ELOQUENT Sensemaking task: LLMs in the Evaluator role

Kateryna Lutsai[1], **Matyáš Thér**[2], **Jonáš Venc**[3], **and Ondřej Bojar**[4]

[1]**Charles University, Faculty of Mathematics and Physics, Institute of Formal and Applied Linguistics**
[2]**Charles University, Faculty of Science**
[3]**Charles University, Faculty of Mathematics and Physics, Institute of Formal and Applied Linguistics**
[4]**Charles University, Faculty of Mathematics and Physics, Institute of Formal and Applied Linguistics**

ABSTRACT

This paper describes our participation in the ELOQUENT Sensemaking Task (2025), focusing on the "Evaluator" role. The task challenges language models to prepare, take, or rate an exam based on provided learning materials. We detail our approach to developing an Evaluator system that scores answers given the materials, a question, and a candidate answer. This involved selecting appropriate large language models (LLMs), designing effective prompts, and conducting some first experiments to refine our methodology. Our work explores the capabilities of LLMs to constrain their knowledge to the given materials and assesses their reliability in understanding and evaluating textual information. We present the results of our experiments, including the performance of different models and prompting strategies, and discuss the challenges encountered, such as handling large contexts and the limitations of automated evaluation.

Keywords: question answering, llm, text understanding, prompt engineering

INTRODUCTION

The ELOQUENT Lab @ CLEF 2025 introduced the Sensemaking as its Task 5 in 2025. Sensemaking is designed to address the question: "Can your language model prep, sit, or rate an exam for you?". The core goal of the task is to automatically generate a quiz for a given learning material, and subsequently to answer the questions and evaluate those answers. The motivation stems from the observation that while LLMs store vast amounts of information and can easily answer questions, it is crucial to test if they can limit their knowledge to information provided in specific materials. The term "sensemaking" in this context refers to extracting the meaning of a text, represented as a set of questions and answers, and assessing how reliably LLMs can perform this extraction (generating a quiz), understand this sense (answering questions), and evaluate others' understanding (evaluating answers).

The task anticipates three main participation types:

- **Teacher** submissions: systems that generate quizzes from input materials.
- **Student** submissions: systems that provide answers to quiz questions based on input materials.
- **Evaluator** submissions: systems that score a given answer based on input materials, a question, and the answer itself.

The dependence of the submission types is summarized in Figure 1.

Our work focuses on the **Evaluator part of the task**. An Evaluator system is expected to accept a document, questions, and responses, and then return scores ranging from 0 (worst) to 100 (best) for each response. This involves assessing the quality of an answer specifically in response to the question, considering only the provided context material. The evaluation by the task organizers for Evaluator submissions involves comparing different systems and manually selecting those with the best results, using test dataset golden question-answer pairs and outputs from Teacher and Student submissions.

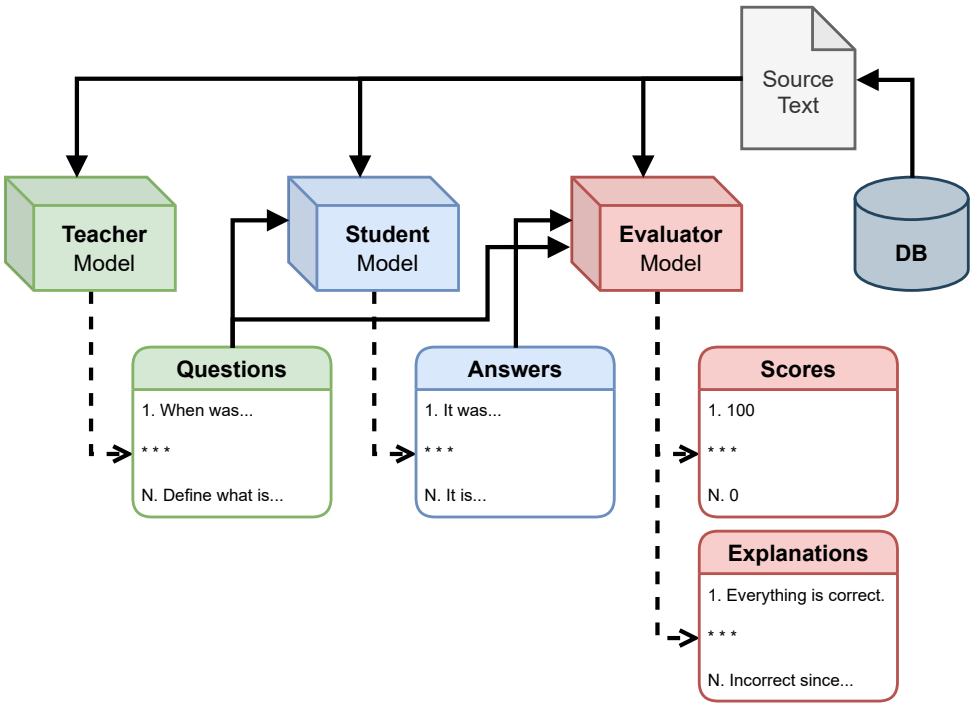

Figure 1. Sensemaking task described in a diagram form

METHODS AND MATERIALS

Input Data and Task Definition

The input data for the Evaluator system, as defined by the task, consists of three main components:

