Team LLMinds Submission for ELOQUENT Sensemaking Task

Anna Sajdoková[1,2,*], Matěj Macek[1,†], Ondřej Hlava[1,†], Marek Štefanec[1,†], Antonín Kříž[2,†], Jakub Kučera[2,†] and Ondřej Bojar[1,†]

[1] *Charles University, Faculty of Mathematics and Physics, Institute of Formal and Applied Linguistics, Malostranské náměstí 25, 118 00 Prague, Czech Republic*

[2] *Czech Technical University in Prague, Faculty of Information Technology, Thákurova 9, 160 00 Prague 6, Czech Republic*

Abstract

This paper presents our system submission for the CLEF 2025 ELOQUENT Sensemaking Task, which focuses on generating and answering questions based solely on provided textual materials, such as lecture transcripts and textbooks. Our approach combines open-source large language models (LLMs) in a two-stage pipeline: a **Teacher** component for question generation, a **Student** for grounded answering using Retrieval-Augmented Generation (RAG). Ideally, the third stage would be an **Evaluator** for an assessment of response quality but we leave this part for the future.

To ensure transparency, privacy, and accessibility, we prioritized using local models such as LLaMA 7B and DistilQwen 1.5B, avoiding reliance on proprietary APIs. For question generation, we implemented both a simple single-question prompting method and an enhanced generator-discriminator pipeline that scores candidates on answerability, context relevance, and diversity. The Student model leverages a RAG system with FAISS to extract relevant context chunks and generate concise answers.

We evaluated several configurations, including standalone model outputs and hybrid combinations (llama + distilqwen), with the final answers polished by OpenAI under strict token and editing constraints. Our findings demonstrate that lightweight local models, when properly orchestrated, can produce competitive results in question answering tasks while respecting data sovereignty and resource limitations.

Keywords

Large Language Models, Question Generation, Question Answering, Retrieval-Augmented Generation, Generator-Discriminator Pipeline, Open-Source NLP, Educational NLP, Local Inference

1. Introduction

The ELOQUENT Sensemaking task, part of the CLEF 2025 evaluation lab, challenges participants to develop systems capable of generating and answering questions based solely on provided textual materials, such as lecture transcripts or textbook excerpts. The primary objective is to assess the ability of large language models (LLMs) to comprehend and reason over specific content provided as input and without being overly influenced by the knowledge accumulated in the model from the training data.

Šindelář and Bojar published the process of making this task as well as its evaluation in the article [1].

Our project, LLMinds, originally aimed to construct a pipeline comprising three components: a **Teacher** that generates questions from the input materials, a **Student** that answers these questions using only the given context, and an **Evaluator** that assesses the quality of the generated question-answer pairs. Eventually, we skipped the **Evaluator** component due to time constraints.

A significant motivation for our approach is the desire to utilize open-source, locally deployable models instead of proprietary solutions like OpenAI's GPT series. Proprietary models mostly require sending data to external servers, raising concerns about data privacy and compliance. This problem can

CLEF 2025 Working Notes, 9 – 12 September 2025, Madrid, Spain

[†] All the authors contributed equally to this work.

sajdokova@gmail.com (A. Sajdoková); matej.macek@email.cz (M. Macek); ohlava@gmail.com (O. Hlava); marek.stefanec@gmail.com (M. Štefanec); krizant2@fit.cvut.cz (A. Kříž); kucerj56@fit.cvut.cz (J. Kučera); bojar@ufal.mff.cuni.cz (O. Bojar)
0000-0002-0606-0050 (O. Bojar)

be even more pressing in educational settings where the school is obliged to protect the privacy of its student's data. Additionally, such models may demand substantial computational resources and incur usage costs, making them less accessible for institutions with limited budgets.

By leveraging local models, we aim to:

- Ensure data privacy by processing all information on-premises.
- Reduce dependency on external services and associated costs.
- Enable deployment on hardware with limited computational capabilities.

This approach aligns with the goals of the ELOQUENT lab to explore trustworthy, controllable, and accessible LLM applications in real-world scenarios.

2. Related Work

The field of question generation and answering has seen significant advancements with the emergence of large language models (LLMs). Transformer-based architectures such as BERT [2] and GPT [3] have demonstrated strong capabilities in processing and generating natural language. Retrieval-Augmented Generation (RAG) [4] further enhances performance by combining dense retrieval mechanisms with generative models, allowing systems to ground their outputs in external knowledge.

In our Student setup, we build on RAG, optimizing both retrieval relevance and answer quality. In the Teacher submission, we try generator-discriminator pipeline much like in generative adversarial networks (GANs).

3. The System

3.1. Teacher Submission

The Teacher component is tasked with creating questions from the provided content, be it lectures or books, for the Student component to answer. We implemented the Teacher in two ways. First by using small local models to generate only one single question and then using larger models available through APIs to generate multiple questions at the same time with large context windows and focus on uniqueness.

