Gbgers at SemEval 2026 Task 8: A Resource-Constrained Self-CRAG Architecture for Multi-Turn RAG using Open-Source LLMs

Marc'Antonio Lopez*, Filippo Simone Iannello*, Nicolò Colle*, Carmine Benvenuto*, Elia Cola*
*Department of Control and Computer Engineering, Politecnico di Torino, Torino, Italy
{s336362, s345220, s336406, s333307, s338009}@studenti.polito.it

***Abstract*—We present our system for SemEval 2026 Task 8: Multi-Turn Retrieval-Augmented Generation (MTRAGEval). Our approach implements a Self-Corrective RAG (Self-CRAG) architecture using LangGraph, designed to operate entirely offline with open-source models on consumer-grade hardware. The system combines: (1) a Parent-Child chunking strategy (1200/400 characters) that balances retrieval precision with contextual coherence; (2) a two-stage hybrid retrieval pipeline using BGE-M3 embeddings with cross-encoder reranking; (3) a cyclic agentic graph implementing both CRAG (document relevance grading) and Self-RAG (hallucination detection with retry loops); and (4) Llama 3.1 8B with 4-bit NF4 quantization for efficient inference. Our telemetry-enabled fallback mechanism provides granular diagnostics for I_DONT_KNOW responses. To ensure rigorous benchmarking without proprietary APIs, we implemented a custom model-based evaluation framework utilizing Qwen 2.5 14B as an offline judge to assess refusal accuracy and faithfulness. On the development set, we achieve a 64% answer rate with detailed failure attribution across four categories: irrelevant documents (16), hallucination loops (12), and LLM refusals (12). All components are fully reproducible using only open-source tools.**

***Index Terms*—retrieval-augmented generation, self-corrective RAG, multi-turn dialogue, LangGraph, open-source LLMs**

I. Introduction

A. The Task

SemEval 2026 Task 8 challenges participants to build conversational RAG systems capable of handling multi-turn dialogues across four domains: government documents (govt), community Q&A (clapnq), financial Q&A (fiqa), and cloud documentation (cloud). The task requires systems to either provide accurate, grounded answers or explicitly refuse with "I_DONT_KNOW" when the retrieved context is insufficient.

B. Challenges

Multi-turn RAG presents unique challenges beyond single-turn question answering:

Context Decay. As conversations progress, coreferences and ellipses accumulate, making later queries increasingly dependent on dialogue history. A query like "How much does it cost?" requires resolution of the implicit referent from previous turns.

Hallucination Risk. LLMs tend to generate plausible-sounding but unsupported answers when retrieved context is marginally relevant. In conversational settings, this risk compounds across turns.

Calibrated Refusal. Systems must balance informativeness with safety—answering when possible while refusing when context is genuinely insufficient, without being overly conservative.

C. Proposed Approach: The "Sovereign" Strategy

Our solution is designed around three constraints that we call the "Sovereign" strategy:

1) **Fully Offline**: No external API calls (GPT-4, Claude, etc.)
2) **Open-Source Only**: All models publicly available on HuggingFace
3) **Consumer Hardware**: Deployable on consumer-grade GPUs with limited VRAM

These constraints drive every technical decision: 4-bit quantization, CPU offloading for embeddings, and a modular architecture that separates concerns by computational requirements.

II. System Implementation

A. Module: Data Ingestion (src/ingestion.py)

The ingestion pipeline is designed to preserve semantic continuity while remaining efficient on constrained hardware. Rather than treating ingestion as a simple pre-processing step, we explicitly optimize it for multi-turn RAG, where context loss triggers broken coreference, retrieval drift, and hallucinations.

a) Corpus processing and metadata preservation.: The mtRAG corpus is delivered as JSONL files across heterogeneous domains (finance, government, cloud). In `load_and_chunk_data`, we parse each record and explicitly retain `title`, `url`, and `source` fields. These metadata are not archival: they are injected into the retrieval context to enable source-aware generation and improve Faithfulness. This choice avoids a common RAG anti-pattern where metadata is discarded early and cannot be surfaced at answer time.

b) Context fragmentation and the Parent-Child strategy.: A critical failure mode in standard RAG systems is *context fragmentation*: small chunks maximize retrieval precision but often omit antecedents needed to resolve pronouns or ellipses (e.g., "He signed the act" without the subject). To address this, we implement **Parent-Child chunking** via `RecursiveCharacterTextSplitter`. Raw documents are first split into Parent chunks of **1200 characters with**

100-character overlap, sized to preserve coherent narrative units. Each Parent is then subdivided into Child chunks of **400 characters with 50-character overlap**, optimized for dense embedding models.

