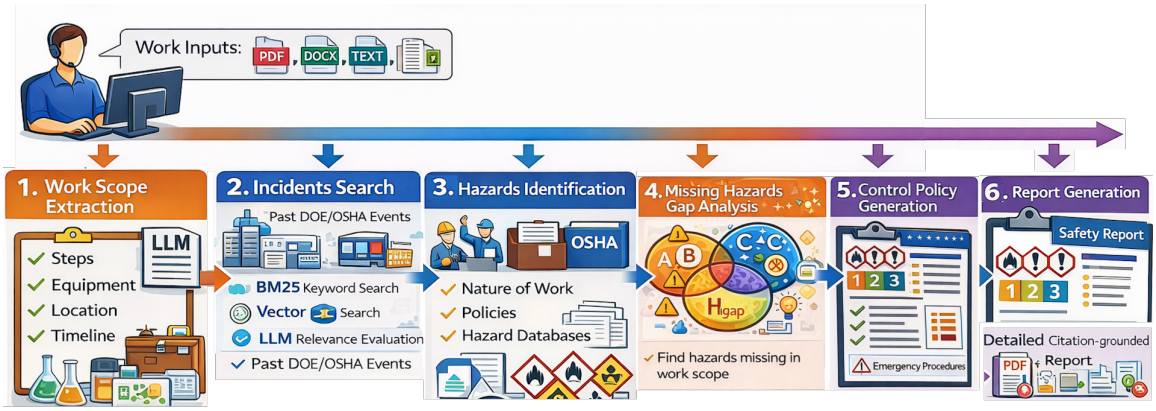

Figure 1: REFSafE system architecture. 1. Researcher uploads/inputs a work plan/order; 2. System queries multiple incidents databases in parallel for incidents 3. Identifies relevant hazards to the work scope and the past events. 4. Analyzes missing hazards gap in work plan, 5. Identifies specific controls and policies and 6. produces a comprehensive safety report.

2.1 Dataset

Our corpus combines four safety and incident reporting datasets from multiple DOE institutions (*e.g.* national laboratories), totaling over **65,000** documents spanning over three decades (1990–2025) of safety-related reporting. These sources include structured reports, narrative-style lessons-learned records, occupational injury and illness reports, and localized site-specific reports. Table 1 showcases example documents from each of the corresponding datasets.

2.1.1 ORPS

The Occurrence Reporting and Processing System (ORPS) dataset is provided by the United States Department of Energy (DOE) and documents any event that may affect public or DOE worker health and safety, the environment, national security, DOE's safeguards and security interests, the functioning of DOE facilities, or the Department's reputation (U.S. Department of Energy, 2003).

We sourced a total of 24,431 documents from the ORPS database, covering reports submitted between 1990 and 2024. Each document consists of two main components: `text` and `metadata`. The `text` field is further structured into three subfields: `title`, `summary`, and `keywords`. The metadata includes attributes such as occurrence date, facility name, reporting level, reporting criteria, process and system tags, outcome categories, and contractor information.

2.1.2 OPEXShare

The OPEXShare dataset originates from the DOE Operating Experience (OPEX) platform, which facilitates the sharing of lessons learned and best practices across DOE sites and contractors (Joseph, 2025). Each document includes a `text` field composed of a `title` and a body, typically describing the context, contributing factors, and corrective actions associated with a safety-related event. In total, we sourced 8,753 documents from the OPEXShare portal. These records serve as informal, narrative-style supplements to more structured reporting systems such as ORPS and CAIRS.

2.1.3 CAIRS

The Computerized Accident/Incident Reporting System (CAIRS) is maintained by the U.S. DOE and contains reports of occupational injuries and illnesses that occur during operations conducted by DOE or its contractors (Fielding, 1983). Each CAIRS document includes a `text` field, which summarizes the incident, and a rich set of metadata fields such as the date of occurrence, location, organization, contractor, injury type, severity (e.g., lost workdays), and relevant system or process tags. CAIRS provided the largest number of documents in our dataset, contributing a total of 31,109 separate reports.

2.1.4 ORNL

The ORNL dataset consists of internal safety and incident reports sourced from Oak Ridge National Laboratory (ORNL). These documents detail laboratory-specific occurrences such as procedural deviations, and equipment failures. Each entry includes a `text` field—typically a narrative description of the event, and `metadata` fields such as site, outcome tags, keywords, and lessons learned. In total, we collected 814 documents from ORNL's internal reporting system. While more localized in scope than DOE-wide systems, these reports offer valuable insight into site-specific operational hazards and institutional safety practices.

2.2 Dataset Statistics

With all four datasets combined into a single corpus, the final dataset contained 65,107 documents. Each document includes metadata such as event name, date, location, a text summary, and a full body text, among other attributes. The summaries had an average length of 231.63 ± 361.36 words, with a median of 73 words and a maximum of 3,476 words. The full body texts

Table 1: Representative Event Documents from Each Dataset.

Dataset	Textual Document Summary (Truncated)	Selected Metadata
ORPS	**Title:** Failure Of Tank 48's Purge Flow Gauge **Summary:** After the facility had entered a Limiting Condition of Operation (LCO) so that a surveillance could be performed, site personnel conducted a calibration of the Tank 48 Purge Exhaust Flow Gauge and determined that it would not hold pressure. Site maintenance personnel suspect that the gauge's diaphragm failed. A work request was initiated to replace the failed gauge, a Safety Significant Component, and restore compliance with the facility... **Keywords:** 11B - Emergency Management System Failure, 12E - Equipment Degradation/Failure	**Facility:** H Tank Farm (Nuclear Waste Operations/Disposal) **PSO:** Environmental Management **Significance:** Minor Impact **Reporting Level:** Low **Outcome Tags:** Equipment/Structural/Property Damage
OPEXShare	**Title:** Prolonged Repairs Complicate Excavation Hazard Control **Body:** Short-term controls for temporary material hazards may be inadequate for longer repair jobs and should be reevaluated periodically as a job stretches out. The excavated soil was placed on a tarp and, because the job took longer than expected, minor amounts of contamination leached out from under the tarp cover and out beyond the posted radiological control area....	**Report:** Prolonged Repairs Complicate Excavation Hazard Control **Site:** Fluor Hanford **Entity:** DOE Company/Contractor **PSO:** Environmental Management **Date:** 2000-01-05 **Outcome Tags:** Environmental Release (Hazmat, Rad, Water, etc.)
CAIRS	The staff member fell on ice while performing snow removal. He sought evaluation and treatment from the site occupational health care provider. Low back pain s/p fall. Revision 1/15/04: Sacroiliac strain. Cold pack administered....	**Title:** HANDLER/LABORER/HELPER experienced STRAIN to his/her LOWER BACK resulting in 69 lost workdays. **Site:** Pacific Northwest National Lab **Outcome Tags:** Injury, Illness, Medical Treatment; Equipment Damage
ORNL	On September 30, 2013, employees initiated a work activity, under an approved work package and two LO/TO Permits, to replace a condensate tank in Building 3047. Affected employees and the Service Supervisor attached their locks to the lock boxes. After the lock-out of the steam supply, the condensate system was drained and the drain valve left open and work activities commenced. On October 15, 2013, safety staff initiated a review of the LO/TO permits and identified several administrative proceeded...	**Lessons Learned:** None **Date:** 2023-01-28 **Outcome Tags:** Operational Safety Vulnerability, Illness, Med Treatments, or Fatalities Impacts Or Unknown

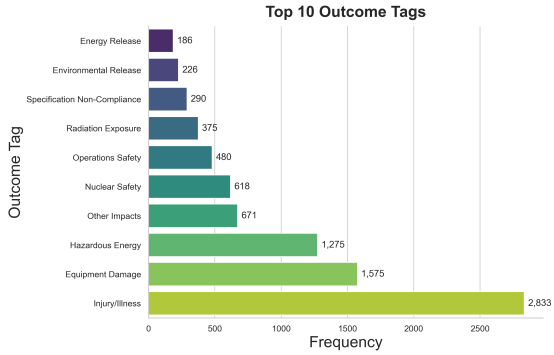

Figure 2: Frequency distribution of the top 10 outcome tags in the dataset. Energy Release (n = 186) and Environmental Release (n = 226) were the least frequent, while Injury and Illness (n = 2,833) was the most prevalent.

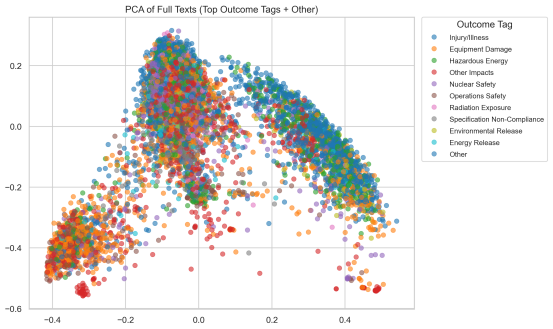

Figure 3: Two-dimensional PCA projection of TF–IDF representations of full-text documents. Points are colored by the top 10 most frequent outcome tags (shortened for clarity), with all remaining categories grouped under Other. The visualization shows clear separation between certain categories, indicating distinctive vocabulary patterns, while others overlap, reflecting shared terminology.

averaged $1,677.22 \pm 3,246.98$ words, with a median of 678 words and a maximum of 63,337 words.

To better understand the distribution of event outcomes, we examined the "outcome tags" associated with each document. Figure 2 presents the ten most frequent outcome tags, with Injury/Illness as the most common and Energy Release and Environmental Release among the least frequent. To examine the semantic organization of the corpus, Figure 3 presents a two-dimensional Principal Component Analysis (PCA) projection of the TF–IDF vector representations derived from the full-text documents. The visualization reveals that several outcome categories form well-separated clusters, indicating distinctive vocabulary and contextual patterns, while other categories exhibit partial overlap due to shared terminology across outcome types. Such insights into the corpus structure enable the analysis to prioritize the critical outcome categories (*e.g.* nuclear safety), thereby guiding the vulnerability assessment toward high-impact cases.

3 System Architecture

The REFSafe workflow is visualized in Figure 1. The system architecture as depicted in Figure 4 follows a microservices design utilizing, FastAPI for the backend server with an exposed API and a StreamLit for the

frontend, which connects to the backend.

3.1 Work Scope Extraction

Work plans or work orders (PDF or free text) are first homogenized via a Python-based pre-processing pipeline. Assume that the raw document be denoted as D_{raw}. A document normalization function $\mathcal{T} : D_{raw} \rightarrow D_{json}$ transforms heterogeneous formats (PDF, DOCX, TXT) into a standardized JSON schema:

$$D_{json} = \{\mathcal{S}, \mathcal{E}, \mathcal{L}, \mathcal{T}, \mathcal{M}\}$$

where $\mathcal{S}$ represents ordered work steps, $\mathcal{E}$ equipment/PPE, $\mathcal{L}$ location metadata, $\mathcal{T}$ timeline/duration constraints, and $\mathcal{M}$ contextual modifiers (environmental, operational conditions).

A schema-constrained LLM (*e.g.* Llama-3.1-8B-Instruct) parser $\mathcal{P}_{LLM}$ is then applied to extract specific work-scope information:

$$(\mathcal{S}, \mathcal{E}, \mathcal{L}, \mathcal{T}) = \mathcal{P}_{LLM}(D_{json})$$

Each work step $s_i \in \mathcal{S}$ is decomposed into atomic, modular actions:

$$\mathcal{S} = \{s_1, s_2, ..., s_n\}, \quad s_i = (a_i, e_i, l_i, t_i)$$

where a_i is the action, e_i the associated equipment, l_i the spatial context, and t_i the temporal constraint.

The Summarization Agent performs this step and the structured output representation enables systematic downstream hazard inference and retrieval.

3.2 Incidents Search

The incident retrieval module searches a rich database of 65,107 reports across DOE repositories using a multi-stage retrieval pipeline (by Smart Retrieval Agent). In the first stage, two independent retrieval channels operate in parallel to maximize recall. A lexical search (R_{BM25}) method captures keyword-based similarity, while a vector search (R_{vec}) identifies semantic similarity using domain-specific embeddings (*e.g.* SFR-Embedding-Mistral). The union of results from both channels forms a broad candidate set of potentially relevant incidents.

Stage 1: Dual Retrieval

$$R_{BM25} = \text{BM25}(q), \quad R_{vec} = \text{VectorSearch}(q)$$

The candidate set is:

$$R_{cand} = R_{BM25} \cup R_{vec}$$

Stage 2: Cross-Encoder Re-ranking

In the second stage, a cross-encoder model performs pairwise relevance evaluation between the work scope representation and each candidate incident. The cross-encoder computes pairwise similarity:

$$S_{ce}(q, r_i)$$

Top-k (k=30) proceed forward. This reranking step evaluates procedural, and contextual alignment, allowing the system to retain only the most relevant candidates.

Stage 3: LLM Relevance Evaluation

In this stage, an LLM (*e.g.* Llama-3.1-8B-Instruct or GPT-4o API (Application Programming Interface) call) evaluates each candidate incident across multiple qualitative dimensions, including relevance (S_{rel}), faithfulness (S_{faith}) to the original report, and contextual quality (S_{ctx}). Each candidate incident r_i is scored on:

$$S_{rel}, \quad S_{faith}, \quad S_{ctx} \in [0, 1]$$

The final ranking combines normalized retrieval (30%), reranking (40%) scores with LLM evaluation scores (40%):

$$S_{final} = 0.3 \cdot \hat{S}_{retrieval} + 0.4 \cdot \hat{S}_{rerank} + 0.3 \cdot \frac{1}{3}(S_{rel} + S_{faith} + S_{ctx})$$

where $\hat{S}_{retrieval}$ is normalized retrieval score and $\hat{S}_{rerank}$ is normalized rerank score. These scores are combined with retrieval signals to produce a final relevance score. Top-5 incidents are returned.

3.3 Hazards Identification

Hazard identification is implemented as a multi-path inference mechanism designed to maximize recall and semantic coverage.

3.3.1 Deterministic rule-based extraction

First, explicitly stated hazards are extracted directly from the structured work scope. These include hazards clearly mentioned in the work plan, such as chemical exposure, fall risk, electrical hazards, or confined space entry. A static hazard identification engine $\mathcal{H}_{plan}$ maps structured elements in D_{json} to hazards using predefined mappings:

$$A = \mathcal{H}_{plan}(D_{json})$$

3.3.2 LLM-driven contextual hazard inference

Second, a contextual inference process uses a large language model to identify latent hazards implied by the procedural context. A prompted reasoning model $\mathcal{H}_{LLM}$ generates hazards conditioned on work scope:

$$B = \mathcal{H}_{LLM}(\mathcal{S}, \mathcal{E}, \mathcal{L}, \mathcal{T})$$

These are hazards not explicitly stated but logically associated with the work activities, equipment, location conditions, or sequencing.

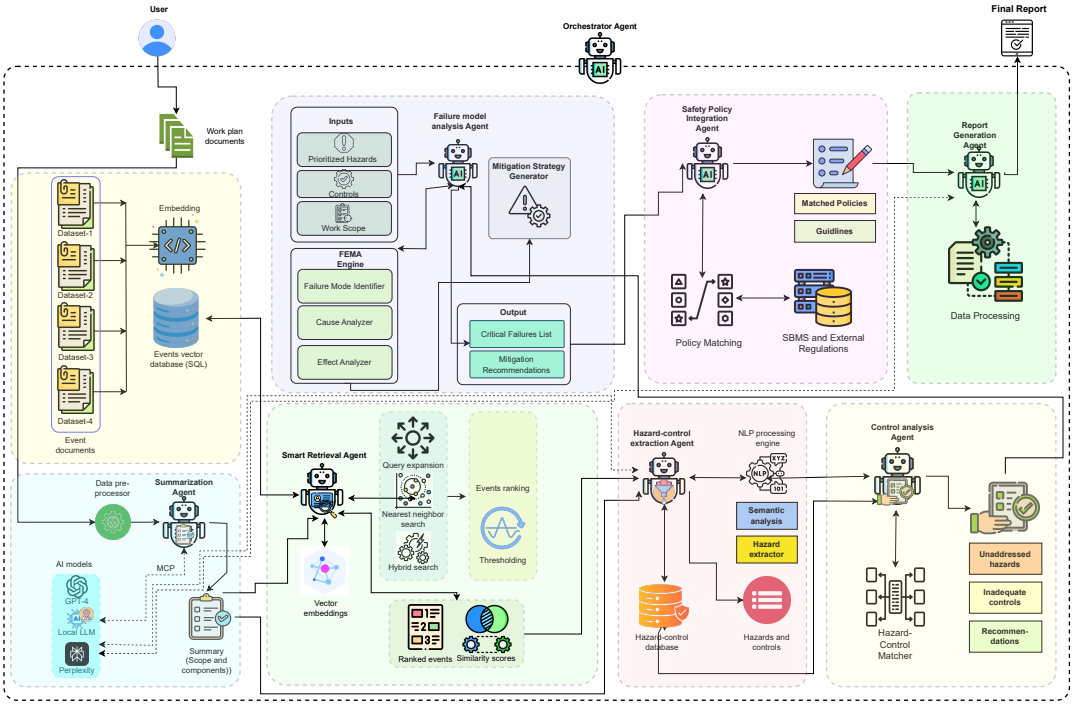

Figure 4: System architecture.

3.3.3 Hazard inference from Past Incidents

Third, hazards are derived from retrieved historical incidents. By analyzing causal factors and lessons learned in past incidents, the system identifies additional hazard patterns relevant to the current work scope.

$$C = \mathcal{H}_{Past}(\mathcal{I}_{\text{top-5}})$$

3.4 Missing Hazards Gap Analysis

The Missing Hazards Gap Analysis stage identifies hazards that were not declared in the original work plan but are historically or contextually associated with similar work. Towards this, a dedicated *Failure Model Analysis Agent* is employed with a domain-specific FEMA engine comprising: Failure Mode Identifier, Cause Analyzer and Effect Analyzer. This subsystem synthesizes a *Critical Failures List*, *causing hazard* and corresponding *Mitigation Recommendations*, which are forwarded for further analysis and policy alignment.

The final inferred hazard space is formed by combining explicitly stated hazards A, incident-derived hazards B, and contextually inferred hazards C. The union produces a comprehensive inferred hazard set:

$$H_{Inc} = A \cup B \cup C$$

The missing hazard set is computed as:

$$H_{gap} = H_{inc} \setminus H_{plan}$$

This set represents hazards historically associated with similar work but not explicitly documented in the current work plan. Such gaps are critical because workers unaware of these hazards may not receive appropriate training, implement required controls.

3.5 Control Policy Identification

For each hazard $h_j \in H_{Inc}$, control retrieval follows two parallel channels.

$$C_j = C_j^{internal} \cup C_j^{external}$$

The first channel retrieves institutional standards ($C_j^{internal}$) from indexed internal Standards-based Management System (SBMS). A cross-encoder similarity model ranks relevant control policy documents based on their alignment with the identified hazard. This, internal controls are retrieved:

$$S_{policy}(q, d_k) = \text{CrossEncoder}(q, d_k)$$

where q is the hazard query and d_k is a candidate SBMS policy document.

The second channel retrieves external best-practice guidance ($C_j^{external}$) through web-based search. This includes regulatory references, industry standards, and publicly available safety recommendations. The combination of internal and external sources ensures comprehensive hazard mitigation coverage.

3.6 Safety Report Generation

All outputs from the previous modules are consolidated into a structured safety report. The report integrates the extracted structured work scope S, explicit and inferred hazards H_{Inc}, retrieved historical incidents R_{top}, lessons learned L, gap analysis results H_{gap}, and recommended control policies C from both internal and external sources. Emergency response procedures and AI-generated safety commentary are also included.

$$\mathcal{R} = \{\mathcal{S}, H_{Inc}, R_{top}, \text{L}, H_{gap}, C\}$$

The final report is a structured and professionally formatted document that can be exported as a PDF artifact. It is suitable for pre-job briefings, regulatory documentation, supervisory review, institutional record keeping, and operational trust.

4 Evaluation and Results

This section details the choice of embedding models and the evaluation results of the different operations of the tool such as events retrieval and vulnerability report generation.

4.1 Platform and Tools

The proposed framework was implemented and evaluated on a system equipped with an NVIDIA A100 80GB GPU. The agent-based architecture was developed in Python using the PydanticAI framework, with retrieval and indexing components implemented via chromadb and LlamaIndex packages. The backend service was deployed using FastAPI with Uvicorn for asynchronous serving, while the interactive user interface was built using Streamlit. For semantic retrieval, the system employs the embedding model "SFR-Embedding-Mistral" and performs cross-encoder reranking using "cross-encoder/ms-marco-MiniLM-L6-v2". We employ the OpenAI API with the GPT-4o model to achieve state-of-the-art performance.

4.2 Evaluation of Embedding Models

To identify the optimal embedding model for REFSafe, we evaluated three high-performing embeddings based on their Hugging Face LLM leaderboard rankings (Face, 2025): SFR-Embedding-Mistral (Meng et al., 2024), OpenAI text-embedding-3-large (OpenAI, 2024), and INF-Retriever-v1 (Yang et al., 2025).

4.2.1 Experiment Setup

We randomly selected 15 ORPS documents and generated 100 QA pairs through manual and AI-assisted annotation as reference sets. Performance metrics included:

1. **Answer Correctness:** Factual agreement with reference answers (RAGAS (Es et al., 2024))

2. **Average Query Time:** Mean latency for retrieval.

4.2.2 Results

Table 2 shows INF-Retriever-v1 achieved the best balance with 68.1% correctness and 22.7s query time. SFR-Embedding-Mistral had 67.1% correctness with the fastest time (19.2s), OpenAI's model had lowest correctness (60.1%) at 20.1s. Based on these results, we selected SFR-Embedding-Mistral for subsequent experiments as it provides balance between correctness and average query time.

Table 2: Answer correctness and average query time for each embedding model.

Model	Correctness (%)	Avg Query Time (s)
SFR-Embedding-Mistral	67.1	19.2
INF-Retriever-v1	68.1	22.7
OpenAI text-embedding-3-large	60.1	20.1

4.3 Retrieval Performance Evaluation

To assess retrieval accuracy without ground-truth datasets, we applied the pooled judgment method (Sparck Jones and Van Rijsbergen, 1975; Voorhees, 2000; Tonon et al., 2015), which builds relevance assessments by merging outputs from multiple retrieval variants (Sanderson and Zobel, 2005). This approach, widely used in large-scale evaluations such as TREC (Harman, 1995; Arguello et al., 2023), is suitable for specialized domains where exhaustive labeling is infeasible.

We compared six retrieval variants: (1) *current_best* – our hybrid system using LLM-generated keywords, document titles, and CrossEncoder reranking; (2) *keywords_only*; (3) *pure_rag* – semantic similarity search; (4) *title_only*; (5) *rule_keywords* – TF-IDF and NER-based extraction; and (6) *extended_keywords* – hybrid with 10 keywords. For five test work plans, each variant retrieved the top 10 documents; up to 25 unique results per query were manually annotated on a three-point relevance scale (0–2). We employ the following metrics:

1. **Precision@5 (P@5)**: Quantifies the proportion of retrieved documents in the top 5 positions that are relevant.

$$\text{P@5} = \frac{|\text{Relevant Documents} \cap \text{Retrieved Top 5}|}{5} \tag{1}$$

2. **Recall@5 (R@5)**: Measures the fraction of the total relevant document set captured within the first 5 results.

$$\text{R@5} = \frac{|\text{Relevant Documents} \cap \text{Retrieved Top 5}|}{|\text{Total Relevant Documents}|} \tag{2}$$

3. **F1-Score@5 (F1@5)**: Provides the harmonic mean of Precision@5 and Recall@5, offering a balanced measure of retrieval performance at a narrow rank.

$$\text{F1@5} = 2 \cdot \frac{\text{P@5} \cdot \text{R@5}}{\text{P@5} + \text{R@5}} \tag{3}$$

As shown in Table 3, the hybrid system achieved the best results (F1@5 = 0.384 $\pm$ 0.080). The *title_only* variant performed closely (F1@5 = 0.369), highlighting the discriminative value of work plan titles.

4.4 Generated Report Evaluation

We assessed report quality using an LLM-as-Judge framework, following prior work showing that large language models can reliably evaluate text quality (Zheng et al., 2024; Dubois et al., 2024; Kim et al., 2024).

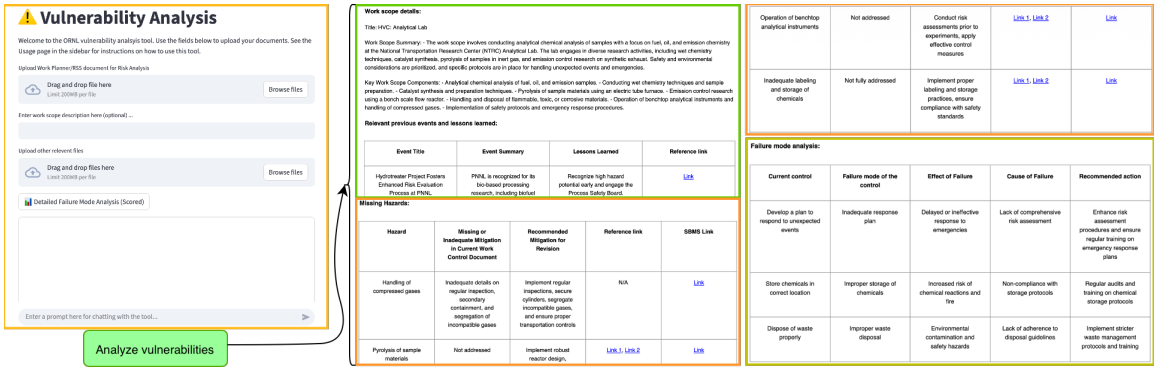

Figure 5: REFSafe UI overview.

Table 3: Retrieval Performance Evaluation Results Across Six System Variants.

System Variant	P@5	R@5	F1@5
RAG + keywords	0.920 ± 0.160	0.243 ± 0.053	0.384 ± 0.080
Title only	0.880 ± 0.160	0.234 ± 0.058	0.369 ± 0.086
Rule + keywords	0.800 ± 0.219	0.212 ± 0.068	0.334 ± 0.104
Keywords only	0.760 ± 0.196	0.196 ± 0.042	0.311 ± 0.069
Extended keywords	0.720 ± 0.271	0.184 ± 0.065	0.293 ± 0.104
Pure RAG	0.680 ± 0.371	0.177 ± 0.101	0.281 ± 0.158

Twenty randomly selected reports were evaluated with GPT-4 as the judge model, each compared to its corresponding work plan. Evaluations covered five criteria: *clarity* (use of technical terms), *completeness* (coverage of hazards, lessons, and mitigations), *usefulness* (support for decision-making), *accuracy* (factual grounding), and *specificity* (relevance to the work plan). Each criterion was rated on a 5-point Likert scale (1 = Poor, 5 = Excellent), with both numeric scores and textual justifications.

Table 4: Mean LLM-as-Judge ratings for generated risk reports (Likert 1–5).

Dimension	Mean Rating
Clarity	4.0
Completeness	3.0
Usefulness	4.0
Accuracy	5.0
Specificity	3.0
Overall	**3.8**

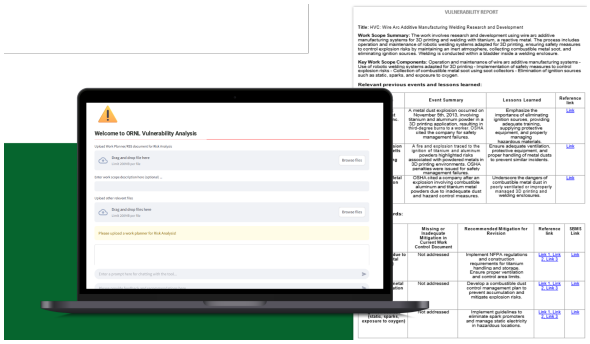

Figure 6: Tool operation example.

Accuracy received a perfect score (5.0), with GPT-4 noting that reports "consistently base hazard identifications on documented events and established safety protocols." This confirms that retrieval grounding effectively prevents unsupported claims.

Clarity* and *usefulness also scored high (4.0), reflecting precise terminology and actionable recommendations. However, lower scores for *completeness* and *specificity* (3.0 each) indicate that while major hazards were captured, some nuanced or facility-specific risks were missed. Future improvements should focus on deeper context extraction and more targeted hazard analysis.

5 Demonstration

REFSafe's user interface is designed to streamline human–AI interaction in all stages of risk analysis, focusing on transparency, traceability, and minimal analyst effort. After setup, users are directed to the main interface where submissions, results, and feedback are managed through a unified workflow, as visible in Fig. 5.

Submission Workflow: Users initiate the vulnerability analysis workflow by uploading a work plan/order with an optional scope information, and then trigger processing with a one-click action via the "Analyze vulnerabilities" button. This workflow supports both individual entries and batch uploads for multiple items.

Outputs: Upon submission, the interface analyses the work plan in the back-end via a complex RAG-enabled LLM-based risk analysis and report generation workflow, and outputs an initial safety report. Additionally the tool provides a 'Chat' option for users to interact with the tool with specific queries. Once satisfied, the users can generate a final report by clicking a "Generate final report" button. The report can be exported as a 'pdf' and downloaded.

