Foundations of Semantic Kernel

We have already told you about LLMs and how to connect to Azure OpenAI Service using .NET and Python SDK. Next we will enter the world of Semantic Kernel. In 2023, we will have a lot of frameworks based on LLMs born. Why should we choose Semantic Kernel? What are the advantages and disadvantages of Semantic Kernel? And how do you use Semantic Kernel as a traditional developer? I will explain it in detail in this chapter.

What is Semantic Kernel

Semantic Kernel is a lightweight open source framework. Through Semantic Kernel, you can quickly use different programming languages (C#/Python/Java) combined with LLMs (OpenAI, Azure OpenAI, Hugging Face and other models) to build intelligent applications. After we entered generative AI, the way humans and machines communicate has changed a lot. We can use natural language to complete conversations with machines, and the threshold has been lowered a lot. Combining hint engineering and LLMs, we can complete different businesses at a lower cost. But how to introduce prompt engineering and LLMs into projects? We need a framework like Semantic Kernel as the basis for opening the door to intelligence. In May 2023, Microsoft CTO Kevin Scott proposed the concept of Copilot Stack, with artificial intelligence orchestration at its core. Semantic Kernel has the ability to be combined with LLMs and plug-ins composed of various prompt engineerings/software, so it is also regarded as the best practice of Copilot Stack. Through Semantic Kernel, you can easily build solutions based on Copilot Stack, and it can also be seamlessly connected to traditional projects.

Semantic Kernel vs LangChain

We have no choice but to make some objective comparisons. After all, LangChain has more user groups. It is undeniable that LangChain now has more features than Semantic Kernel in practical scenarios, especially in the entry-level reference examples. Let's make a more comprehensive comparison:

LangChain is an open source framework based on Python and Javascript and contains many prefabricated components. Developers can complete the development of intelligence applications without writing additional prompt engineerings. Especially in complex application scenarios, developers can quickly integrate multiple predefined components to complete the synthesis. From a development perspective, it is more suitable for developers with a foundation in data science or artificial intelligence.

Semantic Kernel You can use open source frameworks based on C#, Python, and Java. The greater advantage lies in engineering. After all, it is more like a programming paradigm. Traditional developers can quickly master the framework for application development, and can better combine customized plug-ins and prompt projects to complete the enterprise's customized business intelligence work.

The two have a lot in common, and both are still in version iteration. We need to make a choice based on the team structure, technology stack, and application scenarios.

Features of Semantic Kernel

1. **Powerful Plugins** - You can solve intelligentce business problems by combining custom/predefined plugins. Let traditional codes and intelligentce plugins work together to flexibly connect to application

scenarios, simplifying the process of transforming traditional applications into intelligent ones.

2. **Multi-model support** - Configure the "brain" for your intelligent application, which can be from Azure OpenAI Service, OpenAI, and various offline models on Hugging Face. Through the linker, you can quickly connect to different "brains" to make your application more smarter.

3. **Various connectors** - In addition to linking the "brain", the connectors can also link to vector databases, various business software, and different business middleware, allowing more business scenarios to enter the intelligent world possible

4. **Convenient Development** - Simple and easy to use, developers can get started at zero cost