The key design choice is to *index children but retrieve parents*. We index Child embeddings in Qdrant, while storing the Parent content in metadata (`parent_text`, `parent_title`, `parent_url`, `parent_id`). At query time, a relevant Child chunk triggers retrieval of its Parent text, so the LLM receives a coherent block rather than a fragmented snippet. This hybrid approach couples high-recall vector search with generation-ready context.

c) Vector representation and resource-aware storage.: Embeddings are computed with **BGE-M3** (`BAAI/bge-m3`), chosen for state-of-the-art dense retrieval and robustness to multi-granularity inputs. This capability is crucial because user questions are often short and synthetic (e.g., "Who is the CEO?"), while Child chunks are longer and discursive. To prevent VRAM saturation during ingestion, we load the embedding model once (singleton pattern) and run encoding on CPU when necessary.

All vectors are stored in a local Qdrant collection (`mtrag_unified`) using **cosine similarity**. Each chunk carries domain metadata to support filtered retrieval, reducing cross-domain contamination (e.g., "Europa" in finance vs. NASA). Indexing is executed in batches for throughput stability, and the local persistence of Qdrant eliminates the overhead of a separate container during development.

B. Module: Retrieval (src/retrieval.py)

Dense retrieval using bi-encoders is fast but imprecise: the query and documents are embedded independently, limiting the model's ability to capture fine-grained semantic relationships. Cross-encoders provide more accurate relevance scoring by jointly encoding the query-document pair, but are computationally expensive and non-parallelizable.

We combine both approaches in a **two-stage retrieval pipeline** wrapped in LangChain's `ContextualCompressionRetriever`. First, we use **BGE-M3** (`BAAI/bge-m3`) embeddings to retrieve the **top-20 candidate documents** via fast approximate nearest neighbor search against Qdrant. BGE-M3 was selected for its state-of-the-art performance on multilingual retrieval benchmarks and its efficient 1024-dimensional representations with L2 normalization.

Second, the **BGE-reranker-v2-m3** (`BAAI/bge-reranker-v2-m3`) cross-encoder rescores these 20 candidates by jointly encoding each query-document pair, selecting the **top-5 most relevant documents** for generation. This two-stage approach captures the speed benefits of bi-encoders while achieving the accuracy of cross-encoders for the final selection.

1) Domain Filtering: To prevent cross-domain contamination, the retriever applies a **Qdrant metadata filter** when a domain is specified. The filter uses a `FieldCondition` on `metadata.domain` with exact match semantics, ensuring that only documents from the target corpus are considered. This isolation is critical for multi-domain RAG systems.

2) Parent-Text Deduplication: Since multiple Child chunks may share the same Parent, the `format_docs_for_gen()` function extracts unique parent texts from retrieved documents, preventing redundant context from inflating the prompt. This deduplication uses a set-based approach to guarantee each parent appears only once in the final context.

To maximize GPU memory for the LLM, both the embedding model and reranker run on **CPU** (configured via `device='cpu'`). This architectural decision trades retrieval latency for generation capacity—a worthwhile tradeoff given that generation quality is the primary bottleneck for answer accuracy.

C. Module: Graph Orchestration (src/graph.py)

We implement a **cyclic agentic graph** using LangGraph, moving beyond fragile linear pipelines.

1) Workflow Logic: The graph consists of seven interconnected nodes:

1) **Rewrite**: Transforms context-dependent queries (e.g., "How much is it?") into standalone questions by resolving coreferences.
2) **Retrieve**: Fetches top-20 candidates from domain-filtered Qdrant vector store, then reranks to top-5 using cross-encoder scoring.
3) **Grade Documents**: Implements CRAG (Corrective RAG) to discard irrelevant retrieved documents before generation.
4) **Generate**: Produces answer using the LLM with filtered context.
5) **Hallucination Check**: Implements Self-RAG to validate answer grounding against sources.
6) **Increment Retry**: Loops back for regeneration if hallucination is detected (max 2 retries).
7) **Fallback**: Returns "I_DONT_KNOW" with diagnostic telemetry.

The graph's cyclic nature emerges from intelligent routing decisions that distinguish between recoverable and terminal failures. After document grading, the system evaluates whether the retrieved context is sufficient: if relevant documents exist, processing continues to generation; if all documents are deemed irrelevant, the system immediately routes to fallback—there is no value in attempting generation without grounding. Similarly, after generating an answer, the hallucination checker determines the next action: grounded answers proceed to completion, while hallucinated responses trigger a retry loop that gives the model another opportunity to self-correct. Only after exhausting two regeneration attempts does the system concede defeat and return a diagnostic refusal.