Feedback and Review: Users can provide structured feedback through a simple form submitted via a 'Submit Feedback' button. This input is stored for audit and future model refinement.

The application's frontend provides a cohesive, context-aware experience, guiding users seamlessly from submission to analysis, report generation, feed-

back, and export, ensuring both operational efficiency and traceable process.

6 Conclusion

In this work, we proposed an end-to-end agentic workflow, REFSafe, where an SME submits a complex, high-risk work plan for automated risk assessment and hazard forecasting. REFSafe generates interpretable risk profiles and vulnerability reports, enabling experts to identify overlooked hazards and refine mitigations.

Technical Impact: REFSafe advances predictive safety in high-consequence domains by making LLM-based risk analysis *auditable*, *traceable*, and *adaptive* through agentic orchestration.

References

Jaime Arguello, Samarth Bhargav, Fernando Diaz, Evangelos Kanoulas, and Bhaskar Mitra. 2023. Overview of the trec 2023 tip-of-the-tongue track. In *The Thirty-Second Text REtrieval Conference Proceedings (TREC 2023), Gaithersburg, MD, USA, November*, pages 14–17.

Seungwon Baek, Chan Young Park, and Wooyong Jung. 2025. Automated safety risk management guidance enhanced by retrieval-augmented large language model. *Automation in Construction*, 176:106255.

Yann Dubois, Balázs Galambos, Percy Liang, and Tatsunori B Hashimoto. 2024. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*.

Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. 2024. RAGAs: Automated evaluation of retrieval augmented generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 150–158, St. Julians, Malta. Association for Computational Linguistics.

Matteo Esposito, Francesco Palagiano, Valentina Lenarduzzi, and Davide Taibi. 2024. Beyond words: On large language models actionability in mission-critical risk analysis. In *Proceedings of the 18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM '24, page 517–527, New York, NY, USA. Association for Computing Machinery.

Hugging Face. 2025. Embedding models leaderboard. Accessed: 08-2025.

Xudong Feng, Shengnan Zhong, Qianlin Li, Laibin Zhang, and Li Ma. 2021. Application of natural language processing in HAZOP reports. *Process Safety and Environmental Protection*, 155:41–48.

JR Fielding. 1983. Computerized accident/incident reporting system (cairs) update. Technical report, EG and G Idaho, Inc., Idaho Falls (USA).

Donna Harman. 1995. Overview of the second text retrieval conference (trec-2). *Information Processing & Management*, 31(3):271–289.

Taiwo Joseph. 2025. Optimizing operational expenditure (opex) in offshore support vessel (osv) operations: A benchmarking study against international best practices.

Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, and 1 others. 2024. Prometheus: Inducing fine-grained evaluation capability in language models. *arXiv preprint arXiv:2310.08491*.

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, Red Hook, NY, USA. Curran Associates Inc.

Rui Meng, Ye Liu, Shafiq Rayhan Joty, Caiming Xiong, Yingbo Zhou, and Semih Yavuz. 2024. Sfr-embedding-mistral:enhance text retrieval with transfer learning. Salesforce AI Research Blog.

Maroa Mumtarin, Md Samiullah Chowdhury, and Jonathan Wood. 2023. Large language models in analyzing crash narratives – a comparative study of chatgpt, bard and gpt-4. *Preprint*, arXiv:2308.13563.

OpenAI. 2024. Openai text embedding 3 large. Accessed: 2025-10-28.

Mark Sanderson and Justin Zobel. 2005. Information retrieval system evaluation: effort, sensitivity, and reliability. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 162–169.

Johannes I Single, Eva-Maria Schmidt, and Jörn Denecke. 2020. Knowledge acquisition from chemical accident databases using an ontology-based method and natural language processing. *Safety Science*, 129:104747.

Mason Smetana, Lucio Salles de Salles, Igor Sukharev, and Lev Khazanovich. 2024. Highway construction safety analysis using large language models. *Applied Sciences*, 14(4).

Karen Sparck Jones and Cornelis Joost Van Rijsbergen. 1975. Report on the need for and provision of an "ideal" information retrieval test collection. *British Library Research and Development Department*.

Alberto Tonon, Gianluca Demartini, and Philippe Cudré-Mauroux. 2015. Pooling-based continuous evaluation of information retrieval systems. *Information Retrieval Journal*, 18(5):445–472.

U.S. Department of Energy. 2003. *DOE M 231.1-2: Occurrence Reporting and Processing of Operations Information*. Approved: 08-19-03.

Ellen M Voorhees. 2000. Variations in relevance judgments and the measurement of retrieval effectiveness. *Information processing & management*, 36(5):697–716.

Chunling Wang, Shaojun Wu, Jianhui Guo, Qi Wang, Ruidong Zhao, and Shuicheng Tian. 2022. Using text mining to establish knowledge graph from accident/incident reports in risk assessment. *Expert Systems with Applications*, 207:117943.

Junhan Yang, Jiahe Wan, Yichen Yao, Wei Chu, Yinghui Xu, and Yuan Qi. 2025. inf-retriever-v1 (revision 5f469d7).

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2024. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36.

Yujun Zhou, Jingdong Yang, Yifan Huang, Kehan Guo, and 1 others. 2025. Benchmarking large language models on safety issues in scientific laboratories. *Nature Machine Intelligence*. ArXiv:2410.14182.

ORCHID: Orchestrated Retrieval-Augmented Classification of High-Risk Property with Intelligent Decision-Making

Sanjay Das*
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
dass3@ornl.gov

Maria Mahbub*
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
mahbubm@ornl.gov

Vanessa Lama*
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
lamav@ornl.gov

Brian Starks
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
starksbm@ornl.gov

Christopher Polchek
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
polchekcl@ornl.gov

Saffell Silvers
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
silverssc@ornl.gov

Lauren Deck
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
decklm@ornl.gov

Prasanna Balaprakash
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
pbalapra@ornl.gov

Robert Patton
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
pattonrm@ornl.gov

Tirthankar Ghosal
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
ghosalt@ornl.gov

Abstract

High-Risk Property (HRP) classification is critical at U.S. Department of Energy (DOE) sites, where inventories include sensitive and often dual-use equipment. Compliance must track evolving rules designated by various export control policies to make transparent and auditable decisions. Traditional expert-only workflows are time-consuming, backlog-prone, and struggle to keep pace with shifting regulatory boundaries. We propose ORCHID, a modular agentic framework for HRP classification that pairs retrieval-augmented generation (RAG) with human oversight to produce policy based outputs that can be audited. Small cooperating agents—retrieval, description refiner, classifier, validator, and feedback logger—coordinate via agent-to-agent messaging and invoke tools through the Model Context Protocol (MCP) for model-agnostic on-premise operation. The interface follows an "Item to Evidence to Decision" loop with step-by-step reasoning, on-policy citations, and append-only audit bundles (run-cards, prompts, evidence). In preliminary tests on real HRP cases, ORCHID improves accuracy and traceability over a non-agentic baseline while deferring uncertain items to Subject Matter Experts (SMEs). The demonstration shows single item submission, grounded citations, SME feedback capture, and exportable audit artifacts—illustrating a practical path to trustworthy LLM assistance in sensitive DOE compliance workflows.

1 Introduction

The classification of high-risk properties (HRPs) is a mission-critical task for U.S. DOE sites and national laboratories, with implications for national security, export control, and regulatory compliance (Fergusson and Kerr, 2013). Determining whether a property falls under sensitive categories such as the U.S. Munitions List (USML), Commerce Control List (CCL), or Nuclear Regulatory Commission (NRC) regulations requires careful, context-sensitive reasoning. Traditionally, this process has been manual, labor-intensive, and reliant on subject matter expertise—making it slow, error-prone, and difficult to scale.

As property portfolios grow in volume and complexity, there is a growing need for intelligent systems that can augment human decision-making in this domain. While conventional machine learning (ML) and single-prompt large language models (LLMs) offer some automation potential to assist in HRP classification, they are fundamentally limited by issues such as lack of interpretability, inability to incorporate dynamic rule changes, and poor responsiveness to expert correction. Moreover, tightly coupled architectures make it hard to adapt or debug such systems when requirements shift.

Early efforts to automate export-control and security classification relied on rules and ontologies curated by subject-matter experts (sme), typically wrapping the eCFR United States Munitions List (USML)

*Authors contributed equally to this research.

Proceedings of the First Workshop on Multilingual Report Generation via Retrieval Augmented Generation (RAG4Reports), pages 57–64

Figure 1: ORCHID Workflow: An SME inputs model, vendor, item and description of the asset; the system refines the property description, queries multiple policy databases, retrieves relevant information, classifies the asset, validates the classification and reasoning steps and produces the final response, which gets approved by the SME.

[1], eCFR Nuclear Regulatory Commission (NRC) [2], and the eCFR Commerce Control List (CCL) [3] into machine-readable taxonomies(Li and Chen, 2020; Clark, 2008). These systems improved consistency but struggled with ambiguous cross-category items and frequent rule changes. Recent research has moved toward knowledge-centered and ontology-driven modeling of security/export-control concepts, enabling richer reasoning over product descriptions and technical attributes (Rao et al., 2009). For example, ontology-based security/export-control classification approaches demonstrate that standardized concept graphs can reduce ambiguity and support explainable labeling (Rzepka and Obayashi, 2025), though coverage gaps remain for rapidly evolving technologies (e.g., advanced semiconductors, dual-use AI) (Parizadehgan, 2025).

Retrieval-augmented generation (RAG) has become a dominant strategy for keeping models aligned to authoritative texts and reducing hallucinations in law and governance applications (Reuter et al., 2025; Huang et al., 2025). Industry and academic reports alike emphasize dynamic retrieval from up-to-date regulatory repositories and explicit citation in output. Legal-tech guidance and empirical frameworks such as "dynamic legal RAG,"(Ajay Mukund and Easwarakumar, 2025) Gov-RAG(Yu and Chen, 2025), and SemRAG (Zhong et al., 2025) all report improvements in factuality and traceability when generation is grounded in statutes, regulatory notices, and agency FAQs. These capabilities are essential for export-control determinations.

Recent research in AI has explored various methods for combining human expertise with machine learning models. Active learning and human-in-the-loop systems have been widely used in tasks where expert feedback can help refine models, especially in high-stakes domains like medical diagnostics and security (Ruengsurat et al., 2025; Wang et al., 2024). Additionally, retrieval-augmented models like RAG have shown promise in tasks that require both context-specific information retrieval and generation, such as legal document review and scientific research assistance. However, few systems focus on unifying all these elements while integrating SME feedback in real-time to refine predictions. Furthermore, the domain of classifying high-risk properties for national labs requires both high accuracy and clear explanations of predictions, which presents unique challenges for AI models. Our work aims to address this gap by using a human-in-the-loop approach combined with RAG to improve the classification process.

We present *ORCHID: Orchestrated Retrieval-augmented Classification with Human-in-the-loop Intelligent Decision-making*. ORCHID is a modular, multi-agent system designed to deliver explainable and adaptable HRP classifications, guided by expert feedback and grounded in regulatory source material. At the core of ORCHID is an agentic architecture, where specialized agents perform decomposed tasks such as retrieval, reasoning, validation, and feedback integration. Agents interact via an agent-to-agent (A2A) messaging layer, enabling coordination without monolithic prompts or persistent memory (Ray, 2025). To support modular planning and tool invocation, ORCHID implements a lightweight abstraction called the Model Context Protocol (MCP) (Hou et al., 2025). MCP introduces stateless adapters around external models or tools (e.g., dense retrievers, BM25 indexes, LLMs), all accessible through a common invoke(query, params) interface. This decoupling allows the system to flexibly swap components, combine signals through strategies like reciprocal rank fusion (RRF), and maintain clean separation between retrieval, generation, and orchestration logic.

Unlike traditional LLM pipelines, ORCHID enables adaptive learning, by incorporating SME feedback into ongoing refinement; modular, testable components, supporting rapid iteration and system evolution; operational gains, including reduced human burden, improved classification accuracy, and regulatory auditability.

2 Policy Repository

ORCHID restricts search to a versioned policy corpus (eCFR USML, NRC, CCL, plus EAR99 guidance). Policy text is chunked with stable section IDs and embedded with *mxbai-embed-large-v1*(Lee et al., 2024)(Li and Li, 2023). We maintain a hybrid index: BM25 over nor-

[1]https://www.ecfr.gov/current/title-22/chapter-I/subchapter-M/part-121

[2]https://www.ecfr.gov/current/title-10/chapter-I

[3]https://www.ecfr.gov/current/title-15/subtitle-B/chapter-VII/subchapter-C/part-774

malized text for lexical matches and a vector index for semantic matches and results are combined with reciprocal rank fusion. The Vector Store tool (MCP) encapsulates this. Agents pass a query object (with/without description, top-k), and receive a ranked list of snippets with section IDs, confidence. A small citation packer filters to minimally sufficient spans that the model must cite verbatim.

3 System Architecture

The system architecture of ORCHID framework is visualized in Figure 2.

The framework is designed as a modular, multi-agent, retrieval-augmented reasoning system that performs auditable policy-grounded classification with minimally structured user inputs. The architecture decomposes the end-to-end decision process into formally defined transformation stages, each implemented by an independent agent and coordinated by a deterministic orchestrator. The primary design objectives are: (i) strict grounding in authoritative policy sources, (ii) deployment-aware security compliance, (iii) explainable reasoning with citation traceability, and (iv) auditable execution with reproducible metadata tracking.

3.1 Problem Formulation

Here, we formalize the high-risk property classification problem. Let the initial structured user input be defined as:

$$\mathcal{I}_0 = \{V, E, M, d_0\} \tag{1}$$

where V denotes the Manufacturer or Vendor name, E denotes the Equipment or Service identifier, M denotes the Model number (optional), and d_0 represents a short free-text description of the property. The objective of the system is to compute a validated classification decision $\mathcal{D}_f$ grounded in policy evidence.

The complete transformation pipeline is expressed as:

$$\mathcal{D}_f = \mathcal{A}_{VR} \circ \mathcal{A}_{HRP} \circ \mathcal{A}_{IR} \circ \mathcal{A}_{DR}(\mathcal{I}_0) \tag{2}$$

where $\mathcal{A}_{DR}$, $\mathcal{A}_{IR}$, $\mathcal{A}_{HRP}$, and $\mathcal{A}_{VR}$ denote the Description Refiner, Information Retrieval, Classification, and Validation agents, respectively.

3.2 Description Refinement Layer

The Description Refiner (DR) agent performs semantic normalization and ambiguity resolution using a large language model (LLM). This is needed to ensure accurate understanding of the user query for the downstream task. It transforms the raw input $\mathcal{I}_0$ into a canonical structured representation:

$$\mathcal{I}_1 = f_{DR}(\mathcal{I}_0; \theta_{LLM}) \tag{3}$$

where θ_{LLM} denotes the model parameters.

The refinement process includes entity normalization, terminology expansion, contextual disambiguation, and optional clarification queries to the user. The refined representation is:

$$\mathcal{I}_1 = \{V', E', M', d_1\} \tag{4}$$

A deployment-dependent security constraint governs external augmentation to ensure sensitive data protection:

$$\text{OnPrem} \Rightarrow \neg\text{ExternalWebAccess} \tag{5}$$

$$\text{OffPrem} \Rightarrow \text{ControlledWebAccess}(V, E, M) \tag{6}$$

This ensures that sensitive environments prohibit outbound search operations.

3.3 Hybrid Information Retrieval Layer

The Information Retrieval (IR) agent constructs hybrid lexical-semantic queries over a policy-scoped corpus. This step is crucial as it provides relevant policy context for the downstream classification task and ensures evidence grounding. Query generation is defined as:

$$Q = g(V', E', M', d_1) \tag{7}$$

Retrieval proceeds via two complementary channels:

Lexical Retrieval (Sparse Retrieval) : This method relies on token-based matching, looking for exact words, phrases, or their variations within a document. Although, it is excellent at finding precise, rare terminology, proper nouns, and exact phrases, it cannot handle rephrases/synonyms.

$$S_{lex} = \text{BM25}(Q) \tag{8}$$

Dense Semantic Retrieval : This method maps the user query into a dense, low-dimensional vector (an embedding) using a pre-trained neural network. Although, it captures the underlying intent and semantic meaning, enabling it to match synonyms, paraphrases, and context-dependent queries, it struggles with highly specific jargon, rare technical acronyms, or exact entity matching.

$$S_{vec} = \text{Embed}(Q) \tag{9}$$

This identifies nearest neighbor over dense vector embeddings.

Therefore, by combining lexical and semantic retrieval, the system mitigates the weaknesses of one with the strengths of the other, which results in a candidate set is formed by union:

$$S_{cand} = S_{lex} \cup S_{vec} \tag{10}$$

A cross-encoder reranker (Reciprocal Rank Fusion(RPF)) optimizes contextual alignment:

$$S_{ranked} = \text{Rerank}(S_{cand}, Q) \tag{11}$$

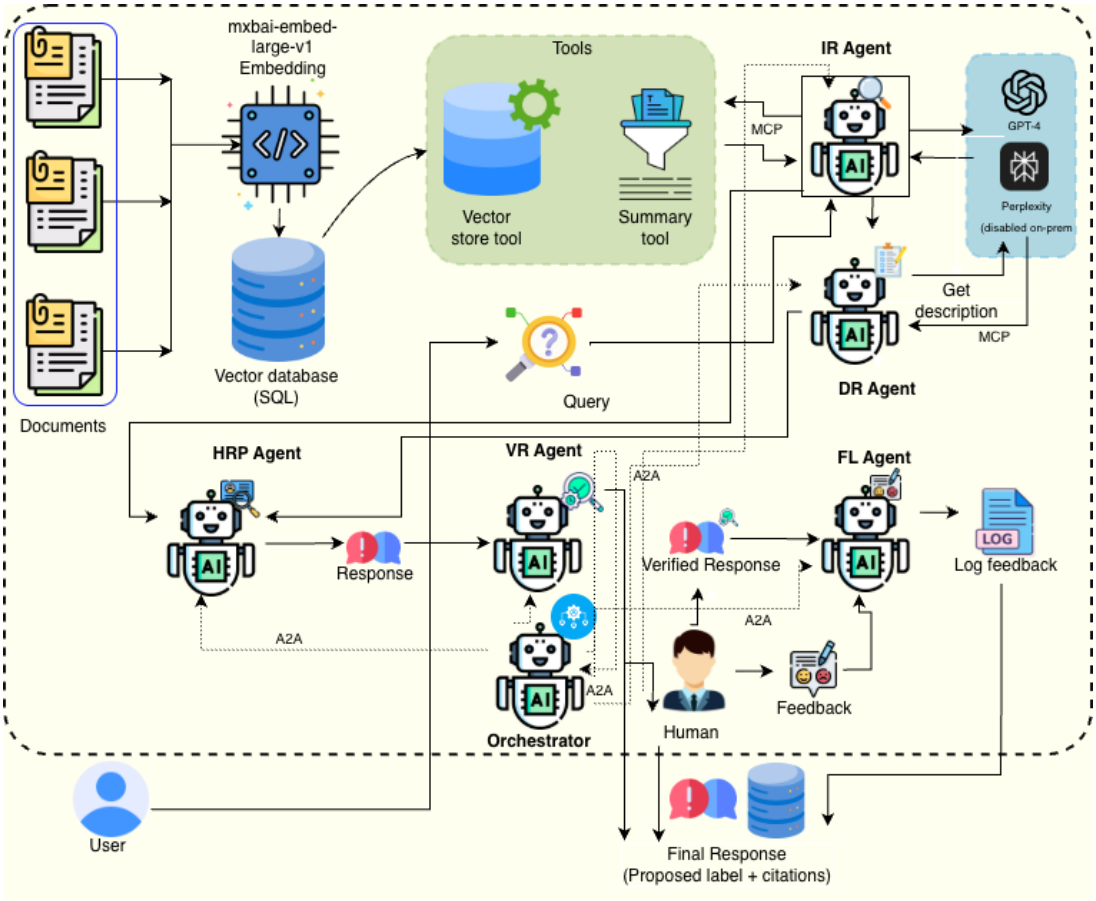

Figure 2: ORCHID agentic architecture. The Orchestrator coordinates agents for Retrieval (IR), Description Refinement (DR), HRP classification, Validation (VR), and Feedback Logging (FL) via agent-to-agent (A2A) messages. IR/DR/HRP access local tools through Model Context Protocol (MCP) adapters (Vector Store, Summary) over a versioned policy corpus. VR either issues a verified decision or routes the case to a human reviewer, whose feedback is recorded in an append-only audit log.

The top-k policy-grounded snippets are selected ($\mathcal{R} = \{s_1, s_2, \ldots, s_k\}$). Each snippet s_i is associated with metadata:

$$s_i = (text_i, doc\_id_i, section_i, version_i) \tag{12}$$

ensuring version traceability and audit compliance.

3.4 Grounded Classification Layer

The HRP classification agent performs citation-grounded classification. It constructs a structured prompt:

$$P = \Phi(\mathcal{I}_1, \mathcal{R}) \tag{13}$$

where Φ enforces explicit inclusion of retrieved evidence. The provisional decision is defined as:

$$\mathcal{D}_p = \{y, c, \mathcal{C}, \mathcal{E}\} \tag{14}$$

where:

- y is the predicted label,
- $c \in [0, 1]$ is the confidence score,
- $\mathcal{C} \subseteq \mathcal{R}$ denotes cited supporting snippets,
- $\mathcal{E}$ represents structured reasoning steps.

A strict grounding constraint is enforced:

$$\forall e_j \in \mathcal{E}, \exists s_i \in \mathcal{C} \tag{15}$$

ensuring that each reasoning step is explicitly supported by retrieved evidence.

3.5 Independent Validation Layer

In this step, we employ a Validation and Review (VR) agent to perform second-order verification of the provisional decision by the previous agent. It evaluates two formal properties.

Coverage Constraint: It checks if each of the reasoning steps ($\forall e_j \in \mathcal{E}$) are supported by any existing retrieved policy snippets ($\exists s_i \in \mathcal{R}$).

$$\text{Coverage} = 1\,(\forall e_j \in \mathcal{E}, \exists s_i \in \mathcal{R}) \tag{16}$$

Conflict Detection: It checks if there exists any policy snippet (S_a) in the citations that contradicts with another

policy snippet (S_b).

$$\text{Conflict} = 1\left(\exists s_a, s_b \in \mathcal{R} \mid s_a \perp s_b\right) \tag{17}$$

Based on these checks, the VR agent emits a verdict:

$$v \in \{\text{AGREE}, \text{REVIEW}, \text{CONFLICT}\} \tag{18}$$

with validation confidence c_v.
A gating mechanism determines routing:

$$\mathcal{D}_f = \begin{cases} \mathcal{D}_p & \text{if } v = \text{AGREE} \wedge c_v > \tau \\ \text{RouteToSME} & \text{otherwise} \end{cases} \tag{19}$$

where τ is a configurable threshold controlling human escalation.

3.6 Feedback Logging and Audit Layer

To ensure traceability and auditability of the system generated outputs, all finalized decisions are appended to an immutable audit store:

$$\mathcal{L}_{t+1} = \mathcal{L}_t \cup \{(\mathcal{I}_1, y, c, v, r_{SME}, \mathcal{RC})\} \tag{20}$$

where r_{SME} denotes reviewer rationale and $\mathcal{RC}$ denotes run-card metadata. An append-only property is ensured to satisfy non-destructive forensic traceability.

3.7 Orchestration and Deterministic Control

To orchestrate and maintain the whole workflow, a non-generative orchestrator coordinates agent execution and maintains state transitions:

$$\mathcal{S}_{t+1} = \mathcal{O}(\mathcal{S}_t, \mathcal{A}_k) \tag{21}$$

Thus, the orchestrator performs, context propagation between agents, execution scheduling, version control annotation, and run-card capture. the run-card metadata captures the LLM_version, Embedding_version, k, τ, and BM25_params.

Importantly, the orchestrator never generates classification content and operates purely at the control plane. Through modular agent decomposition, hybrid retrieval grounding, independent validation, deterministic orchestration, and append-only auditing, the system achieves high-assurance, explainable, and compliance-aligned property identification suitable for regulated environments.

4 Evaluation

In this section, we evaluate the retrieval and classification performance of the framework by comparing tool outputs with SME feedback.

4.1 Experimental Setup

4.1.1 Platform and Tools

The proposed framework was implemented and evaluated on a system equipped with an NVIDIA A100 80GB GPU. The agent-based architecture was developed in Python using the PydanticAI framework, with retrieval and indexing components implemented via chromadb and LlamaIndex and model integration handled through HuggingFace Transformers and PyTorch. The backend service was deployed using FastAPI with Uvicorn for asynchronous serving, while the interactive user interface was built using Streamlit. For semantic retrieval, the system employs the embedding model "intfloat/multilingual-e5-large" and performs cross-encoder reranking using "cross-encoder/ms-marco-MiniLM-L6-v2". Local language model inference is performed using "meta-llama/Llama-3.1-8B-Instruct", while selected reasoning and validation tasks leverage the OpenAI API with the GPT-4o model to achieve state-of-the-art performance.

4.1.2 Data for Evaluation

We evaluate 160 property descriptions with ground truths from SMEs. The ground truths inform if an item is controlled under International Traffic in Arms Regulations (ITAR|USML), Nuclear Regulatory Commission (NRC), or the Commerce Control List (CCL), making them high-risk. If the item poses low-risk, it is labelled EAR99 (i.e., not controlled under any of the lists mentioned above). For each experiment, the model makes a determination of what "category" the item falls under; ITAR|USML, NRC, CCL, or EAR99.

4.1.3 Performance Metrics

The system is currently in a prototype phase, and we have defined a set of core evaluation metrics that are used in assessments. These metrics aim to quantitatively and qualitatively measure the system's adaptability, reliability, and operational efficiency, particularly in the context of human-in-the-loop feedback from Subject Matter Experts (SMEs). To evaluate the retrieval effectiveness and classification precision of our pipeline, we employ the following metrics:

1. **Recall@5**: Measures the ability of the system to retrieve all relevant documents within the top 5 results.

$$\text{Recall@5} = \frac{|\text{Relevant Docs} \cap \text{Retrieved Top 5}|}{|\text{Total Relevant Docs}|} \tag{22}$$

2. **NDCG@5**: Normalized Discounted Cumulative Gain evaluates ranking quality by discounting relevant documents found at lower ranks.

$$\text{DCG}_k = \sum_{i=1}^{k} \frac{rel_i}{\log_2(i+1)}; \quad \text{NDCG}_k = \frac{\text{DCG}_k}{\text{IDCG}_k} \tag{23}$$

3. **Accuracy (4-Class)**: Assesses the model's ability to correctly categorize samples into one of four distinct categories (e.g., USML|ITER, CCL, NRC, EAR99).

$$\text{Acc}_{\text{4-class}} = \frac{\sum_{i=1}^{4} TP_i}{N} \tag{24}$$

4. **Accuracy (Binary/2-Class)**: Evaluates the system's ability to distinguish between "Non-Risk" and any "Sensitive/Risk" class (grouping all 3 classes USML|ITER, CCL, NRC into a single class).

$$\text{Acc}_{\text{2-class}} = \frac{\sum_{i=1}^{2} TP_i}{N} \tag{25}$$

Metric Description: The retrieval metrics (*Recall@5*, *NDCG@5*) focus on the system's ability to surface relevant context, while the **4-class accuracy** provides a granular view of classification performance. The **binary accuracy** specifically isolates the model's capability to act as a safety gate, distinguishing neutral content from any level of sensitive or high-risk information.

4.2 Retrieval Performance

The retrieval performance, summarized in Table 1, illustrates the architectural transition from standalone keyword matching to a multi-stage hybrid pipeline. The baseline Lexical approach utilizing BM25 provides a foundation for exact term matching but exhibits the lowest Recall@5, as it fails to account for semantic variations in policy language. The integration of Semantic Retrieval via a union of candidate sets significantly expands the search breadth, capturing underlying intent and rephrased queries. This performance is further optimized through Reciprocal Rank Fusion (RRF) reranking, which aligns the top candidates more effectively with the user query, thus ensuring that the most contextually relevant and grounded policy snippets are prioritized for downstream tasks.

Table 1: Retrieval Performance Evolution from Lexical Baselines to Hybrid Pipelines.

Configuration	Recall@5	NDCG@5
Lexical (S_{lex} via BM25)	0.17	0.20
Lexical + Semantic ($S_{lex} \cup S_{vec}$)	0.20	0.24
Lexical + Semantic + Rerank (S_{ranked})	0.22	0.27

4.3 Classification Performance

Preliminary results appear in Table 2, and the corresponding confusion matrix is provided in Fig. 4. The system achieves a Weighted Average Accuracy of 63.12% across the four-class taxonomy, with peak performance observed in the NRC (90%) and USML (88%) domains. The diminished accuracy in EAR99 (40%) suggests challenges in classifying baseline or 'catch-all' regulatory content compared to highly specialized policies. Notably, the Binary Accuracy reaches 70.37%, indicating that the IR-driven pipeline is significantly more effective at performing initial risk triage than at granular classification. This performance gap suggests that while the retrieval context successfully surfaces relevant risk indicators, the final classification layer encounters semantic confusion between overlapping regulatory frameworks like CCL and EAR99.

