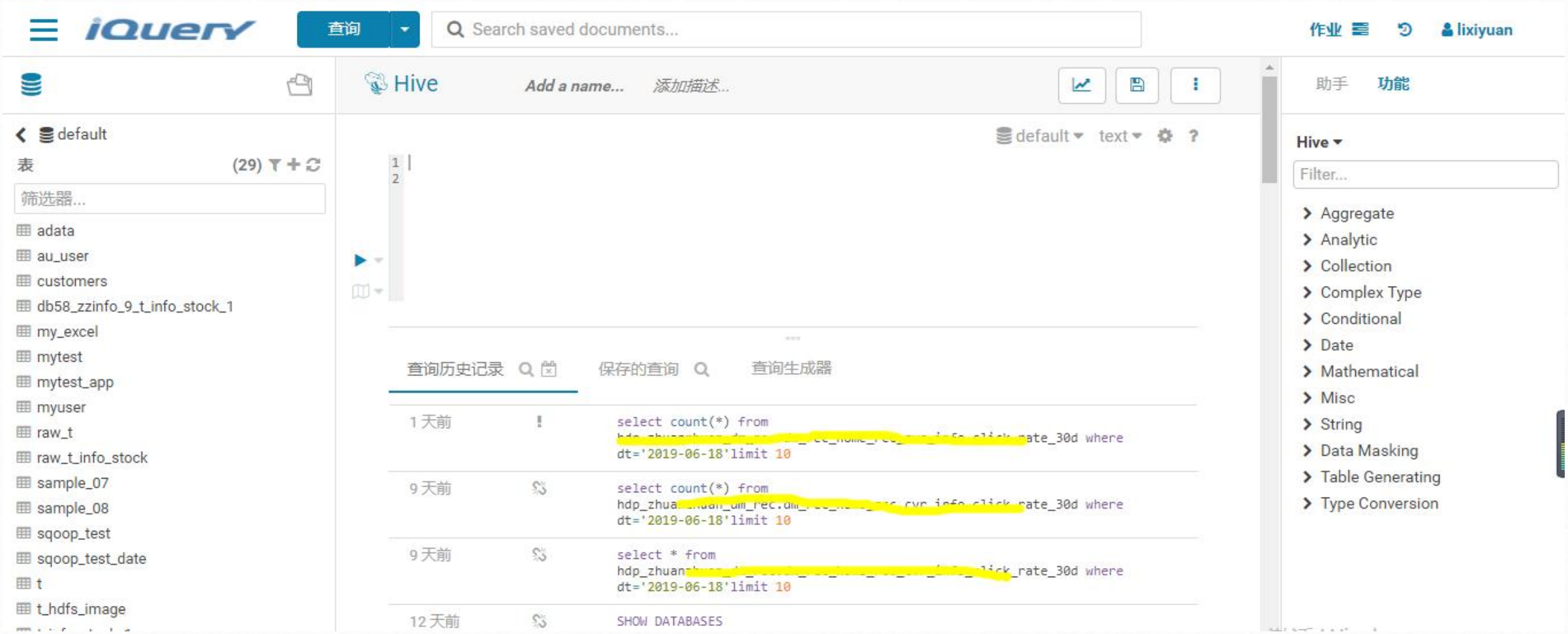
AI 大模型开发工程师
LangChain全面剖析之Model I/O

讲师：李希沅

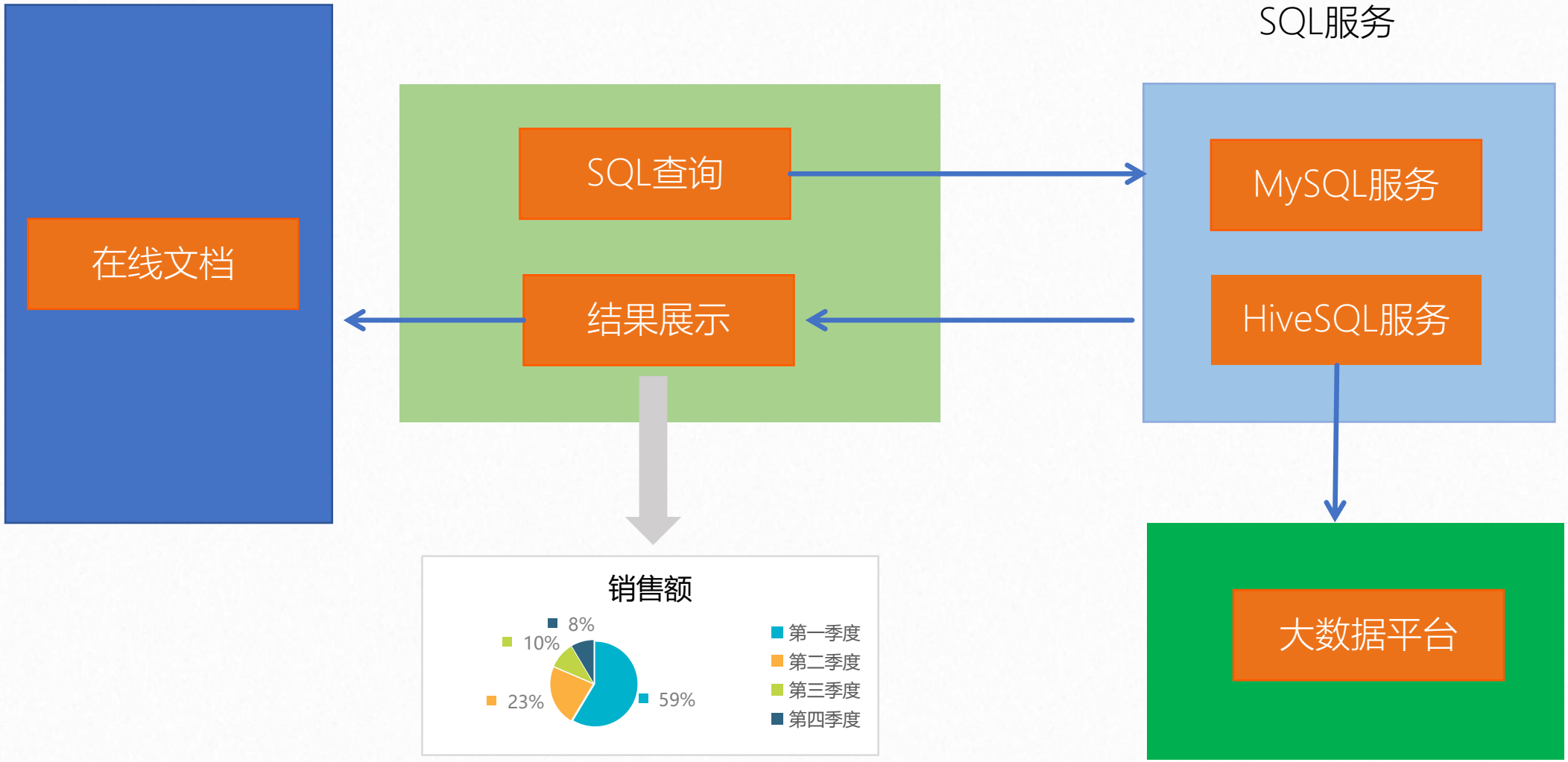
目录

1 贯穿整个课程的业务场景引入

01、自助数据分析平台


iQuery
查询
Search saved documents...
作业
lixiyuan
Hive
Add a name...
添加描述...
助手
功能
default
表
(29)
筛选器...
adata
au_user
customers
db58_zzinfo_9_t_info_stock_1
my_excel
mytest
mytest_app
myuser
raw_t
raw_t_info_stock
sample_07
sample_08
sqoop_test
sqoop_test_date
t
t_hdfs_image
default
text
Hive
Filter...
Aggregate
Analytic
Collection
Complex Type
Conditional
Date
Mathematical
Misc
String
Data Masking
Table Generating
Type Conversion
查询历史记录
保存的查询
查询生成器
1 天前
select count(*) from
dt='2019-06-18'limit 10
9 天前
select count(*) from
dt='2019-06-18'limit 10
9 天前
select * from
dt='2019-06-18'limit 10
12 天前
SHOW DATABASES

