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.

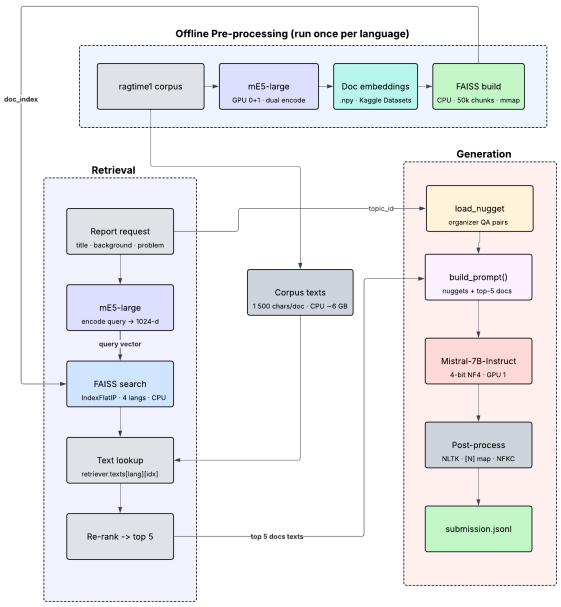

Figure 1: Full system pipeline of `ju_nlp_pg`. *Top (teal):* offline pre-processing, the corpus is encoded with `multilingual-e5-large` on dual GPUs, saved as `.npy` embeddings, and indexed into per-language FAISS indices on CPU. *Bottom-left (blue):* online retrieval, the report request is encoded on 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.

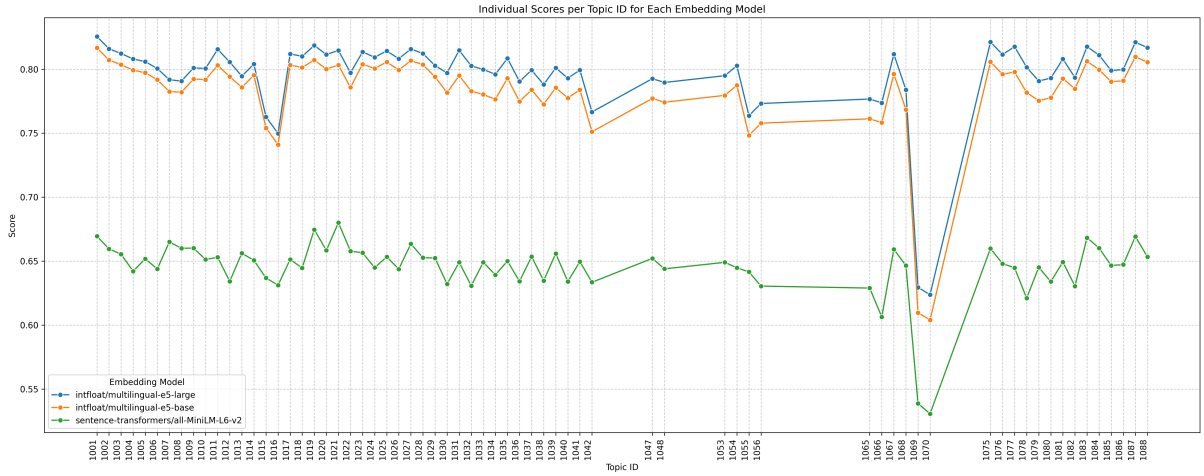

Figure 2: Per-topic retrieval scores for three embedding models across all RAG4Reports 2026 Task B evaluation topics. `multilingual-e5-large` (blue) consistently outperforms `multilingual-e5-base` (orange), which in turn substantially outperforms `all-MiniLM-L6-v2` (green). The performance gap is most 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*.