Open University of Catalonia (UOC) Master's Degree in Data Science

MASTER'S THESIS

Area: Natural Language Processing and Visual Analytics

Automated Correction of Short-Answer Exams Using Language Models

---

Author: Mounir Cheriha Dkik

Tutor: Blas Torregrosa Garcia

Professor: Josep Anton Mir Tutusaus

Barcelona, May 25, 2025

Copyright

FINAL PROJECT RECORD

Title of the project:	Automated Correction of Short-Answer Exams Using Language Models
Author's name:	Mounir Cheriha Dkik
Collaborating teacher's name:	Blas Torregrosa Garcia
PRA's name:	Josep Anton Mir Tutusaus
Delivery date (mm/yyyy):	05/2025
Degree or program:	Data Science
Final Project area:	Natural Language Processing and Visual Analytics
Language of the project:	English
Keywords	NLP, Large Language Models, Automated Grading

Acknowledgements

This project would not have been possible without the support of my beloved life partner *Mariia*, who's constant patience and encouragement were my fuel during countless late nights. And to my family, whose unconditional belief in me has been the foundation of every achievement.

My deepest gratitude to my academic tutor for their guidance and for their invaluable feedback that shaped this work.

Special thanks to my work colleagues for their generous collaboration in providing the essential datasets that made this research possible.

Abstract

This Master's thesis aims to develop and implement a system based on large language models (LLMs) for the automated grading of short-answer exams (Automated Short-Answer Grading, ASAG). The project arises from a pressing need in the educational sector, where the rise of massive open online courses and the increasing digitization of education have exponentially expanded the volume of exams to be graded. In many cases, manual grading has become unfeasible due to the high costs associated with human evaluators. Current natural language processing (NLP) technologies present an opportunity to address this challenge through efficient and scalable solutions.

The proposed system seeks to ensure accurate and objective assessment, reducing instructors' workload while providing students with immediate feedback on their responses. By leveraging various LLMs—including GPT-4o, GPT-4o-mini, Mistral Nemo, and Llama—this project evaluates the most suitable approaches for the task. Key parameters such as temperature tuning and a confidence threshold were tested to optimize performance. The results demonstrate that near-perfect accuracy can be achieved when the LLM evaluates only high-confidence responses, significantly improving reliability—though at the cost of low coverage. Among the tested models, GPT-4o delivered the highest accuracy, though GPT-4o-mini emerged as the most practical choice due to its competitive performance and cost-efficiency.

Keywords: Artificial Intelligence, Natural Language Processing, Language Models, Automated Grading, Education, Machine Learning.

Contents

List of Figures

List of Tables

1 Introduction

Recent breakthroughs in large language models (LLMs) such as GPT-4, Gemini Ultra, and Mistral have revolutionized natural language processing [9], enabling unprecedented accuracy in text understanding and generation. Traditional automated short-answer grading (ASAG) systems relied on keyword analysis, text similarity algorithms (e.g., TF-IDF or cosine similarity), or rule-based approaches. However, these methods suffered from significant limitations and poor performance, resulting in unreliable solutions that were difficult to deploy in real-world educational settings where precision and consistency are critical [3].

Modern language models overcome these limitations by capturing contextual and semantic nuances in student responses, enabling efficient and accurate automation of exam grading.[13].

While multiple-choice questions allow instant scoring, short-answer questions are methodologically superior: they promote critical thinking, knowledge articulation, and conceptual explanation—rather than mere selection of correct options. This reduces the role of chance in assessments and provides deeper insights into student comprehension. However, manual grading of such questions is time-consuming, costly, and prone to subjectivity.

Automating this process not only lowers operational costs but also ensures greater fairness by eliminating subjective grading variations and mitigating human biases. Furthermore, integrating LLMs enables immediate, detailed feedback—allowing students to understand their mistakes and improve their learning, beyond just receiving a score.

Despite these advantages, several key challenges must be addressed when implementing LLM-based grading systems. First, model hallucinations may lead to plausible but incorrect grading rationales, potentially undermining system reliability. Second, performance variability across different subject domains (e.g., STEM versus humanities) raises concerns about consistent application. Third, the opacity of LLM decision-making complicates the validation of grades, as educators and students may require transparent justifications to trust automated evaluations.

This thesis directly addresses these challenges through systematic experiments with state-of-the-art models (GPT-4, GPT-4-turbo, Mistral, and Llama 3). By investigating critical parameters such as confidence thresholds and temperature tuning, and exploring hybrid human-AI workflows, this research develops a systematic approach that optimizes assessment precision without compromising educational principles.

1.1 Context and motivation

This research direction originated from an industry proposal addressing the documented need to reduce labor-intensive grading costs through automated solutions. Over recent years, the field of natural language processing has experienced rapid growth and groundbreaking advancements, creating new opportunities in automated assessment. Recent research in automated short-answer grading has shown particularly promising results, further validating this approach.

Unlike other NLP tasks, ASAG systems cannot tolerate large margins of error. False positives—where the system accepts an incorrect answer—prove especially problematic as they undermine both system credibility and academic assessment validity. A real-world ASAG implementation requires exceptional precision, prioritizing reliability to avoid mistakes that could hinder institutional adoption. This technical challenge, combined with the potential impact, presents an opportunity to develop a practical yet rigorous tool that balances automation efficiency with academic accuracy.

Recent breakthroughs in large language models have demonstrated significant improvements in ASAG performance. These advancements enable more nuanced understanding of student responses while maintaining the precision required for educational assessment. The current project builds upon these developments to create a robust solution that addresses both technical and pedagogical requirements.

1.2 Goals

The primary objective of this project is to develop an LLM-based system for the automated grading of short-answer exam responses. However, during the development process, new needs or limitations may arise, requiring adjustments to the initial goals. Therefore, the planning will remain flexible and iterative, adapting to the results obtained at each project phase. The main objectives are as follows:

- **Create a dataset** suitable for training and/or evaluating ASAG models, ensuring data reliability and relevance.
- **Test various methods, models, and public frameworks** to identify the most promising approaches for ASAG.
- **Adapt an existing framework** or develop a new one tailored to the specific requirements of the project.
- **Implement a functional system** ideally capable of fully automated grading. If the accuracy is insufficient for complete autonomy, the system should assist human graders

by streamlining the evaluation process.

- **Incorporate automated feedback functionality** for students, providing detailed explanations to help them understand mistakes and improve their learning.

In addition to the primary goals, several secondary objectives have been defined to enhance the project:

- **Develop a comprehensive tool** that not only performs automated grading but also includes additional features to optimize the grader's workflow. This tool may include:
  - An intuitive graphical interface for efficient review, validation, and adjustment of automated corrections.
  - Clustering of similar answers to streamline grading consistency.
  - Detection of recurring error patterns for targeted feedback.
  - Generation of detailed reports for performance analysis.

The ultimate goal is to create a solution that not only automates grading but also enhances the grader's efficiency, accuracy, and productivity, thereby improving the overall quality of assessments.

1.3 Sustainability, diversity, and ethical/social challenges

1.3.1 Sustainability

The project's primary sustainability impact stems from computational resource requirements during model training and inference. While LLMs consume significant energy during initial training, our system utilizes existing pre-trained models (GPT-4o, Mistral, etc.) through API calls, minimizing direct ecological footprint. The operational phase may reduce paper usage in educational institutions by enabling digital assessment workflows. However, the energy consumption of continuous model inference requires monitoring, particularly in large-scale deployments.

1.3.2 Ethical Behavior and Social Responsibility

The deployment of automated grading systems necessitates careful consideration of several ethical dimensions. At its core, this approach prioritizes student data protection through strict confidentiality measures. While general dataset characteristics (such as subject domains or response structures) may be discussed, the system ensures no disclosure of individual student responses or personally identifiable information occurs.

A fundamental principle of the system design maintains human oversight as indispensable. The architecture preserves professors' ultimate grading authority, with the AI functioning strictly in an assistive capacity. This human-in-the-loop approach serves multiple purposes: preserving professional judgment in assessment, providing crucial quality control, and ensuring students benefit from educators' expertise. The system is intentionally designed as a tool to enhance rather than replace human grading functions.

1.4 Approach and methodology

The development of this project will follow an iterative research and experimentation cycle. Our approach breaks down into several key phases:

1. **Problem analysis and state-of-the-art review**
    - Literature review of scientific publications on ASAG and existing methods.
    - Identification of current system limitations and most promising approaches.
    - Analysis of requirements for adapting an ASAG system to real educational contexts.
2. **Dataset construction and preparation**
    - Collection of short-answer exam data from open sources or through collaborations.
    - Data normalization, cleaning, and labeling to ensure quality.
    - Creation of a validation set to evaluate model performance.
3. **Model exploration and implementation**
    - Testing of different LLM architectures (GPT-4, Mistral, Llama).
    - Systematically testing and optimizing model prompts and parameters to identify the most effective configuration for our specific evaluation context.