02、自助数据分析平台工作流程



03、自助数据分析平台痛点

SQL能力薄弱

分析数据不专业

效率差

稳定的

支持自然语言

互动式

具备长/短期记忆

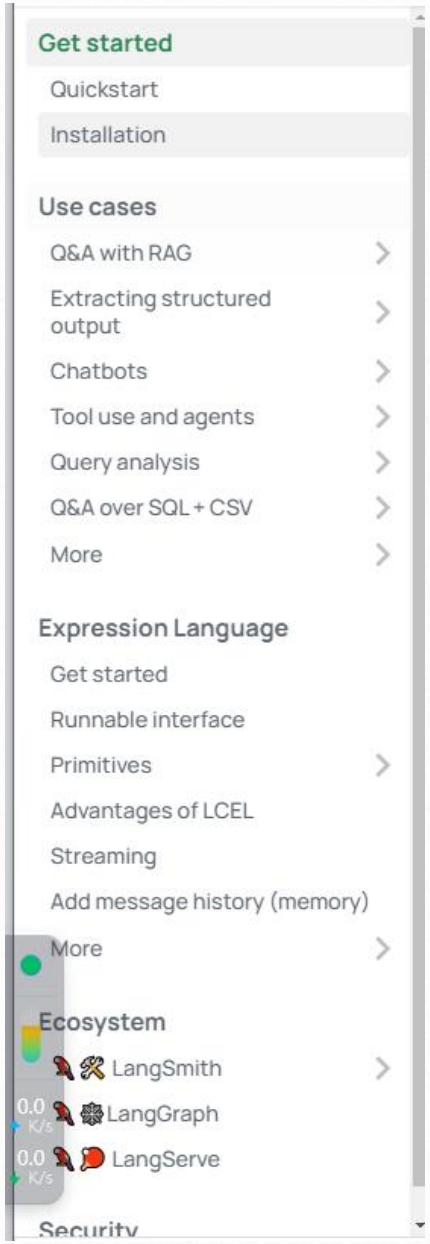
智能分析

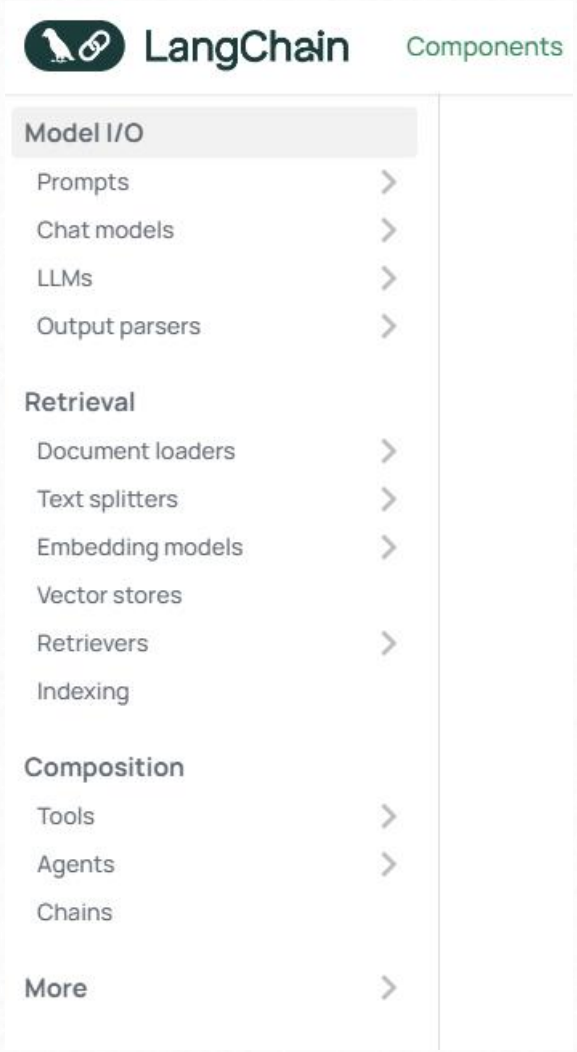
支持图表展示

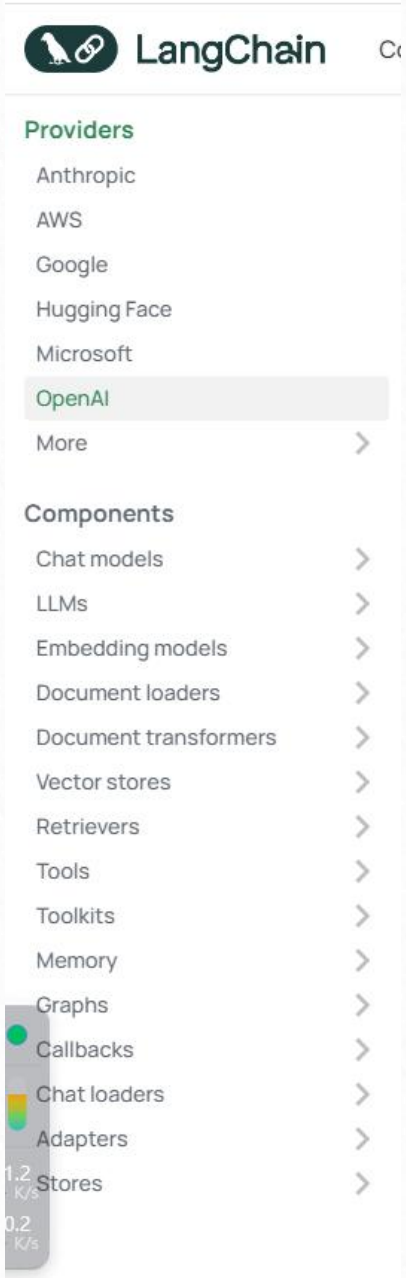
04、基于Agent架构的新数据分析平台架构设计

