int to PDF

Welcome to LangChain

Contents

LangChain is a framework for developing applications powered by language models. We believe that the most powerful and differentiated applications will not only call out to a language model via an API, but will also:

- *Be data-aware*: connect a language model to other sources of data
- *Be agentic*: allow a language model to interact with its environment

The LangChain framework is designed with the above principles in mind.
This is the Python specific portion of the documentation. For a purely conceptual guide to LangChain, see here. For the JavaScript documentation, see here.

Getting Started

Checkout the below guide for a walkthrough of how to get started using LangChain to create an Language Model application.

- Getting Started Documentation

Modules

🦜🔗

⌘ + K

There are several main modules that LangChain provides support for. For each m

provide some examples to get started, how-to guides, reference docs, and conceptual guides. These modules are, in increasing order of complexity:

- Models: The various model types and model integrations LangChain supports.
- Prompts: This includes prompt management, prompt optimization, and prompt serialization.
- Memory: Memory is the concept of persisting state between calls of a chain/agent. LangChain provides a standard interface for memory, a collection of memory implementations, and examples of chains/agents that use memory.
- Indexes: Language models are often more powerful when combined with your own text data - this module covers best practices for doing exactly that.
- Chains: Chains go beyond just a single LLM call, and are sequences of calls (whether to an LLM or a different utility). LangChain provides a standard interface for chains, lots of integrations with other tools, and end-to-end chains for common applications.
- Agents: Agents involve an LLM making decisions about which Actions to take, taking that Action, seeing an Observation, and repeating that until done. LangChain provides a standard interface for agents, a selection of agents to choose from, and examples of end to end agents.

Use Cases

The above modules can be used in a variety of ways. LangChain also provides guidance and assistance in this. Below are some of the common use cases LangChain supports.

- Autonomous Agents: Autonomous agents are long running agents that take many steps in an attempt to accomplish an objective. Examples include AutoGPT and BabyAGI.
- Agent Simulations: Putting agents in a sandbox and observing how they interact with each other or to events can be an interesting way to observe their long-term memory abilities.
- Personal Assistants: The main LangChain use case. Personal assistants need to take actions, remember interactions, and have knowledge about your data.
- Question Answering: The second big LangChain use case. Answering questions over specific documents, only utilizing the information in those documents to cons answer.
- Chatbots: Since language models are good at producing text, that makes the or creating chatbots.

🦜🔗

⌘ + K

- Querying Tabular Data: If you want to understand how to use LLMs to query data that is stored in a tabular format (csvs, SQL, dataframes, etc) you should read this page.
- Code Understanding: If you want to understand how to use LLMs to query source code from github, you should read this page.
- Interacting with APIs: Enabling LLMs to interact with APIs is extremely powerful in order to give them more up-to-date information and allow them to take actions.
- Extraction: Extract structured information from text.
- Summarization: Summarizing longer documents into shorter, more condensed chunks of information. A type of Data Augmented Generation.
- Evaluation: Generative models are notoriously hard to evaluate with traditional metrics. One new way of evaluating them is using language models themselves to do the evaluation. LangChain provides some prompts/chains for assisting in this.

Reference Docs

All of LangChain's reference documentation, in one place. Full documentation on all methods, classes, installation methods, and integration setups for LangChain.

- Reference Documentation

LangChain Ecosystem

Guides for how other companies/products can be used with LangChain

- LangChain Ecosystem

Additional Resources

Additional collection of resources we think may be useful as you develop your application!

- LangChainHub: The LangChainHub is a place to share and explore other prompts, chains, and agents.
- Glossary: A glossary of all related terms, papers, methods, etc. Whether imp 🦜🔗 in LangChain or not!
- Gallery: A collection of our favorite projects that use LangChain. Useful for fi ⌘ + K inspiration or seeing how things were done in other applications.

- Deployments: A collection of instructions, code snippets, and template repositories for deploying LangChain apps.
- Tracing: A guide on using tracing in LangChain to visualize the execution of chains and agents.
- Model Laboratory: Experimenting with different prompts, models, and chains is a big part of developing the best possible application. The ModelLaboratory makes it easy to do so.
- Discord: Join us on our Discord to discuss all things LangChain!
- YouTube: A collection of the LangChain tutorials and videos.
- Production Support: As you move your LangChains into production, we'd love to offer more comprehensive support. Please fill out this form and we'll set up a dedicated support Slack channel.

🦜🔗

⌘ + K