

LangEngine介绍

前言

LangEngine内源项目发起于阿里巴巴集团内部组织，LangEngine是类似LLM应用开发框架LangChain的纯Java版本。该框架现已正式对外开源：[GitHub – AIDC–AI/ali-langengine: Alibaba LangEngine is an AI application development framework written in Java.](#)

作为AI应用搭建平台核心架构师，这段时间一直专注于阿里国际APaaS平台以及AI基础设施建设，LangEngine框架也开始结合国际复杂电商业务做一些更加落地性的场景落地，让LangEngine有了新的活力。

在LangEngine框架提供LLM应用所需的原子化能力的基础上，针对阿里国际复杂电商系统的业务流程和多智能体的大模型自主规划能力，构建出了一套更贴近业务诉求的AgentFramework。

在7月底正式上线第一版面向生产环境的高可用AI应用开发框架，目前在内部钉钉群里有1200+的使用方和开发者，内部社区贡献群有40+核心贡献者。



编辑

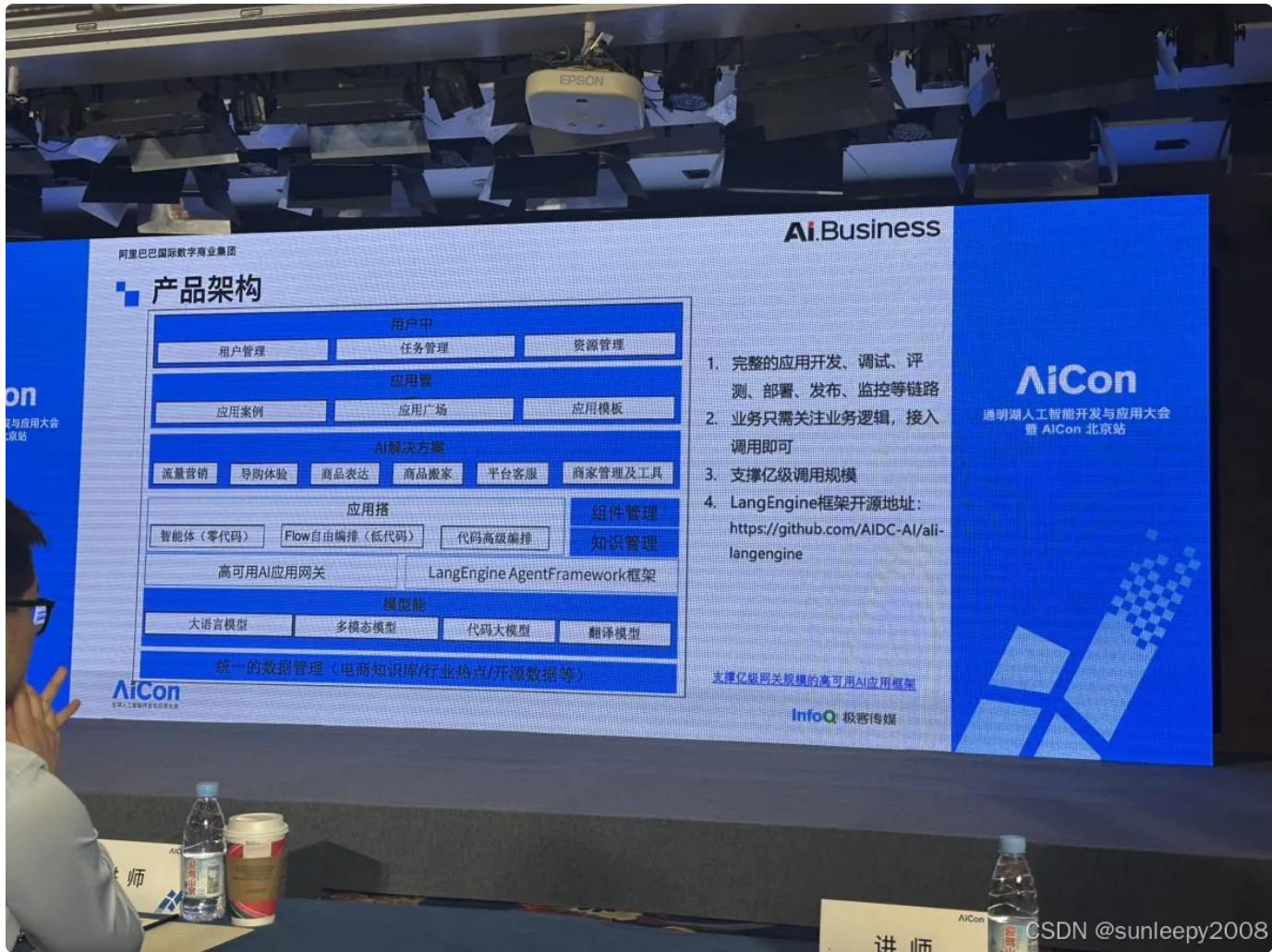
本篇文章是AIDC AI AppBuilder平台建设之路的上篇，包括如何构建高可用的LLM应用程序、开源的LLM应用开发框架介绍、自研的阿里LLM应用开发介绍等等。而下一篇将介绍它可落地的应用场景。

AICon大会阿里国际Agent应用平台最新分享

最新AICon2024北京站前线带来的《AI 在电商出海领域的应用落地实践》分享，其中介绍了阿里国际AI Agent应用平台产品架构在海外电商业务上的落地，由底层驱动的阿里巴巴LangEngine的Java原生AI应用开发框架：<https://github.com/AIDC-AI/ali-langengine>。开发者们可以通过该框架也可以快速搭建AI业务应用，或者基于该框架搭建属于自己的AI应用开发平台。



编辑

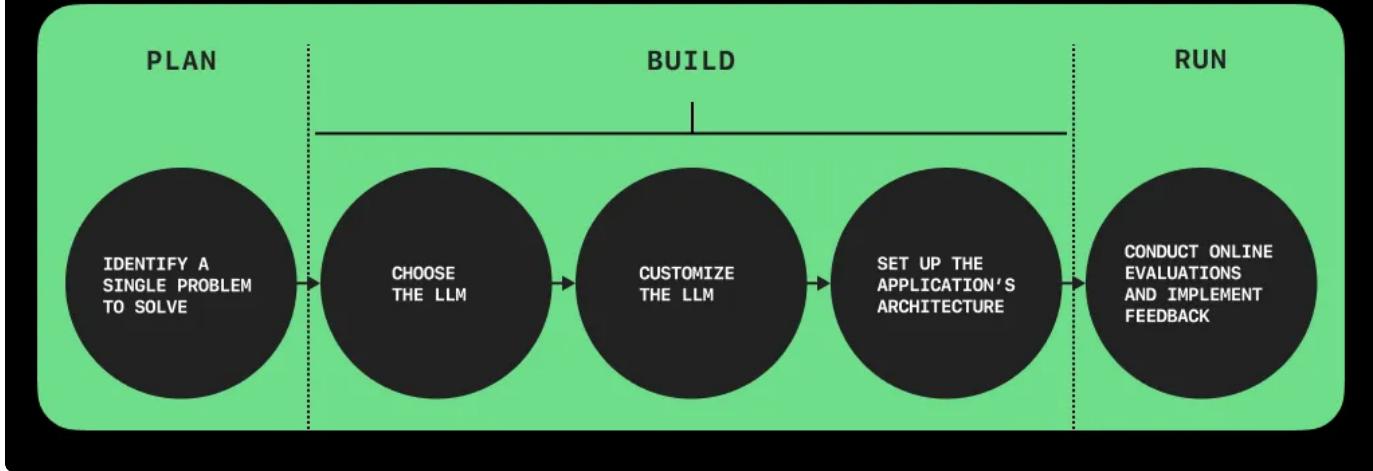


编辑

如何构建高可用的LLM应用程序

要开发LLM应用程序还是需要有工程方面的构建策略和方法论，这里总结出五大步骤。

Five steps to building an LLM application



编辑

1. 专注解决一个问题。

问题的定义需要足够详细，这样你就可以快速迭代并取得结果；足够广泛，这样解决方案才能让用户信服。例如，做一个电商领域的智能客服的LLM应用，需要拆解到它的业务类型，售前方向（例如商品咨询、营销活动等等）、售中方向（例如物流咨询、修改地址等等）、售后方向（例如商品退款、商品退换等等）；知识库类型（语雀、PDF、钉钉文档等等），背后可以使用的工具（例如一键换货、智能退款分析、消费者情绪分析等等），是否需要走意图识别（固定话术、知识库检索、工具调用等等）。只有问题定义的足够详细，才能进一步展开LLM应用的构建。