1. **Source Text (Context)**: plain English text, potentially large (up to 35k tokens in our experiments), derived from diverse sources such as books, presentations, and articles. The source texts provided by the task organizers span various domains, including university lectures, textbooks, and audio transcripts.

2. **Questions**: Provided by the original authors of the material or generated by "Teacher" systems based on the source text.

3. **Answers**: Generated by "Student" systems in response to the questions, using only the provided source text as a reference.

All questions and answers are expected to be in English. The Evaluator system's role is to assess the quality of each answer with respect to its corresponding question and the given context.

Output Format

The required output for the Evaluator system is a JSON object. For each input question-answer pair, the system must output a score and an explanation. Specifically, as detailed in the task documentation, the expected output is a JSON object containing:

- `score`: An integer between 0 and 100, representing the quality of the answer (100 being the best).

- `explanation`: A brief string justifying the assigned score.

For official task submissions, the output is a JSON dictionary where keys indicate the input file location, and values are lists of integer scores for each evaluated answer.

Model Selection and Experimental Setup

Our initial experiments used the **llama3.3** [1] model to generate synthetic training data for the Evaluator. We constructed a dataset from QA pairs, expanding it with incorrect answers (e.g., by shifting the answer to a previous or next item in the sequence) to create a 1:1 ratio of correct and incorrect responses. The llama3.3 model was prompted to evaluate these pairs, with the input context composed of the previous, current (relevant), and next items in the data sequence. The requested output should be a JSON object containing a score.

However, these preliminary experiments revealed significant limitations: the model often assigned zero scores to correct answers and sometimes produced malformed JSON outputs. These findings led us to reconsider our approach, moving away from custom dataset creation and older models, and instead focusing on prompt engineering with more advanced models.

Based on these insights, we selected **Gemma3 (27b)** [2] and **Qwen3 (30b)** [3] as our primary models. Both support a 128k token context window, making them suitable for handling large source texts. Gemma3 is a decoder-only Transformer with sliding window attention and a function-calling head for structured output, while Qwen3 is a mixture of experts (MoE) transformer featuring "Thinking/Not-thinking" modes. All models were run using `ollama` on a cluster equipped with NVIDIA A30 or RTX A4000 GPUs.

Prompt Engineering

Prompt design was a critical factor in achieving reliable and structured outputs. We adopted a two-part prompt structure:

Listing 1. System Prompt (passed as the `system` argument)

```
You are a fair teacher who grades students' answers. Evaluate the quality of
    the Answer specifically in response to the Question considering the
   Context provided. Format your entire response as a single JSON object
   containing 'score' (an integer between 0 and 100, where 100 is best) and
   'explanation' (a string briefly justifying the score).
```

Listing 2. User Prompt (passed as the `prompt` argument)

```
Question: <question>
Answer: <answer>
And given the following context: <text_fragments>
```

In the context of `ollama.generate` method, the **system prompt** Listing 1 sets the overall behavior and role of the model for the session, ensuring that responses are consistent and formatted as required. The **user prompt** Listing 2 provides the specific input for each evaluation instance, supplying the question, answer, and relevant context to be assessed.

The key elements we considered during prompt engineering included:

- Clearly defining the order of information (whether to present the task description or the inputs first).
- Explicitly stating the output format requirements (JSON with exactly two fields: `score` and `explanation`).
- Emphasizing knowledge boundaries (e.g., instructing the model to rely only on the provided context).

This careful prompt design was essential to ensure that the models produced outputs in the required format and focused their evaluation solely on the provided context.

