Integrating Python LLM Code with Unity using a Web API

Overview:

If you have methods written in Python for connecting to an LLM and want to integrate them with Unity, using a Web API is a robust and efficient solution.

This approach allows you to keep your Python code unchanged while enabling Unity to interact with it via HTTP requests.

Steps to Implement:

1. Set up the Python Web API:

Use a framework like Flask or FastAPI to create an API endpoint that Unity can call.

Example Code:

```python
from flask import Flask, request, jsonify
```

```
app = Flask(__name__)

@app.route('/generate', methods=['POST'])
def generate_text():
    data = request.json
    prompt = data.get('prompt', '')
    # Call your LLM function here
    response = your_llm_method(prompt)
    return jsonify({"response": response})

if __name__ == '__main__':
    app.run(port=5000)
```

- Replace `your_llm_method(prompt)` with your actual method to connect to the LLM.
- The API will listen on port `5000` by default.

---

2. Make HTTP Requests from Unity:

Unity's `UnityWebRequest` can send POST requests to the Python API.

Example Code:

```csharp
using UnityEngine;
using UnityEngine.Networking;
using System.Collections;

public class LLMClient : MonoBehaviour
{
    IEnumerator GetLLMResponse(string prompt)
    {
        string url = "http://127.0.0.1:5000/generate";
        WWWForm form = new WWWForm();
        form.AddField("prompt", prompt);

        using (UnityWebRequest www = UnityWebRequest.Post(url, form))
        {
            yield return www.SendWebRequest();

            if (www.result != UnityWebRequest.Result.Success)
```

```
        {
            Debug.LogError(www.error);
        }
        else
        {
            string response = www.downloadHandler.text;
            Debug.Log("Response: " + response);
        }
    }
  }
}
```

- Replace `"http://127.0.0.1:5000/generate"` with the actual URL of your Python server if hosted remotely.
- Use `StartCoroutine(GetLLMResponse("Your prompt here"))` to call the LLM from Unity.

Benefits of This Approach:

1. **Separation of Concerns**:

- Keep your Python LLM code independent of Unity's C# code.

- Makes the system modular and easier to maintain.

2. **Scalability:**

- You can deploy the Python API to a remote server and enable multiple Unity clients to connect to it.

3. **Ease of Debugging:**

- Python and Unity codebases remain separate, making it easier to debug each component individually.

4. **Cross-Platform:**

- Works seamlessly across different platforms supported by Unity.

Next Steps:

- Deploy your Python Web API to a production server (e.g., AWS, Azure, or Google Cloud) for broader access.

- Secure your API using authentication methods like API keys or OAuth if needed.