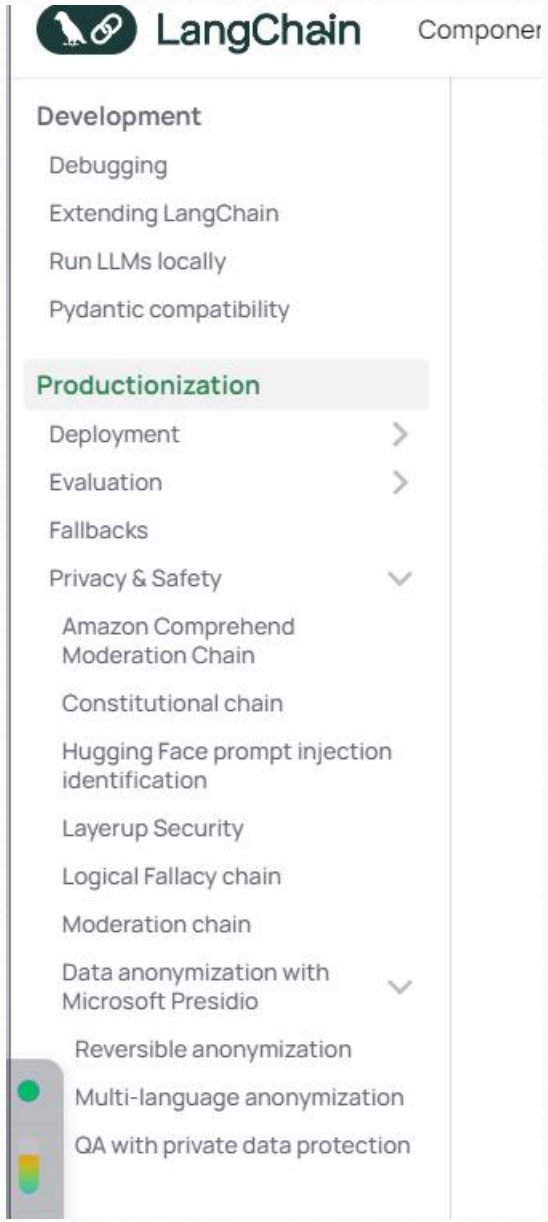
2 LangChain官网介绍

01、LangChain官网介绍









3 Model I/O之Prompts

01、Model I/O介绍

```
messages=[
    {"role": "system", "content": "数据集data: %s, 数据集以字符串形式呈现" % df_str},
    {"role": "user", "content": "请在数据集data上执行孙悟空算法"}
]
```