4. **System evaluation and validation**
    - Application of the model to real exams and comparison with human grading.
    - Performance measurement using metrics such as precision, accuracy, and correlation with human scores.
    - Model adjustment and refinement to improve system reliability.
5. **Functional prototype development**
    - Creation of an interface to test the system in a practical environment.
    - Exploration of integration feasibility with online learning platforms.

This methodology emphasizes adaptability and continuous improvement. By iteratively testing and refining our models and system components, we aim to build a robust ASAG solution that can be realistically deployed in educational settings. Each phase informs the next, creating a feedback loop between design, implementation, and evaluation. The following sections detail the experiments conducted and the insights obtained from applying this methodology.

1.5 Schedule

The project has been structured into seven main phases, each with specific objectives and estimated durations. This timeline spans a total of three months, with flexibility to adjust durations or add/remove steps based on results and time availability. The planning remains adaptable to accommodate unforeseen challenges or necessary pivots during development. Below are the detailed phases and their key characteristics:

Notes:

- The timeline accounts for potential overlaps between phases (e.g., documentation may begin during prototype development)
- Critical paths: Phases 3 (Model Experimentation) and 6 (Prototype Development) require special attention

1.6 Summary of the outputs of the project

The main objective of this project is to determine the highest achievable accuracy of the model using the chosen methodology and the dataset provided by the collaborating company. Once the model's performance has been evaluated and validated, the next step involves developing

Phase	Description	Estimated Duration
1. Literature Review	Comprehensive research on ASAG systems, LLMs, and automated grading techniques.	1 week
2. Dataset Construction	Data collection, cleaning, and labeling. Creation of validation sets.	1 week
3. Model Experimentation	Evaluation of pre-trained models and selection of optimal architecture.	2 weeks
4. Model Development	Optimization of selected model for maximum accuracy.	1 week
5. System Validation	Real-world testing, performance benchmarking against human graders.	1 week
6. Prototype Development	Implementation of basic interface for practical testing.	3 weeks
7. Documentation	Final report writing and presentation preparation.	2 weeks

Table 1: Project timeline and phases

a graphical user interface (GUI). This interface will integrate the optimized model along with additional tools to support the automated grading of exams.

Detailed descriptions of each output, including the model's development, and GUI functionalities, will be presented in the subsequent chapters of this document.

1.7 Brief description of the remaining chapters of the report

The report begins by examining the current state of ASAG systems, reviewing established methodologies and recent advancements in the field. A dedicated section then explores the dataset provided by the industry partner, offering analysis of its composition and preprocessing requirements.

The Methods and Resources chapter is divided into two parts. The Resources subsection outlines the computational setup and the application of quantization to run open-weight models efficiently. It also explains the inclusion of GPT-4 and GPT-4o Mini via API for comparison. The Methodology section presents the evaluation strategy, including the RAG-based prompt design, confidence scoring, and multi-temperature inference to assess model robustness and reliability.

The Results chapter evaluates the performance of the different language models tested. It explores their ability to follow formatting instructions, the influence of temperature settings on output consistency, and how confidence thresholds affect accuracy and coverage. The chapter also reflects on practical challenges and outlines a framework aimed at improving system reliability and adaptability in real-world educational settings.

The development of our graphical interface receives thorough documentation in a later section. Special attention is given to user experience considerations and the integration of supplemental tools designed to assist educators in the grading process.

Concluding sections synthesize the project's contributions to ASAG research while proposing meaningful directions for future work.

2 State of the art

2.1 Classical Feature-Based Approaches

Early approaches to automated short answer grading relied on manually defined lexical, syntactic, and semantic features extracted from text responses. These techniques employed dependency analyses, constituent parsers, or phrase overlap functions to identify relevant patterns in student answers. The extracted features were then processed by traditional machine learning classifiers such as logistic regression [4], support vector machines [10], random forests [15], or Naïve Bayes [8]. Some approaches also combined multiple classifiers through ensemble methods. While these models demonstrated effectiveness across various datasets, the variability in feature selection made it difficult to establish definitive conclusions about the superiority of any particular representation scheme.

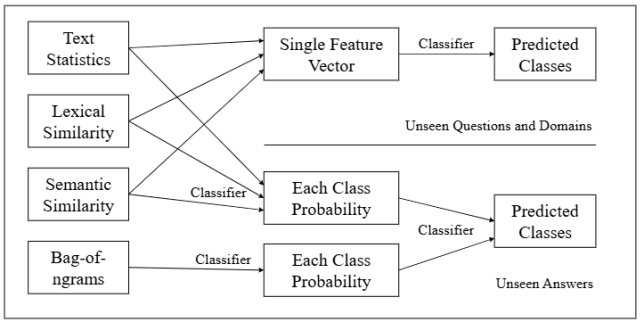

Figure 1: Model proposed by [2]

A representative model of this approach was proposed in [2], which utilized Random Forest and Extreme Gradient Boosting, achieving an accuracy of **0.775** in binary classification (correct/incorrect) on the SciEntsBank dataset and **0.81** on the Beetle dataset, also in binary classification.

2.2 Deep Learning-Based Approaches

With advances in deep learning, traditional feature engineering methods have gradually been superseded by neural models capable of automatically learning text representations. Rather than relying on manually crafted lexical, syntactic, or semantic features, these approaches employ deep neural networks to capture complex textual relationships and improve short answer grading performance.

Early applications of deep learning to ASAG utilized word embedding techniques like Word2Vec and GloVe to represent text in low-dimensional continuous spaces. While demonstrating improved semantic similarity capture at the word level, these methods showed limitations in representing complete sentence meaning. To address this, hybrid approaches emerged that combined neural embeddings with handcrafted features to enrich text representations.

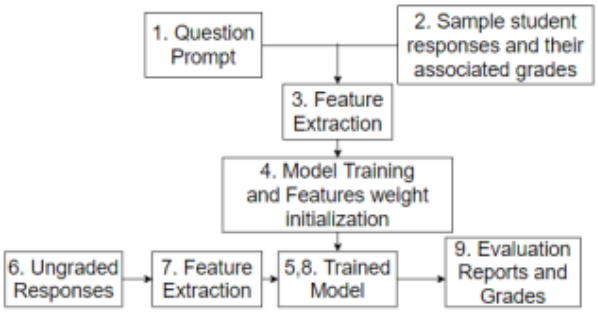

Figure 2: Model combining word embeddings with handcrafted features by [7]

Prior research has demonstrated that hybrid systems combining word embeddings with handcrafted features consistently outperform pure embedding-based approaches in ASAG tasks. Study [7] revealed that integrating traditional techniques (POS tagging and n-gram analysis) with distributed representations (Word2Vec and Doc2Vec) significantly improves answer quality assessment, achieving state-of-the-art performance on the ASAP-SAS benchmark. These findings confirm that while embeddings effectively capture semantic properties, manually engineered features remain indispensable for representing complex syntactic and lexical patterns.

2.3 Sequential Models and Transfer Learning

Sequential models have demonstrated significant potential in ASAG by enhancing semantic text representation and improving system robustness. A notable example is the Siamese BiLSTM architecture proposed by [1], which computes the earth-mover distance between vector representations of reference and student answers. This approach established that sequential models can effectively extract relevant semantic features to enhance classification performance.

Transfer learning has emerged as a powerful paradigm for adapting large pre-trained language models to specific ASAG tasks. The work in [6] achieved state-of-the-art results by employing Universal Sentence Representations trained on the Stanford Natural Language Inference corpus to generate sentence embeddings. Domain adaptation techniques have further advanced the field, as demonstrated by [11], where a BiLSTM model successfully combined general knowledge from Wikipedia with domain-specific question understanding.

Model ensembles have shown particular promise for improving generalization capabilities. The study in [12] integrated eight regression models through a perceptron network, incorporating answer summaries during training to enhance representation quality. While these techniques substantially improve grading accuracy, they simultaneously introduce significant computational complexity that may limit practical deployment.

2.4 Attention-Based Models and Transformers

Recent advancements in ASAG have leveraged sophisticated attention mechanisms and transformer architectures to better capture semantic and structural text features. Attention-based models excel at computing relational weights between words within sentences, effectively modeling long-range dependencies without explicit sequential processing. Transformers, with their encoder-decoder architecture, have proven particularly effective for characterizing long-term dependencies in sequential data through parallelized attention components that learn diverse linguistic relationships.

Hybrid architectures combining LSTM and CNN layers with attention mechanisms have achieved notable results, where careful selection of input embeddings and fine-tuning of pre-trained models emerge as critical success factors. Transformer models like BERT have demonstrated significant performance gains in ASAG tasks through transfer learning and fine-tuning techniques. However, cross-domain generalization remains a persistent challenge. Recent studies indicate that incorporating unstructured domain data and question-answer pairs during fine-tuning can outperform approaches relying solely on task-specific datasets.