The ORCHID framework improves classification reliability, transparency, and reproducibility through evidence-based policy-aware decision-making. Using RAG, each classification is grounded in traceable citations, ensuring verifiable reasoning. Its hybrid retrieval mechanism integrates domain-specific regulatory corpora, ITAR/USML, NRC, CCL, EAR99, for policy compliance, while a human-in-the-loop design incorporates expert feedback to refine performance and prevent recurring errors. The feedbacks are captured in a structured table with its input context, which is queried in later classifications for similar items. This simple structure currently shows promise and we plan to rigorously evaluate it further to analyze and improve. ORCHID's modular, agentic architecture supports scalability and reproducibility, and its single-click interface streamlines the decision process for efficient, auditable outcomes.

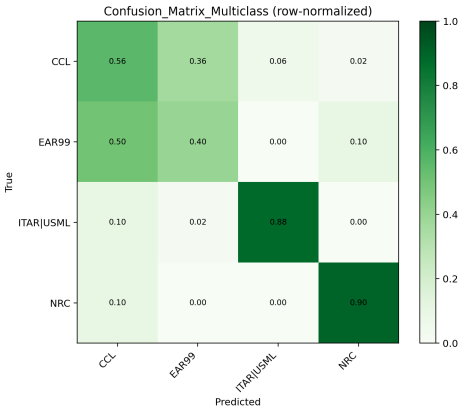

Figure 4: Heatmap with true classes on the y-axis and predicted classes on the x-axis (USML, NRC, CCL, EAR99); values are row-normalized.

The current implementation of ORCHID faces several practical limitations. Its performance depends on curated policy corpora, making it sensitive to coverage gaps and drift when source texts become outdated. Boundary ambiguity persists in fine-grained classifications, particularly in distinguishing CCL and EAR99 items, where validator calibration remains an ongoing effort. The framework currently supports only English text and does not process multimodal inputs such as images or technical specification sheets. In addition, the quality of retrieval and classification is reduced with sparse or poorly written descriptions, and the "no-description" mode exhibits reduced classification relia-

Table 2: Comprehensive preliminary accuracy results.

USML	NRC	CCL	EAR99	Weighted Avg.	Binary Acc
88%	90%	56%	40%	63.12%	70.37%

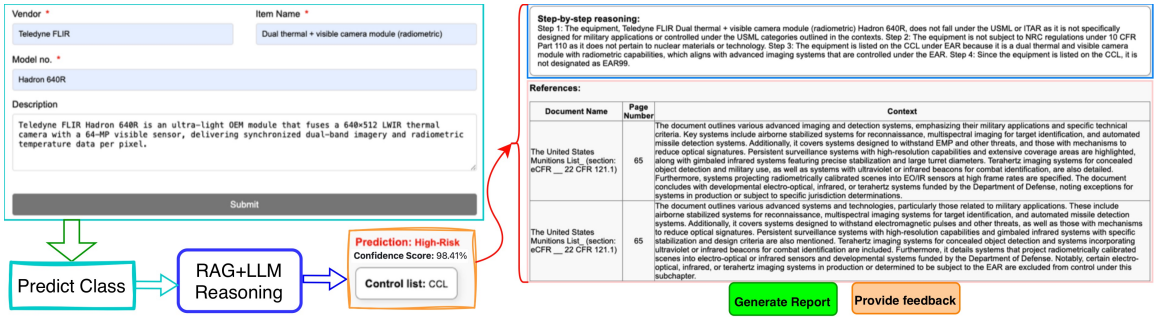

Figure 3: ORCHID UI overview. Submit (vendor, item, model, optional description), inspect policy evidence with citations, review the proposed label and confidence, then record SME feedback.

bility. ORCHID provides decision support but does not constitute legal or regulatory advice, and final determinations must be made by qualified reviewers.

4.4 User Interface

ORCHID's user interface is designed to streamline human–AI interaction in all stages of classification, focusing on transparency, traceability, and minimal analyst effort. After setup, users are directed to the main interface where submissions, results, and feedback are managed through a unified workflow, as visible in Fig. 3.

Submission Workflow: Users initiate the classification workflow by entering a vendor name, item name, and model number, with an optional description, and then trigger processing with a one-click action via the 'Submit' button. This workflow supports both individual entries and batch uploads for multiple items.

Outputs: Upon submission, the interface displays the system's prediction (HRP or Not HRP), the predicted control category, and a single confidence score summarizing model certainty.

Reasoning and Evidence: Each result includes a concise, step-by-step reasoning trace supported by clickable citations. An evidence table presents the underlying documents, sections or pages, and extracted text, with actions for quick copy or open, as well as trace IDs for provenance tracking.

Feedback and Review: Users can provide structured feedback – agreement status, notes, rating, or policy reference – through a simple form submitted via the 'Submit Feedback' button. This input is stored for audit and model refinement.

Batch Mode and Export: For large-scale reviews, the same interface supports batch processing with per-item status indicators and downloadable results. Completed analyses can be exported in JSON, CSV, or PDF formats, each embedding a version strip that records the model identifier, index snapshot, and timestamp to ensure auditability.

The application's frontend provides a cohesive, context-aware experience, guiding users seamlessly from submission to reasoning review, feedback, and export, ensuring both operational efficiency and traceable decision support.

5 Conclusion

In conclusion, we present ORCHID, an agentic framework designed for the scalable, evidence-grounded classification of high-risk properties (HRP). Our initial implementation demonstrates robust performance, particularly within controlled regulatory classes, achieving accuracies up to 90%. Current efforts are focused on refining the architecture to enhance cross-domain classification performance and integrating the framework into mission-critical HRP classification workflows to aid and ease SME burden while maintaining end-to-end traceability and auditability.

References

S Ajay Mukund and KS Easwarakumar. 2025. Optimizing legal text summarization through dynamic retrieval-augmented generation and domain-specific adaptation. *Symmetry*, 17(5):633.

K Clark. 2008. Automated security classification. *Master's thesis, Vrije Universiteit*.

Ian F Fergusson and Paul K Kerr. 2013. The us export control system and the president's reform initiative. Library of Congress, Congressional Research Service.

Xinyi Hou, Yanjie Zhao, Shenao Wang, and Haoyu Wang. 2025. Model context protocol (mcp): Landscape, security threats, and future research directions. *arXiv preprint arXiv:2503.23278*.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55.

Sean Lee, Aamir Shakir, Darius Koenig, and Julius Lipp. 2024. Open source strikes bread - new fluffy embeddings model.

Tong Li and Zhishuai Chen. 2020. An ontology-based learning approach for automatically classifying security requirements. *Journal of Systems and Software*, 165:110566.

Xianming Li and Jing Li. 2023. Angle-optimized text embeddings. *arXiv preprint arXiv:2309.12871*.

Elham Parizadehgan. 2025. Can u.s. export law handle ai?

Jinghai Rao, Alberto Sardinha, and Norman Sadeh. 2009. A meta-control architecture for orchestrating policy enforcement across heterogeneous information sources. *Journal of Web Semantics*, 7(1):40–56.

Partha Pratim Ray. 2025. A review on agent-to-agent protocol: Concept, state-of-the-art, challenges and future directions. *Authorea Preprints*.

Markus Reuter, Tobias Lingenberg, Rūta Liepiņa, Francesca Lagioia, Marco Lippi, Giovanni Sartor, Andrea Passerini, and Burcu Sayin. 2025. Towards reliable retrieval in rag systems for large legal datasets. *Preprint*, arXiv:2510.06999.

Satida Ruengsurat, Jaimai Eawsivigoon, Vidchaphol Sookplang, Karin Sumongkayothin, Prarinya Siritanawan, Razvan Beuran, and Kazunori Kotani. 2025. Human-in-the-loop for machine learning in offensive cybersecurity. pages 0331–0336.

Rafal Rzepka and Akihiko Obayashi. 2025. Effectiveness of security export control ontology for predicting answer type and regulation categories. In *Proceedings of the 2024 8th International Conference on Advances in Artificial Intelligence*, ICAAI '24, page 156–161, New York, NY, USA. Association for Computing Machinery.

Haoran Wang, Qiuye Jin, Shiman Li, Siyu Liu, Manning Wang, and Zhijian Song. 2024. A comprehensive survey on deep active learning in medical image analysis. *Medical Image Analysis*, 95:103201.

Miao Yu and Hailiang Chen. 2025. Gov-rag: A retrieval-augmented generation framework for enhancing e-government services. *Available at SSRN 5111865*.

Kezhen Zhong, Basem Suleiman, Abdelkarim Erradi, and Shijing Chen. 2025. Semrag: Semantic knowledge-augmented rag for improved question-answering. *arXiv preprint arXiv:2507.21110*.

A Pipeline to Bootstrap the Evaluation of Retrieval-Augmented Generation for the Automation of Systematic Reviews in Computer Science

Pierre Achkar
Leipzig University; Fraunhofer ISI
pierre.achkar@uni-leipzig.de

Tim Gollub
Bauhaus-Universität Weimar
tim.gollub@uni-weimar.de

Arno Simons
TU Berlin
arno.simons@tu-berlin.de

Harrisen Scells
University of Tübingen
harrisen.scells@uni-tuebingen.de

Maik Fröbe
Friedrich-Schiller-Universität Jena
maik.froebe@uni-jena.de

Martin Potthast
Kassel University; hessian.AI; ScaDS.AI
martin.potthast@uni-kassel.de

Abstract

Automating systematic reviews (SRs), i.e., evidence-driven analyses under explicit protocol constraints, is a natural target for retrieval-augmented generation and deep research agents, yet existing benchmarks evaluate isolated subtasks or assume fixed evidence inputs. We introduce RAG4SR-CS-200, a benchmark of 200 computer science systematic reviews designed for protocol-driven systematic review automation. Each instance comprises review objectives, research questions, eligibility criteria, cleaned full-text review structure, references, and extracted tables. These elements support evaluation across key tasks in systematic review creation such as literature retrieval, eligibility screening, citation-grounded review generation, and structured table generation, in both stage-wise and end-to-end settings. RAG4SR-CS-200 provides a foundation for developing more reliable and diagnosable deep research agents for scientific evidence synthesis. Code and data are publicly available[1].

1 Introduction

Systematic reviews are the gold standard for scientific evidence synthesis (Mulrow, 1994; Gough et al., 2012), given their protocol-driven, transparent, and reproducible methodology. Yet producing a high-quality systematic review is expensive in both time and effort (Borah et al., 2017). The process also follows clear stages: define the objective and research questions, retrieve candidate studies, screen studies for eligibility, and synthesise accepted evidence into a structured, cited report (Lefebvre et al., 2019; Liberati et al., 2009). This staged process aligns well with multi-stage Retrieval-Augmented Generation (RAG) pipelines (Lewis et al., 2020) and deep research agents (Zhang et al., 2025), where intermediate outputs can be evaluated at each step.

Recent work has made strong progress in SR automation, including end-to-end systems such as otto-SR (Cao et al., 2025) and multi-agent SLR pipelines (Sami et al., 2024). However, existing benchmarks still focus mainly on isolated subtasks. For retrieval and screening, resources such as CLEF TAR (Kanoulas et al., 2017, 2018, 2019), the SysRev Query Collection (Scells et al., 2017), and CSMeD (Kusa et al., 2023) have enabled advances in query formulation and citation screening. For synthesis, datasets such as SciReviewGen, SurveySum, and SumSurvey (Kasanishi et al., 2023; Fernandes et al., 2024; Liu et al., 2024), together with frameworks such as AutoSurvey and SurveyForge (Wang et al., 2024; Yan et al., 2025), mainly target narrative reviews (surveys). Unlike systematic reviews, narrative reviews are generally expert-curated and more subjective, and they do not require protocol-defined eligibility criteria, explicit research questions, or reproducible study selection. Related work also studies structured outputs, such as hierarchical catalogue generation (Zhu et al., 2023) and survey table generation (Chen et al., 2024). Still, some benchmarks assume fixed or pre-selected evidence, leaving protocol-driven retrieval and eligibility decisions outside scope and limiting end-to-end error diagnosis across retrieval, screening, generation, and structured outputs.

To address this gap, we introduce **RAG4SR-CS-200**, a benchmark of 200 computer science systematic reviews for evaluating RAG and deep research agent pipelines for systematic review automation. Each instance combines structured inputs (topic, research objective, research questions, eligibility criteria, and temporal restrictions) with the full re-

[1] https://github.com/webis-de/rag4sr-cs-200

Proceedings of the First Workshop on Multilingual Report Generation via Retrieval Augmented Generation (RAG4Reports), pages 65–70

view text, in-text citation markers, extracted tables, and a resolved reference list. RAG4SR-CS-200 supports evaluation across four stages: **(i)** literature retrieval, **(ii)** eligibility screening, **(iii)** citation-grounded review generation, and **(iv)** structured table generation. The design supports both stage-wise and end-to-end evaluation under realistic retrieval conditions. The benchmark specifies its format, construction pipeline, and evaluation protocol. As the first release in the RAG4SR-X series, designed to expand to additional domains, it provides a foundation for cross-domain benchmarking.

2 Related Work

Prior work on systematic review automation has largely focused on individual pipeline stages. For retrieval and screening, widely used resources such as CLEF TAR (Kanoulas et al., 2017, 2018, 2019), the SysRev Query Collection (Scells et al., 2017), Seed Studies (Wang et al., 2022), SWIFT-Review (Howard et al., 2016), SR Updates (Alharbi and Stevenson, 2019), and AutoBool (Wang et al., 2025a) have enabled progress in Boolean query formulation and citation screening (Scells et al., 2021; Wang et al., 2023, 2025b; Cohen et al., 2006; Wallace et al., 2010; Miwa et al., 2014). However, these datasets mainly evaluate early-stage tasks rather than the full end-to-end review pipeline.

At the synthesis stage, several resources target downstream generation quality. SciReviewGen (Kasanishi et al., 2023) and SurveySum (Fernandes et al., 2024) generate survey sections from cited papers, while SumSurvey (Liu et al., 2024) targets long-document summarisation of full survey papers. SurveyForge (Yan et al., 2025) is a pipeline for producing full survey drafts with structure-aware generation and memory-guided retrieval. SurveyBench (Sun et al., 2025) and SGSimEval (Guo et al., 2025) are evaluation resources that score generated surveys beyond lexical overlap, including structure, content quality, and reference quality. HiCatGLR (Zhu et al., 2023) and Survey Table Generation (Chen et al., 2024) target structured outputs, namely hierarchical outlines and comparison tables. These resources are valuable, but many rely on fixed inputs, pre-collected corpora, or pre-selected references; protocol-driven retrieval and eligibility screening are typically outside the evaluated scope. RAG4SR-CS-200 addresses this gap by targeting protocol-driven systematic review automation, with unified evaluation across retrieval, screening, citation-grounded generation, and structured outputs.

Table 1 summarises representative resources by benchmark coverage and highlights the remaining gap: unified support for stage-wise and end-to-end evaluation under a common protocol.

3 Dataset Construction

This section describes how we construct RAG4SR-CS-200 from an existing large-scale systematic review resource and transform it into benchmark-ready artefacts for SR automation.

3.1 Goal and Scope

RAG4SR-CS-200 is designed for protocol-driven *systematic review* (SR) automation. The benchmark supports unified evaluation across retrieval, eligibility screening, citation-grounded generation, and table generation. In contrast to survey-focused resources, we curate reviews exposing SR-specific method signals, including a stated objective, research questions, and eligibility criteria.

3.2 Collection and Selection Pipeline

The construction starts from a large-scale dataset of systematic reviews that we collected from OpenAlex (Priem et al., 2022); this upstream resource is part of our ongoing unpublished work. OpenAlex indexes a wide range of scholarly documents across domains and provides broad citation metadata coverage (Culbert et al., 2025). On top of this collection, we perform structured extraction over reviews with accessible full text to identify explicitly reported review-protocol artefacts. From this source pool, we derive a computer-science-focused subset by applying strict SR-oriented inclusion filters. Specifically, a review is retained only if it provides: *(i)* an explicit review objective, *(ii)* explicit research questions, and *(iii)* explicit eligibility criteria. We exclude records missing any of these protocol artifacts, outside computer science scope, or lacking usable full text after parsing. The resulting subset contains 200 computer science systematic reviews.

3.3 Data Extraction and Normalisation

For the selected reviews, parsed markdown is already available from the upstream pipeline (generated from PDF using *PaddleOCR-VL*,[2]). However, this parsed text remains noisy for benchmark use,

[2] https://github.com/PaddlePaddle/PaddleOCR

Resource	Domain	R	S	G	T	E2E
CLEF TAR (Kanoulas et al., 2017, 2018, 2019)	Biomedical	X	X			
SysRev Query Collection (Scells et al., 2017)	Biomedical	X				
Seed Studies (Wang et al., 2022)	Biomedical	X				
SWIFT-Review (Howard et al., 2016)	Biomedical		X			
SR Updates (Alharbi and Stevenson, 2019)	Biomedical		X			
AutoBool (Wang et al., 2025a)	Biomedical	X				
SciReviewGen (Kasanishi et al., 2023)	CS/NLP			X		
SurveySum (Fernandes et al., 2024)	CS/NLP			X		
SurveyForge (Yan et al., 2025)	CS/NLP	X		X		
SurveyBench (Sun et al., 2025)	CS/NLP			X		
SGSimEval (Guo et al., 2025)	CS/NLP			X		
HiCatGLR (Zhu et al., 2023)	CS/NLP			X		
Survey Table Generation (Chen et al., 2024)	CS/NLP				X	
RAG4SR-CS-200 (ours)	CS	X	X	X	X	X

Table 1: Representative related work by domain and benchmark coverage. R/S/G/T indicate whether a resource explicitly evaluates retrieval, eligibility screening, citation-grounded generation, or structured table generation. E2E indicates joint evaluation of the full protocol under one benchmark

as headings, citations, and tables are not consistently normalised across reviews. For RAG4SR-CS-200, we therefore apply an additional LLM-assisted cleanup workflow. We first run *DeepSeek-V3.1*[3] for structural cleanup and citation normalisation, then manually compare each cleaned markdown file against the source PDF. During this stage, we enforce consistent heading structure, remove conversion artefacts (e.g., extra whitespace, incorrect line breaks), and normalise in-text citations into numeric markers to make citation grounding comparable across reviews.

After text cleanup, we parse review reference lists with *Anystyle*[4] to extract bibliographic fields, including title, author list, DOI, and publication year. We then normalise reference metadata against *Crossref* to ensure consistent formatting and to obtain OpenAlex identifiers for alignment with the indexed reference corpus. This produces a cleaned, linkable reference layer for retrieval and grounding evaluation.

3.4 Benchmark Schema and Quality Control

Each review is distributed as a structured JSON record, an accompanying markdown file containing extracted tables and the reference list with OpenAlex identifiers. At the JSON level, each instance includes: review identifier, high-level metadata (e.g., section and table counts), section/subsection hierarchy with cleaned text, normalised in-text citation markers, table placeholders in context, and a table index that links placeholders to source locations. The table artefact stores full markdown-renderable table content. Figure 1 shows a compact per-review JSON example.

```
{
"id": "W1505282872",
"metadata": { "title": ..., "n_sections": 3,
"n_subsections": 3, "n_tables": 19 },
"sections": [
{
"section_id": "s_1",
"section_label": "1. Introduction",
"text": ...,
"citations": ["W1993563915", "W2123444177",
...],
"tables_in_text": [ ... ],
"subsections": [ ... ]
}
],
"tables": [
{ "table_id": "tbl_01", "placeholder":
"{{TABLE:tbl_01}}",
"source_section_id": "s_2",
"source_subsection_id": "s_2_1" }
],
"tables_file": "W1505282872_tables.md"
}
```

Figure 1: Compact example of the per-review JSON schema used in RAG4SR-CS-200, based on the instance in `example_data/W1505282872.json`.

Quality control combines automatic and manual checks. Automatic checks validate schema consistency, citation-marker formatting, and section-table linkage integrity. Manual checks compare cleaned markdown with the original PDF to ensure that section boundaries, citation placement, and table references remain faithful to the source document.

3.5 Statistics and Benchmark Coverage

RAG4SR-CS-200 contains 200 computer science systematic reviews with harmonised structure and citation annotations. Each instance supports all

[3] https://api-docs.deepseek.com/news/news250821

[4] https://anystyle.io/

benchmark stages: retrieval (through linked reference metadata), screening-oriented protocol artefacts (objective, research questions, eligibility criteria), citation-grounded generation (through normalised in-text citations and reference alignment), and table generation (through extracted markdown tables). On average, each review contains 77 references, 7 sections, 11 subsections, and 7 tables. In total, the corpus contains 15,498 references, of which 12,871 are matched to OpenAlex metadata, corresponding to 83.1% alignment coverage.

4 Discussion

This section summarises the benchmark scope and discusses why it is useful and challenging for RAG and agentic deep research systems.

4.1 Benchmark Scope

RAG4SR-CS-200 is intended for evaluating RAG and agentic pipelines for protocol-driven systematic review automation. The benchmark supports controlled comparisons at both individual stages and full end-to-end settings across the four tasks. At the retrieval stage, the references aligned with OpenAlex identifiers make it possible to assess whether a system can identify candidate studies relevant to the review objective and research questions. Eligibility screening is supported by protocol-defining inputs such as the review objective, research questions, and eligibility criteria, making it possible to assess whether studies are correctly retained or excluded under explicit review constraints. Citation-grounded generation can be assessed using the cleaned full-text review structure and normalised citation markers, which allow testing of whether generated synthesis remains grounded in identifiable evidence. The extracted markdown tables provide a target for assessing whether a system can produce faithful tabular summaries from the underlying review content. Taken together, these components make RAG4SR-CS-200 suitable for studying deep-research-style agents that must coordinate retrieval, reasoning, grounding, and structured synthesis over long scientific documents.

4.2 Challenges and Evaluation

Systematic review automation is a demanding setting for RAG systems and deep research agents. Unlike short-form question answering, the task requires sustained multi-step reasoning under explicit protocol constraints: a system must identify relevant evidence, respect inclusion criteria, preserve traceability to source documents, and produce outputs whose claims can be inspected against cited references. Errors at early stages, such as weak retrieval or incorrect screening, propagate into later synthesis and can silently degrade the final review. Stage-wise analysis is therefore important for diagnosing where failures begin and how they affect downstream outputs.

RAG4SR-CS-200 is designed to expose exactly these failure modes. By combining protocol signals, cleaned full-text structure, normalised citations, and tables, the benchmark supports analysis of whether a method retrieves the right evidence, grounds claims correctly, and produces faithful structured outputs. This supports both component-level analysis and stress-testing of end-to-end agentic systems for scientific research workflows. To support concrete task use, we release structured per-review JSON files, extracted table files, aligned reference metadata, and pipeline scripts in the public repository. Evaluation can be conducted at both component and end-to-end levels. At the component level, retrieval can be measured with standard retrieval metrics (e.g., precision and recall), screening can be evaluated against the references included in the review, generation can be assessed with lexical or semantic similarity measures, citation quality can be assessed with citation-specific metrics, and table generation can be evaluated at cell-, row-, and table-level granularity using lexical or semantic overlap with reference tables. End-to-end evaluation can then assess faithfulness, completeness, and traceability of the final synthesis.

5 Conclusion

We introduced RAG4SR-CS-200, a benchmark of 200 computer science systematic reviews for evaluating protocol-driven systematic review automation with RAG and agentic systems. The benchmark supports unified evaluation across retrieval, screening, citation-grounded generation, and structured table generation, enabling stage-wise and end-to-end analysis under realistic review constraints. As the first release in the planned RAG4SR-X series, it provides a foundation for broader cross-domain benchmarks and more reliable deep research agents for scientific evidence synthesis.

6 Limitations

The current release is restricted to computer science systematic reviews, so transfer to other scientific domains should not be assumed without further validation. The benchmark also inherits residual noise from PDF parsing, LLM-assisted cleanup, reference parsing, and citation resolution despite manual checks. Because this release prioritises data curation and normalisation rather than subjective annotation, we report procedural quality control instead of inter-annotator agreement. The release provides OpenAlex identifiers for references included in each review, but does not redistribute the corresponding abstracts or full-text documents; users must source this reference content when building retrieval corpora or running end-to-end experiments beyond the released artefacts.

References

Amal Alharbi and Mark Stevenson. 2019. A dataset of systematic review updates. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*, pages 1257–1260. ACM.

Rohit Borah, Andrew W. Brown, Patrice L. Capers, and Kathryn A. Kaiser. 2017. Analysis of the time and workers needed to conduct systematic reviews of medical interventions using data from the prospero registry. *BMJ Open*, 7.

Christian Cao, Rohit Arora, Paul Cento, Katherine Manta, Elina Farahani, Matthew Cecere, Anabel Selemon, Jason Sang, Ling Xi Gong, Robert Kloosterman, Scott Jiang, Richard Saleh, Denis Margalik, James Lin, Jane Jomy, Jerry Xie, David Chen, Jaswanth Gorla, Sylvia Lee, and 14 others. 2025. Automation of systematic reviews with large language models. *medRxiv*.

Po-Chun Chen, Hen-Hsen Huang, and Hsin-Hsi Chen. 2024. Survey table generation from academic articles. In *2024 IEEE/WIC International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, pages 190–197.

Aaron M. Cohen, William R. Hersh, K. Peterson, and Po-Yin Yen. 2006. Research paper: Reducing workload in systematic review preparation using automated citation classification. *J. Am. Medical Informatics Assoc.*, 13(2):206–219.

Jack H. Culbert, Anne Hobert, Najko Jahn, Nick Haupka, Marion Schmidt, Paul Donner, and Philipp Mayr. 2025. Reference coverage analysis of openalex compared to web of science and scopus. *Scientometrics*, 130:2475–2492.

Leandro Car'isio Fernandes, Gustavo Bartz Guedes, Thiago Laitz, Thales Sales Almeida, Rodrigo Nogueira, R.A. Lotufo, and Jayr Pereira. 2024. Surveysum: A dataset for summarizing multiple scientific articles into a survey section. In *Brazilian Conference on Intelligent Systems*.

David Gough, Sandy Oliver, and James Thomas, editors. 2012. *An Introduction to Systematic Reviews*. Sage Publications Ltd.

Beichen Guo, Zhiyuan Wen, Yu Yang, Peng Gao, Ruosong Yang, and Jiaxing Shen. 2025. Sgsimeval: A comprehensive multifaceted and similarity-enhanced benchmark for automatic survey generation systems. In *International Conference on Advanced Data Mining and Applications*.

Brian E. Howard, Jason Phillips, Kyle Miller, Arpit Tandon, Deepak Mav, Mihir R. Shah, Stephanie Holmgren, Katherine E. Pelch, Vickie Walker, Andrew A. Rooney, Malcolm Macleod, Ruchir R. Shah, and Kristina Thayer. 2016. Swift-review: a text-mining workbench for systematic review. *Systematic Reviews*, 5:87.

Evangelos Kanoulas, Dan Li, Leif Azzopardi, and René Spijker. 2017. CLEF 2017 technologically assisted reviews in empirical medicine overview. In *Working Notes of CLEF 2017 - Conference and Labs of the Evaluation Forum, Dublin, Ireland, September 11-14, 2017*, volume 1866 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Evangelos Kanoulas, Dan Li, Leif Azzopardi, and René Spijker. 2018. CLEF 2018 technologically assisted reviews in empirical medicine overview. In *Working Notes of CLEF 2018 - Conference and Labs of the Evaluation Forum, Avignon, France, September 10-14, 2018*, volume 2125 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Evangelos Kanoulas, Dan Li, Leif Azzopardi, and René Spijker. 2019. CLEF 2019 technology assisted reviews in empirical medicine overview. In *Working Notes of CLEF 2019 - Conference and Labs of the Evaluation Forum, Lugano, Switzerland, September 9-12, 2019*, volume 2380 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Tetsu Kasanishi, Masaru Isonuma, Junichiro Mori, and Ichiro Sakata. 2023. SciReviewGen: A large-scale dataset for automatic literature review generation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 6695–6715, Toronto, Canada. Association for Computational Linguistics.

Wojciech Kusa, Óscar E. Mendoza, Matthias Samwald, Petr Knoth, and Allan Hanbury. 2023. Csmed: Bridging the dataset gap in automated citation screening for systematic literature reviews. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Carol Lefebvre, Julie Glanville, Simon Briscoe, Anne Littlewood, Chris Marshall, Maria-Inti Metzendorf, Anna Noel-Storr, Tamara Rader, Farhad Shokraneh, James Thomas, and L. Susan Wieland. 2019. *Searching for and selecting studies*. John Wiley & Sons, Ltd.

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 *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6–12, 2020, virtual*.

A. Liberati, D. Altman, J. Tetzlaff, C. Mulrow, P. Gøtzsche, J. Ioannidis, Mike Clarke, Mike Clarke, P. Devereaux, J. Kleijnen, and D. Moher. 2009. The prisma statement for reporting systematic reviews and meta-analyses of studies that evaluate health care interventions: Explanation and elaboration. *PLoS Med.*

Ran Liu, Ming Liu, Min Yu, He Zhang, Jianguo Jiang, Gang Li, and Weiqing Huang. 2024. SumSurvey: An abstractive dataset of scientific survey papers for long document summarization. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 9632–9651, Bangkok, Thailand. Association for Computational Linguistics.