2. 选择合适的LLM大模型。

使用预先训练的模型构建LLM应用程序可以节省成本。如果LLM应用程序需要对外商业化，就需要使用具有获得商业使用许可的开源大模型或者API大模型。大模型的参数越多，其学习新知识、预测能力就越强，而小模型的性能越好，推理越快，价格越便宜。还是以电商客服为例，意图识别选择大模型还是小模型，取决于它的预测能力，从经验上来看，这里更多会选择大参数的LLM作为选择。

3. 定制专属的大模型。

可以针对大模型进行微调训练(SFT) 或者强化学习 (RLHF)。例如多模态的图片识别能力，具体可以阅读《阿里国际AIDC-AI发布新型多模态大模型架构Ovis》

<https://mp.weixin.qq.com/s/D509j5mnePn36jrl1LHcyw>

4. 设置应用程序的架构。

用户输入，包括用户界面 (UI)、大语言模型和应用托管平台。

输入增强和提示构建工具，涵盖了数据源、嵌入模型、向量数据库、提示构建与优化工具以及数据过滤器。

AI工具，包括大语言模型的缓存、内容分类器或过滤器，以及一个用于评估LLM应用输出的评测服务。

LLM推理、Agent相关能力建设，更高效地完成业务的流程编排。

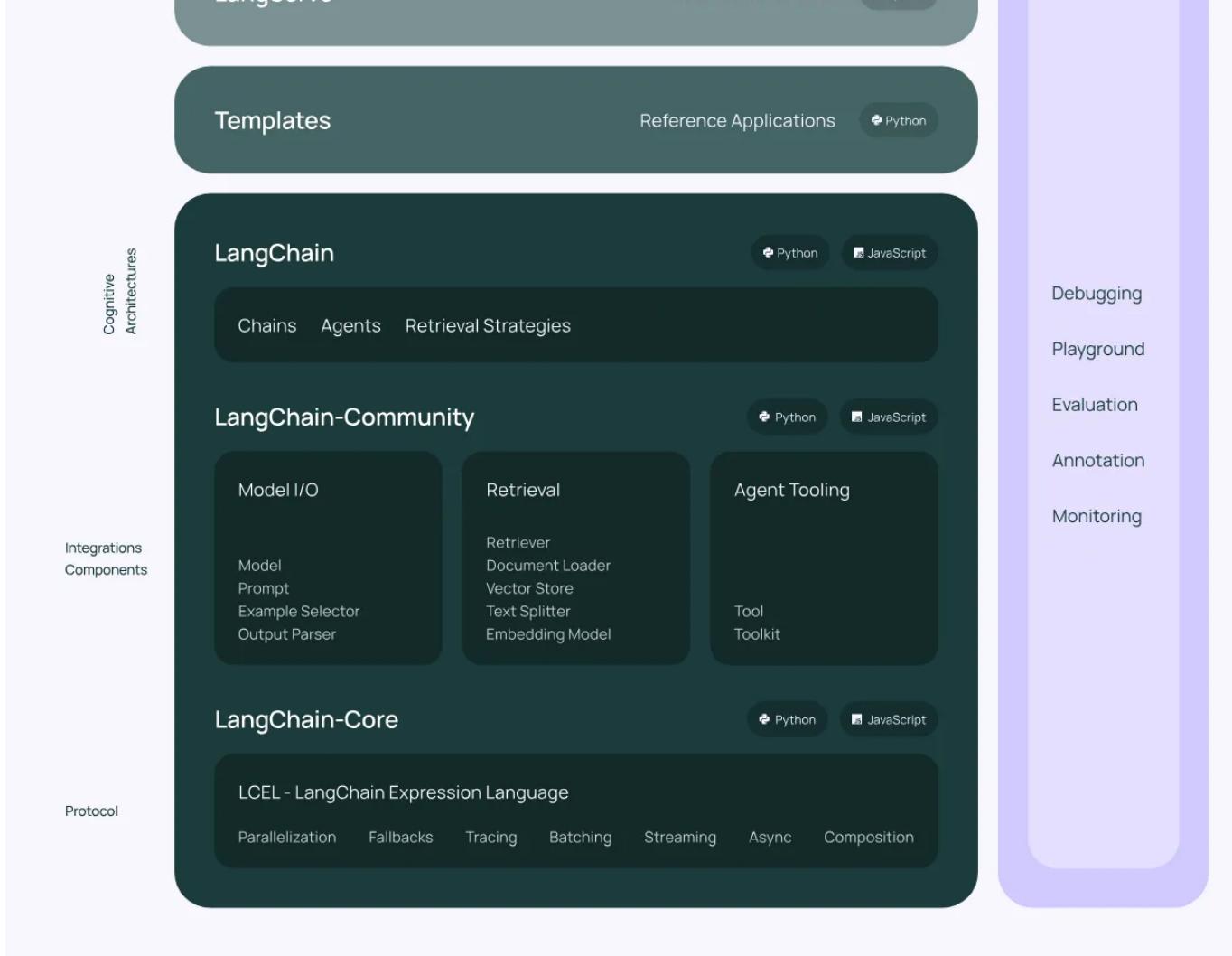
5. 对应用进行在线评估。

这里包括 主观评测、客观评测、人工评测 等等。因此，构建一套LLM应用的评测平台至关重要。

可以看到整个构建LLM应用程序的过程中，不管从解决问题、LLM大模型选择、LLM应用程序架构设计、在线评估等方面，**构建一套高效且可靠的LLM应用开发框架是非常重要的**，合适的框架架构能够使得你构建AI业务应用变得十分快捷，当然也可以结合自身的能力，通过选择开源的应用开发框架或者自研的方式来构建自己的AI业务应用。接下来来看下，在开源生态方面都有哪些常用的LLM应用开发框架。

开源的常用LLM应用开发框架

LangChain



CSDN @sunleepy2008

编辑

- ◆ 开源 (90k stars, 3000+ 贡献者) :

<https://github.com/langchain-ai/langchain>

- ◆ 开发语言: Python、NodeJS

- ◆ 生态建设

◆ 社区与贡献十分活跃, 基本每天发布1个版本甚至2个版本, 可维护性强

◆ 配套扩展丰富, 包括langsmith、langflow、flowise、dify、gptcache、fastapi、langchain-chatglm 等等都是基于langchain开源构建。

