


A close-up, low-angle shot of a bicycle chain and sprocket. The chain is made of metal links and is wrapped around a sprocket. The lighting is warm and focused, creating a sense of depth and texture. The background is blurred, showing more of the bike's frame and components.

Workshop: Prompt Chaining with LangGraph

Understanding and Implementing Prompt Chaining in AI Workflows

A large red circle on the left side of the slide, partially cut off by the edge.

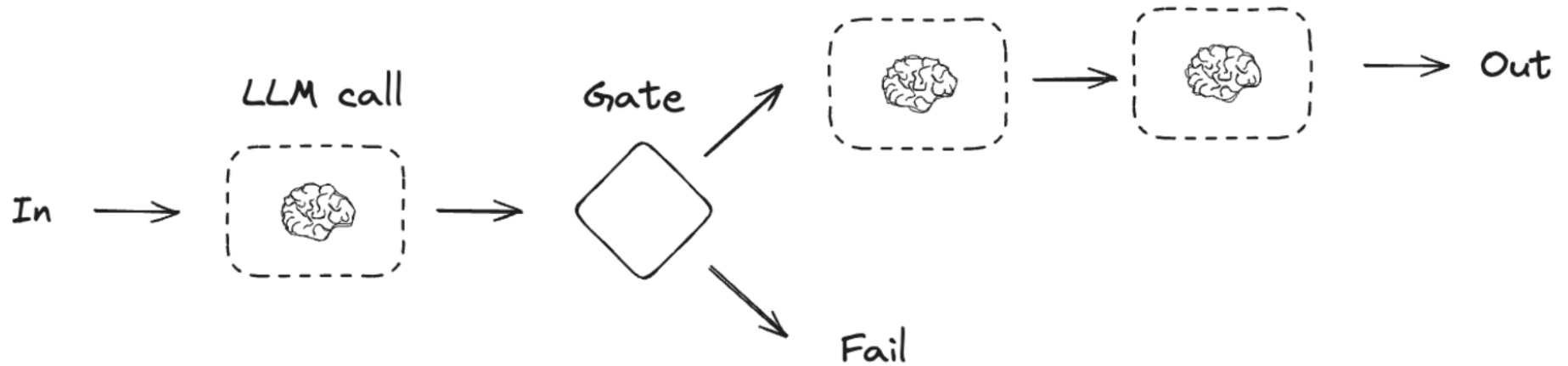
Introduction to Prompt Chaining

- What is Prompt Chaining?
 - Why is it useful in AI workflows?
 - Overview of LangGraph and LangChain
- 
- A decorative purple dashed line in the bottom right corner, consisting of several short, curved segments.

Prompt chaining

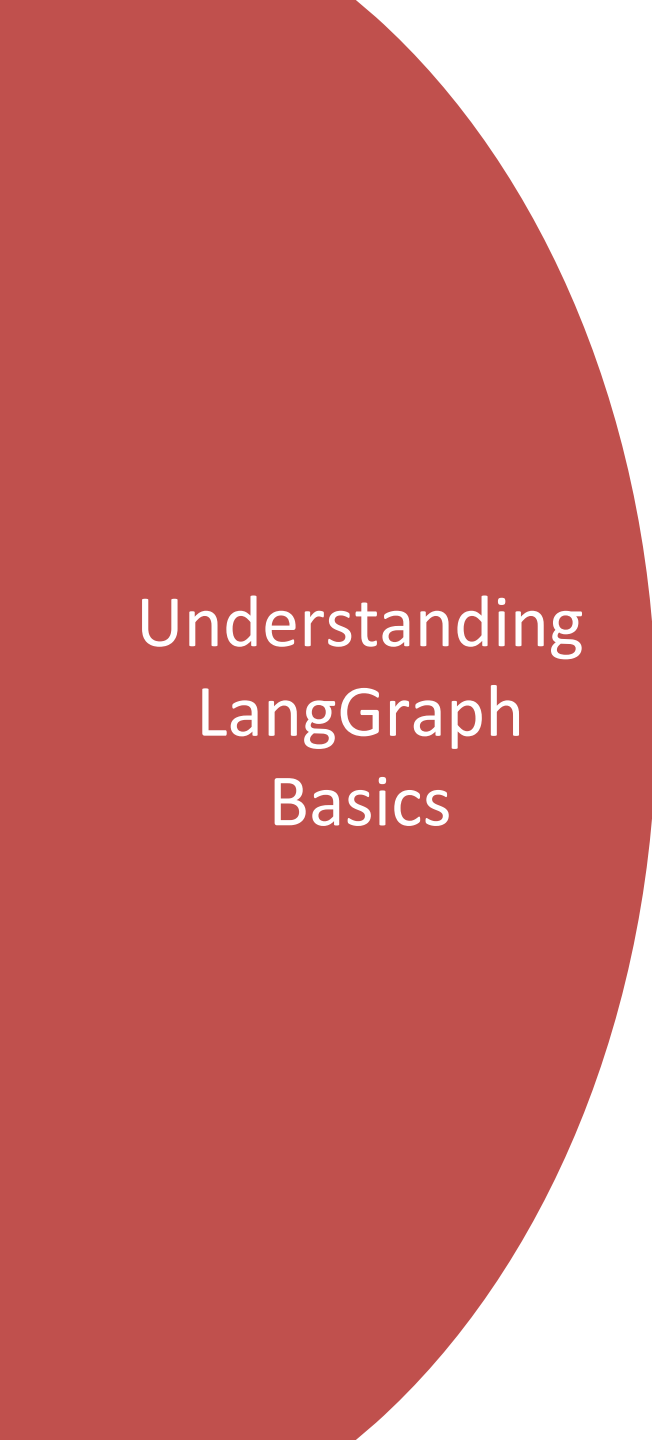
Prompt chaining decomposes a task into a sequence of steps, where each LLM call processes the output of the previous one. You can add programmatic checks (see "gate" in the diagram below) on any intermediate steps to ensure that the process is still on track

