Integrating with function calling

Introduction

This lesson will cover:

- Explain what is function calling and its use cases.
- Creating a function call using Azure OpenAI.
- How to integrate a function call into an application.

Learning Goals

By the end of this lesson, you will be able to:

- Explain the purpose of using function calling.
- Setup Function Call using the Azure OpenAI Service.
- Design effective function calls for your application's use case.

Scenario: improving our chatbot with functions

For this lesson, we want to build a feature for our education startup that allows users to use a chatbot to find technical courses. We will recommend courses that fit their skill level, current role and technology of interest.

To complete this scenario we will use a combination of:

- `Azure OpenAI` to create a chat experience for the user.
- `Microsoft Learn Catalog API` to help users find courses based on the request of the user.
- `Function Calling` to take the user's query and send it to a function to make the API request.

To get started, let's look at why we would want to use function calling in the first place:

Why Function Calling

Before function calling, responses from an LLM were unstructured and inconsistent. Developers were required to write complex validation code to make sure they are able to handle each variation of a response. Users could not get answers like "What is the current weather in Stockholm?". This is because models were limited to the time the data was trained on.

Function Calling is a feature of the Azure OpenAI Service to overcome to the following limitations:

- **Consistent response format**. If we can better control the response format we can more easily integrate the response downstream to other systems.
- **External data**. Ability to use data from other sources of an application in a chat context.

Use Cases for using function calls

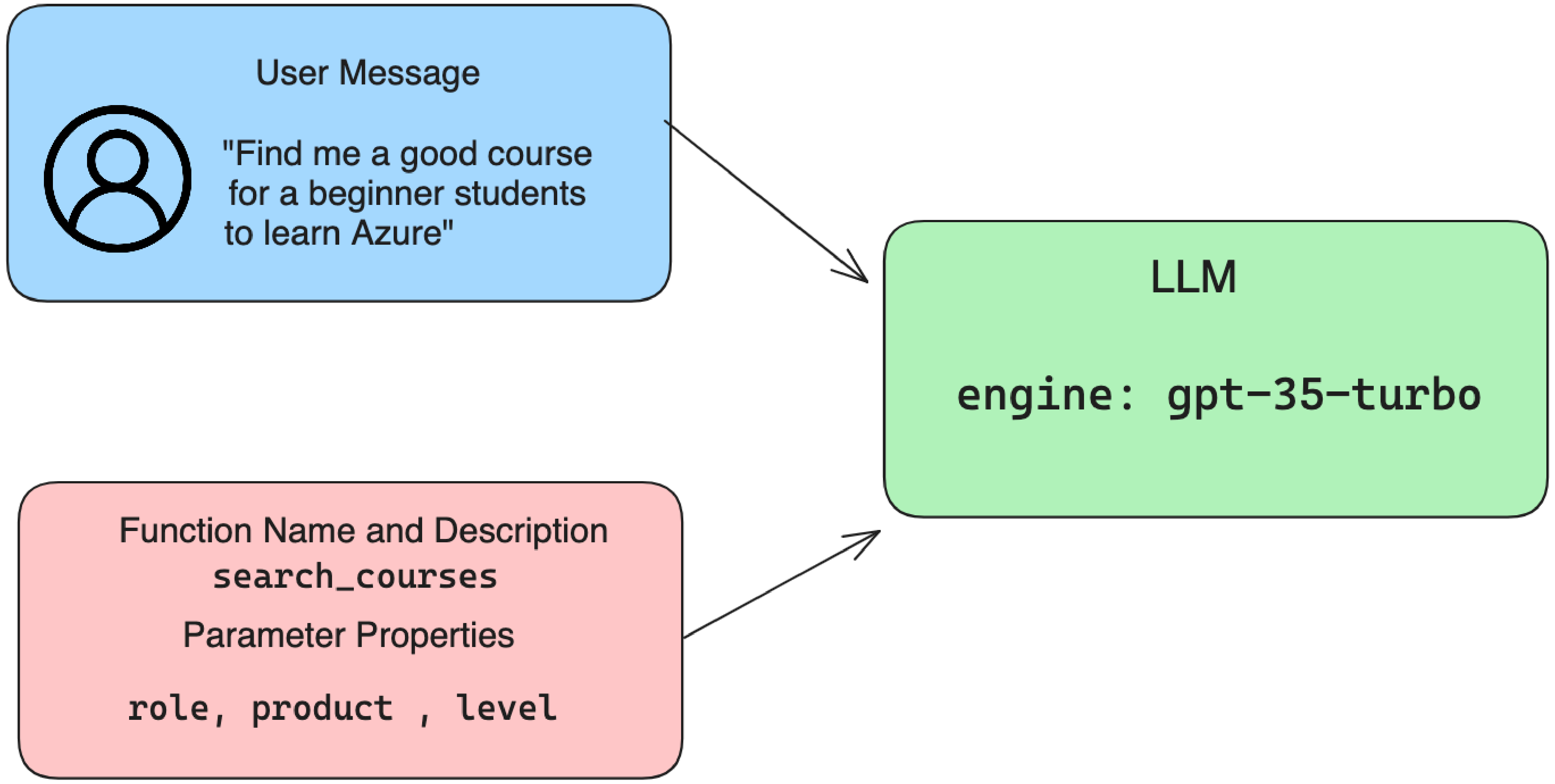
There are many different use cases where function calls can improve your app like:

- **Calling External Tools**. Chatbots are great at providing answers to questions from users. By using function calling, the chatbots can use messages from users to complete certain tasks.
- **Create API or Database Queries**. Users can find information using natural language that gets converted into a formatted query or API request.
- **Creating Structured Data**. Users can take a block of text or CSV and use the LLM to extract important information from it.

Creating Your First Function Call

The process of creating a function call includes 3 main steps:

1. **Calling** the Chat Completions API with a list of your functions and a user message.
2. **Reading** the model's response to perform an action ie execute a function or API Call.
3. **Making** another call to Chat Completions API with the response from your function to use that information to create a response to the user.


User Message
"Find me a good course for a beginner students to learn Azure"
LLM
engine: gpt-35-turbo
Function Name and Description
search_courses
Parameter Properties
role, product , level

Step 1 - creating messages

The first step is to create a user message. This can be dynamically assigned by taking the value of a text input or you can assign a value here. If this is your first time working with the Chat Completions API, we need to define the `role` and the `content` of the message.

The `role` can be either `system` (creating rules), `assistant` (the model) or `user` (the end-user). For function calling, we will assign this as `user` and an example question.

```
messages= [ {"role": "user", "content": "Find me a good course for a beginner student to learn Azure."} ]
```

By assigning different roles, it's made clear to the LLM if it's the system saying something or the user, which helps to build a conversation history that the LLM can build upon.

Step 2 - creating functions

Next, we will define a function and the parameters of that function. We will use just one function here called `search_courses` but you can create multiple functions.

> **Important** : Functions are included in the system message to the LLM and will be included in the amount of available tokens you have available.

Below, we create the functions as an array of items. Each item is a function and has properties `name`, `description` and `parameters`:

```
functions = [
   {
      "name":"search_courses",
      "description":"Retrieves courses from the search index based on the parameters provided",
      "parameters":{
         "type":"object",
         "properties":{
            "role":{
               "type":"string",
               "description":"The role of the learner (i.e. developer, data scientist, student, etc.)"
            },
            "product":{
               "type":"string",
               "description":"The product that the lesson is covering (i.e. Azure, Power BI, etc.)"
            },
```

Let's describe each function instance more in detail below:

- `name` - The name of the function that we want to have called.
- `description` - This is the description of how the function works. Here it's important to be specific and clear.
- `parameters` - A list of values and format that you want the model to produce in its response. The parameters array consists of items where item have the following properties:
    i. `type` - The data type of the properties will be stored in.
    ii. `properties` - List of the specific values that the model will use for its response
        a. `name` - The key is the name of the property that the model will use in its formatted response, for example, `product`.
        b. `type` - The data type of this property, for example, `string`.
        c. `description` - Description of the specific property.

Step 3 - Making the function call

After defining a function, we now need to include it in the call to the Chat Completion API. We do this by adding `functions` to the request. In this case `functions=functions`.

There is also an option to set `function_call` to `auto`. This means we will let the LLM decide which function should be called based on the user message rather than assigning it ourselves.