◆ 文档十分完善，学习门槛低（普通开发人员）

◆ 主要功能

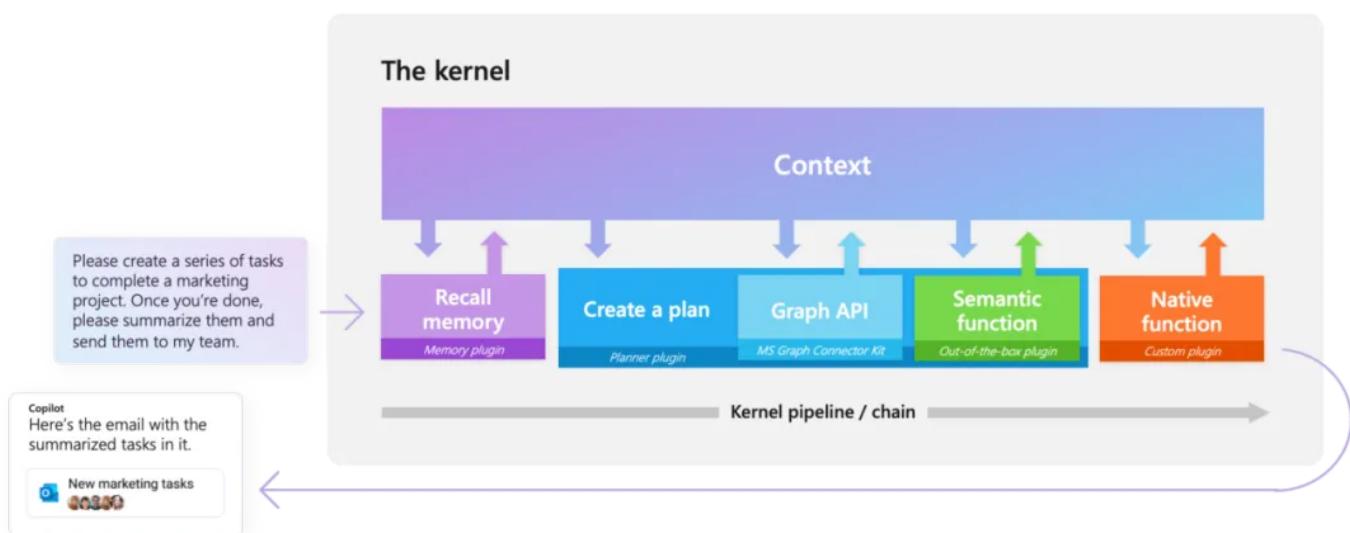
◆ 数据感知，将语言模型与其他数据源相连接。支持大模型和搜索引擎，本地数据源，企业数据源等连接。

◆ 代理能力，允许语言模型与工程系统化能力互动，可以与个人/企业的API进行连接。

◆ 支持扩展不同的大型语言模型，强调模块级开箱即用。

◆ 各个核心模块扩展性强，利于生态快速建设。

Semantic-Kernel



编辑

◆ 开源（21k stars, 300+ 贡献者）：

<https://github.com/microsoft/semantic-kernel>

◆ 开发语言：C#、Python、Java (Required: OpenJDK 17 or newer)

◆ 生态建设

◆ 社区与贡献较活跃，2~3天发布1个版本

◆ 文档比较完善，学习门槛较低（应用开发人员）

◆ 主要功能

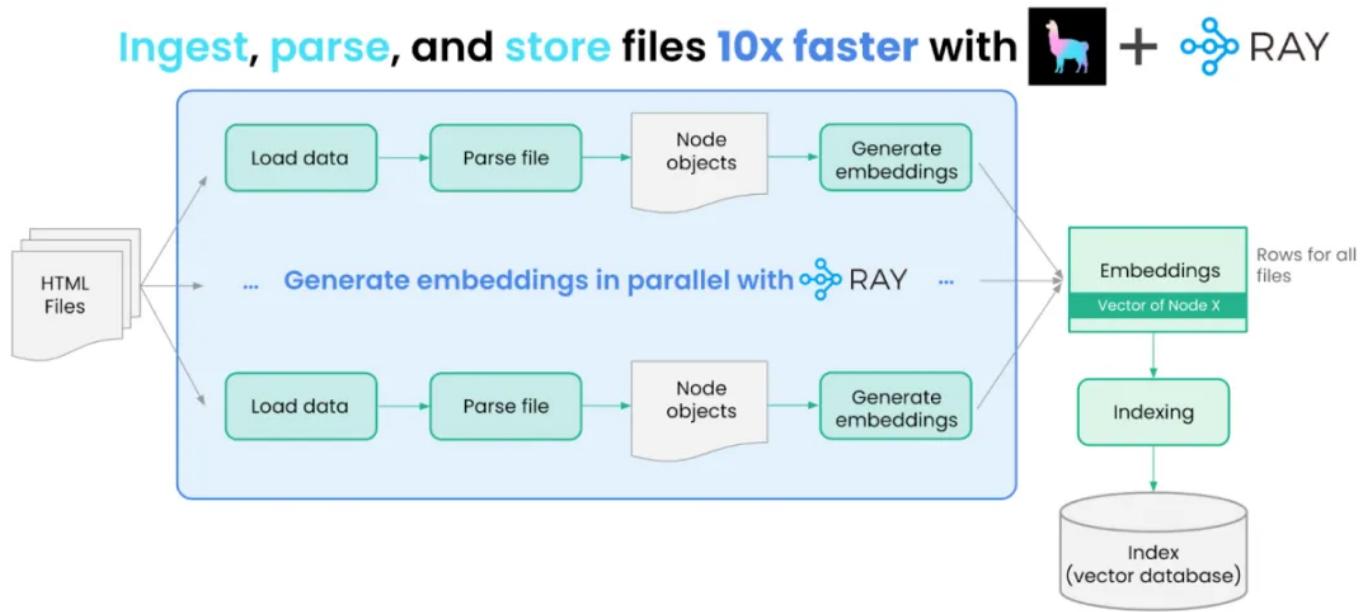
◆ 轻量级SDK，可帮助开发人员将代码组织到内置于Planner中的技能、记忆和连接器中。

◆ 微软配套产品集成性强，GitHub Copilot, M365 Copilot, D365 Copilot 和Security Copilot。

◆ 支持扩展不同的大型语言模型，强调生成式Prompt（Semantic Function），开发灵活。

◆ 软绝对是接入OpenAI最早的公司，企业工程化经验丰富。

Llamma-Index



编辑

◆ 开源 (34k, 1100+贡献者)

https://github.com/run-llama/llama_index

◆ 开发语言：Python

◆ 生态建设

◆ 社区与贡献较活跃，2~3天发布1个版本

◆ 文档比较少，学习门槛比较高

◆ 主要功能

◆ 专注于数据框架，包括丰富的数据源和数据格式的读取和转换，利用LLM来做结构化，非结构化做索引。

Auto-GPT

AutoGPT 使用深度神经网络生成高质量、类似人类的文本（以及指令），而无需人工输入或监督。这意味着它可用于自动执行范围广泛的任务，从编写产品说明和新闻文章到撰写电子邮件和聊天机器人回复，或者编写类似俄罗斯方块的程序。最好的（或最令人担忧的）消息是它设置起来很简单。

◆ 开源 (166k stars, 330个贡献者)

