esurovtsev / langgraph-intro

Code | Issues 2 | Pull requests | Actions | Projects | Security | Insights

An introductory guide to LangGraph, showcasing step-by-step development from basic chains to fully functional AI agents with memory. This repository provides Python notebooks to help you understand and experiment with LangGraph's core capabilities.

31 stars · 9 forks · 1 watching · Branches · Activity · Tags

Public repository

23 Branches · 0 Tags · Go to file · Add file · Code

This branch is 16 commits behind `main`.

esurovtsev Add multi-agent research pipeline example with team formation and rep...

57fc764 · 2 months ago

images	Add multi-agent research pipeline wit...	2 months ago
studio	Add MapReduce agent workflow with...	3 months ago
.gitignore	feat: Implement persistent agent me...	4 months ago
01_chain.ipynb	feat(langgraph): implement basic gra...	5 months ago
02_router.ipynb	added missing inport	4 months ago
03_agent.ipynb	cell outputs removed	5 months ago
04_agent-memory.ipynb	feat: Add memory capabilities to fina...	5 months ago
05_chatbot-messages.ipynb	feat: add chatbot messages notebook	5 months ago
06_chatbot-summarization.ipynb	Add chatbot with message summariz...	5 months ago
07_chatbot-persistence.ipynb	feat: Implement persistent agent me...	4 months ago
08_streaming.ipynb	readme.md updated with the informa...	4 months ago
09_breakpoints.ipynb	Implement human-in-loop approval ...	4 months ago
10_human-feedback.ipynb	feat(examples): Add human feedback ...	4 months ago
11_dynamic-breakpoints.ipynb	updated readme.md	4 months ago
12_replay-fork-state.ipynb	feat: Add lesson on replaying and fork...	4 months ago

13_parallel-ai-execution.ipynb	Add parallel execution lesson and up...	3 months ago
14_ai-agent-subgraphs.ipynb	Add subgraph implementation for AI ...	3 months ago
15_mapreduce-financial-agent....	Add MapReduce agent workflow with...	3 months ago
16_multi-agent-research.ipynb	Add multi-agent research pipeline wit...	2 months ago
README.md	Add multi-agent research pipeline exa...	2 months ago
docker-compose.yml	feat: Implement persistent agent me...	4 months ago
requirements.txt	Add parallel execution lesson and up...	3 months ago
tools_with_summary.ipynb	updared output	4 months ago

LangGraph Introduction

This repository contains a series of Jupyter notebooks that demonstrate how to work with LangGraph - a powerful framework for building structured agents and workflows using LangChain. The project provides a hands-on approach to learning LangGraph concepts, from basic principles to advanced implementations.

Overview

The tutorial is structured in a progressive manner, with each notebook building upon concepts introduced in previous ones. Each concept is accompanied by both code examples, making it easier to understand the underlying principles.

Contents

1. **Chain Basics** (`01_chain.ipynb`)
   - Introduction to basic graph concepts
   - Understanding state management
   - Implementation of simple chains
   - LangGraph Intro - Learn How to Build AI Agents with Tools and Graphs
2. **Router Implementation** (`02_router.ipynb`)
   - Working with routers in LangGraph
   - Flow control and decision making
   - LangGraph Intro - Build Router AI Agents with LangGraph Tools and LLMs
3. **Agent Creation** (`03_agent.ipynb`)
   - Building agents with LangGraph
   - Agent architecture and components

- LangGraph Intro - Build Autonomous AI Agents with ReAct and LangGraph Tools

4. **Agent with Memory** (`04_agent-memory.ipynb`)
    - Implementing memory capabilities
    - State persistence and management
    - Advanced agent patterns
    - LangGraph Intro Build AI Agents with Memory Using LangGraph and LLMs
5. **Chatbot Message Management** (`05_chatbot-messages.ipynb`)
    - Managing conversation context and history
    - Token usage optimization
    - Message state handling
    - Efficient message trimming strategies
    - LangGraph Intro Optimize Chatbot Messages with Memory and Schema Building Blocks for AI Agents
6. **Chatbot Summarization** (`06_chatbot-summarization.ipynb`)
    - Advanced memory optimization techniques
    - Implementing conversation summarization
    - Dynamic context management
    - Memory compression strategies
    - LangGraph Intro - Optimize Chatbot Memory with Summarization: Smarter AI Agents with LangGraph
7. **Chatbot Persistence** (`07_chatbot-persistence.ipynb`)
    - Implementing MongoDB for long-term memory storage
    - LangGraph Studio integration for workflow visualization
    - Persistent state management across conversations
    - Database configuration and connection handling
    - LangGraph Intro Persist AI Agent Memory with MongoDB and LangGraph Studio
8. **Streaming & API Integration** (`08_streaming.ipynb`)
    - LangGraph Studio as local server and API gateway
    - Real-time state streaming techniques (updates/values/messages)
    - API integration patterns for existing workflows
    - Visualizing node-to-node state transitions
    - Hybrid execution (notebook vs studio environments)
    - Preparing for human-in-the-loop workflows
    - LangGraph Intro Streaming AI Agent State and API Calls with LangGraph Studio
9. **Human-in-the-Loop Workflows** (`09_breakpoints.ipynb`)
    - Implementing execution breakpoints for user approval
    - State inspection and modification during pauses
    - Flow resumption using null parameters
    - Studio integration for visual breakpoint management