2) State Management: The graph maintains a shared state object flowing through all nodes, capturing the complete trajectory of each query through the pipeline. This state accumulates not only the user's original question and the full conversation history, but also intermediate artifacts generated at each stage: the standalone reformulated query from the

rewrite step, the retrieved documents and their relevance assessments, the generated answer, hallucination detection results, and a running retry counter. Most critically, the state tracks a diagnostic reason code that categorizes why the system chose to return I_DONT_KNOW rather than an answer. This stateful architecture decouples workflow orchestration from node implementations—each node receives the current state, performs its specialized function, and returns incremental updates that propagate forward through the graph.

D. Module: Generation (src/generation.py)

Running large language models on consumer hardware requires aggressive optimization. We employ **Llama 3.1 8B Instruct** with 4-bit NF4 quantization via bitsandbytes, reducing memory requirements from approximately 16GB to 6GB. This enables the full model to fit on a single GPU while reserving capacity for the retrieval components.

1) Query Rewriting: Multi-turn conversations introduce ambiguity through coreferences and ellipses. A user asking "How much does it cost?" after discussing a specific product expects the system to understand the implicit referent. Our **Rewrite** node addresses this through few-shot prompted coreference resolution, transforming context-dependent queries into standalone questions that the retriever can process independently:

> *Chat History:* Who is the CEO of Apple? / Tim Cook.
> *Last Question:* How old is he?
> *Rewrite:* How old is Tim Cook?

This transformation is critical: retrieval quality depends entirely on the query, and ambiguous queries retrieve irrelevant documents.

2) Self-Correction Mechanisms: The core innovation of our architecture is the integration of two complementary verification stages from recent RAG literature.

Corrective RAG (CRAG) addresses retrieval failures. Before passing documents to the generator, each retrieved document is evaluated for relevance using a binary grader. Documents deemed irrelevant are filtered out. If no relevant documents remain after filtering, the system immediately falls back—there is no point generating an answer with irrelevant context.

Self-RAG addresses generation failures. After the LLM produces an answer, a hallucination grader evaluates whether the answer is actually supported by the source documents. We deliberately use a permissive criterion: answers are acceptable if their main claims can be *found in or reasonably inferred from* the documents. Only clear fabrications—wrong dates, invented names, factual contradictions—trigger a "no" score.

When hallucination is detected, the system does not immediately give up. Instead, it loops back to the Generate node, attempting regeneration up to two times. This retry mechanism captures cases where the LLM initially produces a flawed response but can self-correct given another chance. Only after exhausting retries does the system fall back gracefully.

3) Fallback Telemetry: A key design decision is making failures *observable*. Every I_DONT_KNOW response includes a diagnostic reason that categorizes the failure mode. In our pre-fix runs, the telemetry consistently reports three explicit scenarios: `irrelevant_docs` (all retrieved documents fail relevance grading), `hallucination_loop_exhausted` (answers repeatedly fail grounding checks after retries), and `llm_refusal` (the generator explicitly declines). Successful answers are labeled as `none`. This granular telemetry transforms opaque refusals into actionable debugging signals. A prevalence of relevance grading failures suggests the need for corpus expansion or better chunking strategies; frequent LLM refusals indicate overly conservative prompting that should be relaxed; repeated hallucination loop exhaustion points to fundamental model limitations that might require upgrading to a larger or better-tuned variant.

E. Module: Evaluation (eval/evaluate.ipynb)

To rigorously validate our architecture's effectiveness within a resource-constrained environment where "hallucination is fatal," we developed a semi-deterministic offline evaluation pipeline. Unlike traditional lexical metrics (e.g., ROUGE or BLEU), which fail to capture factual consistency or the validity of a refusal, our system employs a hybrid *Model-Based Evaluation* approach.