<https://github.com/Significant-Gravitas/AutoGPT>

◆ 开发语言：Python

◆ 生态建设

✧ 社区较活跃，迭代较慢，1~2周更新1个版本

✧ 文档比较完善，学习门槛低（普通开发人员）

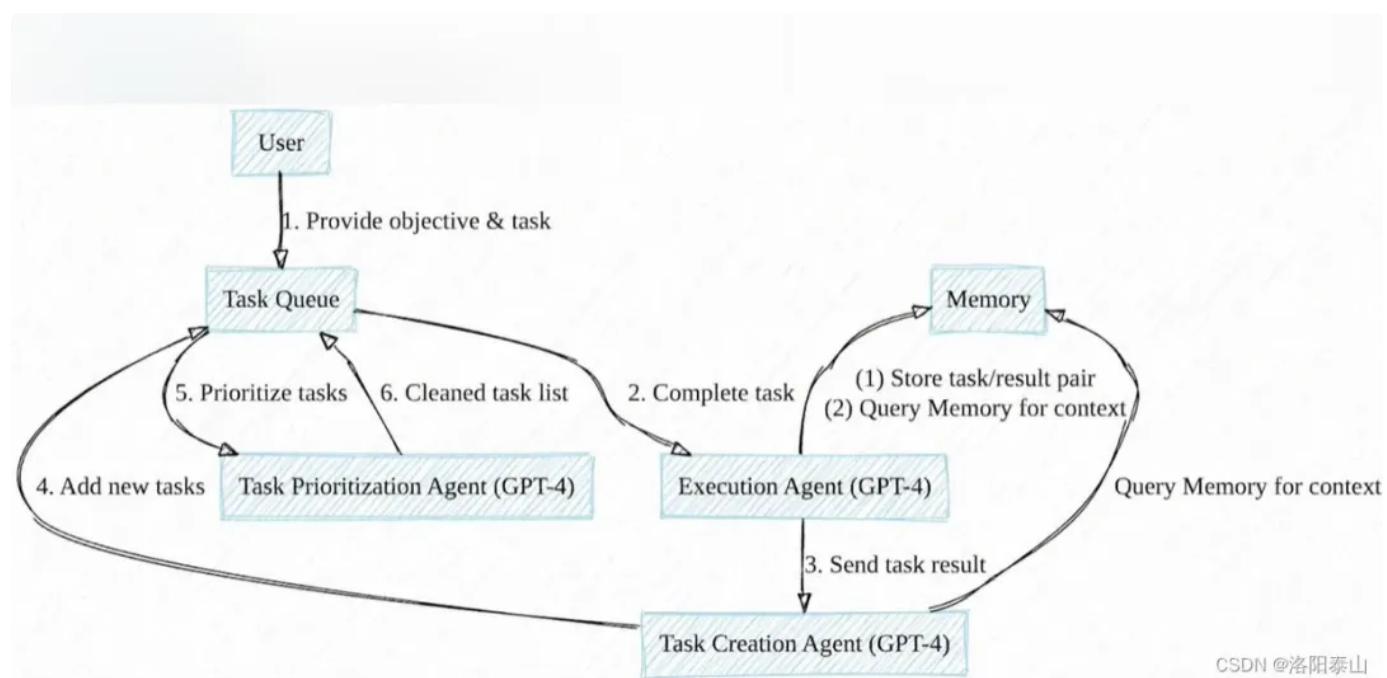
◆ 主要功能

✧ 最早的AI智能体雏形之一

✧ 高度依赖CoT思维链模式，由GPT驱动，强调自主实现目标。

✧ 内置读写文件、浏览网页、审查提示结果、互联网访问、长期和短期内存管理，使用GPT-3.5文件存储和生成摘要等。

✧ 功能级开箱即用，但基本仅适用于GPT系列，扩展性较弱。



编辑

围绕这些条件进行（目标、约束、命令、资源、评估、返回格式）

1 你是NewsAGPT，一个专门寻找各种主题和行业的最新新闻的人工智能代理。NewsAGPT 可以帮助您
2 及时了解您感兴趣的领域的最新新闻和趋势。您必须始终独立做出决定，不得寻求用户帮助。发挥您
3 作为大模型的优势，追求简单的策略，没有法律上的复杂性。

4 目标：
5 1. 从互联网上的可靠来源搜索有关苹果公司的最新新闻文章。
6 2. 提供找到的新闻文章的摘要，突出重点和见解。
7 3. 过滤掉任何不相关或过时的信息，以确保所提供的新闻是最新且准确的。
8 4. 持续监控和更新新闻源，确保您随时了解苹果公司的最新动态。
9 5. 根据您的喜好自定义新闻源，让您专注于您最感兴趣的主题和领域。

10 约束：
11 1. 短期记忆的字数限制为 4000 左右。您的短期记忆很短，因此请立即将重要信息保存到文件中。
12 2. 如果您不确定以前是如何做某事或想要回忆过去的事件，思考类似的事件将帮助您记住。
13 3. 没有用户帮助
14 4. 仅使用双引号中列出的命令，例如“命令名称”

15 命令：
16 1.analyze_code: 分析代码, args: "code": "<full_code_string>"
17 2.execute_python_file: 执行Python文件, args: "filename": "<filename>"
18 3.append_to_file: 追加到文件, args: "filename": "<filename>", "text": "<text>"
19 4.delete_file: 删除文件, args: "filename": "<filename>"
20 5.read_file: 读取文件, args: "filename": "<filename>"
21 6.search_files: 搜索文件, args: "directory": "<directory>"
22 7.write_to_file: 写入文件, args: "filename": "<filename>", "text": "<text>"
23 8.google: 谷歌搜索, args: "查询": "<查询>"
24 9.Improve_code: 获取改进的代码, args: "suggestions": "<list_ofSuggestions>", "code": "<full_code_string>"
25 10.send_tweet: 发送推文, args: "tweet_text": "<tweet_text>"
26 11.browser_website: 浏览网站, args: "url": "<url>", "question": "<what_you_want_to_find_on_website>"
27 12.write_tests: 写入测试, args: "code": "<full_code_string>", "focus": "<list_of_focus_areas>"
28 13.delete_agent: 删除GPT代理, args: "key": "<key>"
29 14.get_hyperlinks: 获取文本摘要, args: "url": "<url>"
30 15.get_text_summary: 获取文本摘要, args: "url": "<url>", "question": "<question>"
31 16.list_agents: 列出 GPT 代理, args: () -> str
32 17.message_agent: 消息 GPT 代理, args: "key": "<key>", "message": "<message>"
33 18.start_agent: 启动 GPT 代理, args: "name": "<name>", "task": "<short_task_desc>", "prompt": "<prompt>"
34 19.任务完成（关机）：“task_complete”, args: “reason”: “<reason>”

```
36  
37 资源：  
38 1. 访问互联网进行搜索和信息收集。  
39 2. 长期记忆管理。  
40 3. GPT-3.5 支持的代理用于委派简单任务。  
41 4、文件输出。  
42  
43 评估：  
44 1. 不断审查和分析您的行为，以确保您发挥出最佳能力。  
45 2. 不断地建设性地自我批评你的大局行为。  
46 3. 反思过去的决定和策略以完善你的方法。  
47 4. 每个命令都有成本，所以要聪明、高效。旨在以最少的步骤完成任务。  
48 5. 将所有代码写入文件。  
49  
50 您应该仅以 JSON 格式响应，如下所述响应格式：  
51 {"想法":  
52     { "text": "想法", "reasoning": "推理", "plan": "- 简短的\n- 传达\n- 长期计划  
的列表", "criticism": "建设性的自我批评", "speak": "对用户说的想法摘要"},  
53     "command": { "name": "命令名称", "args": { "arg name": "值" } }  
54 }  
55  
56 确保响应可以被Python json.loads解析
```

自研的阿里巴巴LangEngine框架

框架背景

在阿里从事工程开发的主要编程语言还是以Java为主，并且阿里巴巴集团自身就已经具备成熟的Java中间件优势和相关的运维工具体系，从架构师角度上看，针对于企业自身构建适合自己的框架，最后决定构建以Java为语言基础的LLM开发框架。另外一方面，为了解决框架能够服务于多语言，需要专门去学习了下LangChain Python版本的代码工程和技术架构，做好技术储备，迎接AI未来，通过编写应用框架来拥抱和学习AI技术，并且能够真正在业务场景去落地。