Attention mechanisms have revolutionized answer representation in ASAG systems by:

- Capturing long-range contextual relationships
- Identifying key concepts within student responses
- Enabling nuanced semantic understanding

While these approaches achieve state-of-the-art performance (e.g., [17] reports a Pearson correlation of 0.82 on the Mohler dataset), they present substantial practical limitations:

1. **Computational Cost:** The fine-tuning process requires intensive, dataset-specific computation
2. **Generalization Challenges:** Performance degrades significantly when applied to unseen question types or domains
3. **Scalability Constraints:** Real-world implementation proves challenging given the inherent variability of educational contexts

These constraints highlight the tension between theoretical performance and practical deployment, suggesting that while attention-based models represent the current frontier in ASAG accuracy, their adoption in authentic educational settings requires careful consideration of computational resources and domain adaptation strategies.

2.5 LLM-Based Models

Large Language Models (LLMs), such as GPT, have revolutionized the field of ASAG thanks to their ability to comprehend and generate text with an unprecedented level of sophistication. Unlike traditional sequential models, which required task-specific architectures and fine-tuning for each dataset, LLMs can leverage the knowledge acquired during their massive training on diverse text corpora. This allows them to generalize more effectively across different types of questions and answers, reducing the reliance on large task-specific datasets.

A straightforward approach to implementing ASAG with LLMs involves defining a prompt that specifies the task, inputting the answers into the system, and processing the model's output, which may include a numerical score, qualitative feedback, or a justification for the assigned grade. This method enables rapid deployment without the need for additional fine-tuning, taking advantage of the zero-shot or few-shot capabilities of LLMs.

Nevertheless, early studies on LLMs concluded that general-purpose models such as GPT-4 do not outperform specialized deep learning ASAG models, though they match or surpass the performance of traditional rule-based systems [5]. Their main advantage lies in their ease of use:

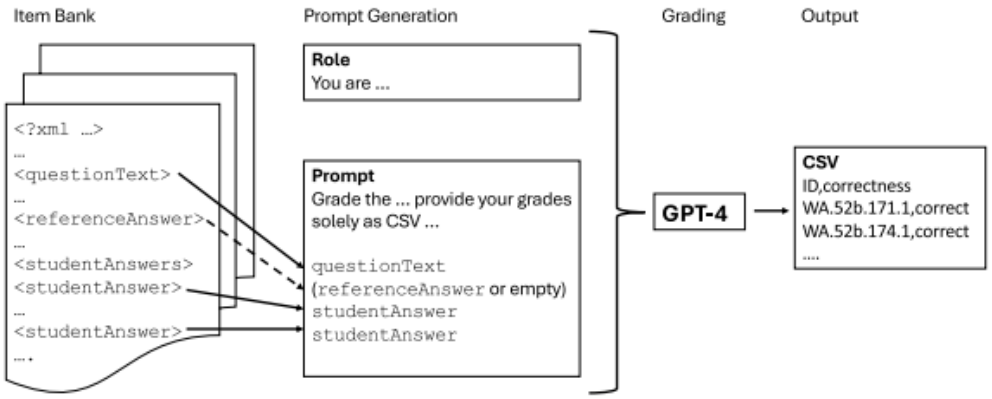

Figure 3: LLM-based model proposed by [5]

they require no task-specific training or complex adjustments, and instead work out-of-the-box, offering reliable performance.

One of the main challenges in using LLMs for ASAG is the phenomenon of hallucinations, where models generate inaccurate or inconsistent responses that do not align with evaluation criteria. This issue is particularly critical when the model lacks domain-specific knowledge, potentially leading to incorrect grading or misleading feedback. To address this, recent strategies such as few-shot learning and Retrieval-Augmented Generation (RAG) have been explored. Few-shot learning provides the model with previously evaluated responses to guide its grading, while RAG enhances contextual accuracy by integrating relevant information from external sources. These techniques have been shown to significantly reduce hallucinations and improve model reliability, achieving accuracy levels comparable to those of human assessors [18].

This work aims to implement and extend the findings of study [18], leveraging its main insights to develop an automatic assessment system tailored to our context. The key conclusions that will guide our implementation are:

- The GPT-4o model stands out for offering an optimal balance between accuracy and cost. However, our study will focus on exploring and prioritizing the use of free LLMs as a viable alternative.
- Including teacher-assessed examples in prompts significantly improves grading accuracy.
- Selecting examples using RAG proves to be superior to random selection, as it ensures that the provided examples are relevant to each specific answer.
- Including structured rubrics in prompts enhances grading accuracy and increases transparency in the assessment process.

3 Dataset

The analysis utilizes a dataset comprising 10 distinct exam questions in Spanish, each with binary-labeled student responses: 1 for correct answers and 0 for incorrect ones.

Question	Domain	Total	Corr.	Incorr.	Avg. words
Q1	STEM	527	140	387	13.98
Q2	STEM	562	99	363	15.14
Q3	Literature	548	314	234	19.65
Q4	Oral comprehension	504	288	216	9.60
Q5	Oral comprehension	531	202	329	16.04
Q6	Oral comprehension	518	288	230	7.36
Q7	Humanities	521	332	189	15.65
Q8	Literature	780	554	226	14.45
Q9	Reading Comprehension	502	193	309	11.87
Q10	Literature	758	397	361	14.67
Total	–	5651	2807	2844	13.84

Table 2: Descriptive statistics per question in the dataset

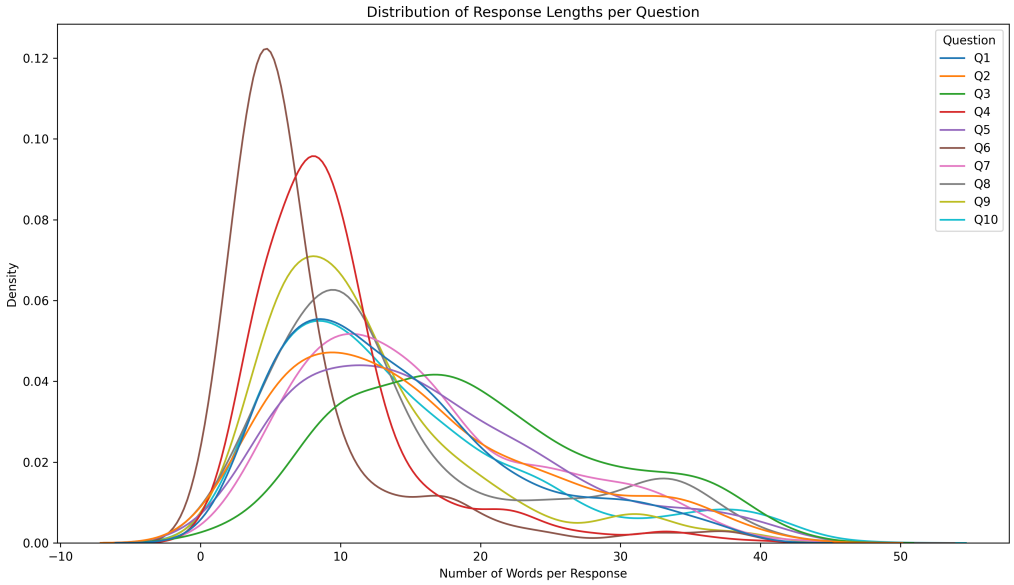

Figure 4: Number of words distribution per question

Given the size of the original dataset, a smaller balanced evaluation subset was constructed to enable feasible experimentation under limited computational resources. The sampling strategy selected 50 correct and 50 incorrect answers per question, resulting in a total of 1,000 responses for testing purposes.

To construct a representative and balanced subset of responses for each question,50 correct and 50 incorrect answers will be selected based on their length. Although the empirical distribution

of response lengths exhibits a right-skewed shape—commonly observed in natural language data—it will be approximated using a Gaussian (normal) distribution for sampling purposes. This choice was made to simplify the selection process and ensure a central concentration of samples around the mean, effectively capturing the most typical responses while preserving variability.

The use of a normal distribution prevented the overrepresentation of extremely short or overly long outliers, select responses that reflect the core linguistic patterns in the dataset and maintain consistency and comparability across different questions.

While this approximation does not fully model the actual length distribution (which is likely closer to a log-normal or gamma distribution), it provided a practical and effective method for subset construction without significantly biasing the evaluation.

Nevertheless, once the best-performing model and configuration are identified, they will be applied to the entire dataset for comprehensive evaluation.

4 Methods and resources

4.1 Resources

The experimental framework utilizes an NVIDIA RTX 3060 GPU with 12GB of VRAM, representing typical mid-range hardware available to many researchers and practitioners. This hardware configuration imposes specific constraints on model selection, particularly regarding memory requirements and computational throughput. Modern large language models often demand substantial resources, with even moderately-sized architectures requiring careful optimization to run efficiently on such hardware. For context, a 7B parameter model in FP16 (16-bit floating-point) needs approximately 17GB of VRAM., which exceeds the used GPU's capacity without optimization techniques.