RESULTS

We processed subsets of the development set using the selected models, **Gemma3** and **Qwen3**, to generate scores and explanations for each question-answer pair. The following examples illustrate the evaluation process and the models' ability to provide structured, context-aware feedback:

Listing 3. Input Example: Relevant Answer

```
Question: What is a result of misplacing punctuation marks in machine
    translation?
Answer: What is a result of misplacing punctuation marks in machine
    translation is not discussed in the provided text.
Context: Computation Graphs For our example neural network from Section ...
    book published in year 2020.
```

Listing 4. Model Output: Relevant Answer

```
{
  "score": 100,
  "explanation": "The given answer is correct, the text does not
     mention misplacing punctuation marks in machine translation."
}
```

Listing 5. Input Example: Unrelated Answer

```
Question: What was the structure of trade in the Roman Empire?
Answer: In hospitals, dehydration is commonly treated with infusions.
Context: CHAPTER OUTLINE 7.1 The Daily Life of a Roman Family 7.2 Slavery in
     the Roman Empire 7.3 The Roman Economy: Trade, Taxes, and Conquest ...
    Jewish population during the imperial period.
```

Listing 6. Model Output: Unrelated Answer

```
{
  "score": 10,
  "explanation": "The answer is entirely unrelated to the question.
     It is the full text of a chapter on the Roman Empire. There is
     no attempt to answer the question at all. Therefore the score
     is the lowest possible."
}
```

These examples illustrate the models' capability to follow the structured output format and provide reasonable scores and explanations based on the provided context and the task instructions. However, a key finding was the difficulty in selecting "the right" model when multiple models of similar quality are available, especially with large and detailed contexts, that is why simple eyeballing is not an option. Furthermore, fine-tuning models was deemed inefficient compared to prompt-engineering the newest models, particularly due to the lack of time and suitably styled training datasets.

Our early efforts to create a custom training dataset by generating synthetic incorrect answers—such as by shifting answers between unrelated QA pairs—highlighted further limitations. In particular, older models like llama3.3 often failed to differentiate correct from incorrect answers and frequently returned malformed or overly simplistic outputs. For that reason, we shifted our attention toward zero-shot prompting with more advanced models.

RELATED WORKS

Question Answering (QA) is a well-established area in Natural Language Processing. Traditional QA systems often rely on large-scale knowledge bases or web corpora to extract answers [4]. In contrast, context-based or reading comprehension QA tasks present models with a specific document and require them to answer questions using only the provided information [5]. These tasks require models to locate and synthesize information solely from the given document(s), which aligns closely with the ELOQUENT Sensemaking task's objective of testing whether LLMs can limit their knowledge to the provided materials. The Evaluator role, in particular, touches upon aspects of automated assessment and answer scoring, which has parallels in educational technology and peer review systems. This framing shares common ground with recent work in automatic answer grading and explanation-based assessment, such as in the domain of Automatic Short Answer Grading (ASAG) [6].

CONCLUSIONS

Participating in the ELOQUENT Sensemaking task, specifically in the Evaluator role, highlighted several challenges and insights. Manually evaluating the nuances of answers against extensive source texts is inherently difficult and time-consuming, underscoring the need for robust automated evaluation methods. However, creating such automated evaluators is also challenging, especially when aiming for human-like judgment.

A significant hurdle is the availability of suitable datasets for task-specific fine-tuning. While general-purpose LLMs are powerful, adapting them to the precise requirements of a specialized evaluation task without extensive, tailored training data relies heavily on prompt engineering. Our experiments showed that newer models with large context windows, combined with careful prompt design, can achieve promising results in scoring answers based on provided contexts. Nevertheless, the process of selecting the best model and refining prompts remains an empirical endeavor. The findings suggest that prompt-engineering with state-of-the-art models is currently a more pragmatic approach than fine-tuning for such specific, limited-duration tasks, especially when appropriately styled training data is scarce.

ACKNOWLEDGMENTS

Thanks to the Institute of Formal and Applied Linguistics ÚFAL) at Charles University, Faculty of Mathematics and Physics (MFF), for providing access to the HPC cluster with GPU nodes, which was essential for running the experiments with large language models. Finally, the grant number is XXX.

REFERENCES

[1] Meta AI. *Meta Llama 3.3 70B Instruction-Tuned Model*. 2024. URL: https://huggingface.co/meta-llama/Llama-3.3-70B-Instruct (visited on 05/26/2025).

[2] Gemma Team. *Gemma 3*. 2025. URL: https://goo.gle/Gemma3Report (visited on 05/26/2025).

[3] Qwen Team. *Qwen3 Technical Report*. 2025. URL: https://arxiv.org/abs/2505.09388 (visited on 05/26/2025).

[4] Danqi Chen et al. "Reading Wikipedia to Answer Open-Domain Questions". In: *ACL* (2017).

[5] Pranav Rajpurkar et al. "SQuAD: 100,000+ Questions for Machine Comprehension of Text". In: *EMNLP* (2016).

[6] Simon Burrows, Iryna Gurevych, and Benno Stein. "Automated Essay Scoring: A Survey of the State of the Art". In: *IEEE Transactions on Learning Technologies* 8.2 (2015), pp. 107–121.