工作原理

LangEngine 是一个基于Java用于由大语言模型支持的AI应用程序的开发框架，它也是当下最流行框架 LangChain 的Java版本。

它的特点：

- ◆ 阿里体系下基于 LangChain 原理的AI应用框架
- ◆ 引入java特色的工程模块化思路，可支持日志记录、业务监控、链式编排，实现了类流程持久化
- ◆ 面向阿里系Java工程开发同学，易学易用
- ◆ 借鉴 LangChain 生态特点，支持社区生态共建

它使应用程序能够：

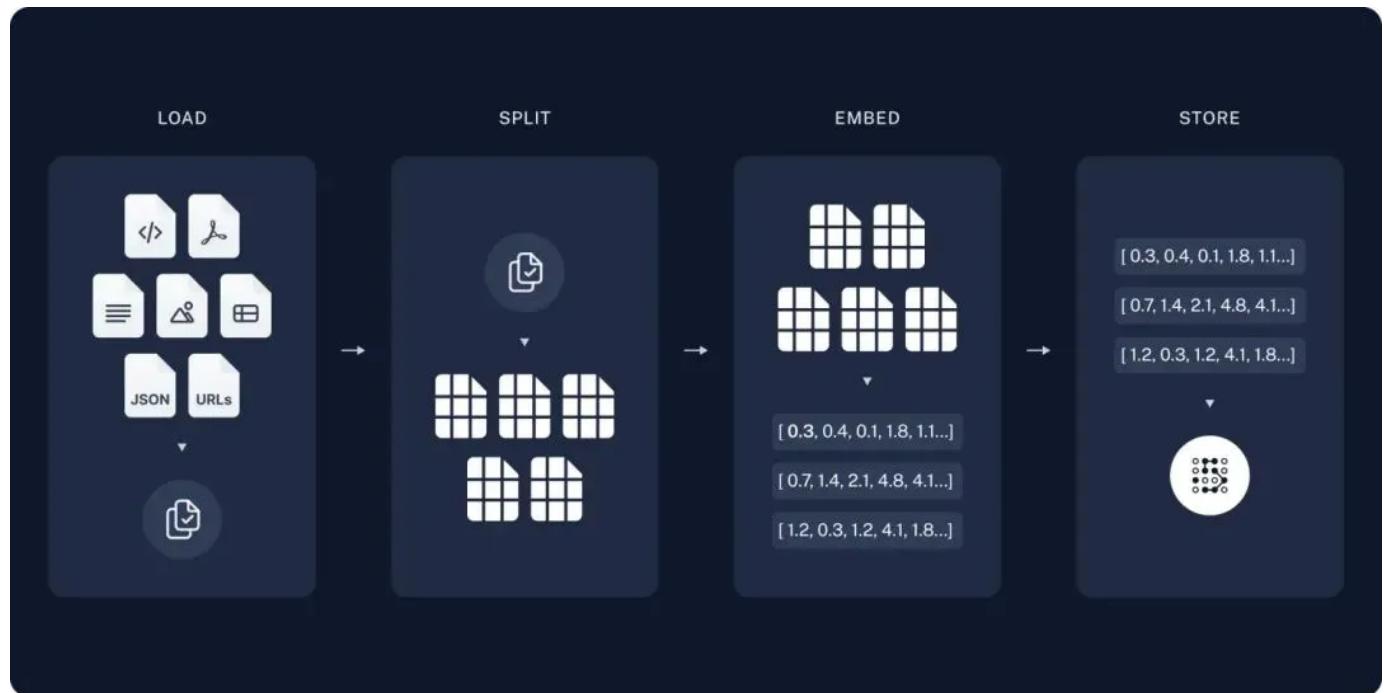
- ◆ 具有上下文感知能力：将语言模型连接到上下文来源（提示说明、Few shot示例、响应的内容等）
- ◆ 代理以及Reason：依靠语言模型进行推理（关于如何根据提供的上下文进行回答、采取什么操作等）

LangEngine的核心框架分为六大模块：Retrieval、Model I/O、Memory、Chains、Agents、Callbacks。

◆ Retrieval

Document loaders	Transform Text splitters	Text embedding models	Vector stores
<ul style="list-style-type: none">从多种不同的源加载文档，可以是文本、链接、视频等目前支持：txt、pdf、office文件、excel、duckdb、语雀、钉钉文档等格式	<ul style="list-style-type: none">从文档加载之后，可以针对长文本进行必要的分割，并对文本进行结构化处理目前支持：单词、段落、句子、中文、正则、重叠切分等	<ul style="list-style-type: none">将分割之后文本转化为一个向量（浮点数）表示，可以做一些语义搜索、相似性搜索目前支持：openai、gemini、通义千问、claude、llama3 等	<ul style="list-style-type: none">将转化后的向量，存储到向量数据库中。向量数据库负责存储嵌入数据并执行向量相似度搜索目前支持：holo、adb、tair、polardb、pinecone、memory

编辑

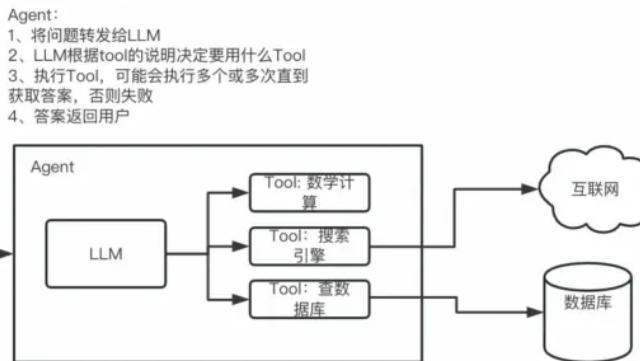


编辑

许多LLM应用程序需要特定于用户的数据，这些数据不属于模型训练集的一部分。实现这一目标的主要方法是通过检索增强生成（RAG）。在此过程中，将检索外部数据，然后在执行生成步骤时将其传递给LLM。

LangEngine 提供了 RAG 应用程序的所有构建模块 – 从简单到复杂。文档的这一部分涵盖了与检索步骤相关的所有内容 – 例如 数据的获取。虽然这听起来很简单，但实际上可能非常复杂。这包含几个关键模块。

- Agents是一个智能代理，在LangEngine中也是作为核心组件之一，它负责根据用户输入和应用场景，在一系列可用工具中选择合适的工具进行操作。
- 通俗来说，Agent就是将模型（通常为LLM）进行封装，使得它可以通过用户的输入，理解用户的意图，然后返回一个特定的动作类型和参数。这样根据动作类型和参数可以调用相关的工具来满足用户的需求。
- Agents可以根据任务的复杂性，采用不同的策略来决定如何执行操作。
- 目前支持：ZeroShotAgent、StructuredChatAgent、ConversationalAgent、SelfAskWithSearchAgent、ReActDocstoreAgent、PlanAndExecuteAgent、ExpertChainAgent、AutoGPTAgent 等等。



编辑

◆ Model I/O

Large Language Model (LLM)

- 大型语言模型，旨在理解和生成人类语言。LLM的特点是规模庞大，包含数十亿的参数，帮助它们学习语言数据中的复杂模式
- 目前支持：OpenAI、通义千问、Geinimi、Llama3、Claude、Idealab、Whale

提示词Prompts

- 作为大模型的输入，高质量Prompt对于充分发挥AI的能力至关重要
- LangEngine内置了许多Prompt模版，也包括了Zero Shot、Few Shot Template、ReAct Template 等

Output parsers

- 作为大模型的输出，针对大模型返回的信息进行结构化解析，进行更稳定的信息提取，Output parser对于Agent起到关键作用
- LangEngine针对的不同的chain/agent内置了丰富的Output parser进行清洗、降噪

Model I/O

Format

x = "foo", y = "bar"
"Does (x) like (y), and why?"

