

Designing an Autonomous Learning Agent with Checkpoint Verification and Feynman Pedagogy

A LangGraph Implementation for Structured Learning Pathways and Adaptive Simplification

Project Objectives

The main goal is to build an **autonomous AI learning agent** using the **LangGraph** framework. This agent will provide a personalized and structured tutoring experience.

Key objectives include:

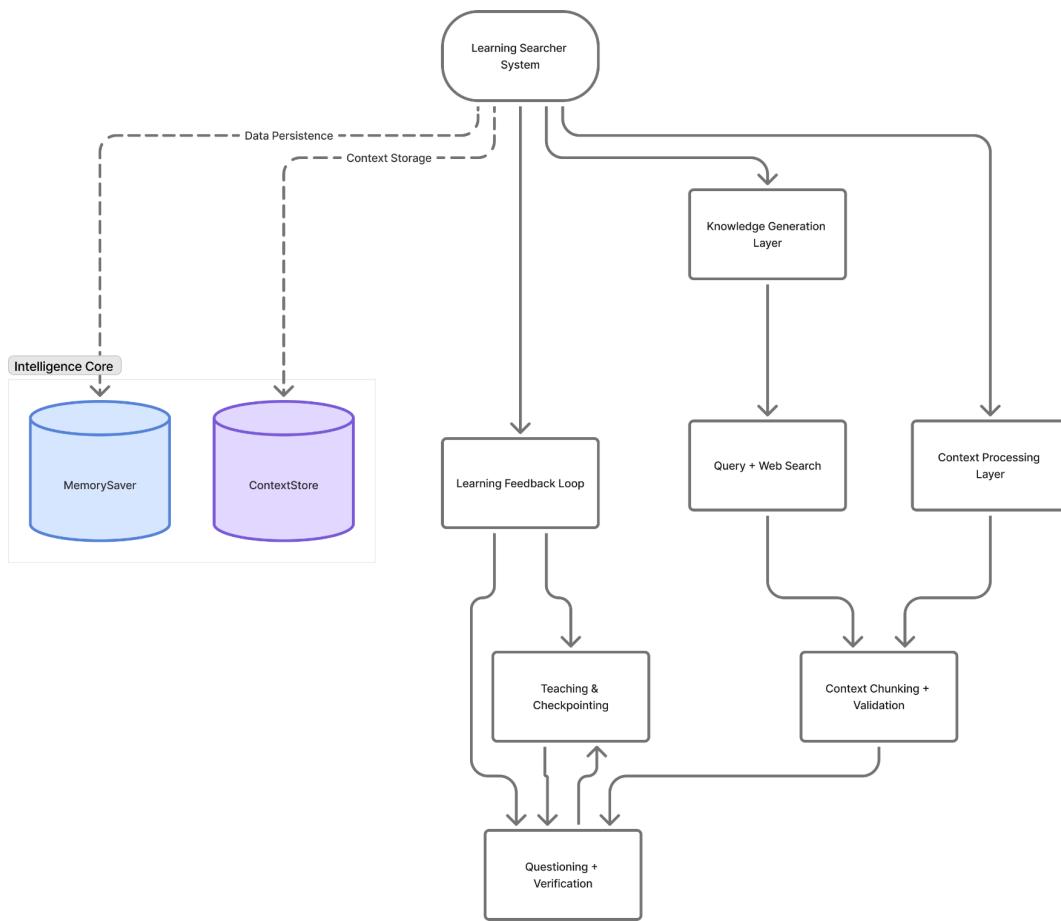
1. **Structured Guidance:** Implement a clear learning path by guiding users through sequential **checkpoints**, each focusing on a specific part of a topic.
2. **Flexible Content:** Utilize both the learner's own **provided notes** and information dynamically retrieved via **web search** as the basis for teaching and assessment.
3. **Rigorous Assessment:** Automatically generate relevant questions at each checkpoint and quantitatively **verify the learner's understanding** against a set threshold (e.g., 70%) before allowing progression.
4. **Adaptive Simplification:** When understanding is insufficient, employ the **Feynman Technique** to re-explain complex concepts using simpler language and analogies.
5. **Mastery-Based Progression:** Ensure deep learning by strictly **enforcing successful completion** of one checkpoint before advancing the learner to the next.
6. Develop a user **interface enabling** direct interaction between users and the application.