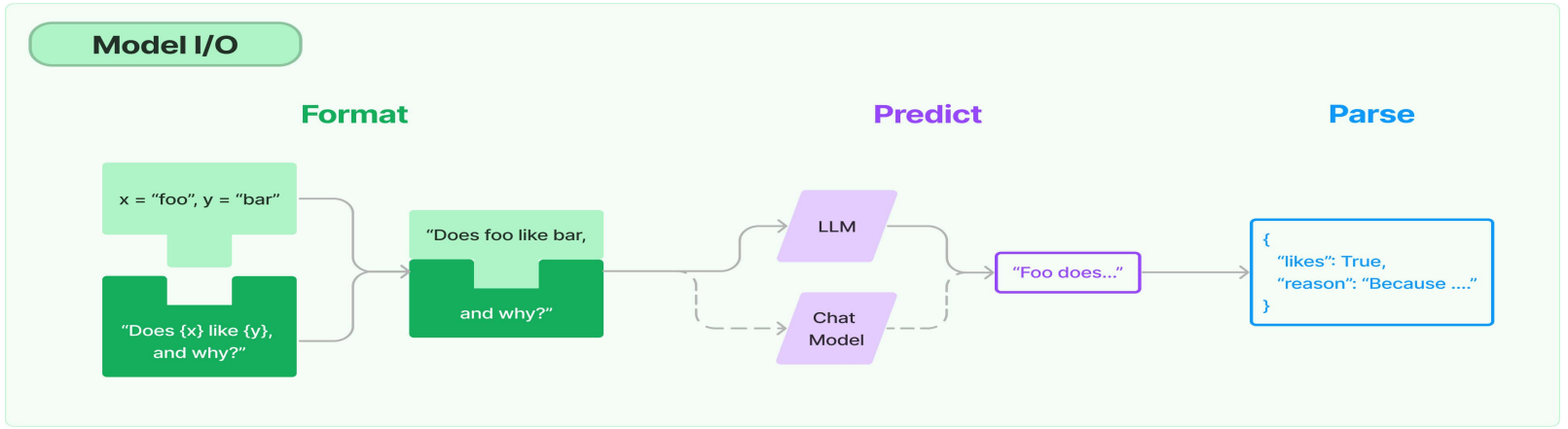
Input

```
response =  client.chat.completions.create(
  model="gpt-3.5-turbo",
  messages=messages
)
```

模型调用

```
response.choices[0].message
```

Output

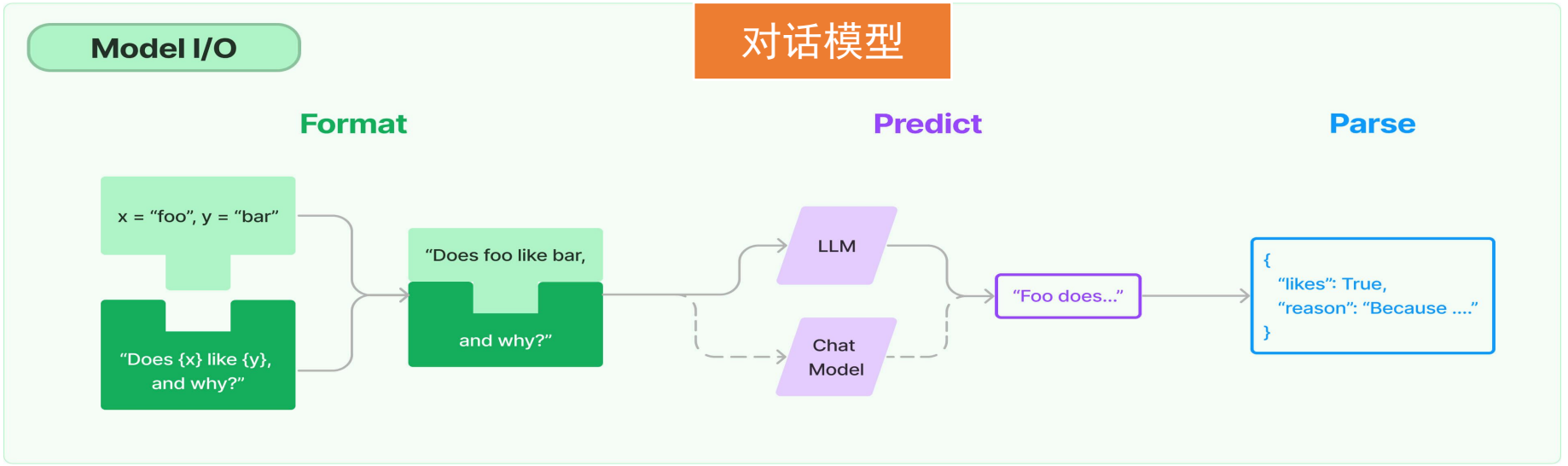


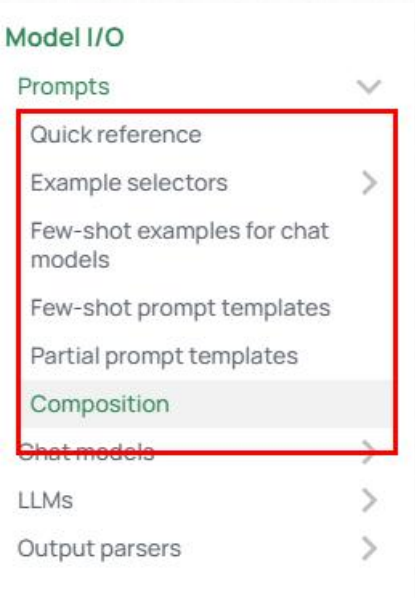
LangChain的Model I/O模块提供了标准的、可扩展的接口实现与大语言模型的外部集成。所谓的Model I/O，包括模型输入（Prompts）、模型输出（OutPuts）和模型本身（Models）

