Route between multiple runnables | Langchain

Skip to main content LangChainDocsUse casesIntegrationsAPIMoreCommunityTutorialsContributingAlso by LangChainChat our docsLangSmithLangChain HubLangServePython DocsSearchCTRLKGet startedIntroductionInstallationQuickstartLangChain Expression LanguageInterfaceHow toRoute between multiple runnablesCancelling requestsUse RunnableMapsCookbookWhy use LCEL?LangChain Expression Language (LCEL)ModulesModel I/ORetrievalChainsMemoryAgentsCallbacksModulesSecurityGuidesEcosystemLangChain Expression LanguageHow toRoute between multiple runnablesOn this pageRoute between multiple RunnablesThis notebook covers how to do routing in the LangChain Expression Language.Routing allows you to create non-deterministic chains where the output of a previous step defines the next step. Routing helps provide structure and consistency around interactions with LLMs.There are two ways to perform routing:Using a RunnableBranch.Writing custom factory function that takes the input of a previous step and returns a runnable. Importantly, this should return a runnable and NOT actually execute.We'll illustrate both methods using a two step

sequence where the first step classifies an input question as being about LangChain, Anthropic, or Other, then routes to a corresponding prompt chain.Using a RunnableBranchA RunnableBranch is initialized with a list of (condition, runnable) pairs and a default runnable. It selects which branch by passing each condition the input it's invoked with. It selects the first condition to evaluate to True, and runs the corresponding runnable to that condition with the input.If no provided conditions match, it runs the default runnable.Here's an example of what it looks like in action:import { ChatAnthropic } from "langchain/chat_models/anthropic";import { PromptTemplate } from "langchain/prompts";import { StringOutputParser } from "langchain/schema/output_parser";import { RunnableBranch, RunnableSequence } from "langchain/schema/runnable";const promptTemplate = PromptTemplate.fromTemplate(`Given the user question below, classify it as either being about \`LangChain\`, \`Anthropic\`, or \`Other\`. Do not respond with more than one word.<question>{question}</question>Classification:`);const model = new ChatAnthropic({ modelName: "claude-2",});const classificationChain = RunnableSequence.from([promptTemplate, model, new StringOutputParser(),]);const classificationChainResult = await classificationChain.invoke({ question: "how do I call Anthropic?",});console.log(classificationChainResult);/* Anthropic*/const langChainChain = PromptTemplate.fromTemplate(`You are an expert in langchain.Always answer questions starting with "As Harrison Chase told me".Respond to the following question:Question: {question}Answer:`).pipe(model);const anthropicChain = PromptTemplate.fromTemplate(`You are an expert in anthropic. \Always answer questions starting with "As Dario Amodei told me". \Respond to the following question:Question: {question}Answer:`).pipe(model);const generalChain = PromptTemplate.fromTemplate(`Respond to the following question:Question: {question}Answer:`).pipe(model);const branch = RunnableBranch.from([[(x: { topic: string; question: string }) => x.topic.toLowerCase().includes("anthropic"), anthropicChain,], [(x: { topic: string; question: string }) => x.topic.toLowerCase().includes("langchain"),

langChainChain,], generalChain,]);const fullChain = RunnableSequence.from([{ topic: classificationChain, question: (input: { question: string }) => input.question, }, branch,]);const result1 = await fullChain.invoke({ question: "how do I use Anthropic?",});console.log(result1);/* AIMessage { content: ' As Dario Amodei told me, here are some tips for how to use Anthropic:\n' + '\n' + "First, sign up for an account on Anthropic's website. This will give you access to their conversational AI assistant named Claude. \n" + '\n' + "Once you've created an account, you can have conversations with Claude through their web interface. Talk to Claude like you would talk to a person, asking questions, giving instructions, etc. Claude is trained to have natural conversations and be helpful.\n" + '\n' + "You can also integrate Claude into your own applications using Anthropic's API. This allows you to build Claude's conversational abilities into chatbots, virtual assistants, and other AI systems you develop.\n" + '\n' + 'Anthropic is constantly working on improving Claude, so its capabilities are always expanding. Make sure to check their blog and documentation to stay up to date on the latest features.\n' + '\n' + 'The key is to interact with Claude regularly so it can learn from you. The more you chat with it, the better it will become at understanding you and having personalized conversations. Over time, Claude will feel more human-like as it accumulates more conversational experience.', additional_kwargs: {} }*/const result2 = await fullChain.invoke({ question: "how do I use LangChain?",});console.log(result2);/* AIMessage { content: ' As Harrison Chase told me, here is how you use LangChain:\n' + '\n' + 'First, think carefully about what you want to ask or have the AI do. Frame your request clearly and specifically. Avoid vague or overly broad prompts that could lead to unhelpful or concerning responses. \n' + '\n' + 'Next, type your question or request into the chat window and send it. Be patient as the AI processes your input and generates a response. The AI will do its best to provide a helpful answer or follow your instructions, but its capabilities are limited.\n' + '\n' + 'Keep your requests simple at first. Ask basic questions or have the AI summarize content or generate basic text. As you get more comfortable, you can try having the AI perform more complex tasks like answering