- Infinite tool call support with approval checks
- Hybrid execution patterns (CLI vs Studio)
- API parameter handling for flow continuation
- LangGraph Intro - Human in the Loop: Breaking and Resuming AI Agent Execution with LangGraph

10. **Advanced Human Feedback** (`10_human-feedback.ipynb`)
    - State manipulation after breakpoints
    - Interactive feedback collection and processing
    - Message content replacement strategies
    - User feedback node patterns
    - Graph state modification techniques
    - LangGraph Intro - Human-in-the-Loop: Collecting and Processing User Feedback in AI Agent Workflows
11. **Dynamic Breakpoints** (`11_dynamic-breakpoints.ipynb`)
    - Implementing conditional execution breaks
    - Intent validation and flow control
    - Dynamic state inspection during breaks
    - Message content validation and correction
    - Graph resumption with modified state
    - Exception-based breakpoint triggers
    - Studio and API integration for dynamic control
    - LangGraph Intro - Human-in-the-Loop: Dynamic Breakpoints for AI Agent Control with LangGraph
12. **State Replay and Forking** (`12_replay-fork-state.ipynb`)
    - Understanding checkpoint-based state management
    - Accessing thread state history and snapshots
    - State replay from historical checkpoints
    - Graph forking and execution branching
    - Thread-based conversation management
    - SDK and API implementation patterns
    - Replay (event playback) vs Fork (real execution)
    - LangGraph Intro - Human-in-the-Loop: Replaying and Forking AI Agent State with LangGraph
13. **Parallel AI Execution** (`13_parallel-ai-execution.ipynb`)
    - Implementing concurrent node execution
    - State conflict resolution with reducers
    - Super step transaction management
    - Synchronizing parallel execution flows
    - Conditional branching in parallel workflows
    - Building multi-source AI assistants
    - Performance optimization techniques
    - LangGraph Intro - Running AI Agent Tasks in Parallel with LangGraph

14. **AI Agent Subgraphs** (`14_ai-agent-subgraphs.ipynb`)
    - Implementing subgraphs as modular components
    - Registering subgraphs as nodes in parent graphs
    - Query classification and optimization techniques
    - Multi-source information retrieval (web search and Wikipedia)
    - State synchronization between subgraphs
    - Structured response generation with source attribution
    - Complex workflow orchestration with specialized subgraphs
    - LangGraph Intro - Structuring AI Agent Workflows with Subgraphs in LangGraph
15. **MapReduce for Parallel Processing** (`15_mapreduce-financial-agent.ipynb`)
    - Implementing the MapReduce pattern for AI agent tasks
    - Dynamic node generation during runtime execution
    - Parallel data processing with mapped nodes
    - State aggregation with reducer functions
    - Building a financial advisor agent with stock analysis
    - Yahoo Finance integration for real-time data
    - Structured output generation with rankings and recommendations
    - Visualizing dynamic parallel workflows in LangGraph Studio
    - LangGraph Intro – Scaling AI Agents with MapReduce in LangGraph
16. **Multi-Agent Research Pipeline** (`16_multi-agent-research.ipynb`)
    - Building complex multi-agent research workflows
    - Team formation with human-in-the-loop feedback
    - Structured output for analyst profile generation
    - Dialogue-based expert interview subgraphs
    - Parallel execution using map-reduce pattern
    - External search integration (web and Wikipedia)
    - Synthesizing research outputs into comprehensive reports
    - Coordinated autonomous multi-agent system design
    - LangGraph Intro - Multi Agent Research Pipelines and Report Writing with LangGraph

Prerequisites

- Python 3.8+
- Jupyter Notebook/Lab environment

Installation

1. Clone the repository:

```
git clone [repository-url]
```

```
cd langgraph-intro
```

2. Create and activate a virtual environment:

```
python -m venv venv
source venv/bin/activate  # On Windows use: venv\Scripts\activate
```

3. Install dependencies:

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

Dependencies

The project relies on the following main packages:

- `langgraph` - Core framework for building graph-based workflows
- `langchain_openai` - LangChain OpenAI integration
- `langchain_core` - Core LangChain functionalities
- `langchain_community` - Community integrations for LangChain
- `python-dotenv` - Environment variable management

- `yfinance` - Financial data integration (for examples)
- `typing_extensions` - Type hints and annotations
- `wikipedia` - Python library for accessing Wikipedia content

Environment Setup

1. Create `.env` files in these locations:
   - Root directory:

     ```
     OPENAI_API_KEY=your_api_key_here
     ```

   - Studio directory (if using LangGraph Studio):

     ```
     OPENAI_API_KEY=your_api_key_here
     ```

2. Add your API keys to both locations

Usage

1. Ensure your virtual environment is activated
2. Launch Jupyter Notebook:

```
jupyter notebook
```

3. Navigate through the notebooks in numerical order
4. Each notebook contains detailed explanations and executable code examples

Video Tutorials

Each notebook has an accompanying video tutorial that walks through the concepts in detail. The video links are provided in each section above.

Contributing

Feel free to submit issues and enhancement requests!

Acknowledgments

- LangChain team for providing the foundational frameworks
- OpenAI for API integration capabilities

Releases

No releases published

Packages

No packages published

Languages

● **Jupyter Notebook** 89.2% ● **Python** 10.8%