Makoto Miwa, James Thomas, Alison O'Mara-Eves, and Sophia Ananiadou. 2014. Reducing systematic review workload through certainty-based screening. *Journal of Biomedical Informatics*, pages 242–253.

C. D. Mulrow. 1994. Systematic reviews: Rationale for systematic reviews. *BMJ*, 309(6954):597–599.

Jason Priem, Heather A. Piwowar, and Richard Orr. 2022. Openalex: A fully-open index of scholarly works, authors, venues, institutions, and concepts. *ArXiv*, abs/2205.01833.

Malik Abdul Sami, Zeeshan Rasheed, Kai-Kristian Kemell, Muhammad Waseem, Terhi Kilamo, Mika Saari, Kari Systä, Anh Nguyen Duc, and Pekka Abrahamsson. 2024. System for systematic literature review using multiple ai agents: Concept and an empirical evaluation a preprint.

Harrisen Scells, Guido Zuccon, and Bevan Koopman. 2021. A comparison of automatic boolean query formulation for systematic reviews. *Information Retrieval Journal*, pages 3–28.

Harrisen Scells, Guido Zuccon, Bevan Koopman, Anthony Deacon, Leif Azzopardi, and Shlomo Geva. 2017. A test collection for evaluating retrieval of studies for inclusion in systematic reviews. In *Proc. of SIGIR 2017*. ACM.

Zhaojun Sun, Xuzhou Zhu, Xuanhe Zhou, Xin Tong, Shuo Wang, Jie Fu, Guoliang Li, Zhiyuan Liu, and Fan Wu. 2025. Surveybench: Can llm(-agents) write academic surveys that align with reader needs? *ArXiv*, abs/2510.03120.

Byron C. Wallace, Thomas A. Trikalinos, Joseph Lau, Carla E. Brodley, and Christopher H. Schmid. 2010. Semi-automated screening of biomedical citations for systematic reviews. *BMC Bioinform.*, 11:55.

Shuai Wang, Harrisen Scells, Justin Clark, Bevan Koopman, and Guido Zuccon. 2022. From little things big things grow: A collection with seed studies for medical systematic review literature search. In *SIGIR*, pages 3176–3186.

Shuai Wang, Harrisen Scells, Bevan Koopman, and Guido Zuccon. 2023. Can chatgpt write a good boolean query for systematic review literature search? In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2023, Taipei, Taiwan, July 23-27, 2023*, pages 1426–1436. ACM.

Shuai Wang, Harrisen Scells, Bevan Koopman, and Guido Zuccon. 2025a. AutoBool: An Reinforcement-Learning trained LLM for Effective Automated Boolean Query Generation for Systematic Reviews. *CoRR*, abs/2602.00005.

Shuai Wang, Harrisen Scells, Bevan Koopman, and Guido Zuccon. 2025b. Reassessing large language model boolean query generation for systematic reviews. In *SIGIR*, pages 3296–3305.

Yidong Wang, Qi Guo, Wenjin Yao, Hongbo Zhang, Xin Zhang, Zhen Wu, Meishan Zhang, Xinyu Dai, Min Zhang, Qingsong Wen, Wei Ye, Shikun Zhang, and Yue Zhang. 2024. Autosurvey: large language models can automatically write surveys. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA. Curran Associates Inc.

Xiangchao Yan, Shiyang Feng, Jiakang Yuan, Renqiu Xia, Bin Wang, Lei Bai, and Bo Zhang. 2025. SURVEYFORGE : On the outline heuristics, memory-driven generation, and multi-dimensional evaluation for automated survey writing. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12444–12465, Vienna, Austria. Association for Computational Linguistics.

Wenlin Zhang, Xiaopeng Li, Yingyi Zhang, Pengyue Jia, Yichao Wang, Huifeng Guo, Yong Liu, and Xiangyu Zhao. 2025. Deep research: A survey of autonomous research agents. *ArXiv*, abs/2508.12752.

Kun Zhu, Xiaocheng Feng, Xiachong Feng, Yingsheng Wu, and Bing Qin. 2023. Hierarchical catalogue generation for literature review: A benchmark. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 6790–6804, Singapore. Association for Computational Linguistics.

UNH @ Rag4Reports: A Broad Exploration of LLM-Judges for RAG

Minna Tran, Ryan McCarthy, Aiden Parsons, Jaren Unzen, Laura Dietz
University of New Hampshire, USA
minna.tran@unh.edu, mryan9393@gmail.com, aidenmp2323@gmail.com, jarenunh@gmail.com, dietz@cs.unh.edu

Abstract

We submitted a breadth of LLM-as-a-Judge approaches to Rag4Reports Task A; our top method ranked first among all submitted systems. We find that citation faithfulness is the most essential signal, and that content is best verified by checking whether cited documents cover nuggets generated from the LLM's internal knowledge.

1 Introduction

An AutoJudge, also called LLM-as-a-Judge, is an automated pipeline for predicting the quality of Retrieval-Augmented Generation (RAG) systems. Given a set of topics, each with a problem statement and background information, and the responses produced by multiple RAG systems, an AutoJudge assigns each system a quality score and produces a leaderboard that ranks systems from best to worst. The Rag4Reports challenge is based on the TREC 2025 RAGTIME dataset, in which each topic consists of a short title, a problem statement, and a user background description.

An AutoJudge evaluates each system output as a triple: topic, retrieved documents, and generated response. The quality score decomposes along the three pairwise relationships among these elements: whether the response addresses the topic, whether the retrieved documents are relevant to the topic, and whether the response is faithful to and properly cited from the retrieved documents. These are aggregated along two dimensions, *nuggets* (atomic pieces of information a good response must contain) and *citation* (whether claims are backed by valid citations to retrieved documents), which we combine into an F1 score.

To evaluate the AutoJudge itself, we compare its system-level ranking against a reference ranking produced by human annotators using Kendall's τ, Spearman's ρ, and Pearson's r. We focus on Kendall's τ as leaderboard quality which it measures pairwise ordering agreement among systems.

The AutoJudge systems in this paper were developed by students in a Spring 2026 undergraduate/graduate course, *Research on Text, Knowledge, and LLMs*. Each student independently designed an AutoJudge and entered it in the Rag4Reports challenge. For method development we used the RAGTIME subset of the TREC AutoJudge Pilot Dataset v0.2. The remainder of this paper presents these systems and compares their leaderboards against the human reference.

2 Related Work

Nugget-based Evaluation. A nugget is an atomic piece of information that a human assessor decides is essential to a response (Voorhees, 2004; Nenkova and Passonneau, 2004). Counting how many essential nuggets a response covers yields a content-relevance metric that is more tractable to annotate consistently than full-response relevance. Voorhees distinguished *vital* nuggets, which any good answer must contain, from other relevant nuggets, which are nice-to-have, and computed recall only over vital nuggets so that importance is embedded into the metric.

Citation Evaluation. Where nuggets ask whether the right content is present, citation evaluation asks whether the content is attributable to a cited source. Liu et al. (2023) found that only 51.5% of sentences generated by commercial search engines were fully supported by their citations, and only 74.5% of citations actually supported the claims they were attached to. Citation quality cannot be assumed. Nugget-based content and citation-based attribution form the dominant axes along which RAG outputs are evaluated in AUTO-ARGUE (Walden et al., 2026), and they are the basis for the scoring criteria used in AutoJudge.

Proceedings of the First Workshop on Multilingual Report Generation via Retrieval Augmented Generation (RAG4Reports), pages 71–76

LLM-as-a-Judge Biases. LLM-based judges are easier to scale than human assessment but exhibit systematic biases: passage length and surface lexical cues inflate scores (Yu et al., 2026), the mere presence of reasoning regardless of its factual content sways judgement (Tu et al., 2026), and judges favor outputs that resemble their own style (Roytburg et al., 2026). These behaviors are artifacts of training on signals (length, fluency, lexical overlap) that correlate with quality without causing it.

Decomposing the Judgement. Several methods make LLM judgement more auditable by decomposing it into checkable parts. Rubric-based evaluation breaks a scalar judgement into explicit, checkable criteria (Dhole and Agichtein, 2026; Dietz, 2024). Chain-of-thought prompting asks the model to produce a reasoning trace before its answer (Wei et al., 2022). The Selection-Inference framework of Creswell et al. (2022) goes further by splitting each step into two modules: Selection picks relevant facts, and Inference produces a new fact from that selection without access to the original question, denying the model the inputs that would let it produce a memorized answer.

3 Approaches

3.1 Citation Accuracy

LLM: `cross-encoder/nli-deberta-v3-base` (DeBERTa-v3 NLI, 3-class).

For each response, fragments with non-empty citations are retained. For each cited fragment, the first cited document that resolves in the response's document set is paired with the fragment text; remaining citations on the fragment are ignored. Each pair is scored by 3-class NLI yielding logits `[contradiction, entailment, neutral]`, and the pair is marked supported iff entailment is strictly the maximum (`s[1] > s[0]` and `s[1] > s[2]`). The final score is supported fragments divided by total cited fragments, or 0 if no fragment cited anything.

3.2 Retrieval Quality

LLM: `gpt-4.1-nano` (used for both nugget generation and grading).

For each topic, the LLM is prompted via the Nugget Creation prompt to decompose the question into atomic sub-questions ("nuggets"). For each response, a single retrieval context is built by taking all retrieved documents, keeping the first 20, truncating each to 1000 characters, and joining with a separator. For each (retrieval context, nugget) pair, the LLM is prompted via the Retrieval Grading prompt to return a single number (2 = fully answers, 1 = partially answers, 0 = does not answer). The final score is the sum of nugget grades divided by $2 \times n_{\text{nuggets}}$.

Nugget Creation prompt.

```
You are an evaluation expert. Given a question, break it into
specific sub-questions that a complete answer must address.

Each sub-question should be atomic (cover only 1 aspect),
independently assessable (don't need further sub-questions to
answer it), and cover a distinct aspect of the question.

Return ONLY a JSON array of strings.

Example: ["What year was the US founded?",
"How many children did Britney Spears have?"]
```

Retrieval Grading prompt.

```
Do these retrieved documents contain information that answers
this question? Reply with a single number:

2 = documents fully answer the question,
1 = documents partially answer the question,
0 = documents do not answer the question.
```

Example Nuggets (topic: "Geoffrey Hinton's AI Concerns"):

- Who is Geoffrey Hinton and what is his significance in artificial intelligence?
- What are Geoffrey Hinton's main contributions to artificial intelligence?
- Why did Geoffrey Hinton leave Google in 2023?
- What specific warnings did Hinton issue about the potential dangers of AI?
- Why did Hinton's warnings about AI become newsworthy or make headlines?

3.3 Attribution Score

LLM: `vectara/hallucination_evaluation_model` (HHEMv2) for scoring; `gpt-4.1-nano` for claim extraction.

For each response, the LLM is prompted via the Claim Extraction prompt to split the report text into self-contained verifiable claims (facts, opinions, predictions, and hedged statements with hedging preserved); claims shorter than 10 characters are dropped. Each claim is then paired with every document retrieved for the response (not truncated) and each pair is scored by HHEMv2, which returns a continuous entailment-style score in $[0, 1]$. For each claim, the maximum score across its document pairings is kept as the "best supporting evidence" score. The final score is the mean of these per-claim max scores, or 0 if no claims were extracted.

Claim Extraction prompt.

```
You are a claim extractor. Break this response into specific
claims that could be verified against a source document.

Each claim should be a single, self-contained statement.
Include all types: facts, opinions, predictions, and hedged
statements. For hedged claims, preserve the hedging language.
Return ONLY a JSON array of strings.

Example: ["The war started in 1914.",
"Everybody in the family agreed that it was a bad meal."]
```

Example Extracted Claims (topic `1001`, system `aloe`):

- Hinton has been worried about the potential of AI to do harm and often talks about the existential threat AI might pose to the human species.
- He has used OpenAI's ChatGPT and this has made him uneasy.
- Dr Hinton told the BBC: "Right now, they're not more intelligent than us, as far as I can tell. But I think they soon may be."
- The Biden White House is concerned about AI being developed by the tech sector and then unleashed on the public with little to no guardrails.
- AI could match human intelligence in five years, according to Google DeepMind CEO Demis Hassabis.

3.4 Final

The FINAL score combines Citation Accuracy (CA), the Attribution Score (AS), and Retrieval Quality (RQ) into a single system quality score. For each (system, topic) pair, the stronger of CA and AS is selected via $\max$ and the weaker is discarded; this value is then averaged with RQ:

$$\text{Final} = \tfrac{1}{2}\big[\max(\text{CA},\, \text{AS}) + \text{RQ}\big].$$

3.5 Checklist

LLM: `gpt-4o-mini`.

For each topic, the LLM is prompted via the Checklist Builder prompt to produce a fixed checklist of five items capturing the most important content a good response should contain. For each response, the LLM is then prompted via the Checklist Evaluation prompt to award one point per checklist item the response satisfies. The final score is the total number of points awarded.

Checklist Builder prompt.

```
You are generating a checklist to evaluate how well a response
satisfies a report request.

Request: {request_text}
Instructions:
- Create EXACTLY {num_items} checklist items.
- Each item must represent ONE distinct requirement.
- Items must be specific, objective, and clearly checkable.
- Avoid overlap between items.
- Cover the most important aspects of the request.
- If needed, merge or prioritize less important details to stay
  within {num_items} items.
- Ensure a balanced checklist (not all items about the same
  aspect).

Output format:
- Return a numbered list (1., 2., 3., ...).
- Each item must be a single sentence.
- Do NOT include explanations or extra text.
Return ONLY the checklist.
```

Checklist Evaluation prompt.

```
You are evaluating how well a response satisfies a checklist.

Checklist: {checklist_request.checklist}
Response:  {response_text}

Instructions:
- Each checklist item is worth 1 point.
- Award 1 point if the response clearly satisfies the item.
- Award 0 points if it does not.
- Be strict: partial or vague matches do NOT count.
- Sum the total points.

Return ONLY the total score as a number.
Do not explain your answer.
```

Example Checklist:

- The report includes a brief biography of Geoffrey Hinton, highlighting his key contributions to artificial intelligence.
- The report explains the context and significance of Hinton's resignation from Google in 2023.
- The report details Hinton's specific warnings about the potential dangers of AI and why they are considered important.
- The report summarizes responses from other experts in the field regarding Hinton's concerns.
- The report provides insights on what the public should understand about the implications of Hinton's warnings for the future of AI.

3.6 Negation Judge

LLM: `gpt-4o-mini`.

For each topic, the problem statement is taken as the original query and a negated query is constructed by prepending the prefix "Not about, related to, or relevant to the following:". The original query, the negated query, and the response are passed to the LLM, which scores the relevance of the response against each query on a 0–3 scale using the prompt below. The final score for the topic is a function of the two relevance scores: the penalty grows as the two scores converge, so a system whose response appears equally relevant to a query and its negation receives the lowest score, while a system that correctly assigns low relevance to the negated query receives no penalty. The intuition is that score separation across inverse prompts indicates a Rag system that generates content of semantic relevance rather than lexical overlap.

Scoring prompt.

```
You are a relevance evaluator that evaluates the relevance
between a query and a response. Score relevance between each
query and response (0 to 3). Consider each query and response
pair separately. Return ONLY one integer that represents the
relevance between each query and response pair. The first
integer is the relevance between the first query and the
response. The second integer is the relevance between the second
query and the response. Example output: 3 1.

Response: {full_response}
Query 1:  {query}
Query 2:  Not about, related to, or relevant to the following:
          {query}
```

3.7 Graded Relevance

LLM: DeepSeek v3.2.

For each topic, a query is constructed by concatenating the title, problem statement, and background. The LLM is prompted to rate the relevance of the response to this query on a 1–5 scale; the scale endpoints are repeated in both the system and user message to reduce malformed outputs. The final score is the LLM's 1–5 rating.

Scoring prompt.

```
query = f"{topic_title} {topic_problem_statement}
          {topic_background}"

System: You are a relevance evaluator. Rate the relevance of
the following output on a scale of 1-5, where 1 is 'Completely
Irrelevant' and 5 is 'Perfectly Relevant'. Provide a score only
in JSON, example: {"score" : "1"}

User: Is this passage: {full_report_text} relevant to this
user's needs? Query: {query}

Rate the relevance on a scale of 1-5, where 1 is 'Completely
Irrelevant' and 5 is 'Perfectly Relevant'. Provide a score only
in JSON, example: {"score" : "1"}
```

3.8 Naive: Length

For each topic, the response is scored by the number of characters in the response text; systems producing longer responses are ranked higher. No LLM used.[1]

3.9 Naive: Random

Each response receives a uniformly random score, providing a sanity-check baseline for the AutoJudge evaluation. No LLM used.

4 Empirical Results

To evaluate our approaches, we use two datasets based on the TREC 2025 RAGTIME data for report generation. For method development we use the RAGTIME subset of TREC AutoJudge Pilot dataset v0.2 (Dietz et al., 2025). Our test dataset is the Rag4Reports challenge.

4.1 Results

Table 1 compares our methods on AutoJudge Pilot and official Rag4Reports data set.

The best results are obtained by CITATIONACCURACY, which placed on first rank in the challenge, obtaining results only slightly below the organizer's AUTO-ARGUE system. This is likely because CITATIONACCURACY directly measures citation faithfulness, which the other judges do not. The official leaderboard places a high emphasis on citation faithfulness, marking CITATIONACCURACY the winner, despite its low nugget-coverage performance. In our internal evaluation with the AutoJudge Pilot dataset, FINAL performed stronger than CITATIONACCURACY.

Table 1: Kendall's τ for each method on the AutoJudge development set ("nugget coverage" and "citation support") and the Rag4Reports test set. Best results in bold. ($\star$) did not finish in time for submission.

Method	AutoJudge Nugget	AutoJudge Citation	Rag4Rep. Kendall
CITATIONACCURACY	0.245	**0.527**	**0.558**
RETRIEVALQUALITY	0.613	0.452	0.468
FINAL	0.669	0.482	0.430
ATTRIBUTION	0.634	0.481	0.555
CHECKLIST	0.669	−0.048	0.299
NEGATIONJUDGE	0.646	0.269	0.284
GRADEDRELEVANCE	**0.686**	0.265	($\star$)
Naive: Length	0.208	0.195	0.222
Naive: Random	-0.072	0.018	-0.143

Across both datasets, NEGATIONJUDGE performed less well, reaching a Kendall score of 0.284. It did not attempt to maximize citation support or incorporate nuggets, which is likely its primary weakness, and its scoring rule is naive. CHECKLIST performed better, especially on nugget measures, suggesting that explicit relevance criteria applied directly by the LLM outperform a rigid hard-coded rule. Tthe naive baselines perform the worst.

GRADEDRELEVANCE obtained the best correlation with AutoJudge nugget coverage, but the method did not finish in time for submission to this challenge.

5 Conclusion

Across our submissions to Rag4Reports two broad takeaways emerge. First, citation faithfulness and topical content coverage (measured via nuggets) appear to be complementary strengths of different judges. Methods that target one axis tend not to lead on the other, and our explicit attempt to combine the two, FINAL, did not dominate either: integrating the two axes into a single judge that is competitive on both remains an open problem.

Second, on a more encouraging note, methods that scored higher on the AutoJudge Pilot development set also tended to score higher on Rag4Reports: the naive baselines came in last on both, and the LLM-based judges led on both. Despite its AUTO-ARGUE-augmented truth on preliminary manual TREC judgments from late 2025, AutoJudge Pilot data serves as a useful proxy for selecting methods to enter into the downstream evaluation.

[1] Test implementation from autojudge-starterkit.

Limitations

Method differences confounded with model and prompt choices. Each method was designed independently by a different author, so differences in Kendall's τ between methods reflect not only the underlying scoring criterion (citation faithfulness vs. nugget coverage, etc.) but also unrelated choices: LLM provider, embedding model, prompt wording, retrieval-set construction, and output post-processing. We did not run ablations to isolate which of these factors drives the observed gaps, so our comparisons are between *systems* rather than between scoring principles.

Single test set. Our only test ranking is the Rag4Reports leaderboard, derived from a single set of human assessments. We cannot say whether the methods that ranked highly here would generalize to other report-generation benchmarks or to non-report RAG tasks. The development set (TREC AutoJudge Pilot v0.2) uses a AUTO-ARGUE-derived reference with very few preliminary human judgments rather than a thoroughly verified human curated truth, so agreement on the development set is itself a noisy signal. GRADEDRELEVANCE did not finish in time for the Rag4Reports submission, so we cannot tell whether its strong AutoJudge Pilot result transfers.

Reproducibility of closed-weight LLMs. All of our LLM-based methods except CITATIONACCURACY rely on commercial APIs (OpenAI `gpt-4o-mini`, `gpt-4.1-nano`, DeepSeek v3.2). Their outputs are non-deterministic and may shift with provider-side updates, so exact replication of our Kendall's τ values is not guaranteed.

Course-project scope. Each method was developed within a single semester course. None has been subjected to the iterative tuning, prompt engineering, or cost analysis that a deployed AutoJudge would require, and our results should be read as a wide first-pass survey rather than a calibrated benchmark of mature methods.

Ethical considerations

Data provenance and anonymization. The datasets used to develop and test our methods originated from the public TREC 2025 evaluation. Participating teams submitted RAG systems. As team identity is immaterial to method comparison, run names are anonymized in both Rag4Reports and the TREC AutoJudge Pilot v0.2. The manual annotations that form the Rag4Reports reference ranking were produced by NIST under their established rules for research involving human subjects.

Risks of LLM-as-a-Judge. The purpose of an AutoJudge is to scale, and in practice often replace, human assessment. An AutoJudge that correlates well with human judges on one benchmark can still embed and amplify biases of its underlying LLM, including length, fluency, and self-preference biases (Section 2). Methods that target a narrow signal, such as the citation-faithfulness focus of our best-performing approach, are auditable but should not be mistaken for general report-quality measures. In consequential settings, automated rankings should not be treated as ground truth without human review.

Compute and access costs. Several of our methods make many LLM calls per topic (e.g., one call per nugget per system, or one call per claim per document). At evaluation scale this incurs monetary cost and energy consumption that we do not measure. Teams without LLM access cannot reproduce these methods on equal footing; future work should consider compute cost alongside Kendall's τ and investigate smaller open-weight alternatives where feasible.

Table 2: Evaluation metrics by team and run, sorted by τ_{gap}. Our team's entries and baselines shown in bold.

Team / Run	τ_{gap}	Kendall	Topic
coordinators / autoargue-f1	*0.470*	*0.636*	*0.607*
unh / citation accuracy	**0.414**	**0.559**	**0.168**
unh / retrieval quality	**0.226**	**0.469**	**0.415**
unh / final	**0.221**	**0.430**	**0.386**
crucible / maxgrade	0.177	0.390	0.356
unh / attribution	**0.173**	**0.291**	**0.306**
crucible / cover4	0.128	0.360	0.190
tiet / f1 (weighted)	0.127	0.334	0.573
unh / negation-judge	**0.109**	**0.284**	**0.260**
crucible / avggrade	0.107	0.355	0.201
ju-nlp-ug / rank-argue-v4	0.099	0.375	0.546
unh / checklist	**0.096**	**0.300**	**0.343**
rgipt / Qwen2-NRB	0.077	0.133	0.221
tiet / nugget coverage	0.075	0.289	0.599
ju-nlp-pg / cite-first	0.026	0.123	0.161
tiet / sentence support	−0.056	0.183	0.247
unh / naive-length	**−0.084**	**0.222**	**0.229**
rgipt / qwen0-5	−0.125	0.154	0.251
rgipt / semiauto-run1	−0.125	0.154	0.251
rgipt / Qwen2-0.5B-NRB	−0.269	−0.297	0.185
unh / naive-random	**−0.277**	**−0.143**	**0.010**

References

Antonia Creswell, Murray Shanahan, and Irina Higgins. 2022. Selection-inference: Exploiting large language models for interpretable logical reasoning.

Kaustubh D. Dhole and Eugene Agichtein. 2026. Rubricrag: Towards interpretable and reliable llm evaluation via domain knowledge retrieval for rubric generation.

Laura Dietz. 2024. A workbench for autograding retrieve/generate systems. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1963–1972.

Laura Dietz, Ian Soboroff, and Coordinators of various TREC tracks. 2025. Trec autojudge pilot data. Licensed under CC BY-SA 3.0. Based on content from TREC DRAGUN, TREC RAG, TREC RAGTIME, their license applies.

Nelson F. Liu, Tianyi Zhang, and Percy Liang. 2023. Evaluating verifiability in generative search engines. In *Findings of the Association for Computational Linguistics: EMNLP 2023*.

Ani Nenkova and Rebecca Passonneau. 2004. Evaluating content selection in summarization: The pyramid method. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pages 145–152, Boston, Massachusetts, USA. Association for Computational Linguistics.

Dani Roytburg, Matthew Bozoukov, Matthew Nguyen, Jou Barzdukas, Mackenzie Puig-Hall, and Narmeen Oozeer. 2026. Are llm evaluators really narcissists? sanity checking self-preference evaluations.

Minzhu Tu, Shiyu Ni, and Keping Bi. 2026. How long reasoning chains influence llms' judgment of answer factuality.

Ellen M. Voorhees. 2004. Overview of the TREC 2003 question answering track. In *Proceedings of the Twelfth Text REtrieval Conference (TREC 2003)*, pages 54–68, Gaithersburg, MD. NIST.

William Walden, Marc Mason, Orion Weller, Laura Dietz, John Conroy, Neil Molino, Hannah Recknor, Bryan Li, Gabrielle Kaili-May Liu, Yu Hou, Dawn Lawrie, James Mayfield, and Eugene Yang. 2026. Auto-argue: Llm-based report generation evaluation.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*.

Chuting Yu, Hang Li, Guido Zuccon, Joel Mackenzie, and Teerapong Leelanupab. 2026. When llm judges inflate scores: Exploring overrating in relevance assessment.

Crucible @ Rag4Reports: Generating Nuggets for Report Generation and Evaluation

Laura Dietz
University of New Hampshire
USA
dietz@cs.unh.edu

Eugene Yang
Johns Hopkins University
USA
eugene.yang@jhu.edu

Abstract

We submit to both tracks of the RAG4Reports challenge with two complementary components: PREFNUGGET, which derives concise nugget banks from pairwise preference judgments between system responses, and CRUCIBLE, a nugget-first pipeline that uses such banks to assemble reports on a given topic. The shared nugget-level representation unifies our approach to report evaluation (Task A) and report generation (Task B).

1 Introduction

Our recent work explored the interplay between LLM-as-a-Judge and Retrieval-augmented Generation systems (RAG) (Dietz et al., 2026c).In particular, we explore different paradigms of automatically generated nuggets (Voorhees, 2003; Nenkova and Passonneau, 2004) in the form of open-ended questions to be used for two purposes:

1. using nuggets in the evaluation of reports via LLM-as-a-Judge,
2. using nuggets to drive the report generation inside a RAG system.

A nugget is a piece of information that should be contained in a good system response. There are different ways of phrasing nuggets, such as via concrete facts or claims that should be mentioned or questions that should be answered. We focus on question-nuggets as prior work found question-nuggets to be better at distinguishing better from best systems than fact nuggets (Farzi and Dietz, 2024a). Question nuggets can either be fact-oriented questions with "known good" gold answers or open-ended questions for which any reasonable answer would count.

The truth. In the experimental setup of Rag4Reports, the truth for both tasks is determined semi-automatically by AUTO-ARGUE (Walden et al., 2025). The evaluation uses manually identified questions with gold answers as nuggets, curated by NIST assessors. Which nuggets are successfully addressed is verified with a Yes/No prompt. Along with verification of faithful citations, this determines the "official" nugget coverage score. A prompt to verify whether a citation supports a sentence of the summary provides the basis for a sentence support score. AUTO-ARGUE's F1 measure is the harmonic mean between the nugget coverage and the sentence support score.

For the report evaluation task (Task A), the official leaderboard of systems previously submitted to TREC 2025 is used as a ground truth.

For the report generation task (Task B), the manual evaluation artifacts collected when assessing the systems submitted to TREC 2025 are applied to evaluate submitted reports.

Dimension	Ours	AUTO-ARGUE
Nugget source	LLM-generated	manually curated
Nugget questions	open-ended	gold answers
Grading	0–5 scale	binary Yes–No
LLM	`gpt-oss-120b`	`llama-3.3-70b`
Grounding	preferences	cited snippets

Table 1: Comparison of our approaches to the AUTO-ARGUE "truth" evaluation pipeline.

Our report evaluation system: PREFNUGGET. The LLM-as-a-judge system we submitted to Task A[1] also uses nuggets, but with critical differences as detailed in Table 1. The most differentiating factor is that nugget banks are automatically derived, then automatically checked against responses—as a result our system runs fully automatically, with no human intervention.[2]

[1]Code for PREFNUGGET evaluation: https://github.com/laura-dietz/prefnugget-starterkit/commits/rag4reports-submission

[2]Despite our submitted approach, the authors believe that evaluation systems must involve human assessors to be reliable (Dietz et al., 2025b).