The framework utilizes **Qwen2.5-14B-Instruct** as the *Judge LLM*—selected for its superior reasoning capabilities over 7B/8B models—and **all-MiniLM-L6-v2** for efficient vector similarity calculations. The evaluation logic is bifurcated based on the agent's output:

a) 1) Refusal Accuracy Protocol: Given that the dataset contains unanswerable questions, the ability to correctly refuse is paramount. When the agent emits the `I_DONT_KNOW` token, the system triggers a specific routine (`evaluate_refusal_justification`) that does not penalize the absence of an answer but evaluates the **justification** of the refusal:

- **Input:** User question and *Retrieved Context*.
- **Logic:** The Judge verifies if the context actually contained the necessary information.
- **Scoring:**
  - *Score 10 (Correct Refusal):* The context does NOT contain the answer. The refusal is validated.
  - *Score 1 (False Negative):* The context CONTAINS the answer. The agent failed to recognize available information.

In this scenario, the *Correctness* metric is marked as "N/A" and the *Hallucination* flag is automatically set to "NO".

b) 2) Answer Assessment Pipeline: When the agent produces a discursive response, evaluation proceeds along two parallel axes to measure both context adherence and semantic correctness:

a. Faithfulness & Hallucination Detection: The Judge evaluates solely if the response is supported by the retrieved documents, isolating the system from external knowledge

leakage. The prompt enforces a structured JSON-like format extracting:

- **Hallucination Flag:** A binary classifier (`YES`/`NO`) triggered if the response cites facts not present in the context.
- **Score (1–10):** A scale that severely penalizes inventions (1–4) and rewards verified details (8–10).

b. Hybrid Semantic Correctness: To assess alignment with the *Ground Truth* while reducing computational costs, we implement a two-stage approach:

1) **Vector Shortcut:** We calculate Cosine Similarity (via MiniLM) between the generated answer and the Ground Truth. If similarity exceeds **0.95**, the system automatically assigns a score of **10/10**, assuming perfect semantic equivalence.
2) **LLM Grading:** Below this threshold, the *Judge LLM* evaluates semantic correctness (1–10), distinguishing between factual errors (1–3), partial answers (4–6), and valid paraphrases (7–10).

c) 3) Robust Parsing & Technical Implementation: The implementation addresses Qwen-specific behaviors, explicitly managing special stop tokens (e.g., `<|im_end|>`) to prevent generative loops. Scores are extracted via **Regular Expressions (Regex)** that isolate the `Reasoning`, `Score`, and `Hallucination` fields from the model's Chain-of-Thought. To guarantee benchmark reliability, any parsing failure or runtime error triggers a *pessimistic fallback* mechanism, assigning the minimum score (1.0) to prevent artificial performance inflation.

III. Experimental Setup

A. Software Stack

Our system is designed to run on consumer-grade GPUs with limited VRAM. Key components:

- **Orchestration**: LangGraph for cyclic graph execution
- **Vector Store**: Qdrant with local persistence
- **LLM**: Llama 3.1 8B with 4-bit NF4 quantization (bitsandbytes)
- **Embeddings**: BGE-M3 via sentence-transformers
- **Evaluation**: A custom offline pipeline leveraging **Qwen2.5-14B-Instruct** as the Judge and **all-MiniLM-L6-v2** for semantic similarity.

B. Dataset

The MTRAGEval dataset comprises conversations across four domains:

- **govt**: Government documents
- **clapnq**: Community Q&A (ClapNQ subset)
- **fiqa**: Financial Q&A
- **cloud**: Cloud documentation

We indexed approximately 100,000 document chunks across all domains in a unified Qdrant collection with domain metadata for filtered retrieval.

TABLE I
Task C Results on Development Set (110 conversations)

Outcome	Count	Percentage
Answered	70	63.6%
I_DONT_KNOW	40	36.4%
Total	**110**	100%

TABLE II
Fallback Reason Breakdown

Reason	Count
none (successful)	70
irrelevant_docs	16
hallucination_loop_exhausted	12
llm_refusal	12

IV. Experimental Results

A. Quantitative Results

The switch to Llama 3.1 8B significantly improved the document grader's recall, halving the `irrelevant_docs` failures from 32 (with 3B model) to 16. **(Separate experiments; not part of the pre-fix run reported here.)** This suggests that the larger model is much better at identifying subtle relevance. The remaining 16 cases likely represent true corpus gaps.

B. Qualitative Case Studies

To contextualize the quantitative metrics, we analyze specific interaction patterns. Table III summarizes three representative cases, while the subsequent paragraphs provide a detailed fluid analysis of the system's reasoning and behavior in each scenario.