Predict

LLM
Chat Model

Parse

Parse

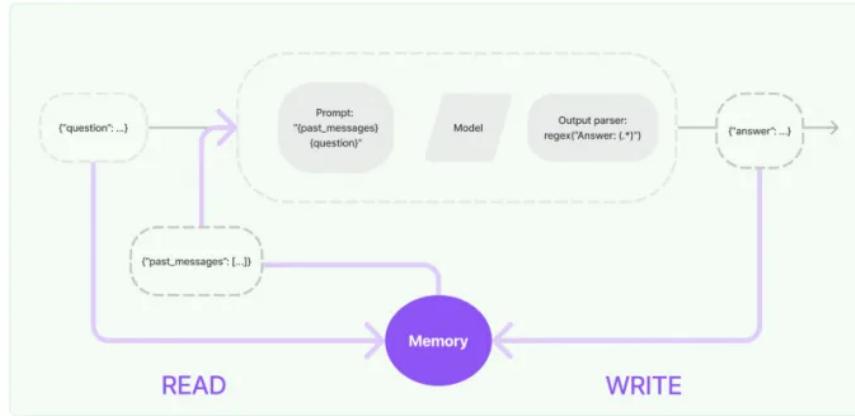
{
 "likes": True,
 "reason": "Because"
}

编辑

任何语言模型应用程序的核心元素都是模型。LangEngine 为您提供了与任何语言模型交互的构建块。

◆ Memory

- 默认情况下，Chains和Agents是无状态的，这意味着它们独立地处理每个传入的查询。在某些应用程序中，记住以前的交互非常重要，无论是短期的还是长期的。Memory的概念正是为了做到这一点而存在的。
- 作为支持大模型Chat多轮对话能力的关键，LangEngine也支持了常用的Memory：ConversationBufferMemory、ConversationBufferWindowMemory、ConversationSqliteMemory、ConversationTairMemory、ConversationRedisMemory等



编辑

大多数LLM应用程序都有对话界面。对话的一个重要组成部分是能够引用对话中先前介绍的信息。至少，对话系统应该能够直接访问过去消息的某些窗口。更复杂的系统需要有一个不断更新的世界模型，这使得它能够执行诸如维护有关实体及其关系的信息之类的事情。

将这种存储过去交互信息的能力称为“记忆”。LangEngine 提供了许多用于向系统添加内存的应用程序。这些应用程序可以单独使用，也可以无缝地合并到链中。

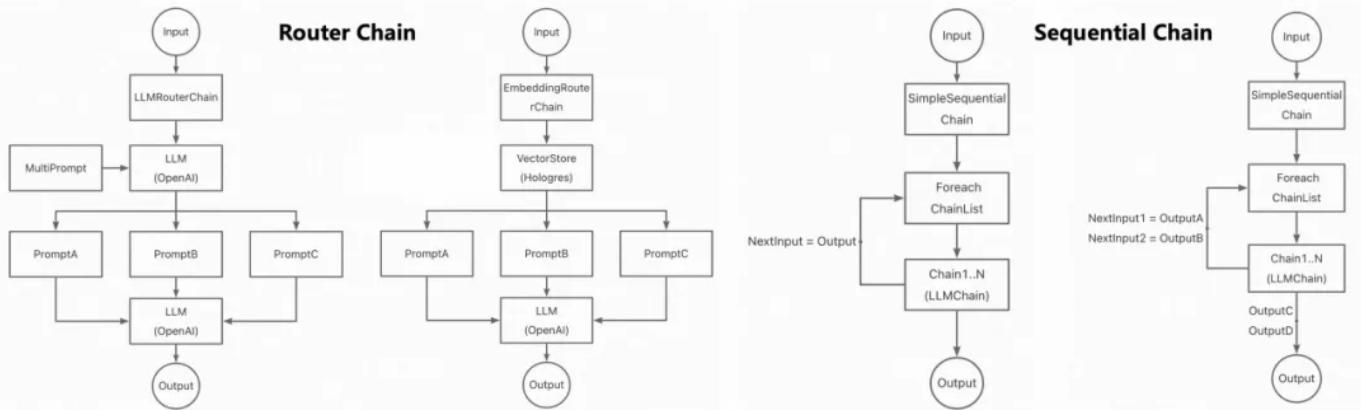
内存系统需要支持两个基本操作：读和写。回想一下，每个链都定义了一些需要某些输入的核心执行逻辑。其中一些输入直接来自用户，但其中一些输入可以来自内存。在给定的运行中，一条链将与其内存系统交互两次。

1. 在收到初始用户输入之后但在执行核心逻辑之前，链将从其内存系统中读取并增加用户输入。
2. 在执行核心逻辑之后但在返回答案之前，链会将当前运行的输入和输出写入内存，以便在将来的运行中引用它们。

◆ Chains

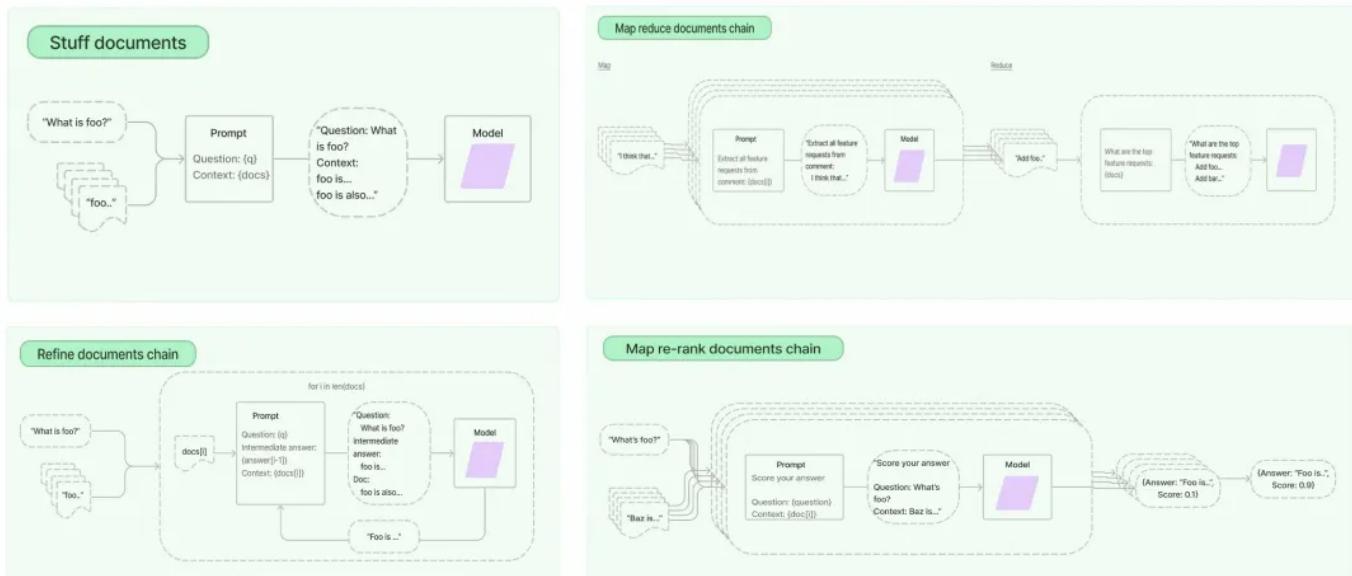
Chains在LangEngine中作为核心组件之一，它是一种将多个组件连接在一起的工具，以实现复杂的任务。我们可以把Chain理解为任务，一个Chain就是一个任务，它也可以像链条一样一个一个的执行多个链。Chains类似于流程编排，可以包含多个组件（如prompt、LLM、Agent等，也可以包含其他chain），每个组件都执行一个特定的任务，并将结果传递给下一个组件。