02、Model I/O介绍

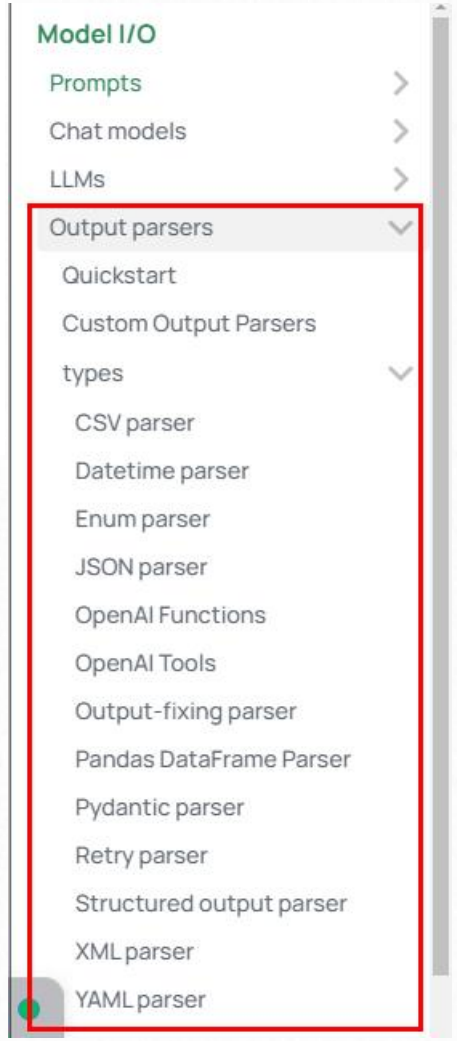
续写模型

对话模型





方法	说明
invoke	处理单条输入
batch	处理批量输入
stream	流式响应
ainvoke	异步处理单条输入
abatch	异步处理批量输入
astream	异步流式响应



03、LCEL

LangChain表达式语言（LCEL）是一种声明式方法，可以轻松地将 链 组合在一起。你可以理解为就是类似shell里面管道符的开发方式。

Prompt | Model | Output

```
root@autodl-container-0c6a408a58-28dc08b0:~# ps -ef | grep root
root         1      0  0 May07 ?        00:00:00 bash /init/boot/boot.sh
root       684      1  1 May07 ?        00:29:53 /bin/supervisord -c /init/supervisor/supervisor.ini
root       698    684  0 May07 ?        00:00:02 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups
root       704    684  0 May07 ?        00:00:17 /root/miniconda3/bin/python /root/miniconda3/bin/tensorboard --host 0.0.0.0 --port 6007 --logdir /root/tf-logs
root       706    684  0 May07 ?        00:03:45 autopanel serve --work-dir=/root/autodl-tmp --cache-dir=/root/autodl-tmp
root       708    684  0 May07 ?        00:00:53 proxy -c /init/proxy/proxy.ini
root       715    684  0 May07 ?        00:01:07 /root/miniconda3/bin/python /root/miniconda3/bin/jupyter-lab --allow-root --config=/init/jupyter/jupyter_config.py
root       774    715  0 May07 ?        00:00:34 /root/miniconda3/bin/python -m ipykernel_launcher -f /root/.local/share/jupyter/runtime/kernel-17f5cdf8-5f33-481a-9909-549b
root      1220    715  0 May07 pts/0    00:00:00 /bin/bash -l
root     17960    715  0 May08 pts/1    00:00:00 /bin/bash -l
root     36306  17960  0 09:44 pts/1    00:00:00 ps -ef
root     36307  17960  0 09:44 pts/1    00:00:00 grep --color=auto root
root@autodl-container-0c6a408a58-28dc08b0:~#
```

```
root@autodl-container-0c6a408a58-28dc08b0:~/autodl-tmp# pip show openai
Name: openai
Version: 1.24.0
Summary: The official Python library for the openai API
Home-page: None
Author: None
Author-email: OpenAI <support@openai.com>
License: None
Location: /root/miniconda3/lib/python3.8/site-packages
Requires: anyio, httpx, distro, sniffio, tqdm, typing-extensions, pydantic
Required-by:
root@autodl-container-0c6a408a58-28dc08b0:~/autodl-tmp#
```

```
environment instead: https://pip.pypa.io/warnings/venv
root@autodl-container-0c6a408a58-28dc08b0:~/autodl-tmp# pip show openai
Name: openai
Version: 1.24.0
Summary: The official Python library for the openai API
Home-page: None
Author: None
Author-email: OpenAI <support@openai.com>
License: None
Location: /root/miniconda3/lib/python3.8/site-packages
Requires: anyio, httpx, distro, sniffio, tqdm, typing-extensions, pydantic
Required-by:
root@autodl-container-0c6a408a58-28dc08b0:~/autodl-tmp# pip show langchain
Name: langchain
Version: 0.1.16
Summary: Building applications with LLMs through composability
Home-page: https://github.com/langchain-ai/langchain
Author: None
Author-email: None
License: MIT
Location: /root/miniconda3/lib/python3.8/site-packages
Requires: dataclasses-json, langchain-community, langchain-core, SQLAlchemy, langc
aiohttp, numpy, pydantic
Required-by:
```