Our second approach to Teacher task uses generator-discriminator pipeline further described in Section 3.1.2.

3.1.1. Small Local Models Approach

We started with multiple smaller local models, ranging from Llama-3.2-1B and Llama-3.2-3B-Instruct to larger "thinking" models like DeepSeek-R1-Distill-Qwen-14B-GGUF.

To obtain questions, we prompted the LLM with the provided input document and a request to generate a question. To battle the repetitiveness of the questions generated by the models, we set the random seed of each run randomly. Since the provided input material can be quite large, we used only the last 16384 characters of each document to fit them in the available VRAM.

3.1.2. Discriminator and Generator Approach

In our second approach to the Teacher task, we use an LLM with different seeds to generate 5 questions per entry. Our question generation system includes a simple *discriminator* component that evaluates and selects high-quality questions from a larger pool of generated candidates.

We aim to ensure that the selected questions are diverse, relevant, and of appropriate difficulty. The discriminator uses the following metrics:

1. Answerability By *answerability*, we aim to verify that a question can be completely and accurately answered using **only** the information provided in the document excerpt. We estimate answerability with the same large language model (LLM) used for generation, by applying the following prompt:

> **Prompt:**
> Can the question be answered completely and accurately using **only** the information provided in the document excerpt?
> Answer with just `Yes` or `No`.

We automatically filter out all questions where the LLM returns `No`.

We optionally use a more advanced version of the prompt to improve the estimate: First, the LLM is asked to select the most relevant passage from the context that supports the answer. Then, in a second step, we prompt the LLM *without any further context* to decide whether that selected passage is sufficient to answer the question. This two-step process acts as a confidence check and helps avoid mistakenly labeling unanswerable questions as valid.

2. Context Relevance *Context relevance* numerically estimates how much the question is related to the provided context. We use lexical overlap between the question and the context:

- Common stop words are removed from the question.
- We compute what percentage of **meaning-bearing words** from the question (excluding interrogative words like *what*, *where*, etc., unless contentful) also appear in the context.
- The score ranges from 0.0 to 1.0, where a higher score indicates that the question is more directly grounded in the given context.

3. Diversity Score We assign a *diversity score* to the entire pool of questions generated for a single input text. This score reflects how much each question differs from the others in both form and content.

Technically, we estimate diversity by:

- Comparing each question with all others in the same pool.
- Calculating *text similarity* as the word overlap between questions.
- Calculating *context similarity* as the word overlap between the associated contexts.
- Computing the final diversity score as:

$$\text{Diversity} = 1.0 - \left(\frac{\text{avg\_text\_sim} + \text{avg\_context\_sim}}{2.0} \right)$$

- Higher scores (closer to 1.0) indicate more unique and diverse questions.

Ranking and Selection Process Overall, the final set of questions is selected using the following process:

1. **Sort all candidate questions** by:
   - *Answerability*: answerable questions first
   - *Diversity score*: higher score first
   - *Context relevance*: higher score first
2. **Ensure question variety** by organizing questions into a 3x3 matrix:
   - Three difficulty levels: *easy*, *medium*, *hard*
   - Three question types: *factual*, *inferential*, *analytical*

   We label each question using an additional classification step with an LLM that assigns difficulty and type labels based on custom prompts. We ensure that at least one question is selected from each matrix cell.
3. **Fill remaining slots** with top-ranked questions not yet selected, ensuring the overall number of questions per document stays within the desired limit.

Handling Large Documents For long documents that exceed input limits, we apply a chunking strategy:

- The document is split into overlapping chunks, typically at sentence or paragraph boundaries. When natural breaks are unavailable, we fall back on token-based limits with overlaps to ensure context continuity.
- We generate a set of candidate questions for each chunk individually and collect all candidates into a common pool.
- The full candidate pool from all chunks is combined and passed through the same ranking and selection pipeline described above.

This method ensures that all parts of a long document are represented, enlarges the initial question pool, and ultimately helps maintain a fixed-size, high-quality final question set.

3.1.3. Future Improvements of Generator Discriminator Pipeline

In the future, we would like to improve method at least in the following ways:

- Use semantic similarity measures (e.g., cosine similarity with Sentence-BERT embeddings) to enhance diversity beyond simple word overlap.
- Score questions on fluency, relevance, and difficulty using prompt engineering.
- Employ better models for question generation and evaluation.

3.2. Student

Our Student component is designed to generate concise answers based solely on the context retrieved from relevant documents.

Model Selection and Loading We dynamically load language models from Hugging Face's Transformers library, specifically GPT-Neo-125M, LLaMA-7B, and Mixtral-8x7B, using PyTorch. Models are loaded onto GPUs with half-precision floating points (float16) when CUDA is available, optimizing inference speed and reducing memory footprint.

