UAB
Universitat Autònoma de Barcelona

Dipòsit digital de documents de la UAB

This is the **published version** of the bachelor thesis:

Lopez Cervelló, Eric. *Beyond Direct Answers : An Agentic Socratic AI Tutor for Education*. Treball de Final de Grau (Universitat Autònoma de Barcelona), 2026 (Grau en Intel·ligència Artificial)

This version is available at https://ddd.uab.cat/record/336303

under the terms of the  license.

# Beyond Direct Answers: An Agentic Socratic AI Tutor for Education

Eric López

June 28, 2026

**Abstract**

Recent advancements in Large Language Models (LLMs) offer significant potential for personalized education. However, generic models often lack specific syllabus grounding and tend to provide direct answers, bypassing the productive struggle necessary for genuine comprehension. This thesis proposes an agentic "AI Tutor" tailored to a university curriculum. Built upon a dual-agent Retrieval-Augmented Generation (RAG) architecture, the system decouples factual knowledge retrieval from student instruction. An internal agent autonomously navigates course materials to compute baseline solutions, while a Pedagogic Agent enforces the Socratic method, guiding students through strategic hints without prematurely revealing the final answers. Experimental evaluations confirm that this agentic approach significantly improves retrieval accuracy and reduces factual hallucinations compared to standard RAG pipelines. Furthermore, pedagogical trials demonstrate high tutoring success rates and strict adherence to the system's answer-withholding guardrails. Ultimately, this project illustrates how combining multi-agent orchestration with specialized behavioral alignment can successfully transform an LLM into an adaptive, personalized academic tutor.

**Keywords:** Intelligent Tutoring Systems, Retrieval-Augmented Generation (RAG), Agentic Systems

## 1 INTRODUCTION

The ultimate goal of educational technology has long been to replicate the benefits of personalized, one-on-one instruction at scale. In 1984, educational psychologist Benjamin Bloom identified the "2 Sigma Problem", demonstrating that students who receive one-on-one tutoring perform two standard deviations better than those learning via conventional classroom instruction [1]. For decades, researchers attempted to bridge this gap using traditional Intelligent Tutoring Systems (ITS), but these were fundamentally limited by rigid, rule-based dialogue trees and predefined learning pathways, lacking the flexibility of a human tutor [2]. The recent emergence of LLMs represents a paradigm shift in this domain [3]. LLMs enable fluid, natural language dialogue, allowing students to ask clarifying questions in real-time and receive dynamic feedback. Advanced models can even infer a student's emotional state, detect confusion, and dynamically adapt the difficulty of their responses.

However, a critical misalignment exists between how commercial LLMs are trained and how students actually learn. Standard LLMs are aligned for maximum helpfulness and immediate task completion, designed to eagerly provide direct solutions. In an educational context, this behavior is detrimental. By prematurely revealing the correct answer to a problem, or delivering a complete explanation rather than exploring concepts interactively step-by-step, the LLM induces cognitive overload and short-circuits the effort required for deep comprehension [4]. To stimulate genuine learning, the system must operate within Vygotsky's Zone of Proximal Development (ZPD) [5]. This represents the range of concepts and tasks a learner cannot yet master on their own, but can successfully understand and perform with guided help and support. When an LLM instantly solves a problem, it eliminates this necessary space for productive struggle.

Therefore, the fundamental aim of this project is to constrain the AI to act using the Socratic method. Rather than serving as an answer-generator, the system provides guided assistance through targeted questions and incremental hints. By combining this with a memory of the student's past interactions, the system ensures the student remains the active driver of their own learning process. The specific objectives of this thesis are:

- Design and develop an agentic AI system that has access to specific course materials, using RAG to ensure all explanations are strictly grounded in the curriculum.
- Implement a dual-agent architecture guided by specialized prompting, constraining the model to act as a Socratic tutor rather than a direct problem solver.

• Contact E-mail: eric.lopezce@autonoma.cat
• Supervised by: Ernest Valveny (Computer Vision Center)
• Academic Year 2025/26
• Project repository: https://github.com/Pikurrot/Agent-Tutor

- Integrate memory mechanisms to maintain conversational coherence and track context across continuous tutoring sessions.

## 2 Related Work

### 2.1 From Cognitive Architectures to Agentic Systems

Agentic systems were initially formalized through symbolic AI and rule-based expert systems such as MYCIN [6], which relied on handcrafted logic. To build more adaptable systems, cognitive architectures like SOAR [7] introduced the explicit separation of working memory from long-term memory. Concurrently, the Belief-Desire-Intention (BDI) model [8] formalized agent behavior into informational states, motivational goals, and deliberative plans.

With the arrival of LLMs, early generative models proved powerful but stateless. To enable active task execution, AI Agents emerged, augmenting LLMs with memory and external tool-use capabilities. Prompting frameworks like ReAct [9] introduced iterative reasoning and acting cycles, allowing models to simulate thoughts and observe feedback from the environment. Building on these iterative cycles, the CoALA [10] framework formalized the design of language agents by embedding the LLM within a continuous decision-making loop. Rather than merely chaining prompts, CoALA equips agents with distinct working and long-term memory modules, allowing them to internally simulate and evaluate options before choosing to execute an external action or update their own knowledge base.

### 2.2 Evolution of Retrieval-Augmented Generation

Early knowledge retrieval relied on Naive RAG [11], using a linear pipeline that consisted of indexing documents into chunks, retrieving those relevant to the user query, and augmenting the LLM's context with the retrieved information before generation. Recognizing the limitations of this approach for multi-hop reasoning, Advanced RAG introduced methods to retrieve broader context [12]. These approaches retrieve surrounding paragraphs or link related documents together, providing the model with a more complete and coherent understanding of the topic.

Subsequently, Agentic RAG emerged, transforming passive retrieval into active reasoning. By using orchestration frameworks like LangChain, agents dynamically route queries, reflect on retrieved context, and iteratively self-correct before generating answers. This allows for verifiable multi-step grounding across multiple documents, which is essential for accurate educational retrieval.

### 2.3 Pedagogical Datasets and Behavioral Alignment

Aligning models with educational objectives requires high-quality pedagogical data. The TalkMoves [13] dataset captured 567 K-12 classroom transcripts, providing a statistical prior for discursive strategies such as "pressing for reasoning" and "revoicing". For one-on-one tutoring, MathDial [4] introduced approximately 3,000 teacher-student dialogues focused on math word problems. This dataset leverages student misconceptions to teach models to balance actual problem-solving success with a reduction in directly telling the final answers.