tricky questions, generating stories, or having a conversation.\n' + '\n' + "Pay attention to the AI's responses. If they seem off topic, nonsensical, or concerning, rephrase your prompt to steer the AI in a better direction. You may need to provide additional clarification or context to get useful results.\n" + '\n' + 'Be polite and respectful towards the AI system. Remember, it is a tool designed to be helpful, harmless, and honest. Do not try to trick, confuse, or exploit it. \n' + '\n' + 'I hope these tips help you have a safe, fun and productive experience using LangChain! Let me know if you have any other questions.', additional_kwargs: {} }*/const result3 = await fullChain.invoke({ question: "what is 2 + 2?",});console.log(result3);/* AIMessage { content: ' 4', additional_kwargs: {} }*/API Reference:ChatAnthropic from langchain/chat_models/anthropicPromptTemplate from langchain/promptsStringOutputParser from langchain/schema/output_parserRunnableBranch from langchain/schema/runnableRunnableSequence from langchain/schema/runnableUsing a custom functionYou can also use a custom function to route between different outputs. Here's an example:import { ChatAnthropic } from "langchain/chat_models/anthropic";import { PromptTemplate } from "langchain/prompts";import { StringOutputParser } from "langchain/schema/output_parser";import { RunnableSequence } from "langchain/schema/runnable";const promptTemplate = PromptTemplate.fromTemplate(`Given the user question below, classify it as either being about \`LangChain\`, \`Anthropic\`, or \`Other\`. Do not respond with more than one word.<question>{question}</question>Classification:`);const model = new ChatAnthropic({ modelName: "claude-2",});const classificationChain = RunnableSequence.from([promptTemplate, model, new StringOutputParser(),]);const classificationChainResult = await classificationChain.invoke({ question: "how do I call Anthropic?",});console.log(classificationChainResult);/* Anthropic*/const langChainChain = PromptTemplate.fromTemplate(`You are an expert in langchain.Always answer questions starting with "As Harrison Chase told me".Respond to the following question:Question:

{question}Answer:`).pipe(model);const anthropicChain = PromptTemplate.fromTemplate( `You are an expert in anthropic. \Always answer questions starting with "As Dario Amodei told me". \Respond to the following question:Question: {question}Answer:`).pipe(model);const generalChain = PromptTemplate.fromTemplate( `Respond to the following question:Question: {question}Answer:`).pipe(model);const route = ({ topic }: { input: string; topic: string }) => { if (topic.toLowerCase().includes("anthropic")) { return anthropicChain; } else if (topic.toLowerCase().includes("langchain")) { return langChainChain; } else { return generalChain; }};const fullChain = RunnableSequence.from([{ topic: classificationChain, question: (input: { question: string }) => input.question, }, route,]);const result1 = await fullChain.invoke({ question: "how do I use Anthropic?",});console.log(result1);/* AIMessage { content: ' As Dario Amodei told me, here are some tips for how to use Anthropic:\n' + '\n' + "First, sign up for an account on Anthropic's website. This will give you access to their conversational AI assistant named Claude. \n" + '\n' + "Once you've created an account, you can have conversations with Claude through their web interface. Talk to Claude like you would talk to a person, asking questions, giving instructions, etc. Claude is trained to have natural conversations and be helpful.\n" + '\n' + "You can also integrate Claude into your own applications using Anthropic's API. This allows you to build Claude's conversational abilities into chatbots, virtual assistants, and other AI systems you develop.\n" + '\n' + 'Anthropic is constantly working on improving Claude, so its capabilities are always expanding. Make sure to check their blog and documentation to stay up to date on the latest features.\n' + '\n' + 'The key is to interact with Claude regularly so it can learn from you. The more you chat with it, the better it will become at understanding you and having personalized conversations. Over time, Claude will feel more human-like as it accumulates more conversational experience.', additional_kwargs: {} }*/const result2 = await fullChain.invoke({ question: "how do I use LangChain?",});console.log(result2);/* AIMessage { content: ' As Harrison Chase told me, here is how you use LangChain:\n' + '\n' + 'First, think carefully about what you want to ask or

have the AI do. Frame your request clearly and specifically. Avoid vague or overly broad prompts that could lead to unhelpful or concerning responses. \n' + '\n' + 'Next, type your question or request into the chat window and send it. Be patient as the AI processes your input and generates a response. The AI will do its best to provide a helpful answer or follow your instructions, but its capabilities are limited.\n' + '\n' + 'Keep your requests simple at first. Ask basic questions or have the AI summarize content or generate basic text. As you get more comfortable, you can try having the AI perform more complex tasks like answering tricky questions, generating stories, or having a conversation.\n' + '\n' + "Pay attention to the AI's responses. If they seem off topic, nonsensical, or concerning, rephrase your prompt to steer the AI in a better direction. You may need to provide additional clarification or context to get useful results.\n" + '\n' + 'Be polite and respectful towards the AI system. Remember, it is a tool designed to be helpful, harmless, and honest. Do not try to trick, confuse, or exploit it. \n' + '\n' + 'I hope these tips help you have a safe, fun and productive experience using LangChain! Let me know if you have any other questions.', additional_kwargs: {} }*/const result3 = await fullChain.invoke({ question: "what is 2 + 2?",});console.log(result3);/* AIMessage { content: ' 4', additional_kwargs: {} }*/API Reference:ChatAnthropic from langchain/chat_models/anthropicPromptTemplate from langchain/promptsStringOutputParser from langchain/schema/output_parserRunnableSequence from langchain/schema/runnablePreviousInterfaceNextCancelling requestsUsing a RunnableBranchUsing a custom functionCommunityDiscordTwitterGitHubPythonJS/TSMoreHomepageBlogCopyright © 2023 LangChain, Inc.