AMU at TREC 2025 RAGTIME: A Minimalist GPT-5 Baseline for Multilingual Retrieval-Augmented Report Generation

Maciej Czajka[1]
maciej.czajka@amu.edu.pl

Piotr Jabłoński[1]
piotr.jablonski@amu.edu.pl

Konrad Pierzyński[1]
konrad.pierzynski@amu.edu.pl

Ryszard Staruch[1]
ryszard.staruch@amu.edu.pl

[1]Center for Artificial Intelligence,
Adam Mickiewicz University, Poznan, Poland

Abstract

This notebook paper presents the AMU system submitted to the TREC 2025 RAGTIME Track. The goal of our participation was to evaluate how well a minimalist, baseline Retrieval-Augmented Generation (RAG) system can perform on the multilingual report-generation task without using reranking, hybrid retrieval, or task-specific optimization techniques. Our pipeline combines a simple Qdrant vector search with BAAI/bge-m3 embeddings and structured GPT-5 generation. Despite its intentionally lightweight design, the system produces valid citation-grounded JSON reports and achieves stable performance in both internal evaluation and the TREC AutoJudge Pilot. We describe the architecture, prompt design, schema constraints, and evaluation results, demonstrating the viability of a quick baseline approach for the RAGTIME task.

1 Introduction

The TREC 2025 RAGTIME Track (Retrieval-Augmented Generation TREC Instrument for Multilingual Evaluation) evaluates systems that generate citation-supported English reports based on multilingual document collections spanning Arabic, Chinese, English, and Russian. Participants must retrieve relevant evidence, assemble a coherent factual summary, and attach supporting citations for each sentence. The evaluation focuses on nugget recall and citation precision [1].

In our submission, we intentionally focused on building a minimal, baseline system designed to assess how well a straightforward RAG pipeline can solve the task without any reranking components, hybrid retrieval strategies, domain-specific models, or additional optimization mechanisms. The purpose of the system was not to push performance boundaries, but rather to establish a clear and interpretable reference point for multilingual retrieval-augmented report generation under the constraints of the RAGTIME Track.

Our work follows the general Retrieval-Augmented Generation (RAG) paradigm, where a generator model is conditioned on documents retrieved from a large external corpus [2]. The AMU system is a multilingual RAG pipeline powered entirely by GPT-5 for generation, Qdrant for retrieval, and the BAAI/bge-m3 embedding

model [3] for multilingual semantic search. The system enforces strict JSON-schema compliance required by the track.

1.1 Task Description

The TREC 2025 RAGTIME track focuses on evaluating systems capable of synthesizing long-form, citation-grounded reports from multilingual sources. The challenge is designed to move beyond factoid question answering by requiring systems to synthesize multi-faceted narratives while maintaining strict attribution to source documents.

1.2 Report Requests and Task Variants

Instead of traditional keyword queries, systems are provided with detailed ”report requests.” Each request is formulated to simulate a real-world information need and consists of three main components:

- Title: A concise topic identifier.
- Background: A persona or scenario description that establishes the context and specific needs of the user requesting the report.
- Problem Statement: Detailed instructions outlining the exact aspects, facts, or questions the report must cover.

Participants could submit runs for two main generation subtasks: Multilingual Report Generation (utilizing the full four-language corpus) and English Report Generation (utilizing the machine translated corpus provided by organizer). Furthermore, the generated reports were strictly constrained by length limits, typically set to either 2,000 characters (short topics) or 10,000 characters (long topics). Systems were required to output the final report in a specific JSON format, where each factual sentence is accompanied by a dictionary of citations mapping to the specific source documents.

2 System Description

The system architecture consists of three main modules:

1. Retrieval using Qdrant vector search.
2. Document reconstruction and context assembly.
3. Structured report generation using GPT-5.

2.1 Retrieval Layer

All documents are preprocessed into semantic chunks using a JSONL batching pipeline. Each chunk is embedded using the **BAAI/bge-m3** model [3] and stored in a Qdrant collection.

At inference time, the system embeds a combined query composed of the report title, background, and problem statement. Qdrant returns the top 15 most similar chunks. Chunk scores are aggregated to determine the highest-ranking original documents.

2.2 Context Reconstruction

To satisfy the requirement that citations refer to entire documents, the system reconstructs top documents using Qdrant's scroll API. Chunks belonging to the same document are sorted by `chunk_id` and concatenated. The GPT-5 model receives context in the following format:

```
Document ID: <id>
Content: <merged-content>
Relevance Score: <score>
```

2.3 Report Generation with GPT-5

GPT-5 is used exclusively for generation. The system relies on JSON schema enforcement and strict instruction-following.

The generator must:

- Produce factual statements supported by the context.
- Assign citations using dictionaries of the form {`doc_id: confidence`}.
- Avoid hallucinations.
- Provide a final standalone conclusion.
- Follow word limits computed from the task's character constraints.

3 Prompt Design and Instruction Engineering

The AMU system relies on carefully designed prompts that enforce factuality, citation correctness, and JSON-schema structure. The prompting layer consists of (a) a system instruction and (b) a user instruction containing the metadata and assembled document context.

