

LLM Evaluation Pipeline for AI Tutor System

Comprehensive System Design & Process Documentation

Repository: [AmanPawar9/llm-eval-assignment](#)

November 13, 2025

Contents

| | | |
|----------|---------------------------------------|----------|
| 1 | Introduction | 3 |
| 2 | Dataset and Input Format | 3 |
| 3 | Evaluation Metrics and Rubrics | 3 |
| 4 | System Architecture | 4 |
| 5 | End-to-End Evaluation Flow | 4 |
| 6 | Configuration and Modularity | 5 |
| 7 | Design Decisions and Strengths | 5 |
| 8 | Running the System | 5 |
| 9 | Conclusion | 6 |

1 Introduction

The rapid adoption of AI in personalized education requires reliable methods for understanding the quality of explanations produced by Large Language Models (LLMs). A tutor model must go beyond correctness—it must teach effectively, at the right grade level, with clarity and accuracy.

This project implements a **modular, rubric-driven LLM Evaluation Pipeline** designed to:

- assess pedagogical quality using multiple evaluation metrics,
- leverage an LLM-as-a-judge framework,
- produce interpretability-rich reports,
- support multiple subjects and grade levels,
- maintain extensibility for future metrics and datasets.

2 Dataset and Input Format

The system assumes a structured JSON dataset containing test cases. Each case includes:

```
{
  "id": "test_001",
  "student_query": "Explain mitosis",
  "grade_level": "high_school",
  "subject": "biology",
  "expected_concepts": ["cell division", "chromosomes", "phases"],
  "ground_truth_answer": "optional reference answer",
  "type": "concept_explanation"
}
```

To ensure broad pedagogical coverage, the dataset spans:

- K–12 + college levels,
- STEM and humanities subjects,
- Several query types: conceptual, analytical, problem-solving.

3 Evaluation Metrics and Rubrics

Every generated tutor response is evaluated across **five core metrics**, each scored on a 1–5 scale:

1. Clarity

Is the explanation understandable and logically structured for the specified grade level?

2. Completeness

Does the response cover all relevant components of the query?

3. Accuracy

Are concepts, steps, and facts correct?

4. Appropriateness

Is vocabulary, tone, and complexity suited to the student's grade?

5. Long-term Retention (Memory Evaluation)

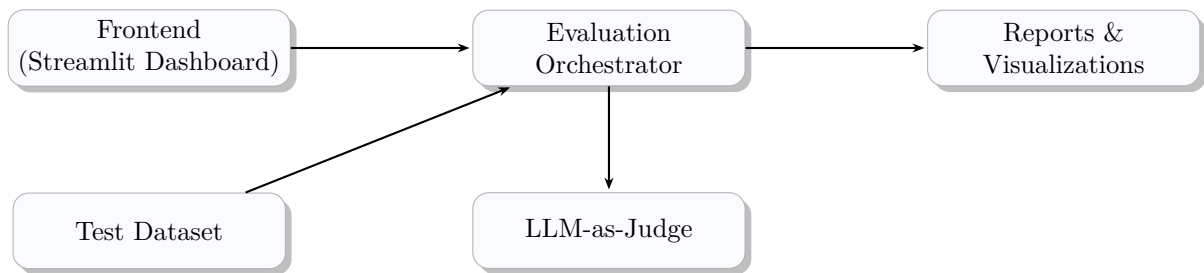
If a query depends on earlier context, is the model consistent?

Each metric has a **rubric-guided judge prompt** such as:

```
"Rate the Clarity of the explanation for a Grade 7 student.  
Return JSON: {score: <int>, justification: <short>}"
```

4 System Architecture

The evaluation system is divided into **frontend**, **backend/orchestrator**, **judge model**, and **report generation** modules.



Orchestrator: batching, prompt templates, retries, aggregation

Figure 1: Compact, A4-safe architecture diagram of the LLM Evaluation Pipeline.

5 End-to-End Evaluation Flow

The end-to-end pipeline executes the following steps:

1. Load the test dataset and validate its structure.
2. For each test case:
 - a) Send the query to the tutor model.
 - b) Generate its explanation/solution.
 - c) For each metric:
 - Construct a rubric-based judge prompt,
 - Query the judge model,
 - Parse JSON output,
 - Save numeric score + justification.

3. Aggregate all scores into dataset-level metrics.
4. Generate CSV/JSON reports and visual summaries.

6 Configuration and Modularity

The evaluation pipeline is controlled through a `config.yaml` file:

```
judge_model: ollama://mistral-7b
dataset_path: data/test_dataset.json
output_dir: outputs/
timeout_seconds: 30
```

The repository is structured with modular components:

| | |
|------------------------|--|
| <code>backend/</code> | Core evaluation logic, judge wrappers, orchestrator. |
| <code>frontend/</code> | Dashboard for running evaluations and viewing reports. |
| <code>data/</code> | Dataset files. |
| <code>infra/</code> | Dockerfile, CI, and environment configs. |
| <code>tests/</code> | Unit tests and integration checks. |

7 Design Decisions and Strengths

Key strengths of the pipeline include:

- **Pedagogy-aware evaluation:** grade-level adaptation built into prompts.
- **Rubric-structured scoring:** reduces LLM subjectivity.
- **Modular components:** easy to plug in new models or metrics.
- **Explainability:** judge provides numerical scores + rationales.
- **Reproducibility:** Docker environment and deterministic config.

8 Running the System

1. Create and activate a virtual environment

```
python -m venv .venv
source .venv/bin/activate
```

2. Install dependencies

```
pip install -r requirements.txt
```

3. Start evaluation

```
python backend/run_evaluation.py --config config.yaml
```

4. View results in `outputs/` or via frontend dashboard.

9 Conclusion

This evaluation system provides a robust and extensible foundation for measuring the pedagogical quality of LLM tutor responses. By combining structured datasets, detailed rubrics, and LLM-as-judge scoring, it enables fine-grained, interpretable assessment suited for real-world educational AI systems.