Table 3: Comparison of LLM Characteristics

Feature	**Llama-3.1-8B**	**Mistral-Nemo**	**GPT-4o**	**GPT-4o-mini**
Parameters	8B	12B	1.8T (est.)	500B (est.)
Context Window	8k tokens	32k tokens	128k tokens	64k tokens
Open Weight	Yes	Partial	No	No
Specialty	Cost-efficient	Long-context tasks	General intelligence	Lightweight

In this study, **Llama-3.1-8B-Instruct** and **Mistral-Nemo-Instruct-2407** will be evaluated, two of the most capable open-weight LLMs that balance performance with practical deployability. These models have demonstrated competitive results on standard benchmarks while

remaining feasible for resource-constrained environments. To overcome the hardware limitations, 8-bit quantization will be implemented through the `bitsandbytes` library, which reduces the memory footprint by converting model weights from 32-bit floating point numbers to 8-bit integers. This quantization process involves carefully scaling and rounding the weights while maintaining their relative distributions, typically preserving about 90-95% of the original model's accuracy. The quantized versions require approximately 7GB of memory, fitting comfortably within the GPU's 12GB VRAM while leaving room for activations and processing buffers.

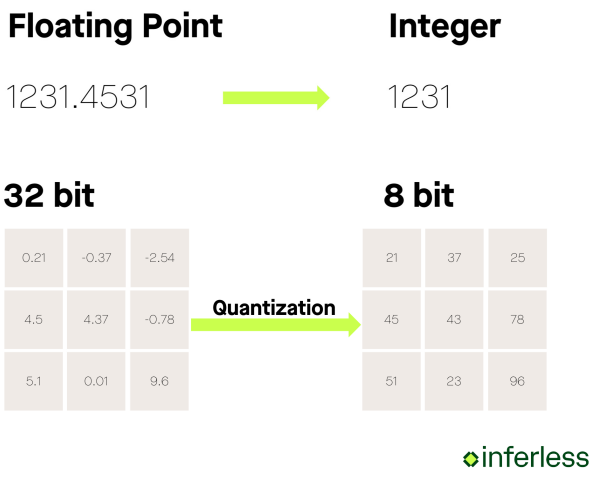

Figure 5: Visual representation of LLM quantization. Source: [16]

While Llama 3.1 8B maintains good performance with 8-bit quantization (approximately 8GB VRAM usage), we employ 4-bit quantization for Mistral-Nemo due to its larger parameter count and enhanced capabilities.

Beyond these open-weight models, the field offers numerous commercial alternatives that could potentially enhance results given sufficient resources. Models like OpenAI's **GPT-4** and **GPT-4o Mini** represent the current state-of-the-art in large language models, offering robust reasoning capabilities and versatility across a wide range of tasks. GPT-4, with its billions of parameters, is renowned for its superior ability to understand complex instructions, generate coherent and contextually relevant responses, and maintain a high level of accuracy even with longer context windows. On the other hand, GPT-4o Mini, a more lightweight version, delivers impressive performance while optimizing for speed and resource efficiency. It retains much of GPT-4's sophistication but at a reduced cost and computational demand. These proprietary models, however, come with higher operational expenses, which may be prohibitive for some setups. In the experiments, both models will also be tested via the OpenAI API, allowing us to evaluate their performance in a cloud-based environment, providing flexibility in terms of

computational resources while maintaining access to these advanced models.

The current hardware-constrained setup serves as an important baseline demonstrating what can be achieved with limited resources. However, the developed methodology is fundamentally scalable. With access to more powerful GPUs or budget, one could deploy larger model variants with potentially improved performance, enhancing both accuracy and overall output quality.

4.2 Methodology

The evaluation process employs the following prompt template for test dataset responses:

```
Eres un profesor que está calificando respuestas de estudiantes en un examen.
Se te proporcionará criterios de evaluación y ejemplos de respuestas evaluadas.
Debes devolver la evaluación en formato JSON con estos tres campos:
- "grade": 1 si la respuesta es correcta, 0 si es incorrecta
- "feedback": una explicación breve del motivo de la calificación
- "confidence": un valor entre 0 y 100 que indique tu nivel de confianza
en la evaluación

Enunciado de la pregunta:
{question}

Criterio de evaluación:
{reference_answer}

Ejemplos de respuestas correctas (grade = 1):
{chr(10).join(examples_correct[:4])}

Ejemplos de respuestas incorrectas (grade = 0):
{chr(10).join(examples_incorrect[:4])}

Respuesta del estudiante a evaluar:
{student_answer}

Devuelve únicamente un texto con el **objeto JSON** con la evaluación según lo indicado.
```

Figure 6: Prompt structure used for Llama and Mistral models

```
system_prompt = """Eres un profesor que está calificando respuestas de estudiantes
en un examen de respuestas abiertas cortas. Se te proporcionará criterios de
evaluación y ejemplos de respuestas evaluadas.
Devuelve la evaluación del alumno en formato JSON con estos tres campos:
- "grade": entero entre 0 y 1
- "feedback": breve explicación
- "confidence": valor entre 0 y 100
Devuelve solo el objeto JSON.
"""
user_prompt = f"""
Enunciado de la pregunta:
{question}

Criterio de evaluación:
{reference_answer}

Ejemplos de respuestas correctas (grade = 1):
{chr(10).join(examples_correct[:4])}

Ejemplos de respuestas incorrectas (grade = 0):
{chr(10).join(examples_incorrect[:4])}

Respuesta del estudiante a evaluar:
{student_answer}
"""
prompt = [
{"role": "system", "content": system_prompt},
{"role": "user", "content": user_prompt}
]
```

Figure 7: Prompt structure used for GPT-4o and GPT-4o-mini

Where:

- **question** represents the question prompt
- **reference_answer** contains the grading criteria
- **chr(10).join(examples_correct[:4])** provides a maximum of four sample correct responses
- **chr(10).join(examples_incorrect[:4])** provides a maximum of four sample incorrect responses
- **student_answer** contains the student response being evaluated

The approach leverages Retrieval-Augmented Generation to dynamically provide the model with semantically similar examples of both correct and incorrect answers from the pre-scored dataset. This method, validated by prior research [18], enhances model performance by grounding evaluations in contextually relevant benchmarks. A pretrained sentence transformer (all-MiniLM-L6-v2) projects all reference answers into a 384-dimensional semantic space. These embeddings are indexed using FAISS (Facebook AI Similarity Search), a highly optimized library for efficient similarity search in high-dimensional spaces.

Additionally, the LLM will be required to output a confidence parameter alongside its score and feedback. This metric quantifies the model's certainty in its judgment, enabling two critical analyses:

- **Precision calibration:** Measuring how accuracy varies with confidence thresholds.
- **Human-in-the-loop filtering:** Automatically flagging low-confidence responses for manual review, thereby improving overall reliability.

The confidence parameter generation will be omitted if empirical results demonstrate no significant improvement in scoring accuracy. This approach intentionally avoids automated scoring of ambiguous responses that exceed the LLM's reliable evaluation capabilities, thereby prioritizing assessment quality over full automation coverage.

For each evaluated model, student responses are input along with the constructed prompt. To support the RAG mechanism, 10 correct and 10 incorrect reference answers from the dataset will be retrieved for each question. However, the prompt only displays the 4 most semantically similar examples for each category (correct/incorrect) to the target student response being evaluated. This similarity is quantified using L2 distance between embeddings produced by 'all-MiniLM-L6-v2'.

In total, the experiment evaluates 800 student responses, evenly distributed across 10 distinct exam questions.

For each model, a comprehensive testing will be conducted using three distinct temperature parameters (0, 0.3 and 1.0). The temperature parameter controls the randomness of output generation through the softmax probability distribution:

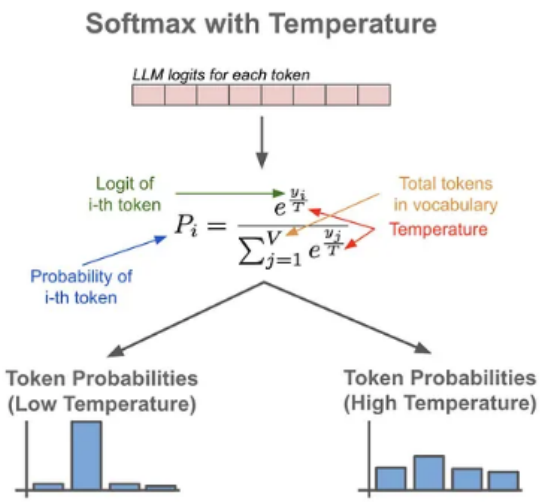

Figure 8: Effect of temperature parameter on LLM output diversity and predictability. Source: [14]

- **Zero temperature (0.0)**: Produces completely deterministic outputs by always selecting the highest-probability token. This setting provides:
  - Maximum reproducibility in grading
  - Baseline performance metrics
  - Essential for high-stakes assessments requiring absolute consistency
- **Low temperature (0.3)**: Produces more deterministic and focused outputs by sharpening the probability distribution, favoring high-confidence tokens. This setting could be optimal for:
  - Grading tasks requiring consistency
  - Factual response evaluation
  - Minimizing creative variations
- **High temperature (1.0)**: Generates more diverse and exploratory outputs by flattening the probability distribution, allowing lower-probability tokens to be selected. This setting is valuable for:
  - Capturing alternative correct expressions
  - Handling creative or unconventional answer phrasings
  - Evaluating model robustness to linguistic variation

This multi-temperature approach will be employed to rigorously assess each model's capability across the spectrum from absolute consistency to flexible evaluation. The temperature analysis is complemented by the confidence threshold mechanism, creating a comprehensive evaluation framework that addresses both response quality and scoring reliability.