Proceedings of the First Workshop on Multilingual Report Generation via Retrieval Augmented Generation (RAG4Reports), pages 77–82

```
You are a highly experienced and accurate assessor for TREC.
Select the passage that answers the query better. Just answer 1
or 2, without any explanation or extra verbiage.
If both passages are similar, answer with 0.
```

Field	Description
In: query	Query title, background, narrative
In: passage_1	Passage 1
In: passage_2	Passage 2
Out: better	1, 2, or 0 (tie)

(a) Phase 1: Preference judging. Prompt inspired by Arabzadeh and Clarke (2025); Yu et al. (2026).

```
Compare Winner vs Loser RAG responses for a query. Focus on
relevance, correctness, completeness. From given_exam_questions,
identify or generate questions the Winner addresses much better
than the Loser. Reuse questions where possible. New
differentiating_questions must be brief, atomic questions about
information the Winner handles much better. Avoid generic
quality questions. Make questions self-contained (e.g.,
"Capital of France?" not "The capital?").
```

Field	Description
In: query	Query title, background, narrative
In: winner_passage	Winning response
In: loser_passage	Losing response
In: given_exam_questions	Prior questions
Out: diff_questions	JSON array of new questions

(b) Phase 2: Iterative contrastive nugget extraction. Runs iteratively; previously extracted questions are provided to avoid redundancy. Stops after 20 unique questions.

```
Grade how well a passage answers a specific question. Can the
question be answered based on the available context? Choose one:
- 5: The answer is highly relevant, complete, and accurate.
- 4: The answer is mostly relevant and complete but may have
  minor gaps or inaccuracies.
- 3: The answer is partially relevant and complete, with
  noticeable gaps or inaccuracies.
- 2: The answer has limited relevance and completeness, with
  significant gaps or inaccuracies.
- 1: The answer is minimally relevant or complete, with
  substantial shortcomings.
- 0: The answer is not relevant or complete at all.
```

Field	Description
In: question	Nugget question
In: passage	Text to evaluate
Out: grade	Grade 0–5

(c) Phase 3: Grading prompt from RUBRIC (Farzi and Dietz, 2024a; Dietz, 2024).

```
Given a question on the given topic/background/problem
statement, find the sections in the provided source document
that support and validate the answer to the question. Provide
the supporting section with complete sentences directly from
the document, with enough surrounding context to justify why
the answer is correct. Respond with the extracted text segment
only. Then condense the extracted text into one concise
sentence that clearly demonstrates how the question is
answered, without referring to the source document.
```

Field	Description
In: nugget_text	Question
In: source_document	Source document
In: title_query	Query topic
In: background	Query background
In: problem_statement	Problem statement
Out: <extracted>	Supporting text from document
Out: <condensed>	One-sentence condensation

(d) Crucible Step 3: Report sentence generation.

Figure 1: LLM prompts and their input/output.

Details of the PREFNUGGET approach (Dietz et al., 2026a) are given below.

Our report generation system: CRUCIBLE. As the Retrieval-augmented Generation (RAG) system submitted to Task B,[3]

we build on our CRUCIBLE system, abbreviated "cru", (Dietz et al., 2026b), which uses an automatically generated nugget-bank to drive the generation of long-form responses.

While the CRUCIBLE RAG system originally used automatic nugget banks generated with DOGMATIQ (Li et al., 2026), for this submission we are using it with the nugget banks generated by PREFNUGGET (our Task A submission). We also submit one variant, called "rubric", that uses a generated nugget bank from LLM's internal knowledge without any grounding.

We do not perform an internal validation to selection submissions, but draw on experience with related datasets: TREC NeuCLIR 2024(Lawrie et al., 2025a) for CRUCIBLE and TREC AutoJudge Pilot Dataset v0.2 (Dietz et al., 2025a) for PREFNUGGET. We do not perform internal validation to select submissions, but draw on experience with related datasets: TREC NeuCLIR 2024 (Lawrie et al., 2025b) for CRUCIBLE and TREC AutoJudge Pilot Dataset v0.2 (Dietz et al., 2025a) for PREFNUGGET.

2 Approach Task A: Automatic Report Evaluation

The goal is to derive compact, discriminative nugget banks from pairwise preference signals and the query. Winner-loser comparisons reveal which information distinguishes strong responses from weak ones; questions targeting these differences naturally capture evaluation-relevant criteria.

2.1 Phase 1: Pairwise Preferences

For each topic, we collect pairwise preference judgments over system responses that are to be evaluated. Each response is compared against four others, with pairs selected via stratified sampling (avoiding periodicity). An LLM judge determines which response better answers the query, resulting in winner-loser pairs. We follow the preference judgment paradigm of Arabzadeh and Clarke (2025), but for this submission, we permitted the

[3]Code for CRUCIBLE report generation: https://github.com/laura-dietz/scale25-crucible/commit/rag4reports-submission

system to indicate ties, based on findings from Yu et al. (2026). After the submission we realized that permitting ties actually reduces the performance slightly.

2.2 Phase 2: PREFNUGGET Banks

For each winner-loser pair where response A is better than response B, we use the prompt in Figure 1b to extract at most two factual questions that capture how response A is better. To avoid redundant nuggets (which are costly to reduce later (Li et al., 2024)), we provide previously extracted questions as candidates for selection, and ask the LLM to generate novel questions only when necessary. Extraction terminates for a topic when either the maximum budget of unique questions has been reached (here: 20) or a maximum number of pairs (here: 100) have been processed.

2.3 Phase 3: Response Grading

The final phase evaluates all RAG responses against the derived nugget bank. Each nugget question is scored on a six-point scale (0–5) indicating how well the provided RAG response answers it. When grading response texts, the nugget-grade is obtained by scoring the nugget against the response text, yielding one grade per nugget.

We predict an evaluation score for each response by one of these measures:

maxgrade: The nugget for which the RAG response obtained the highest grade is used as an evaluation score. This approach was proposed in Farzi and Dietz (2024b).
avggrade: The average grade the response obtained across all nuggets.
cover4: The fraction of nuggets for which the response obtained a grade of 4 or 5.

3 Approach Task B: Multilingual Report Generation

CRUCIBLE is a RAG pipeline which starts by generating question-nuggets and uses them to guide retrieval, extraction, and assembly as follows.

Step 1: Nugget ideation. We begin by generating a bank of open-ended question nuggets. Here these are obtained from PREFNUGGET or generated from the LLM's internal knowledge (`rubric`).

Step 2: Retrieval. While nuggets can be used to retrieve relevant documents (as in CRUXX, Ju et al. (2025)), here we use a one-shot retrieval stage using the multi-lingual retrieval model MILCO (Nguyen et al., 2025) based on a search query that concatenates problem statement, title, and background.

Step 3: Scanning and generation. Using the nugget bank from Step 1, we scan retrieved documents[4] for passages that directly answer each nugget. Using the prompt in Figure 1d, we (1) locate a supporting passage, (2) generate a concise self-contained sentence, and (3) record the LLM's token-likelihood as the extraction confidence.

The original paper makes reference to a verification step which is prone to creating a circularity with AUTO-ARGUE, which we did not use here.

Step 4: Sentence selection. We rank the remaining candidates for each nugget by the extraction confidence and choose the top k sentences, with $k = 1$ for short reports and $k = 5$ for long reports. This ensures that each sentence is tied to exactly one citation.

Step 5: Assembly. Selected sentences are concatenated into a report. Repeated sentences (same stopped/stemmed text) are omitted. Because every sentence is self-contained and atomic, the order of sentences does not affect the readability. Every sentence cites exactly one document.

3.1 Variations

We submit three variations of CRUCIBLE to Rag4Reports.

cru-prefnu: Uses CRUCIBLE on a nugget bank of 20 open-ended questions from PREFNUGGET (as submitted to Task A).
cru-prefnu-extract: As previous, but instead of using abstractive summarization, the supporting passage is used as-is.
cru-rubric: Uses CRUCIBLE on a nugget bank of 20 open-ended questions that is generated only from the LLM's internal knowledge without any form of grounding. (The approach is similar to the RUBRIC as introduced by Farzi and Dietz (2024b), albeit with an adjusted prompt and a more recent LLM.)

We remark that while GINGER (Lajewska and Balog, 2025) also uses nuggets, they define nuggets to be clusters of sentences extracted from source documents. In contrast, CRUCIBLE operates by

[4] Segmented into 1000 character chunks, split at sentence boundary.

Team / Run	τ_{gap}	Kendall	Topic
coordinators / autoargue-f1	*0.470*	*0.636*	*0.607*
unh / citation accuracy	0.414	0.559	0.168
unh / retrieval quality	0.226	0.469	0.415
unh / final	0.221	0.430	0.386
crucible / maxgrade	**0.177**	**0.390**	**0.356**
unh / attribution	0.173	0.291	0.306
crucible / cover4	**0.128**	**0.360**	**0.190**
tiet / f1 (weighted)	0.127	0.334	0.573
unh / negation-judge	0.109	0.284	0.260
crucible / avggrade	**0.107**	**0.355**	**0.201**
ju-nlp-ug / rank-argue-v4	0.099	0.375	0.546
unh / checklist	0.096	0.300	0.343
rgipt / Qwen2-NRB	0.077	0.133	0.221
tiet / nugget coverage	0.075	0.289	0.599
ju-nlp-pg / cite-first	0.026	0.123	0.161
tiet / sentence support	−0.056	0.183	0.247
unh / naive-length	−0.084	0.222	0.229
rgipt / qwen0-5	−0.125	0.154	0.251
rgipt / semiauto-run1	−0.125	0.154	0.251
rgipt / Qwen2-0.5B-NRB	−0.269	−0.297	0.185
unh / naive-random	−0.277	−0.143	0.010

Table 2: Evaluation metrics by team and run, sorted by τ_{gap}. Our team's entries shown in bold.

Team / Run	Sent.	Nugget	F1
crucible / cru-prefnu-extract	**0.965**	**0.470**	**0.586**
genaius / cluster-gpt4	0.814	0.443	0.546
genaius / trial	0.794	0.439	0.538
crucible / cru-rubric	**0.783**	**0.447**	**0.522**
genaius / cluster-gpt5	0.662	0.449	0.501
crucible / cru-prefnu	**0.684**	**0.425**	**0.480**
amu / bge	0.828	0.340	0.435
amu / qwen4b	0.833	0.287	0.383
amu / qwen8b	0.789	0.276	0.350
ju-nlp-pg / ver3-taskb	0.214	0.378	0.231
sgd / subtask2-local-bm25	0.934	0.149	0.228
sgd / subtask2-test-2	0.934	0.149	0.228
ug-tiet-nlp / efsg-t5	0.612	0.126	0.182
ug-tiet-nlp / efsg-t4	0.327	0.065	0.097
ju-nlp-pg / ver1-submission	0.036	0.469	0.046
ju-nlp-pg / ver2-taskb	0.036	0.469	0.046

Table 3: Sentence support, nugget coverage, and F1 by team and run, sorted by F1. Our team marked in bold.

first identifying question-nuggets, then extracting sentences from source documents.

4 Results

All results are generated with `gpt-oss-120b` as the underlying LLM. Results are provided by Rag4Report coordinators via the TIRA leaderboard.[5] Overall, we find that our approaches are working reasonably well. Using PREFNUGGET as Auto-Judge (Task A), we find that the maxgrade approach demonstrates medium-well agreement with manual "truth" leaderboards in Kendall's tau on the F1 measure of 0.39, and τ_{GAP} of 0.177. This is a good result, given that the truth measures emphasize citation accuracy verification, which is not captured in our auto judge approach.

We find that other variants of PREFNUGGET work less well than maxgrade, which is a finding that is in line with other experimental studies (Dietz et al., 2026a; Farzi and Dietz, 2024a).

When PREFNUGGET is used along with our RAG system CRUCIBLE to generate long-form reports (Task B), we obtain the best results with the extractive summarization approach `cru-prefnu-extract`. As sentences are taken verbatim from the cited document, it obtains a near-perfect sentence support score of 0.965. With 0.470 the nugget coverage is slightly above the abstractive approach `cru-prefnu` (0.425) even though the same underlying nugget bank is used to drive the generation. The explanation is that the generated nuggets are not a perfect match with the manually curated nuggets that form the ground truth, but that they match sentences that include additional relevant information. In contrast, the abstractive generation is instructed to only focus on representing the contained answer to the nugget question. So the strength of this approach lies in detecting good sentences with our nugget bank, which also includes other information that was found to be more essential by manual assessors.

We note that among the two abstractive methods, the RUBRIC-style nugget bank, which does not use any form of grounding, performs better than the PREFNUGGET approach. This implies that grounding can miss relevant information, especially when using a strong modern LLM like `gpt-oss-120b`. In order to improve upon strong LLMs, the grounding would need to include knowledge that is not already known by the LLM.

5 Conclusion

We demonstrate that nugget-bank generation methods can yield both high-quality LLM-as-a-Judge approaches as well as retrieval-augmented generation systems. Our results show that modern LLMs can successfully generate nugget banks without grounding, and extractive summarization approaches are still competitive.

[5] archive.tira.io/task-overview/rag4reports/task-a-report-evaluation-20260403-training

Limitations

Our submission is limited by the circularity risk inherent in using LLM-generated nuggets to evaluate or guide LLM-generated reports (Dietz et al., 2026c). This risk is particularly relevant when the same model family, prompts, or intermediate representations are used across generation and evaluation. While our methods are fully automatic and therefore easy to reproduce, they should not be understood as replacing human assessment for reliable evaluation.

Only three submissions were accepted by the challenge. In hindsight, adding a rubric-based experiment to the Task A submission would have led to more conclusive findings on whether grounding is necessary. Our current results therefore compare preference-grounded nuggets against rubric-style nuggets only in Task B, not in Task A.

Ethical considerations

Datasets used to develop and test methods originated from a public competition at TREC 2025. Participating teams submitted methods; as their identities are immaterial, their run names are anonymized (both in Rag4Reports and in TREC AutoJudge). Manual annotations were created by NIST under rules of research with human subjects.

Because of the circularity risk, bad actors can misuse these findings to obtain inflated evaluation results in other data challenges, especially when insider knowledge of the evaluation approach is available. This concern is not specific to our system, but applies broadly to automatic evaluation settings where participants can infer or approximate the evaluation procedure. We therefore recommend that benchmark organizers keep human oversight, use multiple independent evaluation signals, and avoid relying on a single automatically generated nugget bank as the sole source of truth.

References

Negar Arabzadeh and Charles L. A. Clarke. 2025. A human-ai comparative analysis of prompt sensitivity in llm-based relevance judgment. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '25)*. ACM.

Laura Dietz. 2024. A workbench for autograding retrieve/generate systems. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1963–1972.

Laura Dietz, Naghmeh Farzi, Eugene Yang, and Dawn Lawrie. 2026a. Too many questions: Deriving concise and effective nugget banks. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '26)*, SIGIR '26, Melbourne, VIC, Australia. ACM.

Laura Dietz, Bryan Li, Gabrielle Liu, Jia-Huei Ju, Eugene Yang, Dawn Lawrie, William Walden, and James Mayfield. 2026b. Incorporating Q&A nuggets into retrieval-augmented generation. In *Proceedings of the 48th European Conference on Information Retrieval (ECIR 2026)*.

Laura Dietz, Bryan Li, Eugene Yang, Dawn Lawrie, William Walden, and James Mayfield. 2026c. Insider knowledge: How much can rag systems gain from evaluation secrets? In *Proceedings of the 48th European Conference on Information Retrieval (ECIR 2026)*.

Laura Dietz, Ian Soboroff, and Coordinators of various TREC tracks. 2025a. Trec autojudge pilot data. Licensed under CC BY-SA 3.0. Based on content from TREC DRAGUN, TREC RAG, TREC RAGTIME, their license applies.

Laura Dietz, Oleg Zendel, Peter Bailey, Charles LA Clarke, Ellese Cotterill, Jeff Dalton, Faegheh Hasibi, Mark Sanderson, and Nick Craswell. 2025b. 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)*, pages 218–229.

Naghmeh Farzi and Laura Dietz. 2024a. Exam++: Llm-based answerability metrics for ir evaluation. In *Proceedings of LLM4Eval: The First Workshop on Large Language Models for Evaluation in Information Retrieval*.

Naghmeh Farzi and Laura Dietz. 2024b. Pencils down! automatic rubric-based evaluation of retrieve/generate systems. In *Proceedings of the 2024 ACM SIGIR International Conference on Theory of Information Retrieval*, pages 175–184.

Jia-Huei Ju, Suzan Verberne, Maarten de Rijke, and Andrew Yates. 2025. Controlled retrieval-augmented context evaluation for long-form rag. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 21102–21121.

Weronika Lajewska and Krisztian Balog. 2025. Ginger: Grounded information nugget-based generation of responses. In *Proceedings of the 48th International ACM SIGIR Conference (SIGIR '25)*. SIGIR 2025 paper.

Dawn Lawrie, Sean MacAvaney, James Mayfield, Paul McNamee, Douglas W Oard, Luca Soldaini, and Eugene Yang. 2025a. Overview of the TREC 2024 NeuCLIR track. In *Proceedings of TREC*.

Dawn Lawrie, Sean MacAvaney, James Mayfield, Paul McNamee, Douglas W Oard, Luca Soldaini, and Eugene Yang. 2025b. Overview of the TREC 2024 neuclir track.

Bryan Li, William Walden, Yu Hou, Gabrielle Kaili-May Liu, Dawn Lawrie, Jame Mayfield, Eugene Yang, Chris Callison-Burch, and Laura Dietz. 2026. Dogmatiq: Automated generation of question-and-answer nuggets for report evaluation. *Preprint*, arXiv:2605.04458.

Haitao Li, Qian Dong, Junjie Chen, Huixue Su, Yujia Zhou, Qingyao Ai, Ziyi Ye, and Yiqun Liu. 2024. Llms-as-judges: a comprehensive survey on llm-based evaluation methods. *arXiv preprint arXiv:2412.05579*.

Ani Nenkova and Rebecca Passonneau. 2004. Evaluating content selection in summarization: The pyramid method. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2004)*, pages 145–152, Boston, Massachusetts. Association for Computational Linguistics.

Thong Nguyen, Yibin Lei, Jia-Huei Ju, Eugene Yang, and Andrew Yates. 2025. Milco: Learned sparse retrieval across languages via a multilingual connector. *arXiv [cs.IR]*.

Ellen M. Voorhees. 2003. Overview of the TREC 2003 question answering track. In *Proceedings of the Twelfth Text REtrieval Conference (TREC 2003)*, Gaithersburg, Maryland. NIST.

William Walden, Orion Weller, Laura Dietz, Bryan Li, Gabrielle Kaili-May Liu, Yu Hou, and Eugene Yang. 2025. Auto-ARGUE: LLM-based report generation evaluation. *arXiv preprint arXiv:2509.26184*.

Chuting Yu, Hang Li, Guido Zuccon, Joel Mackenzie, and Teerapong Leelanupab. 2026. When llm judges inflate scores: Exploring overrating in relevance assessment. *arXiv preprint arXiv:2602.17170*.

GenAIus at RAG4Reports 2026: Citation-Aware Compression for Multilingual Report Generation

Reyyan Yeniterzi
GenAIus Technologies
reyyan@genaius.tech

Suveyda Yeniterzi
GenAIus Technologies
suveyda@genaius.tech

Abstract

This paper describes the GenAIus submission to RAG4Reports 2026 Multilingual Report Generation Task. Our system builds on our earlier TREC RAGTIME pipeline, reusing the evidence preparation stages for overlapping topics, including question generation, multilingual retrieval, nugget generation, and nugget clustering. For RAG4Reports, we focused on the final generation stage and tested a citation-aware compression strategy: generating the long report first from clustered evidence nuggets and then deriving the short report from it, rather than generating both length conditions independently. Our baseline run, which followed the original TREC-style setup, ranked third overall. Our best run, `genaius-cluster-gpt4`, ranked second overall with an F1 score of 0.5456, achieving the best balance among our submissions between nugget coverage and sentence support. The results suggest that citation-aware compression is a promising strategy for length-constrained, citation-grounded report generation.

1 Introduction

Retrieval-augmented generation (RAG) is increasingly used for producing grounded responses from large document collections. However, long-form report generation poses challenges beyond short-form question answering: systems must retrieve evidence across many documents, organize it into a coherent narrative, cover multiple aspects of the information need, and provide reliable citations for generated claims. These challenges are further amplified in multilingual settings, where relevant evidence may appear in different languages.

The RAG4Reports 2026 Multilingual Report Generation Task directly targets this setting by evaluating systems that generate citation-grounded reports from a multilingual news collection. Our system builds on the question-driven retrieval and evidence-based synthesis framework developed for our TREC 2025 RAGTIME submission (Yeniterzi and Yeniterzi, 2025). Since the RAG4Reports topics overlapped with topics we had previously processed, we reused the earlier question-generation and retrieval outputs and focused on testing generation-side modifications that were not fully explored in our TREC submission.

In particular, we experimented with generating the long report first and then producing the short report through citation-aware compression, rather than generating both length conditions independently. Our best run, `genaius-cluster-gpt4`, ranked second on the official task leaderboard, suggesting that this strategy is effective for balancing nugget coverage and sentence-level citation support.

2 Task Description

RAG4Reports 2026 focuses on retrieval-augmented generation for long-form, citation-grounded report generation. It is a direct continuation of the TREC 2025 RAGTIME track (Lawrie et al., 2025), which studied multilingual, citation-grounded report generation over news documents. As in RAGTIME, the task requires systems to address cross-lingual evidence access, evidence selection, long-form synthesis, and faithful attribution. Compared with short-form question answering, this setting places greater emphasis on retrieving and organizing evidence across multiple documents, producing coherent and information-dense reports, and maintaining explicit sentence-level grounding in source evidence.

Multilingual Report Generation Task uses a news document collection in English, Chinese, Russian, and Arabic, sampled from Common Crawl News from 2021 to 2024. In this task, each report request is written in English and includes contextual back-

Proceedings of the First Workshop on Multilingual Report Generation via Retrieval Augmented Generation (RAG4Reports), pages 83–88

ground about the requester as well as a problem statement describing the information need. Systems are required to generate a report in the same language as the request, with every sentence supported by citations to source documents.

3 System Overview

Our system builds on the question-driven retrieval and evidence-structuring framework developed for our TREC 2025 RAGTIME submission (Yeniterzi and Yeniterzi, 2025). Since RAG4Reports is a direct continuation of the RAGTIME setting, it provided an opportunity to examine generation-side design choices that we were not able to fully explore in the original TREC submission.

For RAG4Reports, we reused the intermediate artifacts produced by our RAGTIME pipeline through the evidence-organization stage. Specifically, we reused the generated search questions, multilingual retrieval results, generated information nuggets, and nugget clusters for the overlapping report requests. Our RAG4Reports-specific modifications therefore begin only at the final report generation stage. Figure 1 provides an overview of the resulting pipeline.

3.1 Evidence Preparation Pipeline

The reused part of the pipeline consists of four stages: question generation, multilingual retrieval, nugget generation, and nugget clustering. We summarize these stages briefly for completeness and refer readers to our TREC RAGTIME system paper for the full prompts and implementation details (Yeniterzi and Yeniterzi, 2025).

First, each broad report request is decomposed into a set of focused search questions. This step transforms a complex long-form report request into multiple targeted information needs, helping the system cover different aspects of the topic more systematically. The generated questions serve as the main intermediate representation for retrieval.

Second, multilingual evidence is retrieved for each generated question. The retrieved documents were then merged across questions, with documents appearing for multiple questions receiving higher aggregate scores. This aggregation emphasizes sources relevant to several dimensions of the report request.

Third, the retrieved evidence is converted into concise factual nuggets. For each generated question, the system uses the top-ranked retrieved document to generate information nuggets conditioned on the original report request. Each nugget captures a single relevant fact or aspect of the topic and serves as an atomic evidence unit for downstream synthesis.

Fourth, the generated nuggets are grouped into thematic clusters using an LLM. The clustering step groups semantically related nuggets into distinct aspects, subtopics, or dimensions of the report. Irrelevant nuggets may be discarded during this step, and each cluster is assigned a descriptive label to improve interpretability. These clustered nuggets provide the structured evidence representation used by the final generation stage.

In RAG4Reports, the report requests were a subset of the topics we had already processed in our TREC RAGTIME experiments. We therefore reused the corresponding outputs from all four stages above. This allowed us to hold evidence preparation fixed and focus on the effect of modifying the final report generation strategy.

3.2 RAG4Reports-Specific Report Generation

Both TREC RAGTIME and RAG4Reports require reports under two length conditions: a shorter report and a longer report. In our TREC experiments, we generated both versions independently from the clustered evidence. We observed, however, that our longer reports generally performed better than our shorter reports, likely because the longer setting allowed the system to preserve more supporting evidence, cover more facets of the topic, and maintain smoother narrative organization.

For RAG4Reports, we therefore changed only the final generation strategy. Starting from the reused nugget clusters, we first generated the long report. Each generated sentence was required to be grounded in the clustered evidence and associated with supporting citations. We then produced the short report by compressing the generated long report, rather than generating the short report directly from the clustered nuggets.

The compression step is designed as citation-aware editing rather than generic summarization. The model is instructed to preserve only the highest-priority factual claims needed to answer the request, merge related claims into compact sentences, remove lower-priority details, and retain only citations that support the preserved claims. This separates comprehensive evidence synthesis from length-constrained content selection. The full prompt used for this compression step is shown in

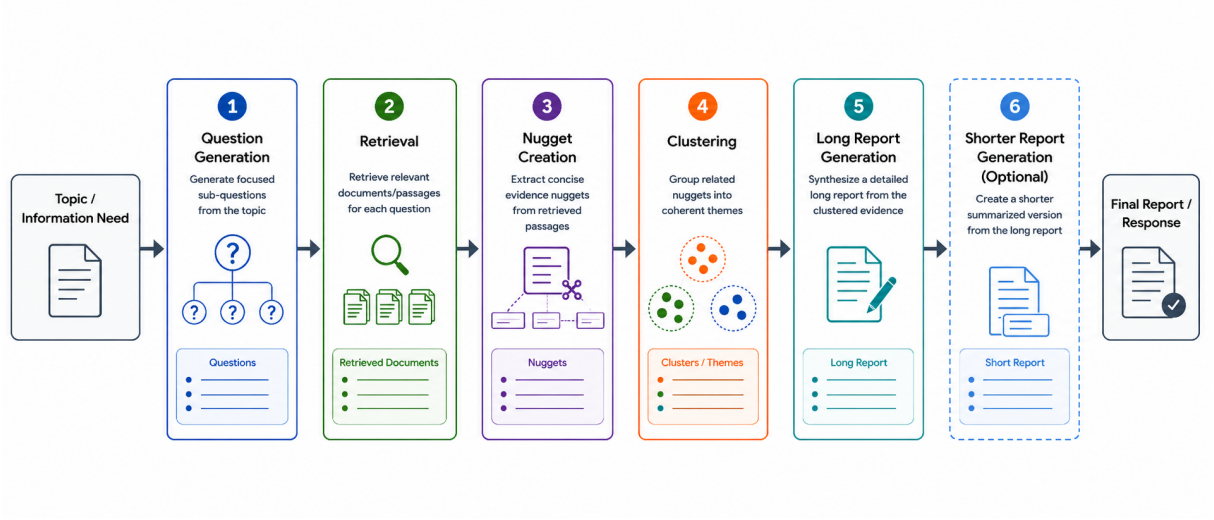

Figure 1: Overview of our RAG4Reports pipeline

Appendix A, Figure 2.

4 Experimental Configuration

We submitted three runs to the official RAG4Reports Multilingual Report Generation Task leaderboard. All runs reused the same evidence-preparation artifacts from our TREC RAGTIME pipeline: generated questions, multilingual retrieval results, information nuggets, and nugget clusters. The submissions therefore differed only in the final report-generation strategy and in the model used for the RAG4Reports-specific generation stages.

4.1 Submitted Runs

`trial`: The `trial` run serves as a baseline approach. It corresponds to the same generation setup used in our earlier TREC system run, `genaius-cluster` (Yeniterzi and Yeniterzi, 2025). In this configuration, the short and long reports were generated independently from the retrieved evidence using GPT-4o (`2024-11-20`) (OpenAI, 2024). We use this run as a comparison point for evaluating the compression-based strategy.

`genaius-cluster-gpt4`: This run also used GPT-4o (`2024-11-20`) for the RAG4Reports-specific generation stages. The long reports were the same as those produced in the `trial` run, generated from clustered evidence nuggets. The short reports, however, were derived from the long reports using citation-aware compression.

`genaius-cluster-gpt5`: This run used the same cluster-based generation and compression strategy as `genaius-cluster-gpt4`, but replaced GPT-4o with GPT-5.4 (`2026-03-05`) (OpenAI, 2026) in the generation stages. Since the clustered nuggets were fixed across runs, differences between these two submissions primarily reflect differences citation-aware generation and compression behavior.

No task-specific fine-tuning or parameter updates were performed in any run. All model calls were conducted in a prompting-based setting, and no additional external datasets were used.