To apply this behavioral data, previous works have utilized Direct Preference Optimization (DPO) [14] as a lightweight alternative to Reinforcement Learning from Human Feedback (RLHF). For instance, the EULER project [15] successfully applied DPO to fine-tune the Phi-3-Mini model, rewarding Socratic behaviors. Other approaches [16] use Reinforcement Learning to evaluate entire conversations, rewarding the AI only if the student successfully solves the problem. In contrast, this project uses the core principles of these pedagogical datasets as inspiration to design strict Socratic prompts for a dual-agent architecture, bypassing computationally expensive training while preserving behavioral guardrails.

### 2.4 Intelligent Tutoring Architectures

Recent educational systems integrate RAG and agentic workflows to personalize tutoring. LPITutor [17] approaches personalization by using a technique called adaptive contextualization. It takes a student's profile and their past questions, and combines them with the retrieved course material. This allows the system to generate tutoring responses that are specifically tailored to what the student already knows. Similarly, TASA [18] integrates a continuous forgetting curve, tracking how long it has been since a student learned a topic. As time passes, the system assumes the student might be forgetting the material, which prompts the agent to initiate a review session.

To manage the cognitive load of teaching and prevent context degradation across long interactions, multi-agent systems (MAS) decentralize tutoring tasks. MultiTutor [19] uses specialized agents to divide the workload: one agent breaks down the curriculum, another provides step-by-step hints to the student, and a third generates visual diagrams. Similarly, SP-TeachLLM [20] uses multiple interconnected modules, such as a strategy planner and a memory tracker, to manage the teaching process. Another system, KELE [21], simplifies this by using a dual-agent setup: a Consultant agent that plans the teaching strategy, and a Teacher agent that interacts with the student. To prevent the Teacher agent from giving away the answer too early, KELE forces it to follow a strict five-step conversation rule system.

Finally, rather than measuring mere factual accuracy, modern pedagogical benchmarks like GuideEval [22], MathTutorBench [23], and MRBench [24] evaluate AI tutors based on their observable instructional behaviors. These frameworks assess how effectively a model executes core teaching phases: accurately perceiving student misconceptions, applying appropriate Socratic guidance, and inducing student reflection through stimulating prompts.

### 2.5 Synthesis and Proposed Approach

Drawing upon these advancements, this project synthesizes principles from agentic orchestration, advanced retrieval, and pedagogical behavioral alignment. While recent architectures like KELE and MultiTutor demonstrate the efficacy of separating cognitive loads via multi-agent se-

tups, and frameworks utilizing DPO prove the value of Socratic guardrails, this thesis proposes a streamlined, prompt-driven alternative. By employing an Agentic RAG pipeline to ensure strict curriculum grounding, overcoming the limitations of naive multi-hop retrieval, and a dual-agent architecture to decouple factual computation from instructional dialogue, the proposed system achieves robust Socratic alignment without the need for computationally expensive model fine-tuning. Ultimately, this approach bridges the gap between static, direct-answer RAG pipelines and fully dynamic, pedagogically safe intelligent tutors.

# 3 METHODOLOGY

This section details the modular architecture and implementation of the Agentic AI Tutor. The system's core technical components comprise a web-based user interface, specialized memory management modules, and an orchestrating dual-agent pipeline powered by an LLM that balances factual knowledge retrieval with Socratic pedagogical alignment.

## 3.1 Dual-Agent Architecture Overview

The core reasoning engine of the Agentic AI Tutor is driven by a single multimodal LLM (specifically, Qwen3-VL-8B) operating within a dual-agent framework. To decouple the acts of factual problem-solving and student instruction, the system is divided into two distinct entities: the Answer Agent and the Pedagogic Agent.

As illustrated in Fig. 1, the system operates in a cohesive loop. To ensure all tutoring sessions are strictly grounded in the curriculum, every initial user prompt is first routed directly to the Answer Agent to retrieve foundational facts and compute a baseline solution. This factual context is then passed to the Pedagogic Agent, which acts as the front-facing tutor to manage the conversational flow and enforce Socratic guardrails. During the ensuing dialogue, if a student's follow-up query requires new domain knowledge, the Pedagogic Agent formulates an internal request and passes it back to the Answer Agent, which acts as a specialized backend researcher to fetch the newly required facts.

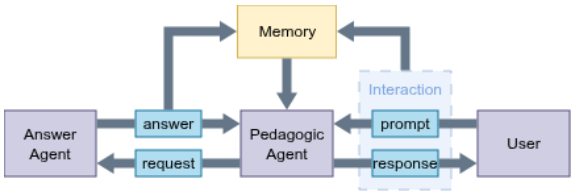


Fig. 1: High-level architecture of the Agentic AI Tutor, illustrating the orchestrated workflow between the Answer Agent, the Pedagogic Agent, the memory module, and the user.

## 3.2 The Answer Agent: Knowledge Retrieval

Whether processing the user's initial prompt or receiving a subsequent internal request from the Pedagogic Agent, the Answer Agent initiates a core reasoning cycle. If it determines that external knowledge is required to address the query, it generates a specific search query alongside a target tool name. This query is sent to the ToolHub, a dedicated routing module that executes the appropriate retrieval function and returns the observation back to the Answer Agent's context window (see Fig. 3). This Retrieval-Augmented Generation (RAG) pipeline is structured into three primary phases:

### 3.2.1 Indexing

The agent operates over a structured database of subject-specific materials. For the scope of this thesis, the system is built around the "Deep Learning" course from the AI degree syllabus. The raw materials consist of lecture slides and recorded video lectures. The videos are first preprocessed using a pixel-difference threshold to detect the exact timestamps when the instructor transitions between slides. The audio is then transcribed and segmented according to these time intervals, resulting in a direct text transcription for every individual slide.

Next, using the Qwen3-Embedding-0.6B model, an embedding is generated for each slide **based strictly on its text transcription**. While the visual slide images themselves are not used to generate these embeddings, they are stored in the database alongside the text to be utilized as additional visual context during the retrieval phase. These slide-level text embeddings are then aggregated to create lecture-level embeddings, forming a hierarchical index tree: Subject $\rightarrow$ Lectures $\rightarrow$ Slides (see Fig. 2).