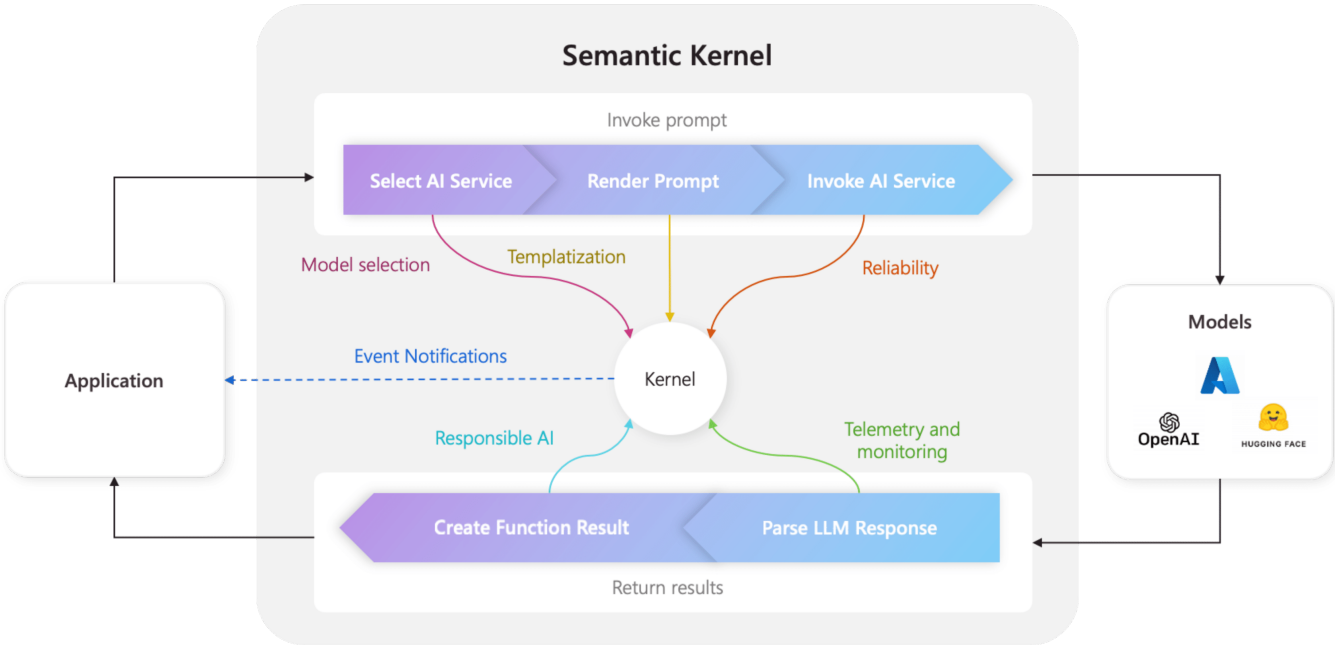
Disadvantages of Semantic Kernel

After all, LLMs are still developing, with many new models being added, many new functions, and new concepts being introduced. Open source frameworks such as Semantic Kernel and LangChain are working hard to adapt to this new Moore's Law, but there will be uncertain changes in version iterations. Therefore, when using it, developers need to pay more attention to the change log on the corresponding GitHub Repo.

In addition, Semantic Kernel needs to take into account multiple programming languages, so the progress is inconsistent, which will also lead to the choice of Semantic Kernel among people with different technology stacks.

Semantic Kernel's Kernel

If Semantic Kernel is regarded as Copilot Stack best practice, then Kernel is the center of AI orchestration and is also mentioned in the official documentation. Kernel can be linked with different plug-ins, services, logs and different models. All Semantic Kernel applications start with the Kernel.



Build a simple translation project with Semantic Kernel

After talking about some basic knowledge, we started to learn how to introduce Semantic Kernel into .NET and Python project projects. We use four steps to complete the implementation of a translation

Step 1: Import the Semantic Kernel library

.NET

Note: We are using the latest Semantic Kernel 1.0.1 version here, and using Polyglot Notebook to complete related learning

```
#r "nuget: Microsoft.SemanticKernel, *-*"
```

Python

Note: We are using the latest version of Semantic Kernel 0.4.3.dev0 here, and using Notebook to complete related learning

```
! pip install semantic-kernel -U
```

Step 2: Create Kernel object

Note here that we need to put the Endpoint, Key and Deployment Name of the model related to Azure OpenAI Service in a file to facilitate calling and setting. In .NET environment I set it in Settings.cs environment, in Python environment I set it in .env environment

.NET

```
using Microsoft.SemanticKernel;
using Microsoft.SemanticKernel.Connectors.OpenAI;


Kernel kernel = Kernel.CreateBuilder()
            .AddAzureOpenAIChatCompletion("Your Azure OpenAI Service
Deployment Name" , "Your Azure OpenAI Service Endpoint", "Your Azure
OpenAI Service API Key")
            .Build();
```

Python

```
import semantic_kernel as sk
import semantic_kernel.connectors.ai.open_ai as skaoai
```

```
kernel = sk.Kernel()
deployment, api_key, endpoint = sk.azure_openai_settings_from_dot_env()
kernel.add_chat_service("azure_chat_competion_service",
skaoai.AzureChatCompletion(deployment,endpoint,api_key=api_key,api_version
= "2023-07-01-preview"))
```

Step 3: Import plugins

In Semantic Kernel, we have different plugins. Users can use predefined plugins or custom plugins. If you want to know more, you can pay attention to the next chapter, where we will explain the use of plugins in detail. In this example, we are using a custom plug-in, which is already in the plugins directory.

.NET

```
var plugin =
kernel.CreatePluginFromPromptDirectory(Path.Combine(pluginDirectory,
"TranslatePlugin"));
```

Python

```
pluginFunc =
kernel.import_semantic_skill_from_directory(base_plugin,"TranslatePlugin")
```

Step 4: Running

.NET

```
var transalteContent = await kernel.InvokeAsync( plugin["Basic"],new()
{["input"] = "你好，我是你的 AI 编排助手 - Semantic Kernel"});

transalteContent.GetValue();
```

Python

```
translateFunc = pluginFunc["Basic"]

result = translateFunc("你好，我是你的 AI 编排助手 - Semantic Kernel")
```

you can visit

.NET Samples Click here

Python Samples Click here

Summary

How did you feel when you first came into contact with Semantic Kernel? In this chapter, you learned the basics of Semantic Kernel and related knowledge. And understanding the comparison between Semantic Kernel and LangChain, as well as the related advantages and disadvantages, will be helpful to you in the process of project implementation. We have also completed a translation through Semantic Kernel in four steps, which is a hello world in the era of large models, giving you confidence when getting started. Next, you can open the next chapter for advanced learning to learn more about Semantic Kernel.