Once the model's outputs—grade, feedback, and confidence—are obtained for each response, the model's precision will be computed using various confidence thresholds to assess performance robustness.

While Section 5.5 addresses methodological limitations, it should be noted that all assessments employ: (a) the instructors' original evaluation rubrics without modification, and (b) identical prompts for experimental consistency.

5 Results

5.1 Model Reliability and Response Parsing

The evaluation begins by examining the models' ability to provide structured, parsable responses. When the LLM fails to output valid JSON (as specified in our methodology), we discard the assessment. These cases typically occur when the model deviates from instructions or produces unexpected errors.

Table 4: Percentage of Unrated Responses by Model and Temperature

Model	**T=0 (%)**	**T=0.3 (%)**	**T=1 (%)**
GPT-4o	1.39	1.75	2.76
GPT-4o-mini	2.26	2.89	3.14
Mistral	4.52	0.13	7.15
Llama 3	11.26	20.75	52.95

As shown in Table 4, significant variability exists in model compliance with the JSON output requirement. Llama 3 demonstrates the highest inconsistency, particularly at higher temperatures (reaching 52.95% unrated responses at T=1), often misunderstanding the instruction as a request for code generation rather than structured assessment output.

The GPT-4o family shows superior instruction adherence, with GPT-4o maintaining the lowest overall unrated percentages (1.39-2.76% across temperatures). While Mistral achieves an exceptional 0.13% unrated rate at T=0.3, this isolated performance requires further validation against accuracy metrics in subsequent analyses. The temperature effect follows expected patterns, with higher temperatures generally increasing non-compliance rates, though this relationship becomes extreme in Llama 3's case.

These initial results position GPT-4o as the most reliable model for structured output generation, though comprehensive evaluation must consider both parsing reliability and grading accuracy in subsequent sections.

5.2 Determining the Optimal Temperature Parameter

Through systematic evaluation of all models (GPT-4o, GPT-4o-mini, Mistral, and Llama 3) across different temperature settings, we identify the optimal balance between assessment reliability and practical applicability. Our evaluation specifically tracks two critical error types: **false positives** (incorrectly marking wrong answers as correct) and **false negatives** (incorrectly marking right answers as wrong). While both errors impact grading quality, false positives

present greater risks in practice: students receiving undeserved credit are unlikely to challenge the result, potentially compromising assessment validity. False negatives, while undesirable, are more likely to be detected through student appeals, allowing for manual correction.

Table 5: Aggregate Model Performance by Temperature (Average across all models)

Temp.	Accuracy	FP Rate	FN Rate
0.0	0.804	46.75	101.5
0.3	0.800	43.00	105.25
1.0	0.770	50.00	94.00

After evaluation, we establish **temperature 0.0** as the optimal baseline for automated grading systems. This configuration delivers maximum assessment reliability with 80.4% aggregate accuracy - a 0.4% improvement over T=0.3. More importantly, T=0 eliminates the unpredictable variations in grading standards observed at higher temperatures, providing three key operational advantages: (1) simplified model calibration through deterministic behavior, (2) consistent performance across diverse question types, and (3) reduced need for post-hoc manual corrections that undermine system efficiency.

While T=0.3 shows a marginally lower false positive rate, this flexibility comes at the cost of greater variance in grading standards - particularly for smaller models like Mistral and Llama 3. Our experiments demonstrate that T=0's inherent stability actually enables more effective confidence thresholding, as detailed in Section 5.3. The deterministic outputs at T=0 allow cleaner threshold application, yielding:

- Better accuracy preservation along confidence thresholds
- More predictable coverage degradation curves
- Lower variance in grading standards across assessment batches

We systematically excluded temperature 1.0 from consideration due to its significantly lower accuracy (77.0% vs 80.4% at T=0) and elevated error rates. This investigation was motivated by the hypothesis that increased creativity at higher temperatures might improve evaluation of diverse answer phrasings. However, our empirical results conclusively demonstrate that the added variability degrades rather than enhances grading quality, confirming the theoretical expectation that deterministic evaluation (T=0) provides optimal reliability for automated assessment systems.

5.3 Accuracy Optimization Through Confidence Thresholding

We now analyze model accuracy under varying confidence thresholds at our selected temperature (T=0). This approach excludes uncertain evaluations (confidence < threshold) from accuracy calculations, revealing the fundamental precision-coverage trade-off inherent in automated grading systems. Note that coverage percentages exclude previously discarded responses (where the LLM failed to produce valid evaluations), as established in Section 5.1.

Table 6: Model Performance Across Confidence Thresholds for Temperature = 0

Model	Thr.	Acc.	Cov.	TP	FP	FN	TN
GPT-4o	100	1	2.94%	7	0	0	16
	95	0.893	75.10%	216	6	57	309
	90	0.849	94.5%	259	12	100	369
	0	0.825	100%	263	14	123	383
GPT-4o-mini	95	1	7.45%	37	0	0	21
	90	0.938	41.07%	219	17	3	81
	80	0.821	96.15%	281	38	96	334
	0	0.816	100%	281	38	105	355
Mistral	100	0.877	45.86%	161	14	29	145
	90	0.796	99.61%	258	39	116	345
	0	0.794	100%	259	41	116	345
Llama	95	1	5.37%	20	0	0	18
	85	0.940	11.58%	57	4	1	20
	0	0.780	100%	287	94	62	265

The threshold analysis reveals distinct patterns across model architectures, with particularly striking differences in how each model responds to confidence filtering:

- **GPT-4o** shows gradual degradation:
  - Maintains 89.3% accuracy at 95 threshold (75.1% coverage)
  - Accuracy declines steadily to 82.5% at full coverage
  - Demonstrates robust error control (FP $\leq$ 14 across all thresholds)
- **GPT-4o-mini** exhibits threshold sensitivity:
  - Perfect accuracy at 95 threshold (7.45% coverage)

  - Sharp 12% accuracy drop between 90 and 80 thresholds

- **Mistral** displays limited threshold utility:
  - Minimal accuracy improvement (8.3%) at 100 threshold
  - Near-complete coverage (99.61%) already at 90 threshold
- **Llama** presents extreme behavior:
  - Perfect accuracy at high thresholds (5.37% coverage)
  - Catastrophic 22% accuracy drop at full coverage

The comprehensive performance analysis reveals a clear hierarchy among the evaluated models. Neither Mistral nor Llama demonstrate sufficient grading reliability to serve as viable alternatives to the GPT series in this automated assessment context. GPT-4 consistently outperforms all other models, maintaining superior accuracy levels across the full range of confidence thresholds. The threshold mechanism proves especially effective with GPT-4, where precision improvements of 8-10% can be achieved while still maintaining practical coverage levels of 75-90%.

The decision between GPT-4 and GPT-4-mini ultimately reduces to an accuracy-cost tradeoff. GPT-4's architectural advantages translate to more stable performance across diverse question types and better accuracy preservation at higher coverage levels. However, these benefits come at a substantially higher operational cost. For institutions with limited budgets or less critical assessment needs, GPT-4-mini represents a reasonable compromise - particularly for pilot programs, developmental testing, or low-stakes evaluations where absolute precision is less critical.

5.4 Full Dataset Evaluation of Optimal Model Configuration

Building upon our threshold analysis, we evaluate GPT-4o-mini's performance across the complete 6,000-response dataset at temperature 0. This comprehensive assessment demonstrates the model's operational characteristics given its significant cost advantage (17× cheaper than GPT-4o) while maintaining reasonable accuracy levels.

Table 7: GPT-4o-mini Performance at T=0 Across Confidence Thresholds (6k Dataset)

Threshold	Precision	Coverage (%)	FP	FN
95	0.958	9.46	21	0
90	0.919	42.40	157	23
85	0.859	84.98	280	347
80	0.841	94.89	282	508
75	0.832	98.66	282	586
70	0.829	100.00	282	616

The analysis reveals three key performance patterns:

- **High-Threshold Reliability**: At 95% confidence threshold, the model achieves near-perfect precision (0.958) while processing 9.46% of responses. This configuration is optimal when false positives must be strictly minimized.
- **Balanced Operation Point**: The 90% threshold provides 91.9% precision with 42.4% coverage, representing the best compromise between accuracy and automation volume.
- **Diminishing Returns**: Beyond 85% threshold, precision gains become marginal while coverage drops exponentially. The 95% threshold filters out 90.54% of potential evaluations, making it suitable only for critical applications.