4.2 Evaluation Metrics

RAG4Reports Multilingual Report Generation Task uses the Auto-ARGUE framework (Walden et al., 2026) to evaluate multilingual report generation with citation-aware quality metrics. The main reported measures are nugget coverage, sentence support, and their combined F1 score.

Nugget coverage measures how well the generated report captures relevant information units expected for the report request. Sentence support measures whether generated sentences are supported by the cited source documents. The F1 score combines these two dimensions, rewarding systems that balance information coverage with citation-grounded faithfulness.

These metrics directly reflect the goals of our system design: the cluster-based long-report generation stage aims to improve coverage by organizing evidence into factual units and themes, while the compression-based short-report generation stage aims to preserve high-priority supported claims under a strict length constraint.

Rank	Run	Sent. Supp.	Nugget Cov.	F1
3	`trial`	0.7936	0.4395	0.5382
2	`cluster-gpt4`	**0.8140**	0.4428	**0.5456**
5	`cluster-gpt5`	0.6620	**0.4487**	0.5006

Table 1: Official RAG4Reports Multilingual Report Generation Task results for our submitted runs. "Rank" refers to the rank within the overall task leaderboard. "Sent. Supp." denotes sentence support and "Nugget Cov." denotes nugget coverage.

5 Results

Table 1 reports the official RAG4Reports Multilingual Report Generation Task leaderboard results for our three submissions. The `trial` run serves as our baseline because it follows the original TREC-style setup, where the short and long reports were generated independently from the evidence. Despite not using the new compression-based strategy, this run ranked third overall with an F1 score of 0.5382. This strong result shows that the reused question-driven retrieval, nugget generation, clustering, and independent report-generation pipeline remained highly competitive in the RAG4Reports setting.

The `genaius-cluster-gpt4` run achieved our best overall performance, ranking second with an F1 score of 0.5456. Compared with the `trial` baseline, it improved both sentence support, from 0.7936 to 0.8140, and nugget coverage, from 0.4395 to 0.4428. This suggests that generating the long report first and then deriving the short report through citation-aware compression helped preserve important information while improving sentence-level grounding. Among our submissions, this run provided the best balance between coverage and citation faithfulness.

The `genaius-cluster-gpt5` run ranked fifth with an F1 score of 0.5006. It achieved the highest nugget coverage among our submissions, with a score of 0.4487, but its lower sentence support score of 0.6620 reduced the final F1. This indicates that the GPT-5 variant preserved or introduced more expected information units, but its generated sentences were less consistently supported by the cited evidence. The result highlights an important tradeoff in citation-grounded report generation: increasing coverage is not sufficient if citation support decreases.

Overall, our results show that the original TREC-style baseline was already strong, while the compression-based variant provided the best improvement over it. The comparison across runs suggests that effective RAG4Reports systems must balance two competing goals: covering relevant nuggets and maintaining reliable sentence-level citation support.

6 Conclusion

We described the GenAIus system for RAG4Reports 2026 Multilingual Report Generation Task. Our approach reused the evidence preparation stages from our TREC RAGTIME pipeline, including question generation, multilingual retrieval, nugget generation, and clustering, and focused on modifying the final report-generation stage.

Our main experiment was to generate the long report first and then derive the short report through citation-aware compression. The original TREC-style baseline already performed strongly, ranking third overall, while the compression-based variant ranked second and achieved the best balance between nugget coverage and sentence support among our submissions.

These results suggest that citation-aware compression can be an effective strategy for length-constrained, citation-grounded report generation. In future work, we plan to study stronger citation-verification methods, and more systematic ablations of the final generation stage, with the goal of improving nugget coverage and sentence support jointly rather than optimizing one at the expense of the other.

References

Dawn Lawrie, Sean MacAvaney, James Mayfield, Luca Soldaini, Eugene Yang, and Andrew Yates. 2025. Overview of the TREC 2025 RAGTIME track. In *Proceedings of the 34th Text REtrieval Conference (TREC 2025)*.

OpenAI. 2024. Gpt-4o. https://developers.openai.com/api/docs/models/gpt-4o. Accessed: 2026-03-10.

OpenAI. 2026. Gpt-5.4. https://developers.openai.com/api/docs/models/gpt-5.4. Accessed: 2026-03-10.

William Walden, Marc Mason, Orion Weller, Laura Dietz, John Conroy, Neil Molino, Hannah Recknor, Bryan Li, Gabrielle Kaili-May Liu, Yu Hou, Dawn Lawrie, James Mayfield, and Eugene Yang. 2026. Auto-argue: Llm-based report generation evaluation. *Preprint*, arXiv:2509.26184.

Suveyda Yeniterzi and Reyyan Yeniterzi. 2025. Question-driven multilingual retrieval and report generation for the ragtime track at trec 2025. In *Proceedings of the 34th Text REtrieval Conference (TREC 2025)*.

A Appendix

```
You are an expert editor for evidence-grounded RAG reports.

Objective:
Rewrite a previously generated report into a substantially shorter version, preserving only the
↪  highest-priority factual content, narrative coherence, and citation faithfulness.

The input report is approximately {original_length_words} words.
The maximum allowed output length is {length} words.
The compression ratio is approximately {compression_ratio}.

Your Task:
Create a shorter report by selecting, merging, deleting, and rewriting only the most important
↪  supported claims.

Mandatory Compression Policy:
- This task is not light editing; it is aggressive selection under a strict word budget.
- You must discard at least 50% of the original report content.
- The final output must contain no more than {length} words.
- If there is a conflict between coverage and length, length compliance wins.
- Do not preserve all claims.
- Do not preserve all themes equally.
- Do not preserve all citations.
- If uncertain whether to keep a claim, omit it.

Content Selection Rules:
- Use only information already present in the Generated Report.
- Do not add new facts, assumptions, explanations, or outside knowledge.
- Preserve only claims directly needed to answer the Description.

Compression Strategy:
1. Identify the main themes required by the Description.
2. Select only the highest-value claims under each theme.
3. Merge closely related claims into compact sentences.
4. Rewrite sentences with high information density.
5. Stop adding content once the report approaches {length}.
6. If the report is still too long, delete lower-priority claims rather than making sentences
↪  longer.

Citation Rules:
- Each output sentence may contain one or more citations.
- If a sentence merges claims from multiple original sentences, include only the citations that
↪  support the retained claims.
- Rank citations by importance, placing the citation supporting the main claim first.
- Remove duplicate citations.
- Do not invent, alter, or normalize citation IDs.
- Do not include citations for claims that were removed.

Final Deletion Pass:
Before output, silently perform a deletion pass:
- Remove any sentence that is not directly necessary to answer the Description.
- Remove any sentence whose main point is already implied by another sentence.
- Remove any low-value detail included mainly for completeness.
- Remove any citation whose supported claim was deleted.
- If the result is above {length}, revise again until it is within {length} words.

Output Requirements:
- Return only a valid JSON list.
- Each object must contain:
  - "text": one coherent sentence in the shortened report.
  - "citations": a ranked list of citation IDs, for example ["3_1_1", "4_2_1"].
- Do not include comments, markdown, explanations, word counts, or metadata outside the JSON
↪  list.
- The final list must collectively contain no more than {length} words.
```

Figure 2: Prompt for citation-aware compression of long reports into short reports

AMU at RAG4Reports 2026 Task B: A Practical Multilingual RAG Pipeline for Citation-Grounded Reports

Maciej Czajka[1,2,3], Piotr Jabłoński[1,2,3], Mateusz Czajka[1,2,3], Konrad Pierzyński[1,2,3], Krzysztof Jassem[1,3]

[1]Adam Mickiewicz University, Poznań, Poland, [2]AMUAI, Poznań, Poland, [3]Center for Artificial Intelligence, Poznań, Poland

Emails: maciej.czajka@amu.edu.pl, piotr.jablonski@amu.edu.pl, matcza11@st.amu.edu.pl konrad.pierzynski@amu.edu.pl, krzysztof.jassem@amu.edu.pl

Abstract

This system paper presents AMU's submission to RAG4Reports 2026 Task B: a practical multilingual retrieval-augmented generation pipeline for evidence-supported report generation. The system combines full-query retrieval, optional query rewriting, dense retrieval with Qdrant, cross-encoder reranking, diversity-aware context selection, and structured generation. The best submitted run uses `BAAI/bge-m3` embeddings, `BAAI/bge-reranker-v2-m3` reranking, and `gpt-5.1` generation with medium reasoning effort, using a partial-coverage prompting strategy. On the official leaderboard, it achieved `F1`=0.4351, `sentence_support`=0.8280, and `nugget_coverage`=0.3403, indicating that the generated reports were well grounded but only partially comprehensive.

1 Introduction

RAG4Reports Task B requires systems to generate long-form, citation-grounded reports from a multilingual news collection. Unlike short-form question answering, the task requires a system to retrieve evidence across several subtopics, synthesize the evidence into a coherent report, and attach citations that support individual factual statements. This makes the task a useful stress test for retrieval-augmented generation (RAG) pipelines (Lewis et al., 2020): a system can be highly faithful to the retrieved context and still fail to cover enough of the expected information.

This paper describes AMU's practical system for Task B. The pipeline was designed as a reproducible multilingual baseline rather than as a task-specific set of hand-written rules. It uses dense retrieval over chunked documents, optional query rewriting, cross-encoder reranking, a diversity-aware context assembly strategy, and constrained JSON generation. The final submitted configuration combines `BAAI/bge-m3` embeddings (`BGE-emb`) (Chen et al., 2024), a `BAAI/bge-reranker-v2-m3` reranker (`BGE-rer`) (Li et al., 2023), and `gpt-5.1` with medium reasoning effort (`gpt-5.1-mid`) for generation, together with a partial-coverage prompt that instructs the model to answer from available evidence while explicitly marking unsupported subtopics.

The main finding is that citation grounding and information coverage behaved differently. Several variants obtained high sentence-level support, but the official F1 was limited by `nugget_coverage`. The best AMU submission reached `sentence_support` of 0.8280 and F1 of 0.4351, while `nugget_coverage` remained 0.3403. This suggests that the system was conservative and well grounded, but did not retrieve or synthesize all nuggets expected by the evaluator.

2 Task and Evaluation

Each Task B instance contains a report request with a title, a user background, and a problem statement. The system must return a structured JSON object whose entries contain report text and citations to source documents. The task is multilingual: requests and evidence may involve different languages, and the generated report must follow the requested output language.

Official evaluation is based on Auto-ARGUE (Walden et al., 2026), using human-curated nuggets and citation-aware diagnostics described in the RAG4Reports shared-task materials (rag, 2026). The official primary score is F1 over `sentence_support` and `nugget_coverage`. Sentence support measures whether generated claims are supported by the cited evidence, while nugget coverage measures how much expected information is covered by the report. Internal citation-oriented diagnostics were additionally tracked on development and full-set runs: `citation_support` and

Proceedings of the First Workshop on Multilingual Report Generation via Retrieval Augmented Generation (RAG4Reports), pages 89–93

Component	Best submitted setting
Query text	Full request: title, background, problem statement
Query rewriting	`gpt-4o-mini-2024-07-18`
Index	Qdrant over chunks
Embedding model	`BGE-emb`
Retrieval depth	top 20
Reranker	`BGE-rer`, top 20
Context selection	Diversity-first chunk selection
Context budget	5 chunks
Generator	`gpt-5.1-mid`
Prompting	Partial-coverage prompt with JSON output

Table 1: Best submitted configuration. Alternative submitted runs used the same high-level pipeline but different reranking stacks.

`correctly_cited_sentences`. These internal metrics were used for model selection but were not optimized directly against the official leaderboard.

3 System Description

3.1 Pipeline Overview

The system consists of 6 stages: query construction, query rewriting, dense retrieval, reranking, context assembly, and structured generation. Table 1 summarizes the configuration used by the best official submission.

3.2 Query Construction and Rewriting

Two query formulations were considered. The *short* formulation concatenates only the title and the problem statement. The *full* formulation also includes the user background. The full formulation was used in the submitted systems because many report requests contain constraints or perspective-setting information in the background field. Removing this field often makes the retrieval query underspecified.

Query rewriting is an optional preprocessing step that converts the full request into a concise search query. The rewriter is instructed to preserve names, dates, numbers, locations, constraints, and the input language, while removing style or persona information. It is also explicitly instructed not to add new facts. In development experiments, rewriting improved the baseline full-query pipeline on the 20-topic subset (Table 2).

3.3 Retrieval and Reranking

Documents were split into chunks and embedded with either `BGE-emb` or, in selected development runs, a `Qwen/Qwen3-Embedding-8B` embedding model (`Qwen8B-emb`) (Zhang et al., 2025). Vectors were stored in Qdrant and retrieved using dense similarity search. Retrieved candidates were then reranked with a cross-encoder, using either `BGE-rer`, `Qwen/Qwen3-Reranker-4B` (`Qwen4B-rer`), or `Qwen/Qwen3-Reranker-8B` (`Qwen8B-rer`).

Most experiments used the `BGE-emb` index because it was the most stable retrieval setup. However, selected development runs also changed the embedding model, reranker, and prompt family at the same time. Therefore, the late-stage model-stack results are not presented as clean one-factor ablations. Instead, query, retrieval-depth, context-budget, and final-stack comparisons are separated.

3.4 Context Assembly

The context assembly stage selects a small number of chunks for the generator. A naive score-only strategy can fill the prompt with several chunks from the same document, which is undesirable for broad report requests. A diversity-first strategy was therefore implemented. Given a reranked list and a context budget K, the first pass selects at most one chunk per document while scanning the reranked candidates from top to bottom. If fewer than K chunks are selected, the remaining slots are filled by the highest-ranked leftover chunks.

The strategy is intentionally lightweight: it does not explicitly model subtopics or cluster evidence. Nevertheless, it reduces prompt domination by a single document and encourages broader evidence coverage across retrieved sources. The best submitted full-set run used 5 diversity-first chunks.

3.5 Prompting and Structured Generation

The generator receives the title, user background, problem statement, and the selected context. It must return valid JSON with a top-level `responses` key. Each response item contains report text and a citation dictionary mapping document identifiers to numeric confidence scores. The prompt enforces three constraints: use only the provided context, cite factual statements with document identifiers from the context, and keep the output language fixed.

A key design decision was the partial-coverage instruction. Early generation prompts sometimes produced broad refusals when evidence was incomplete. This behavior improved caution but reduced report usefulness. The final prompt instead asks the

Query setting	SS	CS
Short, no rewrite	0.6798	0.6354
Full, no rewrite	0.7160	0.6850
Full, rewrite	**0.7610**	**0.7411**

Table 2: Query comparison on 20 development topics. SS denotes `sentence_support` and CS denotes `citation_support`.

generator to produce a partial report from available evidence and to explicitly mark missing subtopics. This is well matched to the official evaluation: unsupported claims hurt grounding, but refusing to answer hurts coverage.

4 Experimental Setup

A two-stage experimental protocol was used. First, rapid experiments were run on a fixed 20-topic development subset sampled from the task requests. Second, the most promising configurations were evaluated on the full 68-topic set used for final validation. The development subset was used to compare architectural choices cheaply, not as a reliable estimate of the final leaderboard ranking.

The experiments varied 5 groups of factors: query formulation and rewriting, retrieval depth, context mode, context budget and diversity-first selection, and final model stacks. Because not all factors were varied independently, compact grouped tables are reported. Each table is intended to answer one local question and should not be read as a fully factorial ablation.

5 Results

5.1 Query and Rewriting on Development Topics

Table 2 compares query construction while keeping the rest of the setup fixed: `BGE-emb`, `BGE-rer`, chunk mode, 20 retrieved candidates, 20 reranked candidates, and 7 chunks passed to the generator. The results support using the full request and conservative rewriting.

5.2 Retrieval Depth and Context Mode

Table 3 compares chunk-level and document-level context modes without query rewriting. Increasing the retrieval and reranking depth from 20 to 40 or 80 did not reliably improve the generated reports. This suggests that the bottleneck was not raw candidate count alone, but the interaction between reranking, context selection, and generation.

Mode and depth	K	SS	CS
Chunk	20	0.7278	0.7038
Chunk	40	0.6842	0.6803
Chunk	80	**0.7303**	0.6886
Document	20	0.7083	**0.6954**
Document	40	0.6685	0.6443
Document	80	0.6400	0.6384

Table 3: Retrieval-depth comparison on 20 development topics. In these runs, retrieval depth and reranking depth were varied jointly (K = 20, 40, or 80). All rows use full queries without rewriting and a context budget of 7 chunks or documents.

Context setting	SS	CS
5 chunks, score-only	0.7843	0.7350
10 chunks, score-only	0.6667	0.6400
7 chunks, diversity-first	**0.8176**	0.7273
5 chunks, diversity-first	0.7881	**0.7525**

Table 4: Context-budget comparison on 20 development topics. All rows use full-query rewriting, `BGE-emb`, `BGE-rer`, and chunk mode.

5.3 Context Budget and Diversity

Table 4 compares context-budget choices after enabling full-query rewriting. The ten-chunk setting performed worse than smaller contexts, which indicates that simply giving the generator more evidence can introduce noise. Diversity-first selection improved sentence support in the seven-chunk setting and gave a stronger citation-support score in the five-chunk setting.

5.4 Final Development Stacks

Table 5 reports the strongest late-stage development variants. This table explicitly separates the embedding model from the reranker. The `Qwen8B-emb` index was evaluated, but it did not outperform the `BGE-emb` embedding index with a `Qwen8B-rer` on this subset. The strongest 20-topic result used `BGE-emb`, `Qwen8B-rer`, `gpt-5.1-mid`, 5 diversity-first chunks, and the partial-coverage prompt.

5.5 Full-Set Internal Validation

Table 6 reports internal full-set diagnostics for the main final candidates. All rows use full-query retrieval, rewriting, 20 retrieved and reranked candidates, 5 diversity-first chunks, and `gpt-5.1-mid` generation. They differ in the reranker and prompt family. The `BGE-emb`/`BGE-rer` configuration with the partial prompt was selected because it was strongest on the full set, even though `Qwen8B-rer` was strongest on the 20-topic subset.

Embedder	Reranker	Prompt	SS	CS
BGE-emb	Qwen4B-rer	standard	0.8170	0.7577
BGE-emb	Qwen4B-rer	partial	0.7861	0.7692
BGE-emb	BGE-rer	partial	0.8253	0.8115
BGE-emb	Qwen8B-rer	partial	**0.8343**	**0.8377**
Qwen8B-emb	Qwen8B-rer	partial	0.7949	0.8000
Qwen8B-emb	Qwen4B-rer	partial	0.7250	0.7538

Table 5: Late-stage 20-topic development stacks. All rows use full-query rewriting, chunk mode, 20 retrieved and reranked candidates, 5 diversity-first chunks, and gpt-5.1-mid.

Embedder	Reranker	Prompt	SS	CS	CCS
BGE-emb	BGE-rer	standard	0.7816	0.7419	408
BGE-emb	BGE-rer	partial	**0.8100**	**0.8178**	**486**
BGE-emb	Qwen4B-rer	partial	0.7757	0.7763	446
BGE-emb	Qwen8B-rer	partial	0.7301	0.7574	422

Table 6: Internal full-set diagnostics on 68 topics. SS is sentence support, CS is citation support, and CCS is correctly cited sentences.

5.6 Official Submitted Runs

Table 7 presents the official Task B scores for the three submitted AMU runs. All three runs used the same high-level pipeline, including the BGE-emb retrieval index and gpt-5.1-mid generation, and differed mainly in the reranking stack: BGE-rer, Qwen4B-rer, or Qwen8B-rer.

6 Discussion

The experiments suggest four practical conclusions. First, report-level retrieval benefits from using the full request. The background field often contains constraints that are not repeated in the problem statement, and omitting it weakens retrieval. Second, query rewriting is useful when it is conservative: it should compress the request into a search query without adding facts or changing the language. Third, more retrieved candidates did not automatically improve the generated report. With a small context budget, adding candidates can increase noise unless the reranker and selector can reliably identify complementary evidence. Fourth, the best development stack was not the best full-set stack. The 20-topic subset favored BGE-emb with Qwen8B-rer, whereas the full set favored BGE-emb with BGE-rer. This motivates validating final candidates on the largest available set before submission.

The official scores also show that the system was more successful at grounding than at coverage. A conservative generator with strict citation requirements can avoid hallucinations, but it may underproduce useful content when evidence is incomplete or scattered across the corpus. For this task, improving nugget_coverage without sacrificing sentence_support is therefore the most important direction. Promising extensions include subtopic decomposition before retrieval, multiple targeted retrieval queries, evidence clustering, and adaptive context budgets for broad requests.

Reranker stack	SS	NC	F1
BGE-rer	0.8280	**0.3403**	**0.4351**
Qwen4B-rer	**0.8334**	0.2866	0.3829
Qwen8B-rer	0.7885	0.2763	0.3498

Table 7: Official RAG4Reports Task B results for AMU submissions. All runs used the BGE-emb retrieval index and gpt-5.1-mid generation; they differed mainly in the reranking stack. SS denotes sentence_support and NC denotes nugget_coverage.

7 Limitations

The main limitation is that the development experiments were not a full factorial ablation. Several factors changed together across runs, especially in later prompt and reranker experiments. The 20-topic results are therefore treated as diagnostic rather than causal. A second limitation is the fixed context budget. Using 5 chunks improved robustness for the selected configuration, but broad report requests may require more evidence than this budget allows. A third limitation is dependence on automatic evaluation. Auto-ARGUE provides scalable feedback, but errors in nugget matching or support annotation can affect model selection. Finally, the system does not perform explicit subtopic planning, which likely contributed to the limited nugget coverage.

8 Conclusion

This paper presented AMU's system for RAG4Reports 2026 Task B. The final pipeline uses full-query retrieval, conservative query rewriting, BGE-emb dense retrieval, BGE-rer reranking, diversity-first context selection, and gpt-5.1-mid JSON generation with partial-coverage prompting. The best submitted run achieved the highest AMU official F1, with strong sentence support but limited nugget coverage. Citation-grounded report generation requires both faithfulness and broader evidence coverage.

References

2026. RAG4Reports: Shared tasks. Accessed: 2026-05-11.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. BGE M3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich K"uttler, Mike Lewis, Wen-tau Yih, Tim Rockt"aschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474.

Chaofan Li, Zheng Liu, Shitao Xiao, and Yingxia Shao. 2023. Making large language models a better foundation for dense retrieval. *Preprint*, arXiv:2312.15503.

William Walden, Marc Mason, Orion Weller, Laura Dietz, John Conroy, Neil Molino, Hannah Recknor, Bryan Li, Gabrielle Kaili-May Liu, Yu Hou, Dawn Lawrie, James Mayfield, and Eugene Yang. 2026. Auto-argue: Llm-based report generation evaluation. *Preprint*, arXiv:2509.26184.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*.

Exploring Capability Thresholds in Ultra-Lightweight LLM Judges for Nugget-Based Report Evaluation

Mann Bajpai* **Pulkit Chatwal** **Priyanshu Deswal**
Santosh Kumar Mishra **Harish Pratap Singh**

Rajiv Gandhi Institute of Petroleum Technology
{21cs2011,21cs2027,21cs2014,
santosh.mishra,harishps23cs}@rgipt.ac.in

Abstract

Reliable automatic evaluation of retrieval-grounded long-form reports typically requires human annotation or frontier-scale proprietary LLMs, both of which are expensive in constrained settings. Team **rgipt** participated in RAG4Reports@ACL 2026 Task 1 with a zero-shot nugget-verification system that runs entirely on a single NVIDIA T4 GPU. We compare three ultra-lightweight decoder-only models:—Qwen2-0.5B, Qwen2-1.5B, and Qwen2.5-0.5B, under identical inference conditions to examine how small an LLM judge can be while retaining human-aligned ranking signal. Both Qwen2 models produced negative τ_{gap}, whereas Qwen2.5-0.5B achieved $\tau_{\text{gap}} = 0.0772$ and Pearson $r = 0.2209$, ranking 13th of 21 teams. Within this family and evaluation setting, model generation appears to matter more than parameter count, although this finding is based on three configurations on a single task and warrants further validation.

1 Introduction

The RAG4Reports shared task evaluates retrieval-grounded report generation and report evaluation under realistic long-form information-seeking settings (Yang et al., 2025). We participated in Task 1: Automatic Report Evaluation, where systems must produce report-level and system-level rankings that correlate with human-derived rankings. Our submission is semi-automatic: it uses organizer-provided human-curated nuggets as evaluation targets.

The central question of our system paper is operational rather than purely architectural: how small can an LLM evaluator be while still preserving useful human-aligned ranking structure? Existing RAG and NLG evaluation pipelines increasingly rely on LLM judges, including RAGAS (Es et al., 2024), ARES (Saad-Falcon et al., 2024), G-Eval (Liu et al., 2023), and Auto-ARGUE (Walden et al., 2025). However, these approaches often assume access to substantially larger models than those available in constrained academic or shared-task settings.

We therefore frame RAG4Reports Task 1 as zero-shot nugget verification with ultra-lightweight instruction-tuned decoder-only transformers. Given a report and an acceptable nugget answer, the evaluator predicts whether the nugget is semantically supported by the report. Nugget recall is then aggregated into report-level and system-level rankings. This design makes the evaluator simple, reproducible, and deployable on a single Kaggle NVIDIA T4 GPU.

Our main empirical finding is that evaluator quality does not scale monotonically with parameter count in this regime. Despite being smaller than Qwen2-1.5B, Qwen2.5-0.5B was the only evaluated model to cross from negative to positive τ_{gap}. This result suggests a possible capability threshold for latent semantic alignment in ultra-small LLM judges and highlights generational model improvements as an important factor in constrained report evaluation.

2 Related Work

RAG and report evaluation. Retrieval-Augmented Generation combines parametric generation with external non-parametric evidence to improve factual, knowledge-intensive generation (Lewis et al., 2020). The RAG4Reports shared task extends this paradigm to long-form, citation-backed report generation and evaluation (Yang et al., 2025), while Auto-ARGUE provides the official framework for report-level assessment through LLM judgments over report evidence and nugget support (Walden et al., 2025).

*Primary author and corresponding author.

Proceedings of the First Workshop on Multilingual Report Generation via Retrieval Augmented Generation (RAG4Reports), pages 94–98

Automatic RAG evaluation. RAGAS evaluates RAG systems along dimensions such as faithfulness, answer relevance, and context use without exhaustive human references (Es et al., 2024). ARES similarly targets scalable RAG evaluation using lightweight judges trained with synthetic data and limited human annotation (Saad-Falcon et al., 2024). Our system instead focuses on semi-automatic report ranking under a strict single-GPU constraint.

LLM-as-a-judge and reasoning prompts. LLM-as-a-judge methods are widely used for scalable evaluation of open-ended generation (Zheng et al., 2023). G-Eval showed that Chain-of-Thought prompting and structured scoring improve agreement with human judgments (Liu et al., 2023), while Chain-of-Thought more generally encourages intermediate reasoning before final decisions (Wei et al., 2022). We adopt this reasoning-first paradigm for binary nugget verification using ultra-lightweight models.

Classical and learned text-generation metrics. Earlier evaluation relied on lexical-overlap metrics such as BLEU and ROUGE (Papineni et al., 2002; Lin, 2004). Learned metrics such as BERTScore and BLEURT improved sensitivity to semantic similarity and human preference (Zhang et al., 2020; Sellam et al., 2020). FActScore further motivates fine-grained atomic evaluation for long-form factuality (Min et al., 2023). Our nugget-verification formulation follows this logic for retrieval-grounded reports.

Compact decoder-only models. The Qwen2 and Qwen2.5 technical reports describe families of instruction-tuned decoder-only transformers spanning compact and large scales (Yang et al., 2024; Qwen Team et al., 2025). Our ablation tests whether Qwen2.5 improves zero-shot semantic verification relative to Qwen2 under identical inference constraints.

3 System Description

3.1 Task Formulation

For each topic t, report r, and nugget $n \in N_t$, the model predicts a binary support label $y_{t,r,n} \in \{0, 1\}$. Report-level scores are computed as average nugget recall across all nuggets associated with a topic:

$$S(t, r) = \frac{1}{|N_t|} \sum_{n \in N_t} y_{t,r,n}$$

System scores are obtained by averaging report scores across topics and are then converted into system-level rankings for submission.

This formulation deliberately avoids learned calibration, supervised training, or additional retrieval. The evaluator is therefore a zero-shot semantic verifier over organizer-provided evidence targets.

3.2 Prompting Strategy

Ultra-small LLMs are brittle under direct score generation, especially when long reports express nuggets through paraphrase, implication, or distributed evidence. We therefore use a reasoning-first Chain-of-Thought prompt inspired by G-Eval (Liu et al., 2023). The model must first produce a short explanation and then emit a JSON score. The explanation is not used directly in scoring; it acts as an intermediate latent alignment step before classification.

System: Precise JSON-formatting evaluation assistant.
`User: Given a question, acceptable nugget answer, and generated report, decide whether the report explicitly or semantically supports the nugget. Count paraphrases as support, but reject vague topical overlap or unsupported inference. Output only JSON: {"reasoning": "...", "score": 1.0}.`

Table 1: Reasoning-first nugget verification prompt.