TABLE III
Qualitative Analysis of Representative Interactions

ID	Outcome	Interaction Summary
govt_4	Success	**Q**: "Are they all by us?" (Mars missions) **A**: Correctly identifies international collaboration (ESA).
govt_7	Safe Failure	**Q**: "Can a 17 year old be the defendant?" **A**: Fallback to I_DONT_KNOW due to ambiguity.
clapnq_4	Retrieval Limit	**Q**: "secondary source" **A**: Fallback due to insufficient query context.

a) Success Case: Multi-Hop Reasoning (ID: govt_4): The user asked "Are they all by us?", referring to Mars missions mentioned in the previous turn. The system successfully resolved the coreference "they" to the missions and "us" to the United States/NASA. By retrieving documents describing international collaborations, it synthesized a correct answer acknowledging the European Space Agency's role.

This demonstrates the effectiveness of the Rewrite-Retrieve-Generate pipeline in handling context-dependent queries.

b) Safe Failure: Ambiguity Handling (ID: govt_7): When asked "Can a 17 year old be the defendant in a claim?", the system returned `hallucination_loop_exhausted` (I_DONT_KNOW). The ground truth answer for this query admits "It is unclear". The system, unable to find a definitive "yes" or "no" in the retrieved legal documents, repeatedly attempted to generate an answer but failed the hallucination check. Rather than fabricating a legal fact, it correctly fell back, confirming that the hallucination grader effectively prevents confidently wrong answers in high-stakes domains.

c) Retrieval Limit: Short Query (ID: clapnq_4): The user provided a generic, keyword-only query: "secondary source". The Rewriter failed to expand this into a distinct question, and the dense retriever failed to find documents exceeding the relevance threshold. This case highlights a limitation in handling extremely telegraphic user inputs and suggests a need for more aggressive query expansion strategies to bridge the gap between keyword search and semantic retrieval.

C. Automated Evaluation Analysis (Judge-Based)

To validate the system's internal telemetry, we executed the offline evaluation pipeline described in Section II-E using **Qwen2.5-14B-Instruct**. The strict evaluation reveals a significant divergence between the agent's confidence and the judge's assessment. While the system achieved a 63.6% answer rate, the granular analysis highlights the complexity of balancing "helpfulness" with strict factual adherence. The judge identified that while some refusals were false negatives, the system successfully avoided hallucination in cases where the context was legitimately completely missing, though it struggled with partial context matches.

D. Evaluation Results

In this section we analyze specific interaction patterns verified by the offline Judge. Table IV details five representative instances, juxtaposing the agent's output with the evaluator's reasoning.

a) Precision Verification (`clapnq_14`): This instance validates the system's end-to-end fidelity. The Judge assigns a maximum score of 10.0, confirming that the Parent-Child chunking strategy successfully preserved the numerical details required to answer the query without introducing extraneous noise.

b) Granular Penalty for Omission (`fiqa_8`): Unlike binary metrics, the Judge demonstrates nuanced scoring by assigning a 7.0. While the generated answer is factually correct in a broad sense, the evaluator penalizes the omission of a specific exemption found in the retrieved documents ("under $10"). This highlights the framework's ability to distinguish between "helpful but incomplete" answers and hallucinations.

c) True Negative Identification (`cloud_3`): This case exemplifies a successful "Safe Refusal." The user requested a command absent from the documentation. Crucially, the Judge (having access to the same context) confirms the absence of the information, validating the agent's decision to fallback to `I_DONT_KNOW` as a correct architectural behavior rather than a failure.

d) Domain Mismatch Detection (`govt_1`): The Judge severely penalizes (Score 1.0) a case of "Knowledge Leakage." The agent, triggered by a vague query, ignored the retrieved technical context (API docs) to generate a response based on pre-training knowledge (space radiation). The evaluation pipeline correctly flags this as a hallucination, enforcing the strict "grounding-only" constraint required by the task.

e) False Negative Attribution (`govt_5`): Conversely to `cloud_3`, here the Judge exposes a limitation in the 8B model's reasoning. Although the agent refused to answer, the 14B Judge successfully identified the relevant information ("improper facility use fines") within the context. This discrepancy (Score 1.0 on a Refusal) provides precise telemetry, attributing the failure to the agent's conservatism rather than a retrieval gap.

V. Ablation Studies and Discussion

A. Impact of Domain Filtering

Initial experiments without domain filtering showed severe cross-domain contamination. For example, a query about "Europa Clipper" (NASA mission in govt domain) retrieved financial documents about "European markets." Adding domain-specific filtering eliminated these errors.