Document Processing and Retrieval We preprocess input documents by splitting them into chunks of 100 words each with an overlap of 20 words to ensure continuity. This is accomplished using spaCy's sentence segmentation. Each chunk is then transformed into semantic embeddings via the `BAAI/bge-base-en-v1.5` SentenceTransformer model. These embeddings are normalized and indexed with FAISS's `IndexFlatIP` for fast inner product similarity search, retrieving the top 20 relevant document chunks per query.

Prompt Construction and Generation For each chunk collected by FAISS, we construct prompts that explicitly include instructions ("Answer the following QUESTION based only on the provided CONTEXT."), context segments retrieved by the FAISS-based search, and the specific question. Context size management dynamically calculates maximum allowable tokens by considering the model's maximum position embeddings minus reserved tokens (set at 256 tokens) and tokens used by instructional text. Responses are generated using GPU acceleration with PyTorch, sampling from a probability distribution at a temperature of 0.7 to maintain a balance between answer variability and consistency.

Advanced Generation Control Generation employs a custom stopping criterion that halts the generation process upon encountering a newline character, thus ensuring concise and coherent single-sentence responses. The system dynamically manages token budgets strictly to avoid overflow and efficiently uses available computational resources.

This detailed, structured approach ensures a robust and efficient Student component, significantly enhancing the overall effectiveness and reliability of our RAG system.

4. Results

4.1. Teacher Submission Results

In the Teacher system, we provided results generated using pure LLMs:

- Deepsek r1 DistilQwen 14B
- Deepsek r1 14B
- LLaMA 3.2 3B Instruct

Additionally, we applied the generator-discriminator approach. Our primary submission is based on the generator-discriminator pipeline run with the LLaMA 7B model.

Teacher Primary Submission Example Output Below is a sample of the generated questions and reference answers:


```
"nmtclass/lecture01-eval/screen80-slide80/text.en.txt": [
  {
    "question": "What units can be used instead of exact word forms for evaluating a
    ↪  fundamental problem?",
    "reference-answers": [
      "So to fix this fundamental problem we can evaluate coarser units, so not exact word
      ↪  forms, but lemmas or deep-lemmas."
    ]
  },
  {
    "question": "What is an alternative metric for calculating the similarity between
    ↪  machine translation outputs and human-generated texts?",
    "reference-answers": [
      "That's something which is equally simple to the BLEU score calculation and it should
      ↪  be popular because it is simpler and correlates better with humans."
    ]
  },
  {
    "question": "What can be used instead of longer sequences for evaluating machine
    ↪  translation outputs?",
    "reference-answers": [
      "We can focus on characters instead of words, so that's chrF3 or chrF3."
    ]
  },
  {
    "question": "How does using references created from MT outputs affect the evaluation of
    ↪  machine translation outputs?",
    "reference-answers": [
      "If you use more references alone that helps, but it is costly. If you use references
      ↪  which are created from the MT outputs, then you will indeed have a mismatch only
      ↪  when there was an error that the post editor had to fix."
    ]
  }
]
```


4.2. Student Results

In the Student system, we used LLaMA 7B and DeepSeek DistilQwen 1.5B, both running on messages augmented with RAG-retrieved results. DistilQwen, being a smaller model, tended to hallucinate less than LLaMA 7B and primarily focused on polishing the retrieved content. In contrast, LLaMA 7B occasionally introduced information not present in the retrieval context—for example, generating equations in the neural machine translation domain.

Our final submission is a combination of Qwen and LLaMA outputs. In cases where LLaMA failed (e.g., due to crashing or missing answers), we defaulted to the DistilQwen response. This combination was then cleaned using OpenAI: we instructed the model to polish the answers without adding new content. This post-processing step was particularly useful because we imposed a token generation limit of 300 tokens to avoid excessively long or off-topic answers and then answers were cut short. The post-processing helped to at least discard the sentence end that did not make any sense, or keep it there if it brings information.

We submitted the following runs:

- LLaMA 7B only
- DeepSeek DistilQwen 1.5B only
- LLaMA + DistilQwen combination
- **Primary submission**: LLaMA + DistilQwen cleaned by OpenAI