Prompt chaining





Setting Up the Environment

- Install dependencies:
- `pip install langchain langchain-openai langgraph pydantic python-dotenv`
- Setting up OpenAI API keys using `dotenv`
- Loading environment variables correctly




Understanding LangGraph Basics

- What is a StateGraph?
 - Defining states
 - Creating nodes and transitions
- 

A large red circle on the left side of the slide, partially cut off by the edge.

When to use this workflow

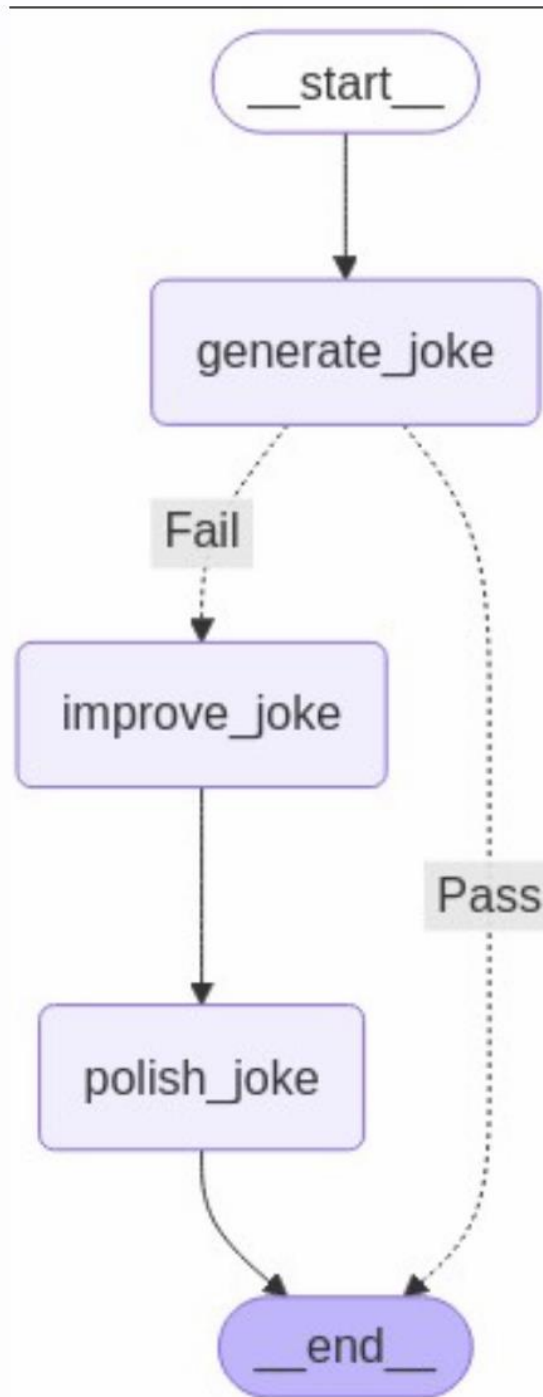
This workflow is ideal for situations where the task can be easily and cleanly decomposed into fixed subtasks. The main goal is to trade off latency for higher accuracy, by making each LLM call an easier task.

A decorative purple dashed line in the bottom right corner, consisting of several curved segments.

Building a Simple Prompt Chain


- Creating a basic LangGraph model
- Adding nodes and defining logic
- Connecting edges and compiling the graph








Running and Debugging

- Executing the model with sample inputs
 - Handling errors and debugging common issues
 - Best practices for efficient prompt chaining
- 

A large red circle on the left side of the slide, partially cut off by the edge.

Q&A and Next Steps

- Open discussion and participant queries
 - Resources for further learning
 - Hands-on challenges for practice
- 
- A decorative purple dashed line in the bottom right corner, consisting of several curved segments.

LLMs can sometimes work simultaneously on a task and have their outputs aggregated programmatically. This workflow, parallelization, manifests in two key variations: Sectioning: Breaking a task into independent subtasks run in parallel. Voting: Running the same task multiple times to get diverse outputs.

- When to use this workflow: Parallelization is effective when the divided subtasks can be parallelized for speed, or when multiple perspectives or attempts are needed for higher confidence results. For complex tasks with multiple considerations, LLMs generally perform better when each consideration is handled by a separate LLM call, allowing focused attention on each specific aspect.

LLM calls



Aggregator



Out

In