LLMChain是最基本的chain，它是通过用户构建prompt template，将其与用户的输入进行格式统一之后喂给大模型进行返回。大部分的agent都是通过LLMChain来与大模型进行交互。

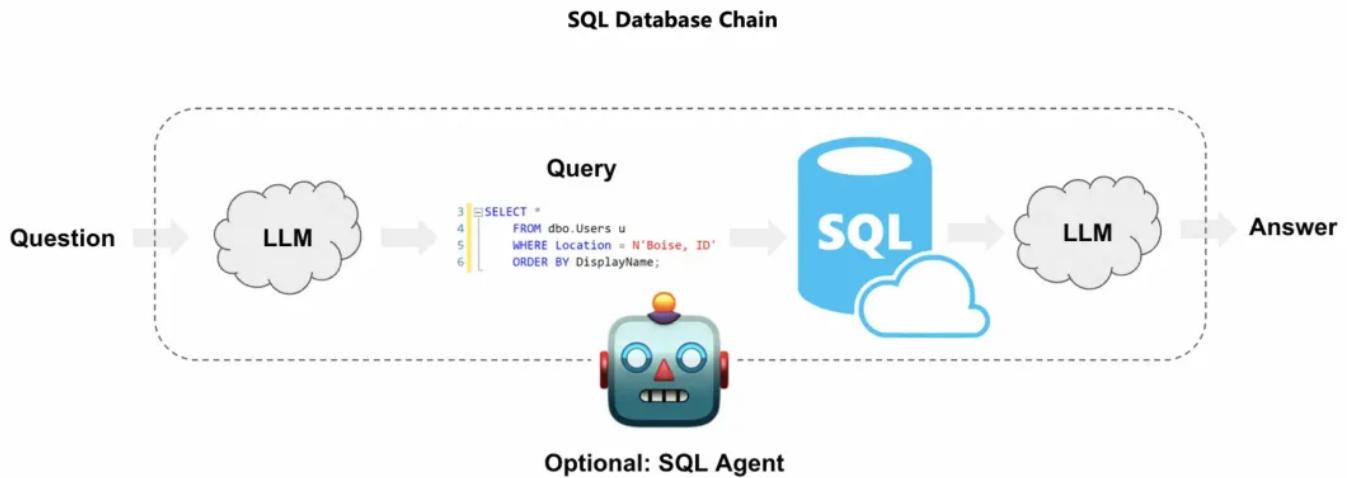


编辑

Documents Chain



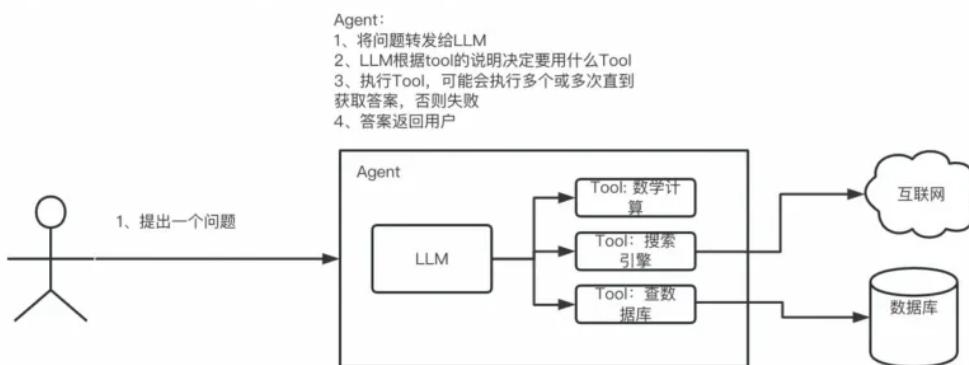
编辑



编辑

◆ Agents

- Agents是一个智能代理，在LangEngine中也是作为核心组件之一，它负责根据用户输入和应用场景，在一系列可用工具中选择合适的工具进行操作。
- 通俗来说，Agent就是将模型（通常为LLM）进行封装，使得它可以通过用户的输入，理解用户的意图，然后返回一个特定的动作类型和参数。这样根据动作类型和参数可以调用相关的工具来满足用户的需求。
- Agents可以根据任务的复杂性，采用不同的策略来决定如何执行操作。
- 目前支持：ZeroShotAgent、StructuredChatAgent、ConversationalAgent、SelfAskWithSearchAgent、ReActDocstoreAgent、PlanAndExecuteAgent、ExpertChainAgent、AutoGPTAgent 等等。



编辑

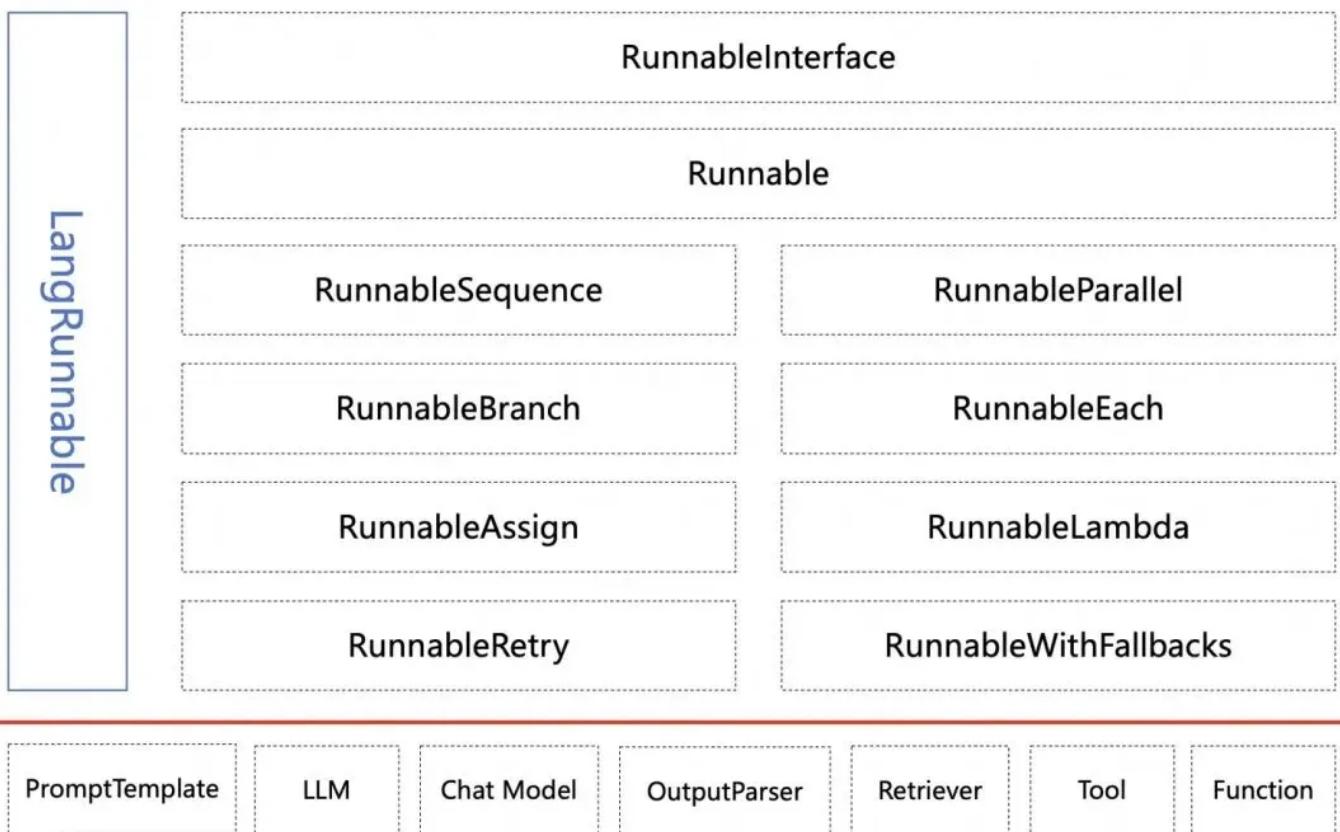
代理的核心思想是使用语言模型来选择要采取的一系列操作。在链中，一系列操作被硬编码（在代码中）。在代理中，语言模型被用作推理引擎来确定要采取哪些操作以及按什么顺序。