Student Primary Submission Example Output


```
{
  "question": "What is the purpose of each decoder in a transformer model?",
  "answer": "The purpose of each decoder in a transformer model is to reconsider and now in
  ↪  this second step deliberation, it is following the dependencies that humans found
  ↪  important because this upper part is made to mimic the syntactic structure of the
  ↪  sentence."
},
{
  "question": "Why are residual connections important in a deep network like a
  ↪  transformer?",
  "answer": "Residual connections allow the network to reconsider and deliberate on the
  ↪  dependencies that humans found important because this upper part is made to mimic the
  ↪  syntactic structure of the sentence."
},
{
  "question": "Why do different types of masks (homemade, surgical, FFP) vary in their
  ↪  impact on airflow and droplet containment?",
  "answer": "Different types of masks vary in their impact on airflow and droplet
  ↪  containment due to differences in their design, materials, and construction. Homemade
  ↪  masks, such as cloth masks, may not provide adequate filtration or protection against
  ↪  airborne pathogens, while surgical masks are designed to filter out larger droplets and
  ↪  may not provide complete protection against smaller droplets or aerosols."
},
{
  "question": "What are the differences in protective capabilities among FFP-1, FFP-2, and
  ↪  FFP-3 masks, and what applications are appropriate for each?",
  "answer": "The differences in protective capabilities among FFP-1, FFP-2, and FFP-3 masks
  ↪  are based on their respective filtration efficiencies against airborne particles. FFP-1
  ↪  masks offer the lowest level of protection, filtering out at least..."
}
```


5. Future Work and Improvements

While our current system successfully implements a Teacher-Student pipeline, several opportunities for enhancement remain. Notably, the Retrieval-Augmented Generation (RAG) paradigm—currently used only in the Student component—could also be incorporated into the Teacher module. By grounding question generation in retrieved contextual segments, RAG-enabled Teachers may produce more focused and relevant questions, particularly for long or complex source documents.

Another promising direction involves the implementation of an automated Evaluator. Although not completed in this iteration, we envision two evaluation strategies: (1) lightweight LLMs used as automatic graders to assess fluency, relevance, and factual accuracy of Student answers; and (2) a pairwise comparison framework, where multiple generated answers to the same question are evaluated head-to-head, with a scoring model selecting the superior response. These approaches could provide meaningful, scalable alternatives to manual annotation.

Future improvements could also explore:

- Enhanced answer evaluation using semantic similarity metrics, such as embedding-based comparison (e.g., cosine similarity using Sentence-BERT).
- Incorporation of explainable evaluation criteria (e.g., highlighting which context passages support an answer).
- Expansion to multilingual materials and models for broader applicability in international educational contexts.

Overall, the integration of retrieval in both generation and evaluation stages offers a compelling path toward more interpretable, accurate, and resource-efficient systems.

6. Conclusion

In this paper, we presented **LLMinds**, our submission for the ELOQUENT Sensemaking Task, which combines open-source large language models in a pipeline designed to generate and answer questions from instructional materials without relying on external knowledge sources. We successfully implemented two core components: the **Teacher**, responsible for generating diverse and context-relevant questions using a generator-discriminator framework; and the **Student**, which answers these questions based solely on RAG-retrieved content using local models and semantic search.

Our experiments demonstrated that smaller, locally deployed models like LLaMA 7B and DistilQwen 1.5B can be orchestrated to produce coherent and useful results with a minimal computational footprint. By refining outputs with OpenAI while preserving the original content, we struck a balance between fluency and factual grounding.

Although we did not implement the **Evaluator** module in this iteration, our framework is designed to support its future integration. We see great potential in extending this pipeline with automatic answer scoring and improved semantic evaluation. This work highlights the feasibility of privacy-respecting, cost-effective alternatives to cloud-based LLMs for educational and research tasks requiring controlled generation and comprehension.

Declaration on Generative AI

LLMs (as a particular type of generative AI tool) are the central element in this study (they create questions, answer them, and evaluate the answers). They were also used in the production of this text. In particular we used **ChatGPT** to rephrase and polish sentences. We also used **Overleaf's suggestions** tool to rephrase our research. After using this tool and service, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] P. Šindelář, O. Bojar, Overview of the Sensemaking Task at the ELOQUENT 2025 Lab: LLMs as Teachers, Students and Evaluators, in: G. Faggioli, N. Ferro, P. Rosso, D. Spina (Eds.), Working Notes of CLEF 2025 – Conference and Labs of the Evaluation Forum, CEUR-WS, 2025.

[2] J. Devlin, M. Chang, K. Lee, K. Toutanova, BERT: pre-training of deep bidirectional transformers for language understanding, in: J. Burstein, C. Doran, T. Solorio (Eds.), Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers), Association for Computational Linguistics, 2019, pp. 4171–4186. URL: https://doi.org/10.18653/v1/n19-1423. doi:10.18653/V1/N19-1423.

[3] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, Language models are unsupervised multitask learners, OpenAI Blog 1 (2019).

[4] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, A. Fan, V. Chaudhary, T. Rocktäschel, D. Kiela, et al., Retrieval-augmented generation for knowledge-intensive nlp tasks, in: Advances in Neural Information Processing Systems, volume 33, 2020, pp. 9459–9474.