LangGraph框架在智能体构建中的应用

一、LangGraph概述

当下计算机行业里最热门的话题便是AI Agent即智能体，将大模型嵌入到软件中赋能传统的应用开发。其中最成功的例子便是Cursor和Perplexity，一个颠覆了程序员的工作方式，另一个改变了人们搜索信息的方式。他们为业界带了了无限的想象，证明了ai agent领域的巨大潜力和机会，现在市面上几乎所有的SAAS都能在大模型的加持下脱胎换骨。然而在开发者们跃跃欲试想要在这个领域大展拳脚时一个难题摆在了所有人的面前，那就是该如何将大模型嵌入到应用中？如何让大模型来驱动整个应用？如何给大模型赋能让其能调度工具完复杂的任务？就在开发者们使劲浑身解数寻找答案时LangChain来了，作为一个工业级开源框架，他为上述问题交出了几乎满分的答卷，而当LangGraph(基于LangChain的升级)诞生后，所有人都知道属于开发者独立开发AI Agent的时代到来了！

二、工作原理

LangGraph顾名思义，其灵感来源于“图”数据结构，他将智能体的工作流通过图的方式定义，将一个复杂的任务拆解成一个个独立的节点，由大模型根据当前的状态决定路由到哪个节点。每一个节点都是高度定制化，模块化，开发者可以全权定义当前节点的逻辑，这个节点可以用来执行外部工具，也可以用来和一个专家模型等，大模型可以根据输出动态路由到指定节点，开发者可以自由编排工作流。

除此之外Langgraph还支持multi agent的开发，它可以在更高层面将agent抽象成一个节点来定制多智能体的工作流，它也可以定义共享状态来让多智能体之间进行信息的高效，无缝传递与协同工作，为构建更庞大、更智能的系统铺平了道路。

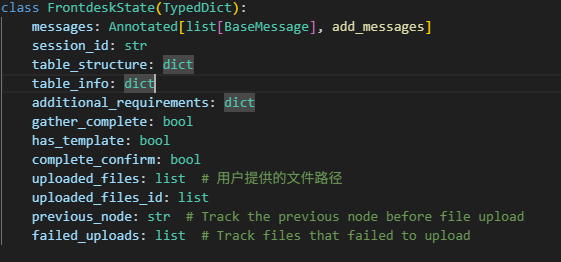
三、LangGraph的优势

LangGraph以图结构重塑智能体开发范式，通过模块化节点、动态路由和共享状态，不仅让单智能体构建更直观灵活，更解锁了复杂多智能体协作的潜力。他降低了构建强大AI Agent的门槛，是开发者迎接智能体时代不可或缺的利器与基石。

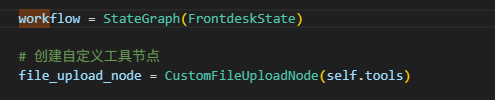
1. 演示
2. 安装Langgraph



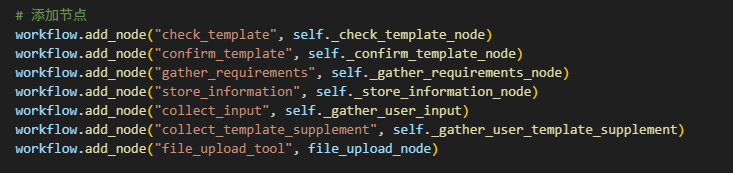
1. 定义graph state



这个状态会在节点之间进行传递，以实现信息的交互

1. 创建Langgraph

将之前创建的状态state作为参数，通过StateGraph()创建graph agent实例

1. 调用add\_node()函数，在graph agent中添加自定义的节点
2. 设置入口节点，结束节点，并添加节点之间的路由





1. 调用compile()函数来编译创建好的graph并配置输出信息存储点



Langgraph提供了多种执行函数如invoke(), stream()其不同在于输出方式， invoke()会在整个工作流执行完后返回结果，而stream()函数会返回流数据