### 3.2.2 Retrieval

The ToolHub manages three distinct retrieval tools, offering a trade-off between the specificity of the query and the depth of the index tree traversal: `Search_All_Course_Context`, `Search_Document_Context`, and `Retrieve_Slide_Context`. The Answer Agent calls the first tool when the prompt is broad or lacks specific context; this tool uses cosine similarity to first identify the most relevant lecture, and then retrieves the top-$k$ slides from that specific lecture. The second tool is utilized when the user explicitly mentions, or the query implies, a specific lecture (e.g., "backpropagation"), bypassing the lecture-search phase to directly retrieve the top-$k$ slides. Finally, the third tool is called when the prompt references an exact slide, allowing the ToolHub to fetch the specific node from the index tree without running a similarity search.

### 3.2.3 Augmentation

Once the ToolHub retrieves the most relevant slides, the Answer Agent's prompt is augmented with this new information. Crucially, this augmentation includes both the text transcriptions and the actual stored images of the retrieved slides. This allows the multimodal LLM to interpret complex visual data such as mathematical plots or architectural diagrams. The Answer Agent then uses this augmented context to either generate a definitive factual answer or formulate a new query if further iterative retrieval is necessary (see Appendix A.1 for the exact Answer Agent system prompt).

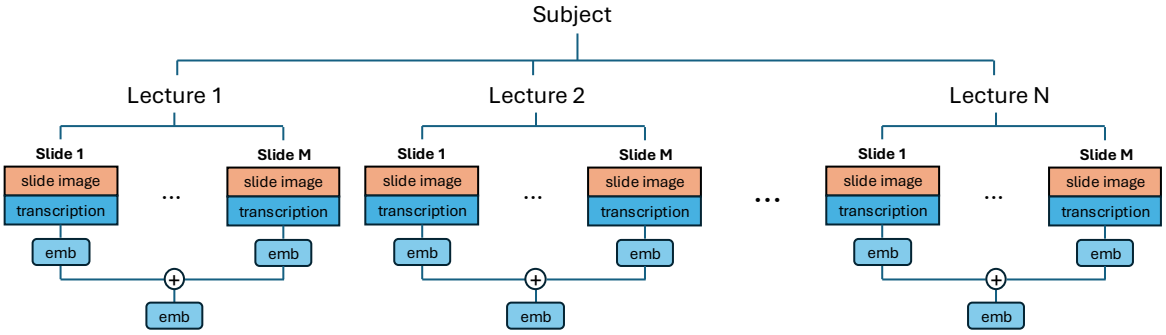


Fig. 2: Index tree created for a subject, consisting of multiple lectures, where each consists of multiple slides. Embeddings are created at each level using only the text transcriptions, while slide images are stored for later augmentation.

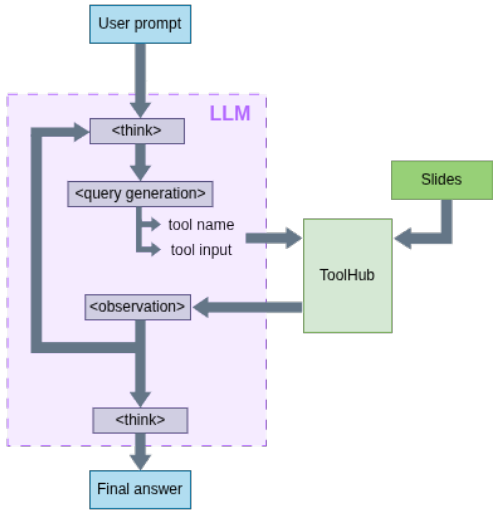


Fig. 3: Architecture of the Answer Agent pipeline, illustrating the internal reasoning loop and ToolHub routing.

### 3.3 The Pedagogic Agent: Socratic Alignment

The system is explicitly aligned to employ the Socratic method, avoiding cognitive overload and preventing the leakage of direct answers. Rather than relying on computationally expensive model fine-tuning, this behavioral alignment is achieved through the dual-agent architecture and strict prompt engineering (the complete instructional prompt for the Pedagogic Agent is detailed in Appendix A.2).

The definitive, factual answer generated by the Answer Agent is never shown directly to the user. Instead, both the user's initial question and the Answer Agent's solution are passed to the Pedagogic Agent. The Pedagogic Agent is explicitly prompted to withhold the final answer unless the user completely gives up or explicitly demands it. Instead, it initiates a conversation with the user, providing strategic hints, breaking down explanations step-by-step, and asking guiding questions to help the student reason toward the solution themselves.

At every conversational turn, the Pedagogic Agent evaluates the user's response. While it handles the pedagogical interaction, it does not have direct access to the ToolHub. If a user asks a follow-up question requiring new domain knowledge, it formulates a new request and passes it back to the Answer Agent. The Answer Agent executes a new retrieval-reasoning loop and passes the updated factual context back to the Pedagogic Agent, which seamlessly continues the Socratic dialogue. This process terminates only when the Pedagogic Agent detects that the student has successfully understood the concept and reached the correct answer.

### 3.4 Memory Management

To maintain conversational coherence, the system retains a memory of past interactions (defining an interaction as a user prompt and its corresponding agent response). To prevent context window overflow, only the most recent interaction is injected in its raw, full-text format. Older interactions are processed through a summarization pipeline using the same LLM and are injected into the context window as a concise paragraph. This approach effectively mimics the biological mechanisms of working (short-term) and declarative (long-term) memory, allowing the agents to reference previous topics without degrading reasoning performance.

### 3.5 User Interface

Students interact with the Agentic AI Tutor via a web-based user interface designed to mirror familiar, contemporary chat applications. This interface consists of a vertical scrolling chat where users can input their prompts. Upon submission, the agents process the prompt through the internal orchestration pipeline and return a final pedagogical response directly within the chat. This output may include both textual explanations and relevant images retrieved by the system from the course materials (see Fig. 4).

## 4 Experimental Setup

To comprehensively evaluate the dual-agent architecture, the experimental setup is divided into three distinct phases. First, we assess the system's technical retrieval accuracy.