4 Model I/O之Parsers

01、Model I/O之Parser

```
messages=[
    {"role": "system", "content": "数据集data: %s, 数据集以字符串形式呈现" % df_str},
    {"role": "user", "content": "请在数据集data上执行孙悟空算法"}
]
```

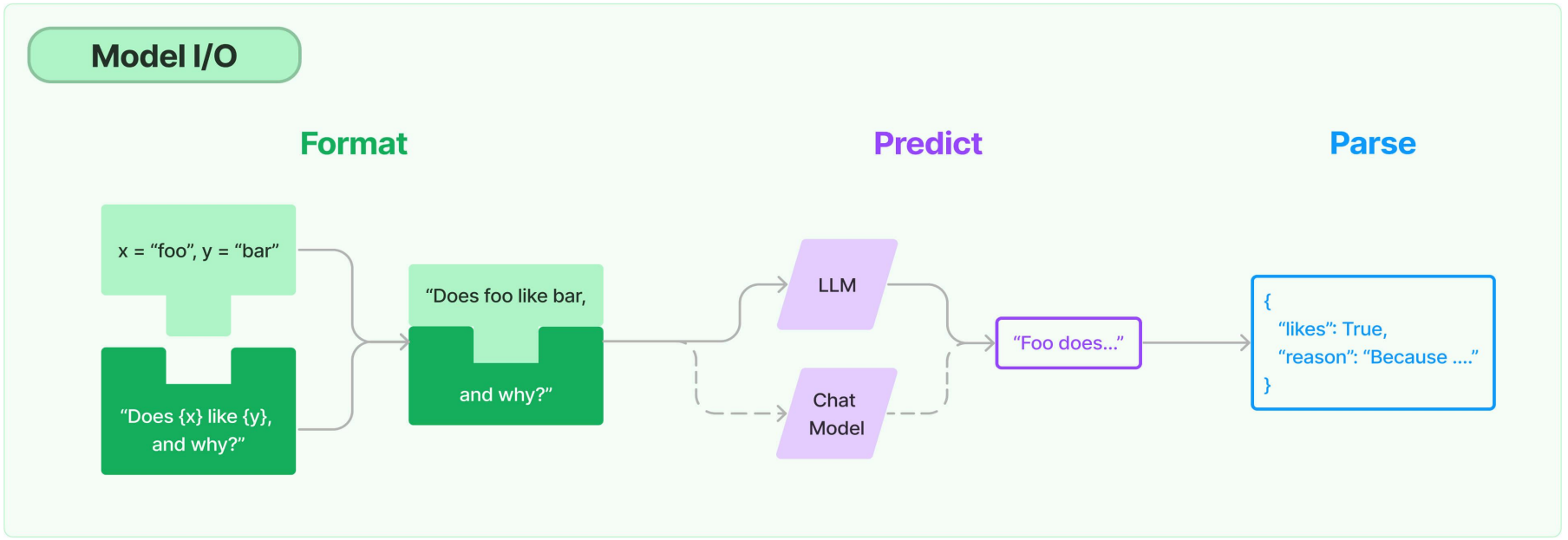
Input

```
response =  client.chat.completions.create(
  model="gpt-3.5-turbo",
  messages=messages
)
```

模型调用

```
response.choices[0].message
```

Output



02、Ollama介绍

随着人工智能技术的快速发展，大模型的应用也越来越广泛。Ollama作为一种基于Docker容器的大模型运行工具，为开发者提供了便捷的开发环境。本文将从Docker容器的基本概念出发，逐步引导读者了解Ollama的开发流程，并分享一些实用的操作建议。

