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*.