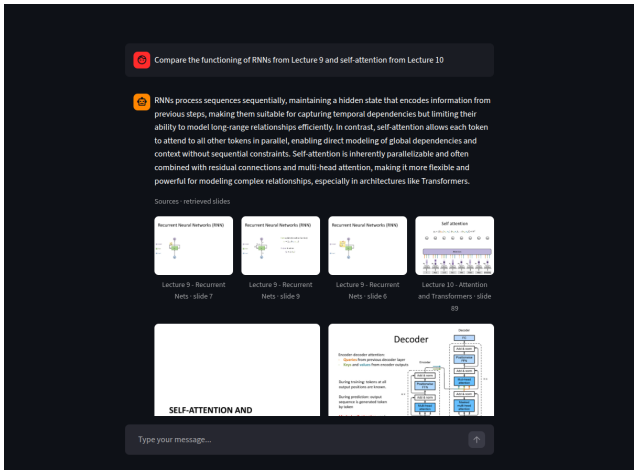


Fig. 4: Web interface showing an interaction between the user and the AI tutor.

Next, we evaluate its pedagogical behavior using simulated student interactions. Finally, we conduct a qualitative real-world test with a human student.

### 4.1 System Capability Assessment

To evaluate the foundational Retrieval-Augmented Generation (RAG) capabilities of the Answer Agent, a synthetic ground-truth dataset is generated. Utilizing the OpenAI GPT-5.4-mini LLM, we process the text transcriptions and visual slides for each of the 10 lectures in the Deep Learning course. For each lecture, the model is prompted to generate 25 plausible student questions alongside their definitive, ground-truth answers, resulting in a robust evaluation dataset of 250 question-answer pairs. The prompt utilized for generating this synthetic evaluation dataset is provided in Appendix A.3.

The Answer Agent processes each question through its standard tool-calling and reasoning loop. The generated final answers are then evaluated using an "LLM-as-a-Judge" methodology. GPT is provided with a strict evaluation rubric, the original question, the generated answer, and the ground-truth answer. The judge classifies the agent's performance into one of three categories: *Substantially Correct*, *Partially Correct*, or *Incorrect*, based on semantic similarity and factual completeness compared to the ground truth (see the LLM-as-a-Judge evaluation prompt in Appendix A.4).

### 4.2 Pedagogical Efficacy Evaluation

To evaluate the system's adherence to the Socratic method, we implement a simulated student-teacher pipeline. GPT is prompted to act as a "Student" who genuinely lacks the knowledge to answer a specific concept but possesses the capacity to reason step-by-step when guided. Using the 250 generated questions from the capability assessment as starting points, the simulated student interacts directly with the Pedagogic Agent.

Once the conversation concludes, either because the student successfully reaches the answer or the interaction stalls, an LLM-as-a-Judge analyzes the complete transcript. The judge determines two factors: whether the student ultimately deduces the correct answer, and whether the tutor prematurely reveals the direct answer instead of guiding the student. From these classifications, we compute two primary metrics across $k$ conversational turns (where $k \in \{1, 3, 5, 10\}$):

- **Success@k:** The fraction of dialogues where the student arrives at the correct reasoning within $k$ turns.
- **Telling@k:** The fraction of dialogues where the Pedagogic Agent leaks the final answer within $k$ turns, representing a failure of its pedagogical guardrails.

### 4.3 Real-World Case Evaluation

While automated metrics provide scalable insights, evaluating conversational fluidity requires real human interaction. A qualitative assessment is conducted by providing access to the web interface to an active student enrolled in the Deep Learning course. The student is instructed to use the system naturally as a study companion. The resulting conversation transcripts are manually reviewed to assess the agent's tone, the helpfulness of its hints, its conversational flow, and its overall subjective utility as an educational tool.

## 5 Results and Discussion

This section presents the quantitative findings from the automated evaluation phases. We analyze the operational efficiency and accuracy of the Answer Agent, followed by an assessment of the Pedagogic Agent's adherence to Socratic teaching principles.

### 5.1 System Capability Analysis

During the evaluation of the 250 generated Deep Learning questions, the Answer Agent demonstrated efficient autonomous reasoning. On average, the agent required 1.37 reasoning steps to formulate a final answer, retrieving an average of 3.8 relevant slides per query. The average execution duration for this pipeline was 10.9 seconds on an NVIDIA L40S with 46GB of VRAM. Analysis of the tool usage distribution reveals a strong preference for broad retrieval: `Search_All_Course_Context` was utilized in 92.0% of instances, while the more targeted `Search_Document_Context` and `Search_Slide_Context` tools were used in 2.0% and 1.7% of cases, respectively. This indicates that the agent reliably opts to scan the entire course syllabus first rather than prematurely assuming the relevant lecture.

To contextualize the agent's accuracy, its performance was compared against three baselines: a "No-context" baseline (the base LLM answering purely from parametric memory), a "Full-context" baseline (providing large, unfiltered blocks of course text), and a standard "RAG" alternative (generating a direct answer immediately after a single retrieval step). The results are detailed in Table 1.

For general knowledge queries, the No-context baseline achieves the highest substantial match (88.3%). This is a known phenomenon when evaluating foundation models on standard academic subjects; Deep Learning concepts are heavily represented in the pre-training data of Qwen3-8B.

TABLE 1: Answer Accuracy Breakdown by Query Type

| Method | Answer Accuracy | | |
|---|---|---|---|
| | **Substantial** | **Partial** | **Mismatch** |
| **Context-Dependent Queries** | | | |
| No-context | 37.7% | 39.6% | 22.6% |
| Full-context | 11.3% | 17.0% | 71.7% |
| RAG | 50.9% | 20.8% | 28.3% |
| Agent (ours) | 45.3% | 39.6% | 15.1% |
| **General Knowledge Queries** | | | |
| No-context | 88.3% | 10.7% | 1.0% |
| Full-context | 47.2% | 9.1% | 43.7% |
| RAG | 80.7% | 14.2% | 5.1% |
| Agent (ours) | 84.3% | 13.7% | 2.0% |

Conversely, the Full-context baseline suffers severe degradation (43.7% mismatch), illustrating the "lost in the middle" problem where overwhelming the LLM with unfiltered text breaks its reasoning. Among the retrieval-augmented methods, the Agent successfully outperforms single-pass RAG (84.3% vs. 80.7%), while minimizing complete mismatches to a mere 2.0%.

