160 lines (123 loc) · 8.72 KB

# Semantic Kernel


> ℹ️ **NOTE**: This project is just like AI and will evolve quickly. We invite you to join us in developing the Semantic Kernel together! Please contribute by using GitHub Discussions, opening GitHub Issues, sending us PRs, joining our Discord community.

**Semantic Kernel (SK)** is a lightweight SDK enabling integration of AI Large Language Models (LLMs) with conventional programming languages. The SK extensible programming model combines natural language **semantic functions**, traditional code **native functions**, and **embeddings-based memory** unlocking new potential and adding value to applications with AI.

SK supports prompt templating, function chaining, vectorized memory, and intelligent planning capabilities out of the box.

Semantic Kernel supports and encapsulates several design patterns from the latest in AI research, such that developers can infuse their applications with plugins like prompt chaining, recursive reasoning, summarization, zero/few-shot learning, contextual memory, long-term memory, embeddings, semantic indexing, planning, retrieval-augmented generation and accessing external knowledge stores as well as your own data.

By joining the SK community, you can build AI-first apps faster and have a front-row peek at how the SDK is being built. SK has been released as open-source so that more pioneering developers can join us in crafting the future of this landmark moment in the history of computing.

## Get Started with Semantic Kernel ⚡

Semantic Kernel is available to explore AI and build apps with C# and Python:

| | |
|---|---|
|  Using Semantic Kernel in C# |  Using Semantic Kernel in Python |

See the Feature Matrix to see a breakdown of feature parity between C# and Python.

The quickest way to get started with the basics is to get an API key (OpenAI or Azure OpenAI) and to run one of the C# or Python console applications/scripts:

### For C#:

1. Create a new console app.
2. Add the semantic kernel nuget `Microsoft.SemanticKernel`.
3. Copy the code from here into the app `Program.cs` file.
4. Replace the configuration placeholders for API key and other params with your key and settings.
5. Run with `F5` or `dotnet run`

### For Python:

1. Install the pip package: `python -m pip install semantic-kernel`.
2. Create a new script e.g. `hello-world.py`.
3. Store your API key and settings in an `.env` file as described here.
4. Copy the code from here into the `hello-world.py` script.
5. Run the python script.

## Sample apps ⚡

The repository includes some sample applications, with a React frontend and a backend web service using Semantic Kernel.

Follow the links for more information and instructions about running these apps.

| | |
|---|---|
| Simple chat summary | Use ready-to-use plugins and get plugins into your app easily. |
| Book creator | Use planner to deconstruct a complex goal and envision using the planner in your app. |

| | |
|---|---|
| Authentication and APIs | Use a basic connector pattern to authenticate and connect to an API and imagine integrating external data into your app's LLM AI. |
| GitHub repository Q&A | Use embeddings and memory to store recent data and allow you to query against it. |
| Copilot Chat Sample App | Build your own chat experience based on Semantic Kernel. |

**Requirements:**

- You will need an Open AI API Key or Azure Open AI service key to get started.
- Azure Functions Core Tools are required to run the kernel as a local web service, used by the sample web apps.
- .NET 6 SDK or .NET 7 SDK
- Yarn is used for installing web apps' dependencies.

## Deploy Semantic Kernel to Azure in a web app service ☁️

Getting Semantic Kernel deployed to Azure as web app service is easy with one-click deployments. Click here to learn more on how to deploy to Azure.

## Jupyter Notebooks ⚡

For a more hands-on overview, you can also check out the C# and Python Jupyter notebooks, starting from here:

- Getting Started with C# notebook
- Getting Started with Python notebook

**Requirements:** C# notebooks require .NET 7 and the VS Code Polyglot extension.

## Contributing and Community

We welcome your contributions and suggestions to SK community! One of the easiest ways to participate is to engage in discussions in the GitHub repository. Bug reports and fixes are welcome!

For new features, components, or extensions, please open an issue and discuss with us before sending a PR. This is to avoid rejection as we might be taking the core in a different direction, but also to consider the impact on the larger ecosystem.

To learn more and get started:

- Read the documentation
- Learn how to contribute to the project
- Join the Discord community
- Attend regular office hours and SK community events
- Follow the team on our blog

## Code of Conduct

This project has adopted the Microsoft Open Source Code of Conduct. For more information see the Code of Conduct FAQ or contact opencode@microsoft.com with any additional questions or comments.

## License



Licensed under the MIT license.