Project Workflow: AI Learning Agent

- **Define Checkpoint:** The workflow starts. The agent identifies the current learning checkpoint and its specific goals based on the learner's progress.
- **Gather Context:** The agent collects relevant learning materials. It first looks for suitable user-provided notes, then performs web searches if necessary.
- **Validate Context:** The agent checks if the gathered materials adequately cover the checkpoint's objectives. If not, it repeats the gathering step.
- **Process Context:** Sufficient context is processed (e.g., chunked, embedded) for efficient use during the verification stage.
- **Generate Questions:** The agent creates specific questions based on the processed context to test understanding of the checkpoint's goals.

- **Assess Learner:** Questions are presented to the learner, their answers are received, and the agent calculates an understanding score against the context.
- **Evaluate Score:** The agent compares the learner's score against the 70% threshold.
- **Apply Feynman Teaching (If Needed):** If the score is below 70%, the agent identifies weak areas, generates a simplified Feynman-style explanation for those concepts, presents it, and then returns to the **Generate Questions** step for reassessment.
- **Mark Complete & Progress:** If the score is 70% or higher, the agent marks the checkpoint as complete. It then checks if more checkpoints exist in the learning path.
- **Continue or End:** If a next checkpoint exists, the workflow loops back to **Define Checkpoint**. If not, the learning path is finished, and the workflow ends

Architecture Diagram



Project Components

- **LangGraph State Graph:** The core engine orchestrating the entire learning workflow, managing state and transitions between different learning stages.
- **Checkpoint Definition Module:** Defines the structured learning milestones, including their specific topics, learning objectives, and criteria for success.
- **Context Management Module:** Gathers learning materials (from user notes or web search), validates relevance, processes text (chunking/embedding), and stores it for retrieval.
- **Web Search Integration:** A tool allowing the agent to dynamically search the web for relevant information based on the current learning checkpoint.
- **Question Generation Module:** Creates targeted questions based on the processed context to specifically assess understanding of the checkpoint's objectives.
- **Understanding Verification Module:** Evaluates learner's answers against the context, calculates a quantitative understanding score, and checks if it meets the predefined threshold (e.g., 70%).
- **Feynman Teaching Module:** Activated when the understanding threshold isn't met; generates simplified explanations using analogies and clear language for concepts the learner found difficult.
- **LLM Integration:** The core Large Language Model used for reasoning, generation, and evaluation tasks throughout the workflow.
- **Learner Interface:** The front-end or method through which the learner interacts with the agent (e.g., text input/output).

Tech Stack

- **Python:** The primary programming language for development.
- **LangGraph:** The core framework used to build the stateful, graph-based learning workflow.
- **LangChain:** Provides essential components for LLM integrations, tool creation, prompt management, text splitting, and embedding interfaces.
- **LLM Provider API & Library:** Specific integration library for the chosen Large Language Model (e.g., `langchain-google-genai` for Gemini, `langchain-openai` for OpenAI models, `langchain-anthropic` for Claude).
- **Web Search API & Library:** Integration for dynamic context retrieval (e.g., Tavily Search API with `langchain-community` tools, SerpApi, DuckDuckGo Search).
- **Embedding Model Library:** Integration for the model used to create embeddings for context storage/retrieval (via LangChain wrappers).
- **Vector Store Library (Optional but Recommended):** Library for efficient storage and retrieval of context embeddings (e.g., `faiss-cpu`, `chromadb`, `pinecone-client`).
- **python-dotenv:** Utility for managing API keys and environment variables securely.
- **Jupyter Notebooks (Development):** Used for iterative development, testing, and demonstrating the agent's components and flow.

Milestone 1: Checkpoint Structure & Context Gathering (Weeks 1-2)

This milestone focuses on setting up the basic workflow and getting the right information into the system.

Goals:

- Set up the project environment (Python, LangGraph, LLM integration, web search API).
- Define the data structure for **learning checkpoints** (topic, objectives, success criteria).
- Implement the initial LangGraph nodes for **starting a checkpoint** and **gathering context** (prioritizing user notes, falling back to web search).
- Implement the **context validation** logic: Does the gathered information seem relevant to the checkpoint's objectives?