Figure 9 illustrates the precision-coverage tradeoff, revealing two optimal operating points depending on requirements:

- **90% threshold** (42.4% coverage, 91.9% precision):
  - Ideal when high accuracy is critical
  - Maintains precision above 90%
  - Processes 4.5× more evaluations than 95% threshold
  - False positive rate of 2.53%

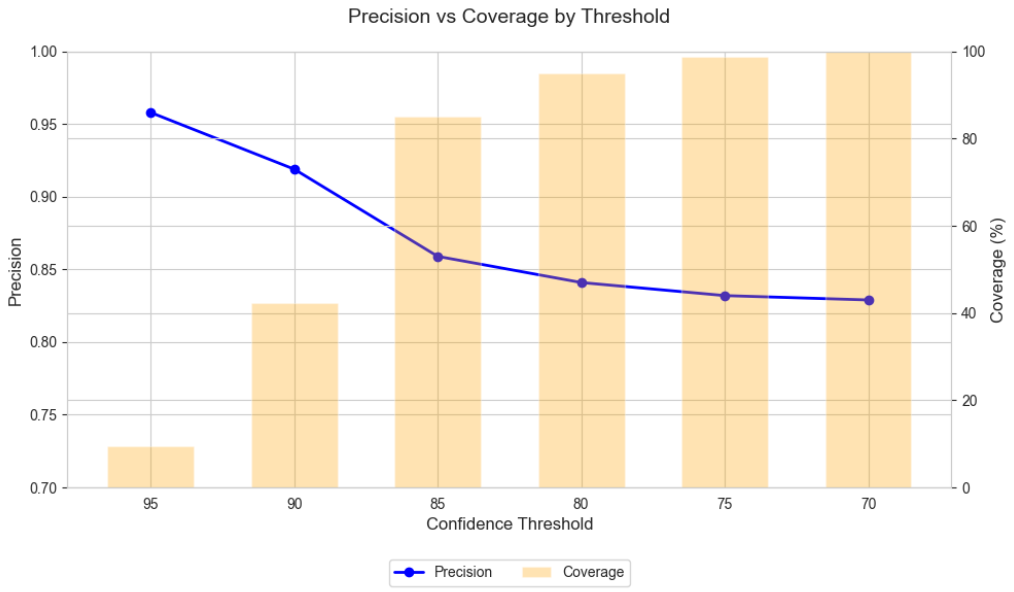

Figure 9: Precision-Coverage Tradeoff for GPT-4o-mini at T=0 (6k Dataset)

- **85% threshold** (84.98% coverage, 85.9% precision):
  - Best for maximum automation
  - Covers 85% of evaluations
  - Only 6% precision drop compared to 90% threshold
  - False positive rate remains moderate at 3.15%

The choice between these configurations depends on whether the priority is accuracy (90% threshold) or evaluation volume (85% threshold). Both options provide significantly better coverage than the 95% threshold while maintaining acceptable precision levels.

While GPT-4o maintains superior absolute performance (Section 5.3), GPT-4o-mini demonstrates sufficient reliability for various applications when properly configured. The model proves particularly effective for:

- Scenarios requiring cost-efficient automated evaluation
- Systems combining automated and manual verification
- Use cases where moderate false positive rates are acceptable

The following section will examine potential enhancements to these results and introduce a framework for optimizing both accuracy and coverage performance.

5.5 Discussion

While we achieved perfect precision through strict confidence thresholds, this approach significantly reduces coverage to impractical levels (2.89-7.45% at 100 threshold). However, our methodology reveals promising avenues for improvement through iterative prompt refinement and feedback analysis.

Before proceeding, it is worth noting that—though marginal—some responses remain misclassified or overly ambiguous, potentially influenced by the professor's subjective interpretation. These errors, present in evaluation, slightly degrade the results. A full manual review was unfeasible due to practical constraints, but this limitation should be acknowledged.

The LLM's reasoning output (provided with each evaluation) offers critical insights into its decision-making process. Qualitative analysis shows the models sometimes adhere too rigidly to certain evaluation criteria that may not be essential to the assessment objectives. Our fixed-prompt experimental design, while methodologically sound for comparative analysis, suggests room for optimization in practical applications through prompt engineering informed by these observations.

Furthermore, the RAG implementation's dependence on limited exemplars (10 correct/incorrect responses per question) likely constrained system performance. This limitation particularly affects questions where answer diversity requires more comprehensive examples for proper contextualization.

5.5.1 Proposed Framework

Our GUI-based implementation framework addresses these limitations through a human-in-the-loop approach:

1. **Initial Human Grading Phase**:
    - Instructors manually grade a diverse sample of responses (both correct and incorrect)
    - This expands and balances the RAG exemplar pool beyond our experimental baseline
2. **Iterative Calibration Phase**:
    - LLM evaluates a controlled subset of responses with reasoning
    - Instructors analyze the feedback to identify:
        - Overly strict or irrelevant evaluation criteria
        - Missing assessment dimensions

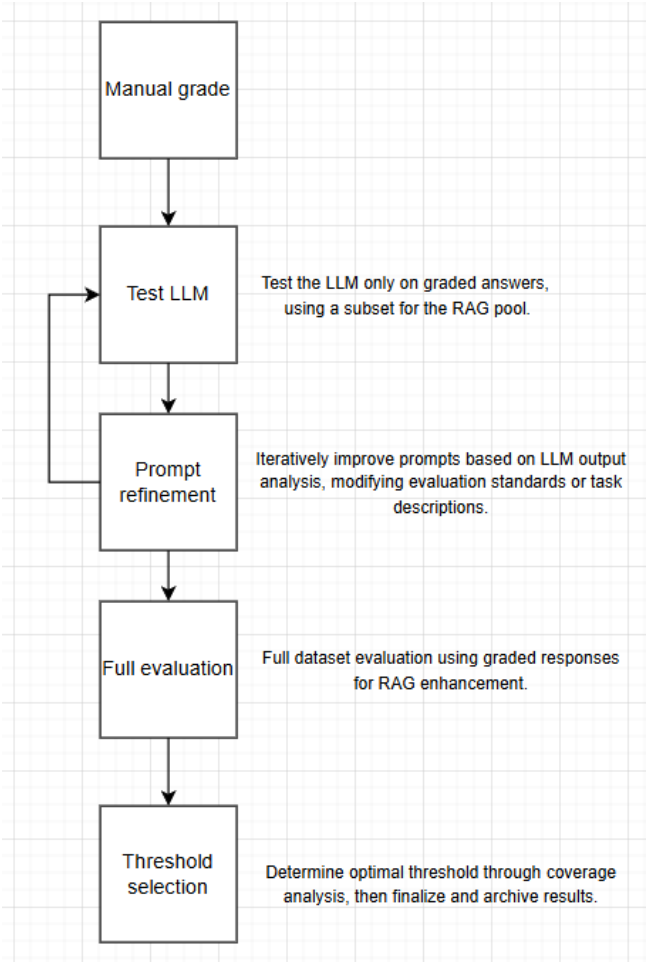

Figure 10: Structural overview of the hybrid assessment framework integrating LLM processing and instructor review

  - Potential biases in the LLM's interpretation

- Prompt refinement based on these insights

3. **Full Deployment Phase**:

- Application to full response set after calibration

- Optimal confidence threshold selection based on:

  - Assessment stakes

  - Required precision levels

  - Acceptable coverage tradeoffs

6 Implementation

6.1 GUI Description

A graphical user interface (GUI) was developed using PyQt6 to streamline the grading workflow by integrating the components and insights established in previous sections. The source code is available on GitHub.[1].

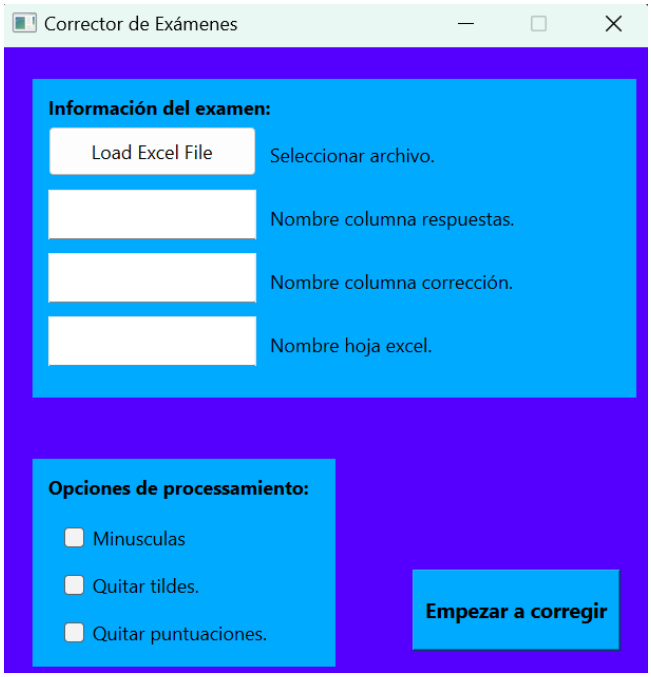

Figure 11: Initial interface window

The first window of the application, shown in Figure 11, allows users to load an Excel file and specify key input parameters: the name of the column containing student responses, the column designated for storing grades, and the worksheet name where the responses are located.