All prompts are formatted with HuggingFace `apply_chat_template` to preserve the models' instruction-tuning conventions.

3.3 Inference Pipeline

All experiments ran on a single Kaggle NVIDIA T4 GPU with 15GB VRAM. We evaluated `Qwen2-0.5B`, `Qwen2-1.5B`, and `Qwen2.5-0.5B`. Each model was loaded natively in `torch.float16`; we avoided 4-bit quantization to prevent additional approximation error in an already low-capacity evaluation regime.

The pipeline used deterministic greedy decoding, left-padded inputs, dynamic batched inference with batch size 4, and a regex fallback parser for malformed JSON. The fallback extracted binary scores using the

pattern `"score"\\s*:\\s*([01]\\.?0?)`. The evaluation set contained 57 generation systems and 68 topics, yielding 3,876 report-topic pairs before nugget expansion.

4 Results and Analysis

4.1 Official Task 1 Results

The official RAG4Reports Task 1 metric measures correlation between submitted rankings and human-derived rankings through the Auto-ARGUE evaluation framework (Walden et al., 2025). Table 2 reports the official scores for our submissions.

Model	τ_{gap}	Kendall	Pearson
`Qwen2.5-0.5B`	**0.0772**	0.1330	0.2209
`Qwen2-1.5B`	-0.1245	**0.1542**	**0.2513**
`Qwen2-0.5B`	-0.2693	-0.2967	0.1852

Table 2: Official RAG4Reports Task 1 results.

`Qwen2.5-0.5B` was our best model under the primary ranking metric, achieving $\tau_{\text{gap}} = 0.0772$, Pearson correlation of 0.2209, and rank 13 of 21 participating teams. The result is modest in absolute magnitude but important operationally: it shows that a 0.5B model can preserve some human-aligned ranking signal when paired with curated nuggets and a constrained verification pipeline.

The secondary metrics reveal a calibration tension. `Qwen2-1.5B` achieved higher Kendall and Pearson values but negative τ_{gap}, indicating that global association alone did not translate into the official gap-sensitive ranking objective. This discrepancy suggests that the older 1.5B model captured some coarse score variance while still misordering systems in the regions most consequential for the leaderboard metric.

4.2 Impact of Model Generation vs. Scale

Figure 1 is the core ablation. Both Qwen2 models fall below zero on τ_{gap}, meaning their induced rankings are inversely aligned with the official human-derived ordering under the primary metric. In contrast, `Qwen2.5-0.5B` crosses into positive correlation despite using fewer parameters than `Qwen2-1.5B`. This pattern is consistent with the hypothesis that model generation, instruction tuning, and latent semantic alignment may matter more than raw parameter count at the ultra-lightweight boundary, although this observation is based on only three models within the Qwen family.

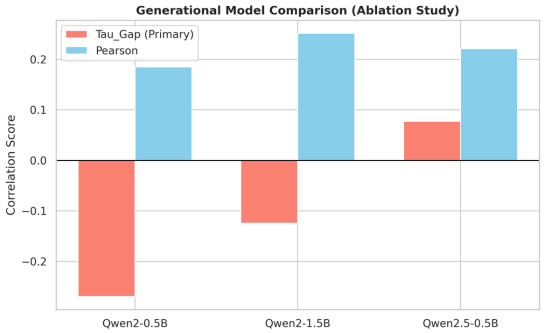

Figure 1: Generational ablation across Qwen variants.

The implication is practical: for constrained evaluator deployment, selecting the most recent compact instruction-tuned model may be more important than selecting the largest model that fits memory. In our setting, added Qwen2 capacity did not compensate for weaker zero-shot nugget verification behavior.

4.3 Score Distribution and Leniency

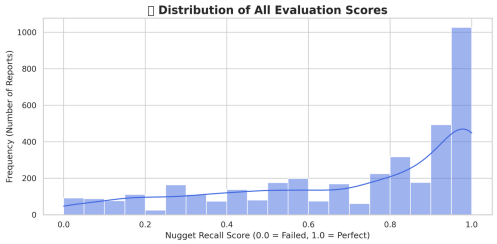

Figure 2: Score distribution for `Qwen2.5-0.5B`.

Figure 2 shows a bimodal score distribution dominated by positive classifications. This behavior indicates that `Qwen2.5-0.5B` often treats partial semantic overlap as sufficient nugget support. Such leniency is beneficial when reports express evidence through semantic paraphrase, but harmful when topical proximity is incorrectly treated as factual entailment.

This positive bias explains why the model can recover a weak ranking signal while remaining poorly calibrated at the nugget level. The evaluator is better interpreted as a high-recall semantic detector than as a strict factual verifier. For future systems, this suggests that lightweight judges need explicit contradiction handling, abstention, or threshold calibration to reduce false-positive nugget matches.

4.4 Ranking Consistency Across Systems

Figure 3 analyzes score dispersion for top-ranked systems. Narrower interquartile ranges indicate

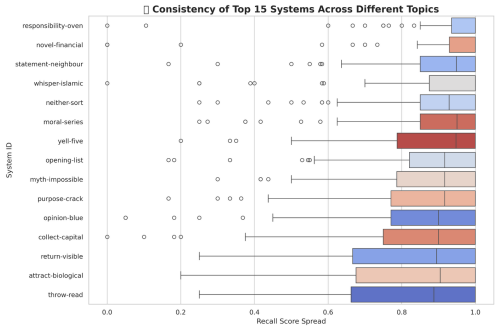

Figure 3: Nugget-score consistency among top-ranked systems.

that strong systems tend to receive stable nugget recall across topics, rather than relying on isolated high-scoring reports. This matters for report evaluation because system-level rankings should reward consistent retrieval-grounded coverage rather than topic-specific overperformance.

At the same time, residual variance among strong systems shows that even the best report generators are uneven across topic types. For the evaluator, this variance creates a difficult ranking problem: small calibration errors at the nugget level can shift aggregate system orderings. The figure therefore supports using consistency analysis alongside headline correlation metrics.

4.5 Topic Difficulty

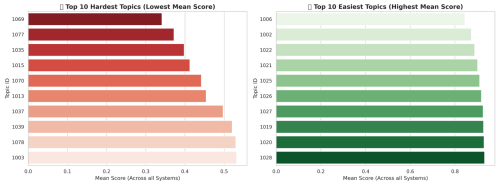

Figure 4: Hardest and easiest topics under nugget verification.

Figure 4 shows substantial topic-level difficulty variation. Easier topics appear to contain nuggets that are recoverable through explicit lexical overlap or localized factual statements. Harder topics require semantic abstraction, multi-paragraph evidence aggregation, or compositional reasoning over dispersed report content.

This result highlights the main failure mode of ultra-small evaluators: they can detect direct factual restatements but struggle when nugget satisfaction requires cross-context inference. This explains why a model can achieve positive aggregate ranking correlation while still failing on semantically demanding topics. Lightweight evaluation is therefore most effective when accompanied by topic-level robustness analysis.

5 Limitations

Our evaluator remains weakly calibrated, exhibiting positive classification bias, limited negation sensitivity, and difficulty with compositional reasoning. Because we use deterministic zero-shot inference with greedy decoding and no supervised calibration, the system cannot learn topic-specific thresholds or reliably distinguish partial from complete nugget support. The approach also depends on organizer-provided nuggets, limiting applicability when high-quality human-curated evaluation targets are unavailable. All experiments were conducted under a single-GPU constraint, so we do not claim competitiveness with frontier-scale proprietary evaluators. Consequently, our conclusion regarding a possible capability threshold is preliminary and should not be interpreted as evidence of a general scaling law.

6 Conclusion

We presented Team **rgipt**'s semi-automatic evaluator for RAG4Reports@ACL 2026 Task 1. Our system reduces long-form report evaluation to zero-shot nugget verification with ultra-lightweight decoder-only transformers and executes entirely on a single NVIDIA T4 GPU.

The central finding is evidence consistent with a possible capability threshold within the evaluated Qwen family: `Qwen2.5-0.5B` was the only tested model to produce positive τ_{gap}, while both Qwen2 baselines remained negative. These findings suggest that evaluator capability in ultra-lightweight LLMs may emerge discontinuously across architectural generations rather than scaling smoothly with parameter count alone.

Use of Generative AI Tools

The authors used generative AI tools solely for language editing and paraphrasing. All research ideas, experiments, analyses, and conclusions were developed and verified by the authors, who take full responsibility for the paper.

References

Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. 2024. RAGAs: Automated evaluation of retrieval augmented generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 150–158, St. Julians, Malta. Association for Computational Linguistics.

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 *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: NLG evaluation using GPT-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2511–2522, Singapore. Association for Computational Linguistics.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12076–12100, Singapore. Association for Computational Linguistics.

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

Qwen Team, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, and 1 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

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

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

William Walden, Marc Mason, Orion Weller, Laura Dietz, John Conroy, Neil Molino, Hannah Recknor, Bryan Li, Gabrielle Kaili-May Liu, Yu Hou, Dawn Lawrie, James Mayfield, and Eugene Yang. 2025. Auto-ARGUE: LLM-based report generation evaluation. *Preprint*, arXiv:2509.26184.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, and 1 others. 2024. Qwen2 technical report. *Preprint*, arXiv:2407.10671.

Eugene Yang and 1 others. 2025. RAG for Reports (RAGTIME) at TREC 2025. https://rag4reports.github.io/.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating text generation with BERT. In *International Conference on Learning Representations*.

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

EFSG: Evidence-First Structured Generation for Multilingual RAG Report Generation

Shaurya Gupta
Thapar Institute of
Engineering and Technology
sgupta8_be24@thapar.edu

Jatin Bedi
Thapar Institute of
Engineering and Technology
jatin.bedi@thapar.edu

Abstract

We describe EFSG (Evidence-First Structured Generation), our submission to Task B of the RAG4Reports@ACL 2026 shared task. Standard retrieval-augmented generation pipelines allow generation models to write from parametric memory and attach citations retroactively: a behaviour we term ***post-rationalization***. EFSG addresses this structurally through a ***phase boundary***: all evidence is retrieved, extracted, and sealed into a fact pool before any generation begins; each sentence then sees only its single committed source passage. Our best run (t5,100k doc corpus) achieved sentence_support of 0.612 and nugget_coverage of 0.126 (F1 = 0.182).

1 Introduction

Retrieval-augmented generation (Lewis et al., 2021) improves factual grounding by supplying a language model with retrieved passages at generation time. In practice, however, large language models draw heavily on parametric memory: a sentence is composed first, and a retrieved document is selected afterward to justify it. Because the generation model has access to the full context window - prior sentences, other documents, and background knowledge, it can write a plausible claim and subsequently identify a passage that approximately supports it. Prompting-based approaches, including Self-RAG (Asai et al., 2023), substantially reduce but not eliminate this tendency, since the fundamental access pattern remains unchanged.

EFSG is built on the hypothesis that post-rationalization is a structural problem requiring a structural solution: the citation must be *committed* before the sentence is written, and the generation model must see *only* the committed passage at the moment of writing. These two constraints together prevent post-rationalization architecturally.

This design is motivated by Bereiter and Scardamalia's distinction between Knowledge-Telling and Knowledge-Transforming in expert writing (Bereiter and Scardamalia, 1987). Knowledge-Telling produces text linearly: each sentence triggers the next without a governing plan. Knowledge-Transforming builds a rhetorical goal structure first, then fills it with evidence. Standard RAG is Knowledge-Telling. EFSG attempts Knowledge-Transforming: explicit epistemic goals are established per section, evidence is retrieved against those goals, sealed, and only then is generation permitted. The epistemic contract in C2 - a verb-led statement of what each section must establish, is the concrete realization of this rhetorical goal structure; it governs both what is retrieved and what may be written.

The paper is structured as follows. Section 2 describes the eight-component pipeline. Section 3 describes the corpus infrastructure. Section 4 presents results and analysis. Section 5 discusses the key design tradeoffs.

2 System Description

Figure 1 shows the full EFSG pipeline. Components C1-C4 constitute *Phase 1: Evidence Construction*. Components C5-C8 constitute *Phase 2: Grounded Generation*. The phase boundary is the explicit seal operation at the end of C4; no new evidence can enter after this point.

2.1 C1+C2: Intent Parser and Section Planner

A single Groq LLaMA-3.3-70B call per topic parses the competition topic JSON into a `ReportIntent` and 4-6 `SectionPlan` objects. Each `SectionPlan` contains (i) a section title, (ii) an *epistemic contract*: a verb-led statement of what the section must establish (e.g., *"Establish the clinical evidence that Tai Chi reduces fall incidence in adults over 65"*) and (iii) 2-3 retrieval queries. Character and sentence budgets are derived from

Proceedings of the First Workshop on Multilingual Report Generation via Retrieval Augmented Generation (RAG4Reports), pages 99–102

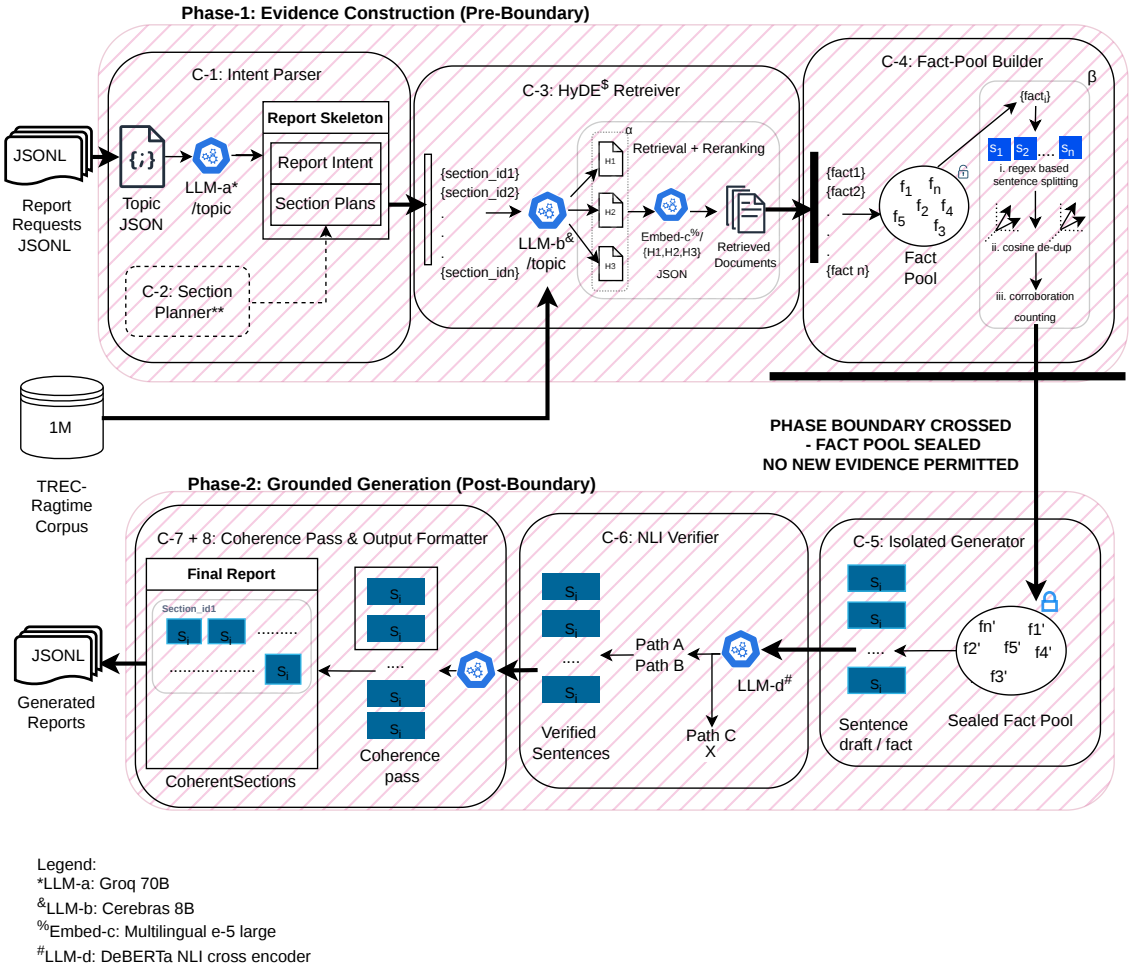

Figure 1: EFSG pipeline. The red line marks the phase boundary. LLM-a: Groq LLaMA-3.3-70B; LLM-b: Cerebras LLaMA-3.1-8B; Embed-c: Multilingual e5-large (text-embedding model); LLM-d: DeBERTa-v3-small NLI cross-encoder (local GPU).

the topic's `limit` field.

These components were originally designed as two separate LLM calls (C1 for intent, C2 for section decomposition). They were merged into one structured JSON response to conserve API quota. **Cost: 1 call/topic.**

2.2 C3: HyDE Retriever

For each section, one Cerebras LLaMA-3.1-8B call generates three hypothetical ideal paragraphs conditioned on the epistemic contract, following the HyDE retrieval strategy (Gao et al., 2022). Each hypothetical is encoded with `multilingual-e5-large` (Wang et al., 2024). The top-20 documents from each hypothetical are pooled (unique by document ID) and re-ranked by cosine similarity against the *epistemic contract embedding* - not the hypothetical. This re-ranking step is deliberate: re-ranking against the contract rather than the hypothetical avoids anchoring retrieval to the model's generated text, instead orienting it toward the section's rhetorical goal. Run logs show that the 8B model occasionally produced malformed JSON, causing fallback to single-hypothetical retrieval for affected sections. **Cost: 1 call/section.**

2.3 C4: Fact Pool Builder (Phase Boundary)

Retrieved documents are segmented into candidate facts using regex sentence splitting. A quality filter discards segments shorter than 30 characters or lacking a common verb form. Within-section semantic deduplication runs at cosine similarity threshold 0.92; cross-section global

deduplication runs at 0.88, preventing the same sentence from appearing in multiple sections. The pool is then **sealed**: `FactPool.sealed = True`. All subsequent components call `pool.get_unused(section_id)`, which enforces the phase boundary by refusing to return facts not present at seal time.

Resource-driven adaptation. The original design called for a Cerebras 8B LLM call per batch of five documents, prompting the model to extract atomic facts conditioned on the section's epistemic contract. At approximately 50 calls per topic for 68 topics, this required roughly 3,400 API calls for C4 alone: infeasible under practical scenarios. Sentence splitting was substituted, reducing C4 to zero API calls. **Cost: 0 API calls.**

2.4 C5: Isolated Generator

Each fact from the sealed pool is used directly as the sentence draft. The committed passage for NLI verification is the fact itself, so NLI(passage, sentence) ≈ 1.0 by construction.

Resource-driven adaptation. The original design called for a per-sentence LLM call that would rephrase the committed passage into appropriate register while remaining informationally contained within it. This step was eliminated to conserve API quota. **Cost: 0 API calls.**

2.5 C6: NLI Verifier

`cross-encoder/nli-deberta-v3-small` (He et al., 2021) runs locally on the T4 GPU. For each sentence draft, the model computes an entailment probability for the (committed passage, sentence) pair. Path A (≥ 0.85): accept as-is. Path B (0.60–0.85): attempt one Groq 70B regeneration with a stricter prompt; keep whichever version scores higher. Path C (< 0.60): substitute the first sentence of the committed passage verbatim. Across the t5 run, 91.9% of sentences took Path A, 8.0% Path B, and 0.1% Path C (1 sentence), with mean NLI = 0.951 and std = 0.042. The near-uniform Path A result is expected given C5's direct-use design: NLI(sentence, sentence) ≈ 1. **Cost: 0 API calls; Path B billed to Groq 70B only when triggered.**

2.6 C7: Coherence Pass

A Cerebras 8B call receives the sentence list for each section and may add minimal transition words (*Additionally*, *However*) at sentence starts only. Rephrasing and factual additions are explicitly prohibited in the system prompt. A post-pass NLI sweep reverts any sentence falling below 0.60. Run logs show that the 8B model frequently produced sentence counts outside the ± 2 tolerance window; for example, splitting one sentence into several or merging adjacent sentences - triggering full section reversion. In the majority of sections across both runs, C7 reverted to the raw C5 output. **Cost: 1 call/section.**

2.7 C8: Output Formatter

Completed sections are serialized to the competition JSONL schema: one object per sentence containing the sentence text, a `citations` field mapping `doc_id` to the NLI score, and metadata (NLI path, fact ID, section title). Character-budget enforcement trims the final section to fit the topic's `limit` field. **Cost: 0 API calls.**

API budget summary. Final configuration: Groq LLaMA-3.3-70B at 4 RPM (C1 only, 1 call/topic); Cerebras LLaMA-3.1-8B at 28 RPM (C3 + C7, ≈13 calls/topic). Total: ≈800 API calls for 68 topics; ≈25 minutes of rate-limit wait. Earlier configurations (with LLM-based C4) required ≈4,100 calls and ≈2.8 hours of wait time across the same 68 topics.

3 Corpus and Infrastructure

The ragtime1 dataset contains 1,000,095 English documents. Documents were sampled directly from the HuggingFace dataset. Documents passing a quality filter (minimum 200 characters, at least three sentences, mean sentence length ≥ 8 words) were embedded using `multilingual-e5-large` and cached to Google Drive. Corpus sizes: 100,000 documents for the t5 run (10% corpus coverage); 50,000 for t4.

4 Results and Analysis

Run	SS	NC	F1
efsg-t5 (100k docs)	0.612	0.126	0.182
efsg-t4 (50k docs)	0.327	0.065	0.097

Table 1: Submission results evaluated with AutoARGUE (Walden et al., 2025). SS = sentence_support; NC = nugget_coverage.

Table 1 shows the two submitted runs. Both metrics scale near-linearly with corpus size: doubling the indexed corpus from 50k to 100k roughly doubles

both sentence_support and nugget_coverage. This scaling pattern isolates the bottleneck as retrieval coverage rather than generation faithfulness.

Sentence_support of 0.612 for t5 is the expected consequence of C5's direct-use design: submitted sentences are verbatim extracts from cited documents, making entailment near-certain by construction. Nugget_coverage of 0.126 reflects document availability: facts absent from the sampled corpus cannot be retrieved regardless of pipeline quality. The two metrics are therefore measuring different failure modes -one of generation, one of retrieval -and should be interpreted independently.

5 Discussion

The central finding is a clean metric decoupling: a system can maximise sentence_support by eliminating generation entirely, while nugget_coverage remains bounded by corpus coverage alone. This suggests that under faithfulness-weighted evaluation, improving retrieval coverage yields larger gains than improving generation quality.

The phase boundary demonstrates that faithfulness guarantees can be structural rather than prompt-based. The invariant: no evidence enters after the pool is sealed holds independently of which models or extraction methods populate the pipeline. Resource constraints affected implementation quality but not the architectural guarantee.

Limitations

Sentence splitting in C4, despite a length and verb-form quality filter remains vulnerable to topical irrelevance, web-boilerplate (cookie notices, navigation text) and non-atomic segments occasionally as direct consequence and trade-off for LLM-based extraction vs API quota.

Direct fact use in C5 means submitted text consists entirely of verbatim source extracts, which maximizes sentence_support by construction but sacrifices fluency and register consistency; C7 was designed to address this through constrained coherence editing, but produced systematic reversion across both runs, leaving most section outputs as raw C5 extracts.

Corpus coverage was limited to 5-10% of the available collection due to compute and storage constraints; this is the primary driver of low nugget_coverage and should be interpreted as a data access limitation rather than a pipeline failure.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *Preprint*, arXiv:2310.11511.

Carl Bereiter and Marlene Scardamalia. 1987. *The psychology of written composition*. Lawrence Erlbaum Associates, Inc, Hillsdale, NJ.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2022. Precise zero-shot dense retrieval without relevance labels. *Preprint*, arXiv:2212.10496.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: Decoding-enhanced bert with disentangled attention. *Preprint*, arXiv:2006.03654.

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. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Preprint*, arXiv:2005.11401.

William Walden, Marc Mason, Orion Weller, Laura Dietz, John Conroy, Neil Molino, Hannah Recknor, Bryan Li, Gabrielle Kaili-May Liu, Yu Hou, Dawn Lawrie, James Mayfield, and Eugene Yang. 2025. Auto-ARGUE: LLM-based report generation evaluation. *Preprint*, arXiv:2509.26184.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Multilingual e5 text embeddings: A technical report. *Preprint*, arXiv:2402.05672.

Adapting AutoARGUE for Automatic Report Evaluation under Missing Citation Annotations

Divrose Kaur **Jatin Bedi** **Jasmeet Singh**
Thapar Institute of Engineering & Technology
divrose19@gmail.com jatin.bedi@thapar.edu jasmeet.singh@thapar.edu

Abstract

We adapt the AutoARGUE framework (Walden et al., 2026) for Task A.2 of RAG4Reports 2026, which requires ranking 57 report generation systems across 68 topics using automated evaluation. The RAGTIME-1 corpus poses a fundamental challenge: all nugget annotations use a `no-reference-doc` sentinel rather than ground-truth document citations, rendering the original citation-relevance gating inoperable. We address this with three adaptations: automatic sentinel detection with forced direct LLM-based nugget matching; a WEAK POSITIVE partial credit mechanism for sentences that correctly answer nuggets but lack attesting citations; and a report-level request alignment check. Our `nugget_coverage_weighted` metric achieves the highest topic-level Pearson correlation (r=0.599) of any non-coordinator submission, closely approaching the coordinator baseline (r=0.607).

1 Introduction

Automatic evaluation of retrieval-augmented generation (RAG) systems is an active area of research, with nugget-based frameworks emerging as the leading paradigm (Walden et al., 2026; Mayfield et al., 2024). RAG4Reports 2026 Task A.2 operationalises this paradigm at scale: given 57 machine-generated reports across 68 topics from the RAGTIME-1 corpus, participants must produce a system ranking that correlates with human annotation rankings, measured by Kendall's τ (Fagin et al., 2004).

The task organizers designated AutoARGUE (Walden et al., 2026) as the official baseline. AutoARGUE implements the ARGUE decision tree, which evaluates each report sentence by checking whether its citations attest its claims and whether it answers key nugget questions. Our work extends this baseline to handle the RAGTIME-1 corpus, which differs fundamentally from the NeuCLIR corpus (Lawrie et al., 2025) that AutoARGUE was designed for: every nugget annotation uses the sentinel `no-reference-doc` instead of a ground-truth document ID, making citation-relevance lookup inoperable.

We describe three principal adaptations, our evaluation setup, and results on the official leaderboard.

2 Task and Data

Task definition. Task A.2 receives as input 57 JSONL files of machine-generated reports (one per system, 68 topics each) from TREC RAGTIME 2025 participants, human-curated nugget files (QA pairs, one per topic), and a `report-requests.jsonl` containing user backgrounds and information needs. The output is a TSV of per-topic metric scores; the primary evaluation metric is gap-aware rank correlation (Fagin et al., 2004) on the macro-averaged F1 rankings against human annotation.

Corpus. The RAGTIME-1 retrieval corpus comprises four multilingual JSONL files: English news articles (`eng-docs`), and Russian-, Chinese-, and Arabic-to-English translations (`rus-trans`, `zho-trans`, `arb-trans`).

3 Methodology

3.1 The `no-reference-doc` Problem

The ARGUE decision tree (Mayfield et al., 2024) at Node B asks: *Is the cited document relevant?* This is answered by looking up the cited document ID in a doc-to-nugget annotation map — each nugget answer is linked to specific documents that attest it. In RAGTIME-1, every nugget answer uses `no-reference-doc` as its citation, intentionally omitting ground-truth annotations. This causes Node B to always return *No*, collapsing every cited sentence to a negative outcome and producing zero

Proceedings of the First Workshop on Multilingual Report Generation via Retrieval Augmented Generation (RAG4Reports), pages 103–107

scores for all systems.

Our solution: (1) detect the all-sentinel condition automatically at load time; (2) strip `no-reference-doc` from all nugget citation lists before building the doc-to-nugget map; (3) enable `always_check_all_nuggets` mode, forcing the LLM to evaluate every nugget against every sentence regardless of citation annotation. This bypasses annotation-based gating and substitutes direct LLM judgment throughout, at the cost of increased LLM call volume per topic.

3.2 Adapted Decision Tree

Figure 1 illustrates our adapted ARGUE decision tree. Table 1 summarises the outcome cases; the principal changes from the original framework are the WEAK POSITIVE branch and the report-level alignment check.

Condition	Outcome	Credit
Cited + attested + nugget match	Strong Positive	1.0
Cited + attested + no nugget	Neutral	0.0
Cited + not attested + nugget	Weak Positive	0.5
Cited + not attested + no nugget	Negative	0.0
No citation + requires citation	Negative	0.0
No citation + no citation needed	Neutral	0.0
Report misaligned with request	Align. mult.	$\times 0.7$

Table 1: Adapted decision tree outcome matrix.

WEAK POSITIVE partial credit. During development, we observed that some systems correctly answered nuggets — demonstrating genuine information coverage — but received F1 = 0.0 because their cited documents were not attested by the corpus. This discards a meaningful quality signal: the system found relevant information but cited imperfectly. We implement $0.5\times$ partial credit in sentence precision for WEAK POSITIVE outcomes, making the scorer sensitive to information coverage independent of citation quality.