Evaluation Plan (End of Week 2):

- **Metric:** Context Relevance Score.
- **Method:** For 5-10 different checkpoints, manually trigger the context gathering. Review the gathered text (from notes or web search). Score its relevance (e.g., 1-5 scale) to the checkpoint's defined objectives.
- **Tool:** Manual review and **LangSmith Tracing** (to see search queries and retrieved documents).
- **Success Criteria:** Context gathered achieves an average relevance score of 4/5 or higher across test checkpoints. Agent correctly identifies and attempts to re-fetch context if initial results are irrelevant.

Milestone 2: Context Processing & Initial Verification (Weeks 3-4)

This milestone focuses on preparing the context and implementing the core assessment loop.

Goals:

- Implement the **context processing** node (chunking, embedding, storing in a temporary vector store for the session).
- Implement the **question generation** node: Based on the processed context, generate 3-5 relevant questions per checkpoint.
- Implement the initial **understanding verification** node: Evaluate simulated learner answers against the context and calculate a percentage score.

- Implement the basic conditional logic: If score $\geq 70\%$, proceed; if $< 70\%$, temporarily halt (Feynman node is placeholder).

Evaluation Plan (End of Week 4):

- **Metrics:** Question Relevance & Scoring Accuracy.
- **Method:** For several checkpoints with processed context:
 - Review the generated questions. Are they relevant to the context and checkpoint objectives?
 - Provide predefined "good" and "bad" answers to the questions. Does the scoring logic correctly assign high scores to good answers and low scores to bad ones, consistently meeting the 70% threshold logic?
- **Tool:** Manual review, unit tests for scoring, [LangSmith Tracing](#).
- **Success Criteria:** $>80\%$ of generated questions are relevant. Scoring logic correctly differentiates good/bad answers and applies the 70% threshold accurately in $>90\%$ of test cases.

Milestone 3: Feynman Teaching Implementation (Weeks 5-6)

This milestone implements the adaptive teaching mechanism.

Goals:

- Implement the **Feynman teaching module**:
 - Identify knowledge gaps based on incorrect answers from the verification step.
 - Generate simplified explanations (analogies, simple terms) for those specific gaps using the LLM.
- Integrate the Feynman node into the LangGraph workflow: Trigger it when the score is $< 70\%$.
- Implement the loop-back mechanism: After presenting the Feynman explanation, the workflow should return to the **question generation/verification** stage.

Evaluation Plan (End of Week 6):

- **Metrics:** Explanation Simplicity & Workflow Logic.
- **Method:** Force the workflow into the $< 70\%$ path for several checkpoints/concepts.
 - Manually review the generated Feynman explanations. Are they genuinely simpler than the original context? Do they use analogies? Do they avoid jargon?
 - Verify the workflow correctly loops back to re-assessment after the explanation.

- **Tool:** Manual review of LLM outputs, **LangSmith Tracing** (to confirm the graph correctly routes to Feynman node and then back to verification).
- **Success Criteria:** Feynman explanations are rated as "simpler" than original context in >80% of cases. The workflow correctly executes the Feynman loop in 100% of sub-threshold test cases.

Milestone 4: Integration & End-to-End Testing (Weeks 7-8)

This final milestone integrates all components and tests the complete learning journey.

Goals:

- Ensure seamless state transfer and logic flow between all nodes in the LangGraph.
- Implement the logic for progressing through **multiple sequential checkpoints**.
- Conduct end-to-end testing simulating a learner progressing through a short learning path (e.g., 3 checkpoints).
- (Optional) Connect to a basic front-end interface for learner interaction.

Evaluation Plan (End of Week 8):

- **Metric:** Successful Learning Path Completion.
- **Method:** Simulate a full learning session:
 - Start at Checkpoint 1. Provide answers to pass. Verify progression to Checkpoint 2.
 - At Checkpoint 2, provide answers to fail. Verify Feynman explanation is triggered. Provide answers to pass the re-assessment. Verify progression to Checkpoint 3.
 - At Checkpoint 3, pass. Verify the workflow ends correctly.
- **Tool:** End-to-end simulation, **LangSmith Tracing** (to monitor the full state transitions and logic flow across multiple checkpoints and loops).
- **Success Criteria:** The agent successfully guides the simulated learner through the multi-checkpoint path, correctly applying verification, Feynman teaching, and progression logic in >80% of end-to-end test runs.