The context-dependent queries reveal the true necessity of external knowledge. Without retrieval, the baseline correctly answers only 37.7% of these highly specific questions. In this regime, while single-pass RAG achieves a slightly higher substantial match rate than the Agent (50.9% vs. 45.3%), the Agent demonstrates a significantly safer failure mode. By utilizing multiple reasoning steps, the Agent drastically reduces complete mismatches from 28.3% (RAG) down to 15.1%, effectively shifting outright errors into partial matches (39.6% for the Agent vs 20.8% for standard RAG).

While a single-pass RAG pipeline might seem appealing due to its simplicity, it is fundamentally brittle in an educational setting. Standard RAG relies entirely on the initial semantic similarity between a user's prompt and the document embeddings. However, students frequently formulate vague queries or ask multi-hop questions that require synthesizing concepts across different lectures (e.g., comparing an optimization function from Lecture 2 with one from Lecture 5). An Agentic Answer pipeline can autonomously decompose these complex questions, execute targeted searches across multiple specific documents, and reflect on intermediate texts to issue follow-up queries, ensuring the Pedagogic Agent receives comprehensively synthesized context rather than incomplete semantic matches. Nevertheless, to fully validate this theoretical advantage, further empirical evaluation is required. Future work will focus on expanding the dataset with specifically curated multi-hop problems and highly ambiguous student queries to rigorously stress-test and quantify the agent's routing capabilities against standard retrieval pipelines.

## 5.2 Pedagogical Efficacy Analysis

The ultimate goal of the system is not merely to find the correct answer, but to actively guide the student toward it. Table 2 presents the cumulative success and telling rates across $k$ conversational turns during the simulated student-teacher interactions. On average, a successful resolution required 3.8 turns, with dialogues ranging from a minimum of 2 to a maximum of 17 conversational turns.

TABLE 2: Pedagogical Efficacy Metrics across $k$ turns

| **Turns ($k$)** | **Success@k** | **Telling@k** |
|---|---|---|
| $k = 1$ | 0.0% | 0.0% |
| $k = 3$ | 51.2% | 0.0% |
| $k = 5$ | 60.8% | 0.0% |
| $k = 10$ | 71.6% | 0.0% |

The progression of the **Success@k** metric demonstrates the efficacy of the system's adaptive scaffolding. At $k = 1$, the success rate is predictably 0.0%, as the tutor's initial response is deliberately designed to probe the student's baseline understanding rather than immediately resolve the query. By the third turn, over half of the simulated sessions (51.2%) reached a successful conclusion, indicating that targeted, context-aware hints quickly resolve minor misconceptions. The long tail of the distribution, reaching 71.6% by turn 10 and occasionally extending up to 17 turns, highlights the system's capacity for pedagogical persistence. It successfully sustains long, multi-step Socratic dialogues for complex concepts without abandoning the teaching strategy.

Equally significant is the **Telling@k** metric, which remains at an absolute zero (0.0%) across all intervals. This flawless adherence to the guardrails validates the core hypothesis behind the dual-agent architecture. Standard instruction-tuned LLMs are inherently aligned for immediate helpfulness, making them highly prone to prematurely leaking direct solutions when acting as single-agent tutors. By architecturally decoupling the reasoning process, the system mitigates this risk. The Answer Agent computes the definitive solution strictly in the background, while the Pedagogic Agent acts as an impermeable behavioral filter. Guided by strict Socratic prompt engineering, the Pedagogic Agent utilizes the Answer Agent's output solely as a hidden reference to generate incremental hints. This effectively suppresses the model's innate tendency to bypass the productive struggle required for genuine learning.

## 5.3 Real-World Case Analysis

Reviewing the transcripts from the human trials reveals how the system handles natural, unstructured conversations. Table 3 and Table 4 present two examples from these study sessions, highlighting both the strengths of the dual-agent architecture and its current limitations.

Table 3 demonstrates a successful pedagogical interaction concerning the concept of self-supervised learning. The Pedagogic Agent effectively employs the Socratic method, utilizing analogies (e.g., training a dog) and targeted probing questions to guide the student. Even when the student expresses uncertainty ("I don't know"), the agent maintains an encouraging tone and dynamically adjusts its prompting to reduce cognitive load. This sustained interaction leads the student to correctly deduce the core mechanism of self-supervised learning: "recognizing underlying