Below we present the exact prompts used in our system.

3.1 System Prompt (Exact Version)

```
You are a multilingual RAG report generation assistant. Your task is to
   generate accurate, well-structured English reports based solely on the
   provided document context.

    For each factual sentence:
    - Do NOT include document IDs or confidence scores in the text.
    - Assign citations separately, using a dictionary format: {document_id:
       confidence_score}.
    - Confidence scores must reflect how strongly the document supports the
       sentence:
        - Use 100.0 only for exact, unambiguous support.
        - Use 60.0 - 90.0 for partial or inferred support.
        - Use 20.0 - 59.0 for weak or indirect support.

    At the END of the report:
```

```
    - Add a short, clear summary sentence that directly answers the user's
       question or problem statement.
    - This should be a standalone paragraph or sentence that serves as the
       final conclusion.
    - Do NOT include document IDs or scores in this conclusion.

    Avoid hallucinations and only use facts explicitly found in the context.
    Return the response in strict JSON format as defined by the schema.
```

3.2 User Prompt (Exact Version)

```
You are an AI assistant that generates accurate, citation-supported reports
   based ONLY on the given document context.

    Main thought: {title}

    User Profile: {background}
    User Query: {problem_statement}

    Context Information:
    {context}

    Instructions:
    1. Write a clear and informative report that covers all relevant facts
       from the context related to the user's query.
    2. Structure your answer using logical sentences or short paragraphs.
    3. Do NOT include any document IDs or confidence scores in the text itself
       .
    4. For every factual sentence, assign citations as a dictionary: keys are
       document IDs (strings), values are confidence scores (float from 0.0
       to 100.0).
    5. Only use citations when the source provides full or partial support.
    6. At the end of your report, **in the last response segment**, add a
       short, direct answer to the user's problem statement in a separate
       sentence or paragraph - clearly summarizing your conclusion based on
       the context.
    7. If no relevant information is available, say so clearly in the final
       answer.
    8. Never hallucinate or assume facts not directly present in the context.
    9. Your entire output must match the JSON schema exactly.
    10. Divide your answer into simple sentences.
    11. **The total number of words across all text fields in the JSON
       responses array must not exceed {limit//6} words.**
    Return ONLY valid JSON that matches the schema.
```

4 Structured Output Schema

The TREC RAGTIME Track requires systems to produce citation-grounded reports formatted as structured JSON. To ensure correctness, our system enforces a strict schema at generation time using Pydantic. The schema consists of two components: individual response items and the overall container object.

Each response item contains the generated text and a dictionary of citations, where keys correspond to document identifiers and values represent confidence scores assigned by GPT-5. The structure is defined as follows:

```
class ResponseItem(BaseModel):
    text: str = Field(
        description="One sentence or short paragraph with information"
    )
    citations: Dict[str, float] = Field(
        description="List of document IDs used for this information",
        default={}
    )
```

A full system output is represented as a list of such items:

```
class StructuredResponse(BaseModel):
    responses: List[ResponseItem] = Field(
        description="List of response items with references"
    )
```

The schema ensures that:

- each factual statement is represented as a separate element,
- citations are always present in dictionary form,
- confidence scores are numeric and bounded between 0.0 and 100.0,
- the final answer is delivered in the last `ResponseItem` without citations.

During generation, GPT-5 is constrained to output strictly valid JSON conforming to this schema using the `json_schema` response format. After decoding, an additional post-processing step converts any list-style citations into dictionary form to satisfy the required format. This guarantees compliance with both the RAGTIME specification and the internal system requirements.

5 Implementation Details

Table 1 lists the main components used in the system.

5.1 Retrieval Parameters

- Embedding model: BAAI/bge-m3 [3],
- Top chunks retrieved: 15,
- Max reconstructed documents: 5–7,

Component	Technology
Retrieval Engine	Qdrant Vector Database
Embedding Model	BAAI/bge-m3 [3]
Backend Framework	FastAPI
LLM Generator	GPT-5, GPT-5-mini
Schema Enforcement	Pydantic JSON Schema

Table 1: System components.

- Similarity metric: cosine.

5.2 GPT-5 Generation Parameters

- Model name: gpt-5-mini
- Temperature: 0.0 (deterministic),
- JSON schema output,
- Max tokens: ~ 0.75 of allowed limit,
- Citation format: {`doc_id: score`}.

Although both gpt-5 and gpt-5-mini were tested, manual inspection indicated that gpt-5-mini was sufficient for the task. For simplicity, we refer to the generator generically as "GPT-5", denoting the model family used in the system.

6 Results

Since the AMU system was intentionally designed as a minimalist baseline without reranking, hybrid retrieval, or internal tuning, all evaluation results come exclusively from the official scoring performed by the TREC organizers using the ARGUE (Automated Report Generation Under Evaluation) framework [4, 5]. The assessment process is designed to measure both the completeness of the information provided and the reliability of the citations. The evaluation revolves around two primary dimensions:

- Nugget Coverage: This metric assesses the content completeness of the report. Human assessors first identify critical pieces of information, formulated as question-answer pairs called "nuggets," from the source documents. A generated report is then evaluated on how many of these essential nuggets it successfully incorporates.

