LangChain

Building applications with LLMs through composability

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.

Quick Install

```
pip install langchain
```

What is this?

Large language models (LLMs) are emerging as a transformative technology, enabling developers to build applications that they previously could not. But using these LLMs in isolation is often not enough to create a truly powerful app - the real power comes when you can combine them with other sources of computation or knowledge.

This library is aimed at assisting in the development of those types of applications. Common examples of these types of applications include:

Question Answering over specific documents

- Documentation
- End-to-end Example: Question Answering over Notion Database

Chatbots

- Documentation
- End-to-end Example: Chat-LangChain

Agents

- Documentation
- End-to-end Example: GPT+WolframAlpha

Documentation

Please see here for full documentation on:

- Getting started (installation, setting up the environment, simple examples)
- How-To examples (demos, integrations, helper functions)
- Reference (full API docs)
- Resources (high-level explanation of core concepts)

What can this help with?

There are six main areas that LangChain is designed to help with. These are, in increasing order of complexity:

LLMs and Prompts:

This includes prompt management, prompt optimization, generic interface for all LLMs, and common utilities for working with LLMs.

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.

Data Augmented Generation:

Data Augmented Generation involves specific types of chains that first interact with an external datasource to fetch data to use in the generation step. Examples of this include summarization of long pieces of text and question/answering over specific data sources.

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.

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.

Evaluation:[BETA] 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