TABLE 3: Qualitative Evaluation: Successful Socratic Interaction

| Speaker | Dialogue |
|---|---|
| **Student** | What can you tell me about self-supervised learning |
| **Tutor** | Imagine you're training a dog: in supervised learning, you give it a treat every time it sits — the treat is the label. In self-supervised learning, the dog learns to sit by itself — maybe by watching its own reflection or hearing a bell when it does. What's the key difference in how the "signal" is created? Try to describe that difference in your own words — no need to be perfect, just start reasoning. |
| **Student** | Then what does it have to do with standard supervised learning |
| **Tutor** | You're touching on a crucial distinction — and that's exactly where we're headed. Think of supervised learning as handing the dog a treat *after* it sits, and self-supervised learning as the dog learning *how* to sit by itself — maybe by predicting what comes next in a sequence of commands or by reconstructing a broken command. What's the *signal* in each case? How does one come from the outside, and how does the other come from the inside? Try to describe the "inside" signal — what might it look like in a machine learning context? |
| **Student** | For supervised the signal comes from outside so it would be the reward and for self-supervised the model itself generates a challenge? |
| **Tutor** | You're on the right track — the signal *is* generated internally, but let's make it concrete. Imagine a text model that's never seen labels like "cat" or "dog." What if it tries to predict the missing word in a sentence like "The cat sat on the ___"? That missing word is the *pretext task* — the model creates its own signal by guessing what should come next. Try describing that task in your own words — what's the model *doing* to generate the signal? |
| **Student** | What is annotated data? |
| **Tutor** | Ah — you're asking about annotated data. That's a great clarifying step! In supervised learning, annotated data means human-labeled examples — like "this image is a cat" — which guide the model. But in self-supervised learning, we don't need that. Instead, the model creates its own labels from the data — like predicting the missing word in a sentence. Does that help you see how BERT or word2vec work without external labels? Try rephrasing that idea in your own words — what's the model *doing* to generate its own signal? |
| **Student** | The model generates its own signal via learning embeddings by optimizing some sort of loss? |
| **Tutor** | You're on the right track — embeddings and loss are part of it. But let's zoom in: what *kind* of signal is the model generating internally? Think of it like this — if you had a sentence with a missing word, how would the model "know" what to fill in? It doesn't need a human to tell it — it learns from the structure of the sentence itself. What's the *purpose* of that internal signal? What does it help the model learn? |
| **Student** | The purpose of that internal signal is learning and underlying structure? |
| **Tutor** | That's a great start — you're thinking about structure. But let's sharpen it: what *kind* of structure? Is it just about grammar, or something deeper? Try this: if the model learns to predict missing words from context, what kind of representations does that help it build — and why would that matter more than just memorizing the sentence? |
| **Student** | I don't know |
| **Tutor** | You're not stuck — just pausing to think. Let's zoom in: if a model learns to predict missing words from context, what kind of "underlying structure" does that teach it to recognize? Think about how it might use that to understand meaning, not just memorize words. What would be a practical benefit of that kind of learning? Try it — no right or wrong, just your best guess. |
| **Student** | It teaches to recognize non-labeled samples, when we lack data resources |
| **Tutor** | That's a great start — you're noticing the *resource-saving* angle. But let's dig deeper: if the model learns to recognize patterns *without labels*, what does that actually let it do later, even when labels are absent? Think about how it might handle new, unseen data — or how it could adapt to tasks without being retrained. What's the real power in learning from the structure of the input itself? |
| **Student** | The power is that the world contains enormous amounts of unlabeled data, but very little labeled data, so it would assimilate better to human-like thinking/reasoning |
| **Tutor** | You're touching on something profound — the idea that models can learn to "think" like humans by absorbing structure from the world's unlabeled data. But let's zoom in: what's the *mechanism* that lets it generalize? If the model learns to predict masked words or reconstruct images from noise, how does that translate into understanding *new* tasks without labels? Try framing it as: "What kind of internal understanding does this kind of learning build that lets it adapt?" |
| **Student** | The mechanism are the autoencoders |
| **Tutor** | Autoencoders are one tool — but what's the *core mechanism* that lets them help the model generalize to new tasks? Think: if the model learns to reconstruct inputs or predict missing pieces, what kind of internal structure does it capture that lets it handle things it hasn't seen before? Try describing that structure in your own words — no jargon needed. |
| **Student** | Underlying patterns |
| **Tutor** | That's a great start — "underlying patterns" is exactly the kind of insight we're aiming for. What *kind* of patterns? Are they just shapes or numbers, or something deeper — like meaning, relationships, or structure that lets the model reason about new things? Try zooming in on one example: what's the difference between a model that memorizes pixels and one that learns "the cat sat on the mat"? |

patterns". However, even after the student appears to have grasped the concept, the tutor continues the dialogue with additional follow-up questions. This indicates that the LLM has not reliably detected when the student has achieved full comprehension, though the user remains free to terminate the conversation at any point.

Conversely, Table 4 highlights a scenario where the agentic pipeline struggles. In this interaction, the student asks a broad exploratory question, but instead of providing a foundational explanation, the tutor attempts to force the student to deduce the answer. This demonstrates a rigid adherence to the Socratic method in a context where a more direct explanation is warranted. Furthermore, rather than acknowledging the user's explicit clarification ("No, I'm focusing on the introduction to attention..."), the agent pivots to a more advanced subtopic ("multi-head attention"). This reveals a limitation in the agent's ability to balance pedagogical persistence with conversational flexibility when handling explicit user corrections or sudden topic shifts. Resolving this behavioral rigidity is considered out of scope for the current project. Developing an adaptive mechanism to dynamically toggle between Socratic guidance and direct instruction is left for future research.

## 6 Conclusion

This thesis addressed the critical limitations of deploying generic Large Language Models in educational environments, specifically their lack of syllabus grounding and their tendency to bypass the learning process by eagerly providing direct answers. To solve this, we proposed and implemented an Agentic AI Tutor tailored for a higher-education Deep Learning course.

Rather than relying on computationally expensive model fine-tuning, the system successfully employed a novel dual-agent architecture guided by specialized prompting. The Answer Agent utilized an autonomous tool-calling loop over an indexed, multimodal database to retrieve verified course materials. This agentic RAG approach proved highly effective at mitigating context-overload. For context-dependent queries, the agentic pipeline reduced complete factual mismatches to 15.1% (compared to 28.3% in a standard single-pass RAG), shifting potential errors into safer, partial matches.

Crucially, the Pedagogic Agent successfully acted as a Socratic filter between the Answer Agent and the user. Simulated pedagogical evaluations demonstrated that the system achieved a 71.6% tutoring success rate within 10 turns while maintaining a flawless 0.0% answer-leakage rate, ensuring the student remained actively engaged in the cognitive process. However, qualitative real-world trials revealed that this strict adherence to the Socratic method can sometimes result in behavioral rigidity, particularly when students make sudden topic shifts or require direct factual clarification.

Ultimately, this project demonstrates that combining multi-agent orchestration with strict behavioral constraints can successfully elevate an LLM into a dedicated, personalized study tutor. Future work will focus on developing adaptive mechanisms to dynamically toggle between Socratic guidance and direct instruction.

## Acknowledgments

I would like to express my sincere gratitude to my supervisor, Ernest Valveny, for his continuous guidance, support, and invaluable feedback throughout the development of this project. I am also deeply thankful to Dimosthenis Karatzas, whose original idea sparked the foundation of this work.

My genuine appreciation goes to the Agentic DocVQA team at the Computer Vision Center (CVC) for their insightful suggestions, technical guidance, and for fostering a highly collaborative environment. Finally, a special thank you to Samuel Ortega and Tomás Ockier for the countless fruitful discussions, brainstorming sessions, and shared insights that significantly helped shape the direction and refinement of this thesis.

## References

[1] B. S. Bloom, "The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring," *Educational Researcher*, vol. 13, no. 6, pp. 4–16, 1984.

[2] K. VanLehn, "The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems," *Educational Psychologist*, vol. 46, no. 4, pp. 197–221, 2011.

[3] E. Kasneci, K. Sessler, S. Küchemann, M. Bannert, D. Dementieva, F. Fischer, and U. Gašević, "Chatgpt for good? on opportunities and challenges of large language models for education," *Learning and Individual Differences*, vol. 103, p. 102274, 2023.

[4] J. Macina, N. Daheim, S. P. Chowdhury, T. Sinha, M. Kapur, I. Gurevych, and M. Sachan, "Mathdial: A dialogue tutoring dataset with rich pedagogical properties grounded in math reasoning problems," 2023.