B. Impact of Prompt Permissiveness

We iterated on generator and grader prompts to balance precision with recall:

Strict Prompt: "Only answer if the context DIRECTLY contains the answer" **(Separate experiments; not part of the pre-fix run reported here.)**

- Result: High I_DONT_KNOW rate (85%), many valid answers rejected

Permissive Prompt: "Use context as PRIMARY source; reasonable inferences allowed" **(Separate experiments; not part of the pre-fix run reported here.)**

- Result: 64% answer rate, substantial reduction in false negatives

C. Impact of Parent-Child Chunking

Compared to flat chunking (400 chars), Parent-Child chunking:

- Preserved document-level coherence for complex queries
- Reduced context fragmentation in retrieved passages
- Enabled multi-sentence answers requiring broader context

D. Limitations

a) Corpus Gaps vs. System Failures: Our error analysis reveals that the system's performance ceiling is significantly lowered by data sparsity. Specifically, 16 out of the 40 failures (40%) in the development set were classified as "Valid Refusals" or verified corpus gaps. **(Separate experiments; not part of the pre-fix run reported here.)** This indicates that the raw answer rate of 63.6% is partially a reflection of

TABLE IV
SELECTED ROWS FROM EVALUATION LOGS

ID	Question	Generated Answer	Judge Reasoning (Excerpt)	Score	Verdict
clapnq_14	"How many teams?"	"The NFL consists of 32 teams, divided into two conferences: AFC and NFC, each with 16 teams."	"The answer accurately reflects the information... mentioning the division into two conferences... No hallucinations."	**10.0**	Perfect Match
fiqa_8	"Is interest taxable?"	"Yes, interest income is generally taxable and must be reported on your federal tax return."	"The answer correctly states that interest is taxable, but fails to mention the specific exemption for amounts under $10 mentioned in Doc 2."	**7.0**	Partial Accuracy
cloud_3	"Force restart command"	"I_DONT_KNOW"	"The retrieved context discusses graceful shutdown procedures but does not contain the specific force restart command requested. The refusal is justified."	**10.0**	Valid Refusal
govt_1	"Sollutions"	"Solving this issue is crucial for sending humans into the solar system. Better shielding is needed."	"The answer discusses solutions to protect astronauts from space radiation... The context talks about API operations. Completely unrelated."	**1.0**	Severe Hallucination
govt_5	"Fines and penalties"	"I_DONT_KNOW"	"The context contains information related to fines for improper facility use... It specifies fines for not abiding by organic waste requirements. The refusal is a False Negative."	**1.0**	Bad Refusal

the dataset's incompleteness rather than purely an architectural limitation. In these cases, the system correctly identified that no document contained the answer, behaving exactly as designed.

b) Model Capacity Constraints: The use of a 4-bit quantized 8B model (Llama 3.1) introduces distinct reasoning bottlenecks. As seen in the "False Negative" refusals (Table IV, govt_5), the compressed model struggles to perform multi-hop deductions or synonym matching that the unquantized 14B Judge handles effortlessly. This confirms that while consumer hardware allows for deployment, it imposes a hard limit on semantic reasoning depth.

c) Agent-Judge Alignment Gap: A significant factor affecting our quantitative scores is the intentional strictness mismatch between the runtime agent and the offline evaluator. The agent is prompted to be "helpful" and permissive (allowing reasonable inferences to maximize user satisfaction), whereas the Qwen-14B Judge is prompted to be "strict" and penalize any information not explicitly present in the text. This divergence explains the high Hallucination Rate (87%). **(Separate experiments; not part of the pre-fix run reported here.)** This highlights the trade-off between conversational fluency and strict RAG adherence.

VI. CONCLUSION

We presented a Self-CRAG architecture for multi-turn RAG that operates entirely with open-source components on consumer hardware. Key contributions:

1) **Cyclic Agentic Graph**: LangGraph-based workflow enabling self-correction through document grading (CRAG) and hallucination detection (Self-RAG)
2) **Parent-Child Chunking**: Two-level indexing strategy balancing retrieval precision with contextual coherence
3) **Telemetry-Enabled Fallback**: Granular diagnostics for I_DONT_KNOW responses enabling targeted debugging
4) **Resource Efficiency**: Full system runs on a single consumer-grade GPU using 4-bit NF4 quantization

Our 64% answer rate demonstrates that carefully engineered open-source systems can achieve competitive performance on complex RAG tasks without proprietary APIs.

Future Work. Directions include: (1) larger open-source LLMs when hardware permits, (2) query routing to domain-specific retrievers, and (3) active learning to expand corpus coverage for frequent failure patterns.

ACKNOWLEDGMENT

This work was conducted as part of the Large Language Models course at Politecnico di Torino.