Additionally, the interface provides three optional preprocessing settings: converting all text to lowercase, removing diacritics (accent marks), and stripping punctuation. These options are user-selectable, as certain questions may rely on capitalization or punctuation for meaning. This cleaning process is particularly useful when dealing with short responses, where normalization often reveals repeated identical answers, thereby reducing the number of unique cases requiring individual evaluation.

After clicking the "Start Grading" button, the following window will appear:

[1]https://github.com/MounirCheriha/semi-auto-grading-gui

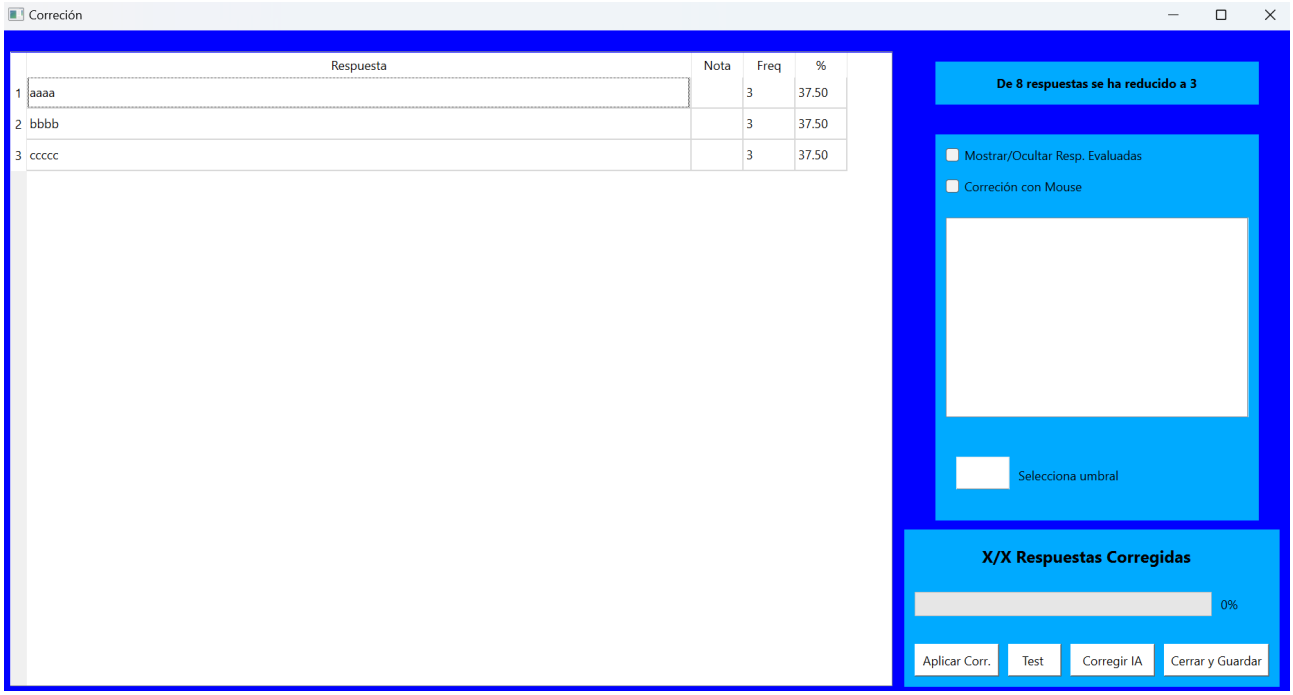

Figure 12: Correction Interface Window

This interface is divided into three main sections:

1. **Main Table:** The central area displays a table with the students' responses. The columns include:
   - The normalized student answer
   - The assigned score
   - The frequency (Freq) indicating how many times that exact response appears (After normalization).
   - The percentage of that response relative to the total

   The table can be sorted by clicking on the column headers.

2. **Options Panel** (on the right-hand side): At the top, a label shows how many responses remain after text normalization and grouping. Below that, two interactive options are available:
   - **Show/Hide Evaluated Responses:** If checked, previously evaluated responses are hidden from the table.

- **Mouse-based Grading:** When enabled, the user can assign scores directly with the mouse:
  - Left click on the score cell assigns a value of 1
  - Right click assigns a value of 0
  - Middle click clears the current score

  This functionality helps speed up the grading process by avoiding extra steps like manually editing each cell or clicking buttons.

3. **Threshold Table:** Beneath the options, there's a table (currently empty in the image) that will display confidence thresholds and the percentage of responses covered by each threshold. This assists the grader in selecting an appropriate confidence cutoff.

At the bottom, a button panel contains the following actions:

1. **"Apply Corrections"**: When clicked, this button writes the assigned scores back to the Excel file using openpyxl, without modifying the original file structure. Even if several responses were grouped during normalization, the program ensures that all equivalent responses are scored appropriately. After applying the corrections, the progress bar and status text are updated to reflect the total number of responses and how many are still pending evaluation.

   Important Note: Clicking this button captures a "snapshot" that confirms all your corrected values. Subsequent operations (LLM testing or AI-assisted correction) will use these confirmed values as reference points.

2. **"Test"**: This button performs a test evaluation using the LLM on the already graded responses (either graded during the current session or previously stored in the Excel file). For the RAG system, it uses half of the correct responses and half of the incorrect ones as the retrieval pool. The remaining responses are evaluated using the LLM (e.g., 'gpt-mini'). The prompt for this evaluation is stored in a '.txt' file within the project directory, allowing users to easily customize it. The program automatically parses and applies the prompt correctly. Once the evaluation is completed, a new window opens showing a table with the following information:
   - The student's response
   - The original grade assigned by the user
   - The AI-generated grade

- The feedback provided by the model
- The confidence score

This window is read-only and meant for informational purposes only; it does not modify the original Excel file. To continue, the user must close the window.

3. **"AI Correction"**: Similar to the "Test" button, this option evaluates the entire dataset. It uses all graded responses to build the RAG retrieval pool and then evaluates the ungraded responses using the LLM. The results—AI-assigned scores, feedback, and confidence scores—are displayed in a new column within the main table. Additionally, the threshold table on the right will be populated with each unique confidence score, showing how many responses would be accepted if that value were chosen as a threshold. After reviewing the results, the user can manually select the desired confidence threshold by entering it into the input box.
4. **"Close and Save"**: This button writes the AI-assigned scores back to the Excel file, similarly to the "Apply Corrections" button. However, it only includes responses whose confidence score is equal to or greater than the selected threshold. A new Excel file is saved containing the AI-assigned grade, feedback, and confidence score.

6.2 How to use

The interface is designed to be used in the following workflow:

First, in the initial window, load the Excel file you wish to work with and configure any desired text-cleaning options. Once the data is loaded and displayed in the main table, you should begin manually evaluating a set of student responses. It's important to include both correct and incorrect answers in this initial manual grading step. Note that this grading can also be done directly within the Excel file before launching the interface.

After you have manually graded a reasonable number of responses, click the "Apply Corrections" button. This ensures that your scores are written to the correct column in the Excel file. Since these scores were manually assigned, it is assumed that they are accurate and reliable

Next, click the "Test" button to evaluate the consistency and performance of the LLM. A new window will open, comparing the model's predictions with your manual grades. Review any discrepancies between the AI-generated grades and your own, and read the corresponding feedback to understand the model's reasoning.

At this point, you may notice that the model is focusing too heavily on a specific concept

or overlooking important criteria. If needed, you can refine the prompt used for evaluation by editing the prompt.txt file, which is located in the project directory. The program will automatically use the updated prompt in the next test, without requiring a restart.

You can repeat the testing process as many times as needed—adjusting the prompt and re-evaluating—until you are satisfied with the model's grading logic and feedback.

At this point, you can click the "AI Correction" button. This process may take some time depending on the number of ungraded responses, as the LLM will evaluate all remaining unanswered items using the RAG system and the prompt you previously configured.

Once the evaluation is complete, several new columns will appear in the main table: AI Score, Feedback, and Confidence. Additionally, a threshold table will be populated on the right side of the interface. This table shows how many responses would be accepted for different confidence threshold values.

The user is now expected to analyze this threshold-performance tradeoff. As seen throughout this study, higher confidence thresholds usually correspond to more accurate predictions, but fewer evaluated responses. Based on the importance of coverage vs. precision for the current use case, the user should select an appropriate confidence threshold and click the "Close and Save" button.

This will generate a new corrected Excel file, saved to the outputs folder. In the exported file, human and AI-assigned scores are saved in separate columns. This separation is intentional to avoid confusion between the two sets of scores. If desired, the user can merge the columns or apply further logic in Excel.

6.3 Additional Features and Future Work

This section outlines potential future improvements and optional features that could be added to the tool. These ideas may be implemented progressively and published via the project's GitHub repository if deemed useful.

1. **Quick Correction Mode**: A possible addition would be a "Quick Correct" button available directly from the initial file-loading window. This would allow users to immediately launch automatic correction, provided that a sufficient number of graded responses exist in the Excel file for the RAG system. The correction would proceed without opening any windows or filtering by confidence, and the resulting file would contain all evaluated answers. Users could then manually filter or analyze results based on confidence thresholds if they are already confident in the quality of the prompt.