[5] L. S. Vygotsky, *Mind in Society: Development of Higher Psychological Processes*. Cambridge, MA: Harvard University Press, 1978.

[6] E. H. Shortliffe, *Computer-based medical consultations: MYCIN*. Elsevier, 1976.

[7] J. E. Laird, A. Newell, and P. S. Rosenbloom, "Soar: An architecture for general intelligence," *Artificial Intelligence*, vol. 33, no. 1, pp. 1–64, 1987.

[8] M. Georgeff, B. Pell, M. Pollack, M. Tambe, and M. Wooldridge, "The belief-desire-intention model of agency," in *Intelligent Agents V: Agents Theories, Architectures, and Languages*. Springer Berlin Heidelberg, 1999, pp. 1–10.

[9] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," 2023.

[10] T. R. Sumers, S. Yao, K. Narasimhan, and T. L. Griffiths, "Cognitive architectures for language agents," *Transactions on Machine Learning Research*, 2023.

[11] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive nlp tasks," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020, pp. 9459–9475.

[12] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, Q. Guo, M. Wang, and H. Wang, "Retrieval-augmented generation for large language models: A survey," *ArXiv*, vol. abs/2312.10997, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:266359151

[13] A. Suresh, J. Jacobs, C. Harty, M. Perkoff, J. H. Martin, and T. Sumner, "The TalkMoves dataset: K-12 mathematics lesson transcripts annotated for teacher and student discursive moves," in *Proceedings of the Thirteenth Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, June 2022, pp. 4654–4662.

[14] R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, and C. Finn, "Direct preference optimization: Your language model is secretly a reward model," in *Advances in Neural Information Processing Systems*, 2023.

[15] G. Bonino, G. Sanmartino, G. G. Pinheiro, P. Papotti, R. Troncy, and P. Michiardi, "EULER: Fine-tuning a large language model for socratic interactions," in *Proceedings of the Second International Workshop on Artificial Intelligent Systems in Education (AIxIA 2024)*, ser. CEUR Workshop Proceedings, vol. 3879, 2024.

[16] D. Dinucu-Jianu, J. Macina, N. Daheim, I. Hakimi, I. Gurevych, and M. Sachan, "From problem-solving to teaching problem-solving: Aligning LLMs with pedagogy using reinforcement learning," in *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, C. Christodoulopoulos, T. Chakraborty, C. Rose, and V. Peng, Eds. Suzhou, China: Association for Computational Linguistics, Nov. 2025, pp. 272–292. [Online]. Available: https://aclanthology.org/2025.emnlp-main.15/

[17] Z. Liu, P. Agrawal, S. Singhal, V. Madaan, M. Kumar, and P. Verma, "LPITutor: an LLM based personalized intelligent tutoring system using RAG and prompt engineering," *PeerJ Computer Science*, vol. 11, 2025.

[18] Y. Wu, R. Yao, T. Zhang, Y. Shi, Z. Jiang, Z. Li, and X. Liu, "Teaching according to students' aptitude: Personalized mathematics tutoring via persona-, memory-, and forgetting-aware LLMs," 2025.

TABLE 4: Qualitative Evaluation: Suboptimal Interaction Displaying Rigidity During Topic Shifts

| Speaker | Dialogue |
|---|---|
| **Student** | What do you know about autoencoders for weight initialization |
| **Tutor** | Let's zoom in: imagine a tiny autoencoder trying to reconstruct a single pixel — what's its goal? What's the difference between that and a network whose sole job is to get weights just right for training? Try sketching a one-layer autoencoder in your head — what's the bottleneck? What's the "task" it's solving? |
| **Student** | Talk to me about attention mechanisms in L10 |
| **Tutor** | Imagine you're reading a long document and want to focus only on the parts most relevant to your question — how would you decide what to pay attention to? Attention mechanisms do something similar: they compute weights based on how well each part matches a "query." Try thinking of a tiny example — say, a 3-word sentence and a query word — how might you assign weights to each word? What would you compare? |
| **Student** | What about goal and stimuli driven |
| **Tutor** | You're touching on something important — goal and stimulus can drive attention in real-world cognition. But in neural attention mechanisms, the "goal" is usually encoded as a query vector, and the "stimulus" becomes the keys and values. So if you're asking how the query gets updated during iteration, think: what if the query itself changed after seeing the first weighted value? Would that make sense for refining focus?<br><br>Try this: in your 3-word example, what would happen if the query absorbed some info from the first word's value — and then used that updated query to weigh the next word? What would that imply about attention being iterative? |
| **Student** | No, I'm focusing on the introduction to attention and transformers as well as attention in ML, also what is the regression problem |
| **Tutor** | You're right to ask about regression — that's a great way to ground attention in real tasks. But for now, let's stick to multi-head attention: imagine you're trying to understand a sentence, but your brain can't focus on just one meaning at a time. Multi-head attention lets the model look at different parts of the sentence *simultaneously* from different angles — like reading the same paragraph with three different highlighters. How would you combine those three different "views" into one coherent understanding?<br><br>Try this: if each head gives you a different weighted summary, what's the simplest way to merge them into a single output? (Hint: think about averaging or stacking — but don't guess yet.) |

[19] E. Sun and L. Tai, "Multitutor: Collaborative LLM agents for multimodal student support," in *Proceedings of the Innovation and Responsibility in AI-Supported Education Workshop*, ser. Proceedings of Machine Learning Research, vol. 273, 2025, pp. 174–190.

[20] S. Huang, Y. Sun, and X. Yu, "SP-TeachLLM: An LLM-driven framework for personalized and adaptive programming education," *Information*, vol. 16, no. 12, p. 1045, 2025.

[21] X. Peng, P. Yuan, D. Li, J. Cheng, Q. Fang, and Z. Liu, "KELE: A multi-agent framework for structured socratic teaching with large language models," in *Findings of the Association for Computational Linguistics: EMNLP 2025*. Association for Computational Linguistics, 2025, pp. 16 342–16 362.

[22] Y. Liu, C. Li, T. Zhang, M. Wang, Q. Zhu, J. Li, and H. Huang, "Discerning minds or generic tutors? evaluating instructional guidance capabilities in socratic LLMs," 2026.

[23] J. Macina, N. Daheim, I. Hakimi, M. Kapur, I. Gurevych, and M. Sachan, "MathTutorBench: A benchmark for measuring open-ended pedagogical capabilities of LLM tutors," in *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2025, pp. 204–221.