- Sentence Support (Citation Accuracy): This metric evaluates the groundedness of the report. For every factual sentence generated by the system, the provided citations are checked to verify whether the referenced document actually supports the claim made in the sentence.

An overall F1 score is computed as the harmonic mean of Nugget Coverage and Sentence Support, balancing the trade-off between providing comprehensive information and ensuring strict factual attribution. For the 2025 track, short topics underwent full manual evaluation by human assessors, while long topics were evaluated using Auto-ARGUE, an LLM-based automated evaluation system that utilizes human-curated nuggets as a reference.

6.1 Auto-ARGUE

Table 2 summarizes the aggregate ("all-topic") metrics for our English-only (ENG) and multilingual (ML) runs.

Metric (micro-average)	ENG	ML
Sentence support	0.797	0.810
Nugget coverage	0.396	0.357
F1	0.529	0.496
Citation support	0.831	0.823

Table 2: Official Auto-ARGUE results for the AMU baseline system.

Overall, the system demonstrates strong citation behavior—citation support is around 0.82–0.83 and sentence support reaches 0.80-0.81 — indicating that used model reliably attaches correct evidence when provided retrieved documents. Nugget coverage remains moderate, reflecting the limitations of a purely vector-based retrieval baseline without task-specific optimization.

6.2 Almost Human Evaluation

In this phase, human assessors manually verify the generated reports against the source documents to determine citation support and nugget coverage. However, due to the scale of the evaluation and potential missing judgments for certain sentences, the final scores are calculated using three distinct interpretation strategies.

Table 3 presents the Almost Human evaluation results. The system achieved a very high optimistic sentence support score of 0.813 for english translated dateset submission (ENG) and 0.821 for multilingual submission (ML), but this metric drops significantly under the pessimistic and LLM-filled interpretations (0.490 - 0.496, and 0.496 - 0.505 respectively). This substantial gap highlights the uncertainty in areas where human assessors did not provide explicit judgments. Interestingly, the official fully automated Auto-ARGUE score for sentence support (0.797 and 0.810 micro-average, as shown in Table 2) aligns much closer to the optimistic Almost Human variant. This suggests that the LLM evaluator used in the fully automated Auto-ARGUE pipeline exhibits a lenient tendency, frequently accepting citation support in ambiguous cases where explicit human confirmation is absent.

Metric	ENG	ML
Sentence Support (Optimistic)	0.813	0.821
Sentence Support (Pessimistic)	0.490	0.490
Sentence Support (LLM-filled)	0.496	0.505
Nugget Mentioned	0.289	0.269
Nugget Coverage	0.178	0.181

Table 3: Almost Human evaluation results for the AMU English-only (ENG) baseline.

Regarding content completeness, the Almost Human evaluation reports a Nugget Mentioned score of 0.289/0.269 and a strict Nugget Coverage of 0.178/0.181. These figures are lower than the automated Auto-ARGUE nugget coverage estimates 0.396/0.357, further confirming that the automated framework

tends to be more generous in recognizing semantic equivalence between generated text and reference nuggets than human assessors.

7 Conclusion

We presented the AMU baseline system for the TREC 2025 RAGTIME Track: a minimalist multilingual Retrieval-Augmented Generation pipeline based entirely on GPT-5 for structured, citation-aware report generation. The design intentionally avoids reranking, hybrid retrieval, and task-specific optimization, in order to provide a clean reference point for future systems. Qdrant performs vector-based retrieval, BAAI/bge-m3 supplies multilingual embeddings, and GPT-5 generates schema-compliant reports with sentence-level citation dictionaries.

As a baseline, the system establishes a foundation against which more sophisticated RAGTIME approaches can be compared. Future work will explore improved retrieval recall, enhanced citation precision, hybrid scoring strategies, and document-level evidence aggregation tailored to multilingual settings.

References

[1] TREC RAGTIME Organizers. "TREC 2025 RAGTIME Track Guidelines." 2025. https://trec-ragtime.github.io/.

[2] 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. "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks." arXiv:2005.11401, 2021. https://arxiv.org/abs/2005.11401.

[3] Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. "BGE M3-Embedding: Multi-Lingual, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation." arXiv:2402.03216, 2024. https://arxiv.org/abs/2402.03216.

[4] 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. "Auto-ARGUE: LLM-Based Report Generation Evaluation." arXiv:2509.26184, 2025. https://arxiv.org/abs/2509.26184.

[5] James Mayfield, Eugene Yang, Dawn J. Lawrie, Sean MacAvaney, Paul McNamee, Douglas W. Oard, Luca Soldaini, Ian Soboroff, Orion Weller, Efsun Kayi, Kate Sanders, Marc Mason, and Noah Hibbler. "On the Evaluation of Machine-Generated Reports." In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2024. https://api.semanticscholar.org/CorpusID:269502216.