Request alignment check. The ARGUE framework states that reports should be tailored to the specific user's information need. We add a report-level alignment check: the full report text is evaluated against the user background and problem statement from `report-requests.jsonl` by the LLM judge. Misaligned reports receive a $0.7\times$ multiplier on their F1 score. In practice, most systems scored alignment $= 1.0$; the multiplier correctly flagged near-empty or off-topic reports.

3.3 Scoring Metrics

$$\textit{sent\_sup} = \frac{\sum_s \text{credit}(s)}{|\text{sentences requiring citation}|} \tag{1}$$

$$\textit{nug\_cov} = \frac{|\text{nuggets answered}|}{|\text{total nuggets}|} \tag{2}$$

$$\text{F1} = \frac{2 \cdot \textit{sent\_sup} \cdot \textit{nug\_cov}}{\textit{sent\_sup} + \textit{nug\_cov}} \times \textit{alignment} \tag{3}$$

The primary submission metric is `f1_macro`: the mean F1 across all evaluated topics per system.

3.4 LLM Judge and Evaluation Setup

We use Llama-3.3-70B Instruct (Grattafiori et al., 2024) (4-bit quantisation, `unsloth/Llama-3.3-70B-Instruct-bnb-4bit`) served locally via vLLM (Kwon et al., 2023), matching the judge model used in the AutoARGUE paper (Walden et al., 2026). Prior to local vLLM deployment, we evaluated several provider options. Cloud-based APIs (Llama-3.3-70B and smaller variants via Groq) were constrained by rate limits that made full-dataset evaluation infeasible. Locally-hosted smaller models via Ollama (`qwen2.5:7b`) lacked the nuanced judgment capability required for reliable nugget matching. These constraints motivated local deployment of the full 70B model. Document text is truncated to 800 characters for LLM attestation calls to reduce per-topic compute cost.

Due to resource constraints, we evaluated 5 representative topics per system (1001: AI/technology, 1009: science/weather, 1017: entertainment, 1069: technology history, 1083: environmental science) rather than all 68.

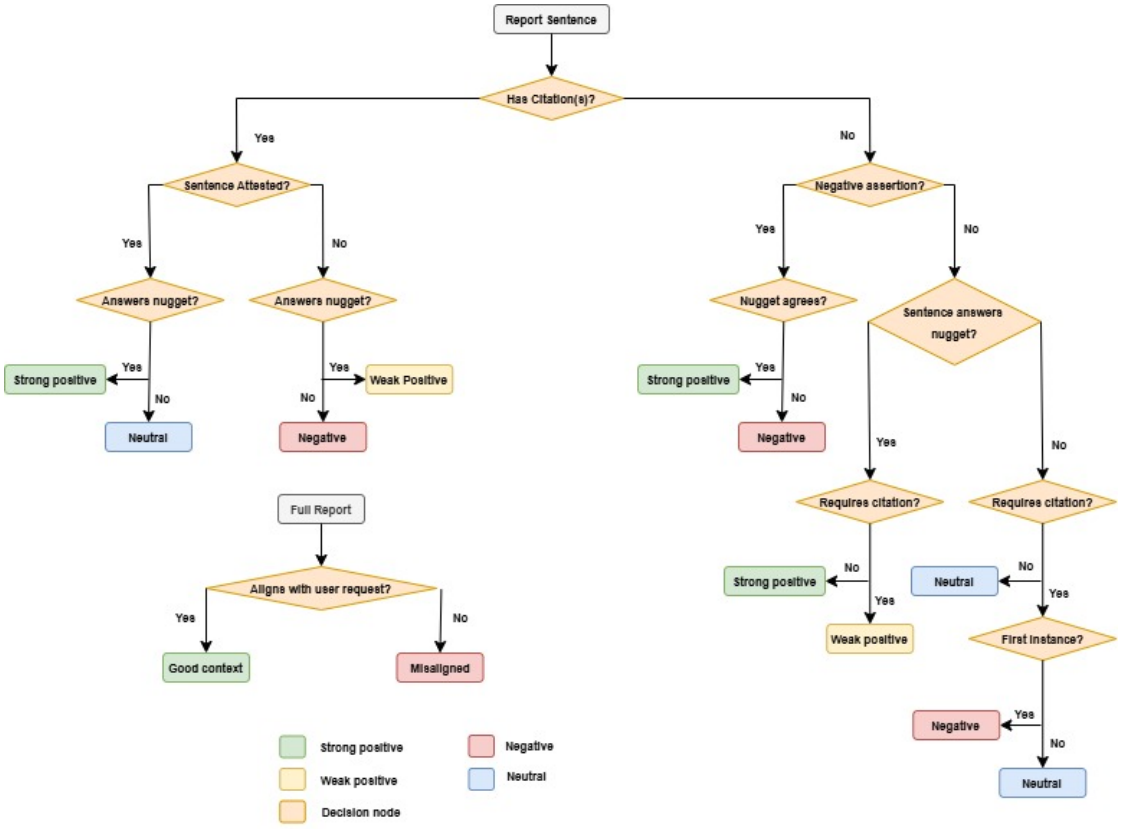

Figure 1: Adapted ARGUE decision tree for RAGTIME-1. The left subtree handles sentences *with* citations; the right subtree handles sentences *without* citations. The bottom section shows the report-level request alignment check, which applies a $0.7\times$ score multiplier to misaligned reports.

4 Results

4.1 Leaderboard Results

Table 2 shows our submissions against the coordinator baseline on the Task A leaderboard.

Team	Metric	tau_gap	τ	r
Coord.	autoargue-f1	0.470	0.636	0.607
Ours	f1_weighted	0.127	0.334	0.573
Ours	nugget_cov_w	0.075	0.289	**0.599**
Ours	sentence_support	−0.056	0.183	0.247

Table 2: Task A results. tau_gap is the primary metric (higher is better); r = topic-level Pearson. `nugget_cov_w` achieves the highest topic-level Pearson of any non-coordinator submission.

Our `nugget_coverage_weighted` metric achieves the highest topic-Pearson correlation (r=0.599) of any non-coordinator submission, closely approaching the coordinator baseline of 0.607. `sentence_support` alone performs below random (tau_gap < 0), confirming that citation quality in isolation is insufficient to rank systems without the nugget coverage signal.

4.2 System-Level Score Distribution

Our pipeline produced clear differentiation across the 57 systems, with F1 macro ranging from 0.611 (`friend-proceed`) to 0.000 for systems with zero citations or zero nugget coverage. Notably, `possibility-prime` had 27 citations but 0.000 nugget coverage — citing documents without addressing the information need. Citation support was consistently lower than sentence support across all systems, indicating a pervasive pattern of citing documents that do not fully attest their sentences — a known failure mode in RAG systems.

4.3 Per-Topic Analysis

Table 3 shows per-topic scores for the top system (`friend-proceed`). Topic 1069 (LiveJournal) was consistently the hardest across all evaluated systems, suggesting ambiguous or poorly-specified requests. Topic 1083 (mammoth extinction) achieved 100% nugget coverage for this system.

Topic	F1	Nug.	Sent.	Cit.
1001 (AI)	0.683	0.600	0.792	0.522
1009 (Weather)	0.750	0.667	0.857	0.289
1017 (Disney)	0.587	0.500	0.711	0.378
1069 (LiveJrn.)	0.296	0.364	0.250	0.222
1083 (Mammoth)	0.741	1.000	0.588	0.250
Aggr.	**0.611**	0.626	0.640	0.332

Table 3: Per-topic scores for top system (`friend-proceed`). Nug. = nugget coverage; Sent. = sentence support; Cit. = citation support.

5 Analysis

Nugget coverage vs. F1. The higher topic-Pearson of `nugget_coverage_weighted` (r=0.599) compared to `f1_weighted` (r=0.573) suggests nugget recall is a stronger per-topic signal than the combined F1. That `sentence_support` alone yields a negative tau_gap (-0.056) confirms this component is unreliable in isolation, while nugget coverage directly measures information coverage.

Partial evaluation scope. The strong score differentiation across systems (F1 macro 0.00–0.61) suggests the 5-topic sample was sufficient to surface the most salient quality differences. Full 68-topic evaluation was infeasible under our resource constraints.

The `no-reference-doc` adaptation. Our sentinel detection and direct LLM-matching approach generalises to any dataset with incomplete annotation coverage. The primary tradeoff is increased LLM call volume, as every nugget must be checked against every sentence regardless of document overlap.

Sentence support as a standalone signal. The negative tau_gap for `sentence_support` (-0.056) confirms that citation quality alone is an unreliable ranking signal. The F1 harmonic mean combining nugget coverage and sentence support is substantially more reliable than either component alone.

6 Conclusion

We adapted AutoARGUE for the RAGTIME-1 corpus by addressing the `no-reference-doc` sentinel problem, adding WEAK POSITIVE partial credit, and implementing report-level request alignment checking. Our `nugget_coverage_weighted` metric achieves the highest topic-Pearson of any non-coordinator submission (r=0.599), closely approaching the coordinator baseline. These adaptations generalise to evaluation scenarios with incomplete nugget annotation coverage.

Limitations

Evaluation covers 5 of 68 topics per system due to resource constraints; full coverage may alter system rankings at the margins. Document truncation to 800 characters is effective for inverted-pyramid news text but may not generalise to corpora with different information density. The request alignment multiplier ($0.7\times$) was set heuristically; a calibrated value may perform better. The 4-bit quantised Llama judge may produce different judgments than the full-precision model used in the official coordinator evaluation.

Ethical Considerations

This work develops automated evaluation metrics for RAG systems. Automated metrics influence system development and benchmarking; over-reliance on any single metric without human validation carries the risk of rewarding systems that optimise for the metric rather than genuine quality. Our results show that individual metric components (e.g., sentence support alone) can be poor proxies for human judgments, underscoring the importance of multi-faceted evaluation.

Acknowledgments

This work was conducted as part of undergraduate research at Thapar Institute of Engineering & Technology. The authors thank the RAG4Reports 2026 organizers for their support and guidance throughout the shared task.

References

Ronald Fagin, Ravi Kumar, Mohammad Mahdian, D. Sivakumar, and Erik Vee. 2004. Comparing and aggregating rankings with ties. In *Proceedings of the 23rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 47–58.

Grattafiori et al. 2024. The Llama 3 herd of models. *arXiv preprint*.

Woosuk Kwon, Zhuowei Li, Siyuan Zhu, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the ACM SIGOPS*

29th Symposium on Operating Systems Principles, pages 611–626.

Dawn Lawrie, Sean MacAvaney, James Mayfield, Paul McNamee, Douglas W. Oard, Luca Soldaini, and Eugene Yang. 2025. Overview of the TREC 2024 NeuCLIR track. In *Proceedings of the Thirty-Third Text REtrieval Conference (TREC 2024)*.

James Mayfield, Eugene Yang, Dawn Lawrie, Sean MacAvaney, Paul McNamee, Douglas W. Oard, Luca Soldaini, Ian Soboroff, Orion Weller, Efsun Kayi, Kate Sanders, Marc Mason, and Noah Hibbler. 2024. On the evaluation of machine-generated reports. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1904–1915.

William Walden, Marc Mason, Orion Weller, Laura Dietz, John Conroy, Neil Molino, Hannah Recknor, Bryan Li, Gabrielle Kaili-May Liu, Yu Hou, Dawn Lawrie, James Mayfield, and Eugene Yang. 2026. Auto-ARGUE: LLM-based report generation evaluation. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

JU-NLP-PG at RAG4Reports 2026: Memory-Efficient Multilingual Report Generation with 4-bit Quantized LLMs

Swayam Chatterjee
Jadavpur University
Department of Computer Science and Engineering
swayam.internship@gmail.com

Prof. Dipankar Das
Jadavpur University
Department of Computer Science and Engineering
dipankar.dipnil2005@gmail.com

Abstract

In the present article, we have described our system developed for participating in Task B on Multilingual Report Generation under RAG4Reports 2026 at ACL 2026 [1] with submitted run ID ju_nlp_pg. The problem statement is given a report request[2] in English, the system retrieves relevant passages from a four million multilingual document corpus [3](English, Chinese, Russian, Arabic) and generates a grounded, citation-bearing report. Our core challenge was how to fit a large retrieval corpus along with a capable generative model on a two-GPU node with ≈29 GB RAM. We addressed the challenge employing three different techniques: (1) 4-bit NF4 quantization of Mistral-7B-Instruct-v0.3, shrinking the LLM from ≈14 GB to ≈4 GB; (2) memory-mapped, chunked FAISS index construction over pre-computed multilingual-e5-large embeddings; and (3) strict model-loading order to prevent heap fragmentation. On the other hand, the reports are structured around topic nuggets to directly target the Auto-ARGUE evaluation signal.

1 Introduction

Retrieval-Augmented Generation (RAG) has become one of the utmost important approaches for long-form, knowledge-intensive text generation (Lewis et al., 2020). RAG4Reports 2026 advances this further by requiring systems to produce paragraph-level reports from a four-language news corpus, with every sentence carrying inline citations to source documents; a setting the organizers call "deep research" or "agentic search."

If we highlight the challenges, participating with limited hardware is one of them and its genuinely hard. A competitive system needs cross-lingual retrieval indices over four million documents, a multilingual query encoder, and a generative model capable of instruction following and citation insertion all at the same time. On a Kaggle 2×T4 node (≈29 GB RAM, 15 GB VRAM per GPU), this does not fit naively.

In contrast, this paper describes how we made it fit. Our system, ju_nlp_pg, combines 4-bit NF4 quantization, memory-mapped FAISS indexing, and a careful model-loading sequence to keep the full pipeline within budget. We also use nugget-guided prompting to align generation with the Auto-ARGUE metric (Walden et al., 2025).

2 Background

RAG for Long-form-Generation Lewis et al. (2020) introduced RAG framework as a hybrid parametric/non-parametric framework. Later, the work was extended to open-domain QA (Karpukhin et al., 2020) and iterative generation (Shao et al., 2023). Our system follows the standard single-pass retrieve-then-generate pattern and is applied to the multilingual, citation-required report generation setting.

Cross-lingual Retrieval We use multilingual-e5-large[4] (mE5-large) (Wang et al., 2022), which maps text from all the four languages and combined into a shared 1024 dimensional embedding space via weakly supervised contrastive pre-training.

LLM quantization. 4-bit NF4 quantization (Dettmers et al., 2023) maps weights to 16 points derived from the standard normal quartile function theoretically optimal for normally distributed parameters. It combines with double quantization

[1] https:https://rag4reports.github.io/
[2] https://huggingface.co/datasets/hltcoe/rag4reports-2026
[3] https://huggingface.co/datasets/trec-ragtime/ragtime1
[4] https://huggingface.co/intfloat/multilingual-e5-large

Proceedings of the First Workshop on Multilingual Report Generation via Retrieval Augmented Generation (RAG4Reports), pages 108–112

of scale constants, it reduces a 7B model from ≈14 GB (fp32) to ≈4 GB.

Evaluation Auto-ARGUE (Walden et al., 2025) scores reports by checking whether curated *nuggets* essential QA pairs for each topic is answered and correctly cited. We structure our prompts directly around these nuggets.

3 System Description

The whole system was designed in a modularized approach as described in Figure 1 that shows the full end-to-end pipeline.

3.1 Offline Embedding

Before inference, we compute document embeddings for all the four languages. For each of the languages, the corpus is loaded from `trec-ragtime/ragtime1` on HuggingFace Hub. The dataset is embedded using `multilingual-e5-large` (`normalize_embeddings=True`).
In order to speed up encoding, we use `SentenceTransformer`'s `start_multi_process_pool` to distribute across both T4 GPUs (batch size 256), roughly halving wall-clock time (e.g., from ≈25 to ≈13 hours for English). The resulting matrices are saved as `.npy` files on Kaggle Datasets (`swayamc/{eng,arb,zho,rus}-rag4reports`).
We use FAISS to identify 50k chunks and pass those chunks in the retrieval phase of the pipeline for FAISS lookup.

3.2 Retrieval Methodology

Now, we pass such report requests into multilingual m5 large to create query vectors. At inference time, the report request, title, background, and problem statement are concatenated and finally sent for encoding on GPU 0:

$$\mathbf{q} = \text{Encode}_{\text{mE5}}(Q) \in \mathbb{R}^{1024}$$

Q = raw dataset,
mE5 = multilingual E5 large model,
$\mathbb{R}^{1024}$ = 1024 dimension vectors generated
The query vectors are used to retrieve related corpus embeddings. Inner-product search over each of the languages `faiss.IndexFlatIP` returns the top-10 candidates per language; the top-5 by score across all the four languages are then finally passed for generation.

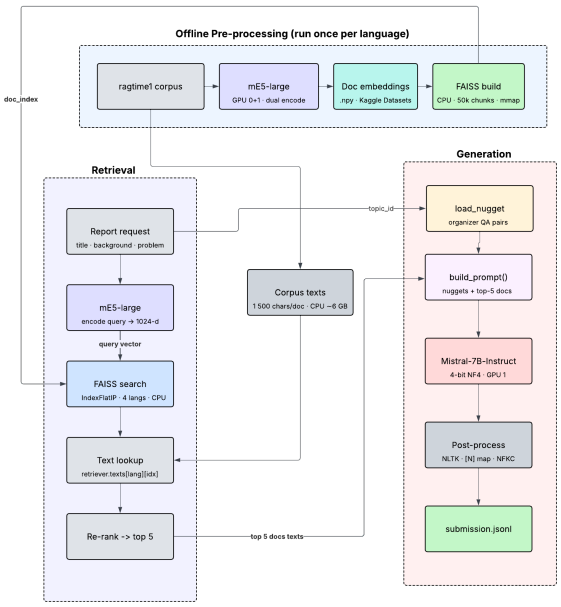

Figure 1: Full system pipeline of `ju_nlp_pg`. *Top (teal):* offline pre-processing, the corpus is encoded with `multilingual-e5-large` on dual GPUs, saved as `.npy` embeddings, and indexed into per-language FAISS indices on CPU. *Bottom-left (blue):* online retrieval, the report request is encoded on GPU 0 and searched against all four FAISS indices; the top-5 documents by inner-product score are retrieved. *Bottom-right (red):* generation, topic nuggets and retrieved context are merged into a structured prompt, fed to Mistral-7B-Instruct-v0.3 (NF4, GPU 1), and post-processed into the JSONL submission.

3.3 Semantic Similarity and Query Expansion

Our retrieval framework employs dense semantic embeddings generated using `sentence-transformers` (`MiniLM-L6-v2`[5], `multilingual-e5-base`[6], `multilingual-e5-large`) to measure semantic relevance between user queries and candidate document chunks. Both the query and document embeddings are L2-normalized during encoding, enabling efficient similarity computation through vector dot products. Since the embeddings are normalized to unit length, the dot product is mathematically equivalent to cosine similarity. The relevance score between a query embedding $\mathbf{q}$ and a document embedding $\mathbf{d}$ is:

$$\text{score}(\mathbf{q}, \mathbf{d}) = \frac{\mathbf{q} \cdot \mathbf{d}}{\|\mathbf{q}\| \, \|\mathbf{d}\|}$$

[5] https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2
[6] https://huggingface.co/intfloat/multilingual-e5-base

After normalization, this simplifies to:

$$\text{score}(\mathbf{q}, \mathbf{d}) = \mathbf{q} \cdot \mathbf{d}$$

It was observed that the document chunks are ranked in descending order of cosine similarity, where the higher similarity indicates greater semantic relevance to the query. In order to improve the retrieval robustness, the system performs multi query expansion, generating multiple semantically related query variants. Each expanded query independently retrieves candidate documents, and the final relevance score for a document is determined using max score fusion across all the expanded queries:

$$\text{final\_score}(\mathbf{d}) = \max_i \ \text{score}(\mathbf{q}_i, \mathbf{d})$$

where $\mathbf{q}_i$ denotes the i^{th} expanded query. It was found that this approach improves recall by preserving the strongest semantic match obtained from any query variant.

Additionally, cosine similarity is used during citation validation to verify semantic alignment between the statements of generated report and supporting evidence retrieved. Candidate citations whose similarity exceeds a predefined threshold are considered semantically supported and retained in the final report generation pipeline.

3.4 Nugget-Guided Generation

We load the topic nuggets provided by the organizers and build a structured prompt:

> Answer these questions using only the provided documents:
> 1. {nugget 1} $\cdots$ n. {nugget n}
> Documents: {retrieved texts, 1 500 chars each}
> Write complete sentences. Cite sources as [N].
> Skip unanswerable questions.

This directly targets nugget recall under AutoARGUE. We generate with Mistral-7B-Instruct-v0.3[7] (NF4, GPU 1) at temperature 0.7, up to $\min(\lfloor L/3.5 \rfloor, 2048)$ tokens, where L is the per-topic NFKC character limit and 3.5 approximates the average characters per token for mixed multilingual text.

[7] https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3

Component	Device	Size
Mistral-7B (NF4, 4-bit)	GPU 1	≈4 GB
mE5-large encoder	GPU 0	≈2.2 GB
FAISS indices (4 langs)	CPU	≈16 GB
Text arrays (4 langs)	CPU	≈6 GB
Total		≈28 GB

Table 1: Memory allocation. Components are loaded in this order (top to bottom) to avoid heap fragmentation.

3.5 Post-Processing

Raw output is converted into the required JSONL format: (1) sentence segmentation via NLTK `sent_tokenize`; (2) regex extraction of `[N]` markers mapped to doc IDs and retrieval scores; (3) greedy character-budget enforcement under the NFKC limit L; (4) serialization into `metadata`, `responses`, and `references` fields.

4 Experimental Setup

All runs were uploaded on Kaggle's 2×T4 environment (CUDA 12.x, Ubuntu 22.04, ≈29 GB CPU RAM). Dependencies: `transformers` ≥4.40, `bitsandbytes` ≥0.46.1, `sentence-transformers` ≥2.2, `faiss-gpu`.

Document texts originated from `trec-ragtime/ragtime1` were truncated to 1,500 chars per document.

4.1 Memory Optimization

Table 1 shows how we allocate ≈28 GB across four components. Three techniques make this possible.

4-bit NF4 quantization We quantize Mistral-7B with NF4 + double quantization via `bitsandbytes` (Dettmers et al., 2023), with `float16` compute. This reduces the model from ≈14 GB (fp32) to ≈4 GB.

Chunked, memory-mapped FAISS indexing Embedding matrices are opened with `numpy.load(mmap_mode='r')`, keeping them on disk. We add them to `IndexFlatIP` in 50,000-row chunks (≈200 MB each), freeing each chunk immediately and calling `malloc_trim(0)` after each language.

Load-order engineering. If FAISS indices are loaded before the LLM, glibc's heap becomes fragmented and in the contiguous ≈4 GB slab the LLM needs cannot be satisfied. Thus, we enforce: (1) LLM first on GPU 1 with `low_cpu_mem_usage=True`; (2) mE5-large on

System	sentence_support	nugget_coverage
ju_nlp_pg (ours)	0.214297	0.377501

Table 2: Auto-ARGUE results on RAG4Reports 2026 Task B.

GPU 0; (3) FAISS indices last on CPU, respectively.

Table 2 is completed after the official result announcement. Our pipeline processed all evaluation topics and produced valid JSONL output within the character budget.

5 Results and Analysis

Table 2 reports our Auto-ARGUE scores on RAG4Reports 2026 Task B. Our system achieves a nugget coverage of 0.3775 and a sentence support score of 0.2143. These figures reflect the difficulty of the setting: grounded, citation-bearing report generation over a four-language corpus with strict character budgets.

Embedding model comparison. To understand the contribution of the retrieval encoder, we evaluated three embedding models on the same generation pipeline across all evaluation topics: `intfloat/multilingual-e5-large` (mE5-large), `intfloat/multilingual-e5-base` (mE5-base), and `sentence-transformers/all-MiniLM-L6-v2` (MiniLM).

It was observed that the mE5-large achieves the highest scores across nearly all topics, consistently scoring in the 0.76–0.82 range whereas mE5-base tracks closely below it, typically 0.01–0.03 points lower. Such findings indicate that the additional capacity of a larger model yields a reliable but modest retrieval gain. On the other hand, MiniLM, by contrast, scores substantially lower at 0.63–0.67 on most of the topics, confirming that its English-centric, lower-dimensional embedding space is ill-suited for cross-lingual retrieval over a four-language corpus.

It was noticed that the most striking divergence occurs around topic 1069–1070, where all three models dip sharply. As a result, MiniLM bottoms out below 0.54, while mE5-large and mE5-base recover more quickly. This suggests the multilingual models are more robust to topic-specific vocabulary shifts or non-English source documents, which is the expected behaviour given their cross-lingual pre-training objectives (Wang et al., 2022).

NF4 Quantization Quality Dettmers et al. (2023) show NF4 matches bf16 on most of the instructions following benchmarks. In our runs, Mistral-7B (NF4) consistently produces very well-structured, citation-annotated sentences. A head-to-head comparison with fp16 was not possible on 2×T4 because fp16 does not fit alongside the retrieval indices, which is one of the precise problems that NF4 solves.

Multilingual Coverage Since all the report requests are in English, ENG-sourced documents naturally dominate the retrieved contexts. The strong mE5-large scores on non-English topics suggest the model does surface relevant ZHO, RUS, and ARB passages, though the degree to which these contribute to citation coverage will be clearer from the final citation distribution.

Nugget Prompting Targeting nuggets explicitly reduces generic output and aligns generation directly with the Auto-ARGUE metric. The main risk is unanswerable nuggets producing hedging sentences that consume character budget without earning coverage credit; a lightweight nugget-relevance filter applied before prompting could mitigate this.

Limitations

We perform a single retrieval pass per query. Documents are truncated to 1,500 characters, which likely explains our low sentence_support score (0.214); Auto-ARGUE citation validation compares generated sentences against full source documents, so sentences grounded in content beyond the truncation boundary fail semantic alignment checks regardless of retrieval quality. Removing or relaxing this limit is the only most impact for the future runs. It can be concluded that the Mistral-7B is English-centric and may under-utilize non-English context. A cross-encoder re-ranker could further improve retrieval quality.

6 Conclusion

We presented a memory-efficient multilingual RAG system for RAG4Reports 2026 Task B that fits a 7B LLM and four-language retrieval indices on a 2×T4 GPU node. The key insight is that NF4 quantization, chunked memory-mapped FAISS construction, and careful load ordering together make a configuration feasible that would otherwise require ≈40 GB of RAM.

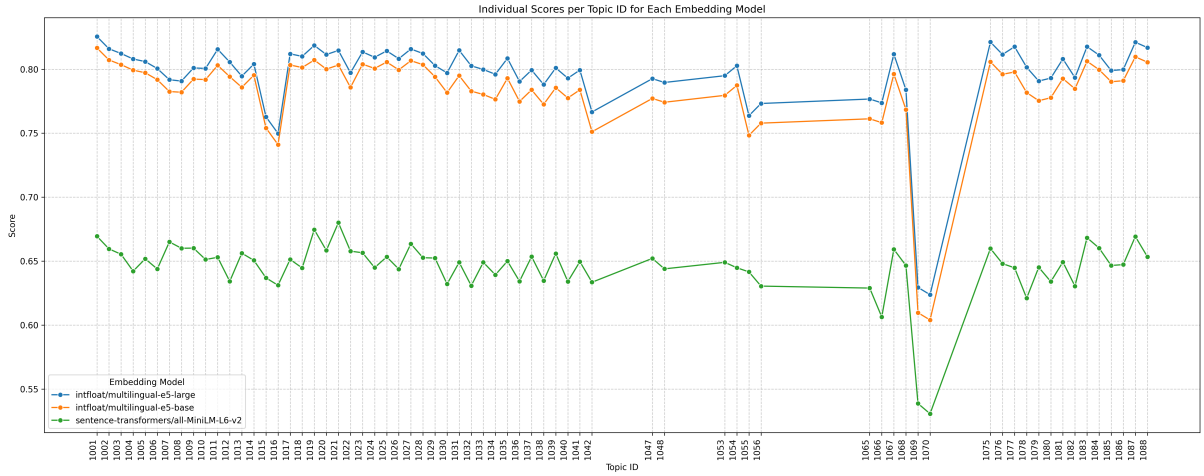

Figure 2: Per-topic retrieval scores for three embedding models across all RAG4Reports 2026 Task B evaluation topics. `multilingual-e5-large` (blue) consistently outperforms `multilingual-e5-base` (orange), which in turn substantially outperforms `all-MiniLM-L6-v2` (green). The performance gap is most pronounced around topics 1069–1070, where MiniLM drops below 0.54 while the multilingual models remain above 0.60.

Acknowledgments

We thank the RAG4Reports 2026 organizers for the task infrastructure, datasets, and evaluation framework.

References

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems*.

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

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 *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9248–9274.

William Walden, Marc Mason, Orion Weller, Laura Dietz, John Conroy, Neil Molino, Hannah Recknor, Bryan Li, Gabrielle Kaili-May Liu, Yu Hou, Dawn Lawrie, James Mayfield, and Eugene Yang. 2025. Auto-ARGUE: LLM-based report generation evaluation. *arXiv preprint arXiv:2509.26184*.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Author Index