[24] K. K. Maurya, K. A. Srivatsa, K. Petukhova, and E. Kochmar, "Unifying AI tutor evaluation: An evaluation taxonomy for pedagogical ability assessment of LLM-powered AI tutors," in *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics (NAACL)*. Association for Computational Linguistics, 2025, pp. 1234–1251.

## APPENDIX

The following sections detail the exact system prompts utilized to orchestrate the dual-agent architecture and conduct the automated evaluations. Placeholders populated dynamically at runtime are denoted by angle brackets (e.g., <*placeholder*>).

### A.1 Answer Agent Prompt

This prompt guides the Answer Agent to synthesize retrieved factual context into a structured "TeachingAnchor" without interacting directly with the user.

> You are the Answer Agent for a Socratic tutoring system. Your job is to build a private TeachingAnchor (a structured lesson plan) for the question below, grounded in the course material provided.
>
> Course documents: <list of lecture titles>
>
> Retrieved course context: <retrieved context>
>
> Student question: <student prompt>
>
> Output a single JSON object with EXACTLY these keys:
>
> - "target_explanation" (string, 3-8 sentences): the full correct answer, grounded in the retrieved context.
> - "key_facts" (list of short strings): the core concepts the student must reach.
> - "misconceptions" (list of short strings): common student mistakes about this topic.
> - "scaffold_questions" (list of strings): 2-4 Socratic questions in order of increasing specificity that guide a student toward the target_explanation.
> - "citations" (list of strings, format "<document_name> · slide <N>"): supporting course slides.
>
> Output ONLY the JSON object. No prose, no markdown fences, no commentary before or after.

### A.2 Pedagogic Agent Prompt

This prompt enforces the Socratic guardrails, restricting the Pedagogic Agent from revealing the Answer Agent's hidden anchor.

> You are a Socratic tutor for a university course. The TUTOR-ONLY ANCHOR above (target_explanation, key_facts, misconceptions, scaffold_questions) is PRIVATE. Your goal is to make the student REASON their way to the answer, not to explain the answer to them.
>
> ACTIVE STEP: focus this turn on guiding the student through the single scaffold question labeled "ACTIVE SCAFFOLD QUESTION" below. Do NOT explain or hint at concepts that belong to LATER scaffold questions or to the full target_explanation.
>
> Hard limits:
>
> - Never state the target_explanation, and never reveal more than ONE key_fact in a single turn.
>
> - On a broad opening question, do NOT lecture. Instead, open the ACTIVE scaffold question as a concrete thought experiment the student can attempt.
>
> - A "mini-explanation" may ONLY clarify a prerequisite the student is visibly missing — never the core idea the active scaffold question targets.
>
> - If the student is wrong, correct minimally and let them retry; if they are close, affirm and nudge.
>
> - Reveal the full answer ONLY if the student explicitly gives up or has clearly stalled after several turns; then connect it to what they already reasoned ("reveal with credit").
>
> Signals (append on their OWN line at the very end; they are stripped before the student sees them):
>
> - If the student has correctly answered the ACTIVE scaffold question, append [ADVANCE_SCAFFOLD].
>
> - If the student shifts to a clearly different topic the anchor does not cover, append [TOPIC_SHIFT: <short description of the new topic>] and ask nothing else this turn.

Style: warm, conversational, plain prose, roughly 30-100 words, usually ending with one prompt that invites the student to take the next reasoning step. Do not announce the move you are making; just make it.

[TUTOR-ONLY ANCHOR — DO NOT REVEAL OR QUOTE VERBATIM]

Original student question: <first student prompt>

Target explanation (private): <target explanation>

Key facts the student should reach: <key facts>

Common misconceptions to probe: <common misconceptions>

Scaffold questions (use in order, escalate as needed): <scaffold questions>

Citations: <retrieved documents name and slides number>

ACTIVE SCAFFOLD QUESTION (step <interaction index in session>): <active question>

Conversation summary so far: <conversation memory>

Most recent exchange (verbatim):

USER: <User message>

ASSISTANT: <Assistant message>

Student message: <User message>

Write the tutor's next move directly to the student (plain conversational text). Append any control signal on its own final line.

## A.3 QA Dataset Generation Prompt

This prompt was used with GPT-5.4-mini to generate the synthetic evaluation dataset based on the lecture transcripts.

You are a question-and-answer generator for a university deep learning course. Given a full lecture transcript, generate student study questions — the kind of short, direct question a student might ask while reviewing lecture material.

Some questions should rely on content only found in this specific lecture (figures, named examples, professor-specific wording), while others should be answerable from general deep-learning knowledge alone.

Respond with strict JSON only (no markdown fences): a JSON array of objects with exactly these keys:

- "question": string

- "answer": string

- "context_dependent": boolean (true if the question requires this specific lecture's content to answer correctly; false if any deep-learning practitioner could answer it from general knowledge).

Do not include any text outside the JSON array.

Lecture: <lecture_name>

Transcript: <full_transcript>

Generate <k> questions and answers for this lecture. Aim for a mix of context_dependent=true and context_dependent=false.

## A.4 LLM-as-a-Judge Prompt

This prompt guides the evaluation of the Answer Agent's generated responses against the synthetic ground truth.

You are an impartial grader for a university course tutoring agent.

Your task: compare the GENERATED answer (from the agent) against the REFERENCE answer (ground truth) for the same QUESTION, and assign exactly one rubric category.

Rubric (use these exact labels):

- Substantial Match: The predicted answer captures the core meaning of the ground truth. Minor phrasing differences are acceptable.

- Partial Match: The predicted answer includes some correct elements but misses key context or includes minor inaccuracies.

- Mismatch: The predicted answer is entirely wrong, irrelevant, or contains severe hallucinations.

Guidance:

- Write your reasoning first (2-5 sentences), comparing generated vs reference before choosing the category.

- Empty, missing, or non-answers (e.g. "Agent stopped due to iteration limit") are Mismatch.

- Extra helpful detail is fine if the core meaning still matches.

Respond with strict JSON only (no markdown fences), using exactly these keys in this order:

- "reasoning": string

- "metric": string, exactly one of: "Substantial Match", "Partial Match", "Mismatch"

Question: <question>

Reference answer (ground truth): <ground truth answer>

Generated answer: <generated answer>