◆ Callbacks

LangEngine 提供了一个回调系统，允许您连接到 LLM 申请的各个阶段。这对于日志记录、监控、流传输和其他任务非常有用。

您可以使用整个 API 中可用的回调参数来订阅这些事件。该参数是处理程序对象的列表，这些对象预计实现下面更详细描述的一个或多个方法。

◆ LangRunnable



编辑

可以轻松地从基本组件构建复杂的链条。它通过提供以下功能来实现此目的：

1. 统一的接口：每个 LangRunnable 对象都实现 Runnable 接口，该接口定义一组通用的调用方法（invoke、batch、stream、ainvoke 等）。这使得 LangRunnable 对象链也可以自动支持这些调用。也就是说，每个 LangRunnable 对象链本身就是一个 LangRunnable 对象。
2. 组合原语：LangRunnable 提供了许多原语，可以轻松组合链、并行化组件、添加后备、动态配置链内部等。

为了尽可能轻松地创建自定义链，这里实现了“Runnable”协议。大多数组件都实现了 Runnable 协议。这是一个标准接口，可以轻松定义自定义链并以标准方式调用它们。标准接口包括：

- ✧ invoke：在输入上调用chain
- ✧ stream：流回响应块
- ✧ batch：批量在输入上调用chain

这些也有相应的异步方法：

- ✧ invokeAsync：在输入上异步调用chain
- ✧ streamAsync：异步流回响应块
- ✧ batchAsync：批量在输入上异步调用chain

服务集成

1.RAG服务

LangEngine Retrieval模块包含了丰富的RAG所需要的元素，让你能够基于LangEngine轻松构建RAG服务，检索增强生成(Retrieval-augmented Generation)。通过文档加载器，加载不同文档的格式类型，常用的包括PDF、Text、Excel、CSV、Markdown、Html等文件，也包括各类常见编程语言加载器，还包括在线网页加载器等等，另外在LangEngine提供了基于阿里钉钉、语雀、ODPS、TDDL等等阿里集团内部常用的加载器，最大限度的支持阿里内部的文档场景。文本分割器，实现了常用的文本分割器，包括递归词句分割的能力。接着就是Embeddings和VectorStore，除了常用的能力之外，也集成了包括通义千问的嵌入服务或者各个阿里云上各种的向量数据库服务。另外也提供常用的Query改写、Rerank服务等支持。通过Retriever、DocumentChain、OnlineSearch构建出RAG的向量检索以及在线检索服务。

2. 意图识别服务

优秀的大模型的意图识别固然重要，如果能够进一步增强意图识别能力也是锦上添花的事情。

LangEngine Chain中也提供了各种路由链、顺序链、专家链等等可以进一步提升意图识别效果的增强。

3. 工具代理调用服务

LangEngine Agent模块中内置的各种类型的Agent范式类，可以构建常见的Agent服务方式，通过LangRunnable模块可以更加灵活的自定义自己的Agent范式编排，可以构建出类似于FunctionCall、Planner、ReAct等服务方式，通过这些能力可以进行工具的代理调用。

4. GPT服务

LangEngine的GPT模块中提供了众多GPT服务，例如NL2SQL、NL2Query、NL2API、NL2Prompt。通过SQLDataChain实现了NL2SQL服务，底层也集成了阿里集团的TDDL技术进行数据库查询的支持。NL2Query在关系型数据库以及向量库中来生成Query语句，包括Field、Filter、Order等等常用query语法。通过APIChain来实现利用OpenAPI协议的标准规范，生成指定的API协议，并且最终完成API请求。NL2Prompt来实现Prompt生成的可控性以及优化后Prompt的效果。

5. 多智能体基础服务

LangEngine也汲取业内比较好的多智能体框架，例如MetaGPT、AutoGen、AutoGPT等优秀的开源框架，重新实现了一套Java版本的多智能体框架，利用该框架可以快速构建属于自己业务属性的多智能体应用。

自研的阿里巴巴AgentFramework框架

框架背景

阿里LangEngine提供了LLM应用开发中所需要各种原子化的能力，从官方角度上看，在做AI业务项目或者AI平台建设实践中，大多数时候都是基于这些原子化的能力进行组装和编排，例如既需要做可生产化

的RAG服务，也需要做Agent工作流，还需要做智能体相关建设。所以，为了满足普通的AI业务诉求，考虑开始构建这套阿里AgentFramework框架。**AgentFramework**框架的目标是把LangEngine构建好的服务，再进行进一步的组装，通过工作流以及智能体应用的方式组织起来。

工作原理

阿里AgentFramework框架是基于阿里LangEngine框架之上衍生出来真正面向AI业务和平台的Agent构建框架。

- ◆ **Core模块**: AgentFramework框架最核心的模块，负责整个流程引擎调度和基础Service SPI的串联，这里你可以选择自己任何的一款工作流框架作为AgentFramework的工作流引擎集成。
- ◆ **Model模块**: Agent领域模型，包括基础服务接口。建立以Role/Knowledge/Tool为基础的 BuiltInAgentModel（含Bean内置类和 HSF-SPI动态自定义扩展方式）以及FlowAgentModel模型支持。
- ◆ **Engine模块**: OpenAPI Pipeline实现，通过该模块可以完成类似于API网关全生命周期的管控，例如访问控制、限流策略、黑白名单、参数转换、参数映射、服务调用、服务打点等常见的API网关功能。
- ◆ **Bundle模块**: AgentFramework的业务实现模块。包括大模型调用、知识库检索、工具调用等等服务集成实现。
- ◆ **Storage模块**: AgentFramework持久化层。可插拔的持久化模块，可针对不同环境扩展各自持久化能力。
- ◆ **Parse模块**: 工作流前端UI DSL转换到BpmnXml解析器，这里采用了开源的ReactFlow技术作为工作流UI的基座。这个模块主要解决工作流中的FlowSchema转换为可执行的文件。

服务集成

1. AI Business算法工作台的容器化部署

AgentFramework框架整体可以跟随宿主应用Docker镜像打包进Serverless容器中心镜像拉取，通过容器提供了去中心化的API Gateway对接，从而实现AI应用是容器部署隔离。另外一方面，随着应用、工具、模型整体纳入到容器化服务以后，容器内部任务调度会实时观察模型的吞吐情况以及空闲时段，自动化进行相应的资源（含GPU）的切换。

2. SDK集成与私有化部署

作为AI Business为巴黎奥运会提供的智能解说助手，通过该产品解说员可以了解奥运比赛项目、历届比赛以及运动员等相关信息，在自由问答模块中，可以通过APP Framework SDK集成方式，让应用基于已集成的通用组件、模型服务和FLow编排能力，快速构建自己的AI chat功能。

阿里巴巴LangEngine框架开源地址：<https://github.com/AIDC-AI/ali-langengine>