2. **Automatic Report Generation**: A key feature identified for potential implementation was the automatic generation of a post-correction report; however, time limitations prevented its development. This report would summarize key statistics and insights extracted from both the student answers and AI feedback. Possible contents include:

   - Frequency analysis of common keywords in correct (score = 1) and incorrect (score = 0) responses using TF-IDF or BERT-based embeddings.
   - Prompt quality analysis: flagging patterns in feedback that may suggest missing evaluation criteria.

3. **Sentence Embeddings and Similarity-Based Selection**: Another potential feature would involve embedding student responses and the reference answer provided by the instructor using sentence-transformer models. After computing the similarity scores, The system could provide instructors with the option to manually review responses falling within intermediate similarity ranges - those that are neither clearly correct nor clearly incorrect.

   By including these borderline cases as examples in the prompt, The system guides the LLM toward more accurate decisions by explicitly defining criteria for correct and incorrect responses. This technique could enhance the model's sensitivity to subtle distinctions in answer quality. However, this would introduce additional processing time and complexity. It's still debatable whether this level of granularity justifies the added effort, or if the current approach is already sufficient for our goals. Further research could investigate whether this enhances the outcomes and the degree of improvement.

An immediate priority for future interface development is to improve error handling. The system should be robust enough to manage unexpected inputs or behaviors gracefully. In general, any functionality that users might expect to "just work" in order to streamline their experience should be implemented.

Furthermore, aesthetic design and overall usability will also be improved. These aspects will be addressed iteratively, based on feedback received from real users once the system is deployed. Incorporating this user feedback will be crucial to refining the interface and ensuring that it aligns with practical needs and expectations.

In the end, the main objectives were successfully achieved: providing an interactive interface that uses LLMs for automatic grading, enabling prompt refinement and confidence threshold selection within the app, and reducing the number of answers through basic preprocessing and normalization.

7 Conclusions

Our results demonstrate that GPT-4o stands out as the most reliable model in terms of both structured output generation and grading performance. Its deterministic behavior at temperature 0 ensures low parsing failure rates (under 3%) and consistent accuracy. Although GPT-4o-mini offers slightly lower accuracy, it provides a compelling trade-off with dramatically reduced computational cost, making it suitable for cost-sensitive applications.

Temperature analysis confirmed that deterministic sampling (T=0) offers the best grading reliability. Attempts to leverage higher temperatures (e.g., T=1) for increased linguistic diversity led instead to degraded accuracy and higher error rates, particularly with Llama 3.

Confidence thresholding emerged as a mechanism for precision control. For instance, GPT-4o-mini reached up to 95.8% precision at the 95% threshold while covering nearly 10% of the dataset. At the operational sweet spot of 90%, it achieved over 91% precision and 42% coverage—highlighting a feasible point of balance between automation and reliability.

It is hardly necessary to emphasize that LLaMA and Mistral cannot compete with GPT. However, it was important to evaluate them empirically. The possibility of using a local LLM for this task remains highly appealing, especially considering the significant cost savings it would entail.

The results could potentially be improved through prompt refinement and the incorporation of additional (or more representative) examples in the RAG pipeline. As discussed in the Future Work section 6.3, one promising direction involves constructing the prompt based on manually

designed evaluation criteria tailored to each question. Additionally, selecting responses within a defined range of ambiguity can lead to more precise retrieval. These refinements are likely to yield considerable performance gains. Employing a more advanced model such as GPT-4o with comprehensive implementation could yield substantially improved results.

It is also important to address a key limitation: the current evaluation framework is binary—each response is classified as either correct or incorrect. This setup causes small errors to have a disproportionately large effect on accuracy metrics. A valuable direction for future research would be to implement a more granular scoring system, such as a 0–100 scale. This would give the LLM room to express varying degrees of correctness, and the final evaluation could then determine whether the score falls within an acceptable range. However, this approach was not adopted in the present study, as the initial goal was to assess short, exam-style responses using a strict correct/incorrect scheme.

8 Glossary

Definitions of key terms and acronyms used in this report:

AI (Artificial Intelligence) Systems designed to perform tasks typically requiring human intelligence, including learning and problem-solving.

ASAG (Automated Short Answer Grading) Computer systems that evaluate and score student-written responses to open-ended questions.

BERT (Bidirectional Encoder Representations from Transformers) A transformer-based language model that processes words in relation to all other words in a sentence.

BiLSTM (Bidirectional Long Short-Term Memory) A type of recurrent neural network that processes sequence data in both forward and backward directions.

CNN (Convolutional Neural Network) A deep learning architecture particularly effective for processing grid-like data such as images or text.

FAISS (Facebook AI Similarity Search) A library for efficient similarity search and clustering of dense vectors.

GPT (Generative Pre-trained Transformer) A family of autoregressive language models developed by OpenAI.

GPU (Graphics Processing Unit) Specialized hardware for parallel processing, essential for deep learning computations.

GUI (Graphical User Interface) Visual interface allowing users to interact with software through graphical elements.

JSON (JavaScript Object Notation) Lightweight data interchange format used for structured data representation.

LLaMA (Large Language Model Meta AI) Meta's family of foundational language models.

LLM (Large Language Model) AI systems trained on vast text data to understand and generate human-like text.

LSTM (Long Short-Term Memory) A recurrent neural network architecture capable of learning long-term dependencies.

Mistral A series of efficient open-weight language models developed by Mistral AI.

NeMo (NVIDIA NeMo) An end-to-end framework for building, training, and deploying conversational AI models.

RAG (Retrieval-Augmented Generation) Architecture combining information retrieval with text generation.

VRAM (Video Random Access Memory) Specialized memory on GPUs used for storing and processing graphical data.

Bibliography

[1] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark, 2017. Association for Computational Linguistics.

[2] Lucas B. Galhardi, Helen Senefonte, Rodrigo de Souza, and Jacques Brancher. Exploring distinct features for automatic short answer grading. In *Anais do XV Encontro Nacional de Inteligência Artificial e Computacional*, pages 1–12, Porto Alegre, RS, Brasil, 2018. Sociedade Brasileira de Computação (SBC). In Portuguese.

[3] S. Haller, A. Aldea, C. Seifert, and N. Strisciuglio. Survey on automated short answer grading with deep learning: From word embeddings to transformers. 2022.

[4] Michael Heilman and Nitin Madnani. ETS: Domain adaptation and stacking for short answer scoring. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 275–279, Atlanta, Georgia, USA, 2013. Association for Computational Linguistics.

[5] Gerd Kortemeyer. Performance of the pre-trained large language model GPT-4 on automated short answer grading. *arXiv preprint*, 2023.

[6] Sachin Kumar, Soumen Chakrabarti, and Shourya Roy. Earth mover's distance pooling over siamese LSTMs for automatic short answer grading. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI-17)*, pages 2046–2052, Melbourne, Australia, 2017. IJCAI Organization.

[7] Yaman Kumar, Swati Aggarwal, Debanjan Mahata, Rajiv Ratn Shah, Ponnurangam Kumaraguru, and Roger Zimmermann. Get it scored using autosas - an automated system for

scoring short answers. In *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI Press, 2019. AAAI-19, Honolulu, Hawaii, USA.

[8] Omer Levy, Torsten Zesch, Ido Dagan, and Iryna Gurevych. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 285–289, Atlanta, Georgia, USA, 2013. Association for Computational Linguistics.

[9] A. Matarazzo and R. Torlone. A survey on large language models with some insights on their capabilities and limitations. 2025.

[10] Michael Mohler, Razvan Bunescu, and Rada Mihalcea. Learning to grade short answer questions using semantic similarity measures and dependency graph alignments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 752–762, Portland, Oregon, USA, 2011. Association for Computational Linguistics.

[11] Swarnadeep Saha, Tejas I. Dhamecha, Smit Marvaniya, Peter Foltz, Renuka Sindhgatta, and Bikram Sengupta. Joint multi-domain learning for automatic short answer grading.

[12] Ankit Sahu and Partha Pratim Bhowmick. Feature engineering and ensemble-based approach for improving automatic short-answer grading performance. *IEEE Transactions on Learning Technologies*, 13:77–90, 2020.

[13] J. Schneider, B. Schenk, and C. Niklaus. Towards llm-based autograding for short textual answers. 2024.

[14] Aman Singhal. Understanding temperature in LLMs. https://medium.com/@amansinghalml_33304/temperature-llms-b41d75870510, 2023. Accessed: May 25, 2025.

[15] Md Arafat Sultan, Cristobal Salazar, and Tamara Sumner. Fast and easy short answer grading with high accuracy. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1070–1075, San Diego, California, 2016. Association for Computational Linguistics.

[16] Inferless Team. Quantization techniques demystified: Boosting efficiency in large language models (llms).

[17] Manan Thakkar. Finetuning transformer models to build asag system.

[18] Chen Zhao, Miriam Silva, and Simon Poulsen. Language models are few-shot graders. *arXiv preprint*, 2025.