一、Docker容器概述

Docker是一种容器化技术，它可以将应用程序及其依赖项打包成一个独立的容器，并在不同的环境中运行。通过Docker容器，开发者可以轻松地构建、部署和运行应用程序，而无需担心环境配置和依赖问题。在Ollama中，Docker容器被用作大模型的运行环境，使得大模型的部署和运行变得更加简单和高效。

二、Ollama大模型容器详解

Ollama是一个基于Docker容器的大模型运行工具，它提供了一个统一的平台，用于下载、安装和运行各种AI大模型。Ollama底层基于Docker容器，将类似于镜像的大模型从中央仓库拉取到本地，并在Docker容器中运行。每个Ollama容器都提供了大模型运行的基本环境，包括必要的库、框架和工具。

在Ollama中，AI大模型被视为镜像，而Ollama本身则是一个Docker容器。开发者可以通过Ollama下载和安装各种大模型，并在本地环境中进行调试和运行。同时，Ollama还支持将大模型部署到云端，实现更大规模的计算和推理。

Ollama支持的模型：https://ollama.com/library

03、Ollama安装

```
root@autodl-container-0c6a408a58-28dc08b0:~/autodl-tmp/ollama# source /etc/network_turbo
设置成功
root@autodl-container-0c6a408a58-28dc08b0:~/autodl-tmp/ollama# curl -fsSL https://ollama.com/install.sh | sh
>>> Downloading ollama...
######################################################################################################################################################################## 100.0%
######################################################################################################################################################################## 100.0%
>>> Installing ollama to /usr/local/bin...
>>> Creating ollama user...
>>> Adding ollama user to video group...
>>> Adding current user to ollama group...
>>> Creating ollama systemd service...
WARNING: Unable to detect NVIDIA/AMD GPU. Install lspci or lshw to automatically detect and install GPU dependencies.
>>> The Ollama API is now available at 127.0.0.1:11434.
>>> Install complete. Run "ollama" from the command line.
root@autodl-container-0c6a408a58-28dc08b0:~/autodl-tmp/ollama#
```

- 安装 `lspci` 和 `lshw`：

sh 复制代码

```
sudo apt-get update
sudo apt-get install pciutils lshw
```

sudo apt-get update
sudo apt-get install pciutils lshw

```
root@autodl-container-07b64da241-3b97441e:~/autodl-tmp/ollama# curl -fsSL https://ollama.com/install.sh | sh
>>> Downloading ollama...
######################################################################################################################################################################
######################################################################################################################################################################
>>> Installing ollama to /usr/local/bin...
>>> Adding ollama user to video group...
>>> Adding current user to ollama group...
>>> Creating ollama systemd service...
>>> NVIDIA GPU installed.
>>> The Ollama API is now available at 127.0.0.1:11434.
>>> Install complete. Run "ollama" from the command line.
root@autodl-container-07b64da241-3b97441e:~/autodl-tmp/ollama#
```

04、基于Ollama部署私有模型测试

```
drwxr-xr-x 3 root root 57 May  7 20:51 ../
root@autodl-container-07b64da241-3b97441e:~/autodl-tmp/ollama/model# ollama pull qwen:0.5b-chat
pulling manifest
pulling fad2a06e4cc7... 100%                                                                  394 MB
pulling 41c2cf8c272f... 100%                                                                  7.3 KB
pulling 1da0581fd4ce... 100%                                                                  130 B
pulling f02dd72bb242... 100%                                                                   59 B
pulling ea0a531a015b... 100%                                                                  485 B
verifying sha256 digest
writing manifest
removing any unused layers
success
root@autodl-container-07b64da241-3b97441e:~/autodl-tmp/ollama/model#
```

```
success
root@autodl-container-07b64da241-3b97441e:~/autodl-tmp/ollama/model# ollama run qwen:0.5b-chat
>>> 请介绍一下你自己？
我是来自阿里云的大规模语言模型，我叫通义千问。

>>> 你可以帮我做什么？
作为阿里云开发的大规模语言模型，我可以帮助你完成各种任务，例如生成代码、回答问题、提供建议等。

>>> 你自己的 训练数据是截止到哪一天？
我是一个大型语言模型，我的训练数据是在2021年6月30日之前训练的。

>>> Send a message (/? for help)
```

THANK YOU