Build Agents Models for Agents

Ollama model host for ADK agents

Supported in ADK Python v0.1.0

Ollama is a tool that allows you to host and run open-source models locally. ADK integrates with Ollama-hosted models through the LiteLLM model connector library.

Get started

Use the LiteLLM wrapper to create agents with Ollama-hosted models. The following code example shows a basic implementation for using Gemma open models with your agents:

```
root_agent = Agent(
    model=LiteLlm(model="ollama_chat/gemma3:latest"),
    name="dice_agent",
    description=(
        "hello world agent that can roll a dice of 8 sides
and check prime"
        " numbers."
    ),
    instruction="""
      You roll dice and answer questions about the outcome of
the dice rolls.
    """,
    tools=[
        roll_die,
        check_prime,
    ],
)
```

Warning: Use `ollama_chat` interface

Make sure you set the provider `ollama_chat` instead of `ollama`. Using `ollama` can result in unexpected behaviors such as infinite tool call loops and ignoring previous context.

Use `OLLAMA_API_BASE` environment variable

Although you can specify the `api_base` parameter in LiteLLM for generation, as of v1.65.5, the library relies on the environment variable for other API calls. Therefore, you should set the `OLLAMA_API_BASE` environment variable for your Ollama server URL to ensure all requests are routed correctly.

```
export OLLAMA_API_BASE="http://localhost:11434"
adk web
```

Model choice

If your agent is relying on tools, make sure that you select a model with tool support from Ollama website. For reliable results, use a model with tool support. You can check tool support for the model using the following command:

```
ollama show mistral-small3.1
  Model
    architecture        mistral3
    parameters          24.0B
    context length      131072
    embedding length    5120
    quantization        Q4_K_M

  Capabilities
    completion
    vision
    tools
```

You should see **tools** listed under capabilities. You can also look at the template the model is using and tweak it based on your needs.

```
ollama show --modelfile llama3.2 > model_file_to_modify
```

For instance, the default template for the above model inherently suggests that the model shall call a function all the time. This may result in an infinite loop of function calls.

```
Given the following functions, please respond with a JSON for
a function call
with its proper arguments that best answers the given prompt.

Respond in the format {"name": function name, "parameters":
dictionary of
argument name and its value}. Do not use variables.
```

You can swap such prompts with a more descriptive one to prevent infinite tool call loops, for instance:

```
Review the user's prompt and the available functions listed
below.

First, determine if calling one of these functions is the
most appropriate way
to respond. A function call is likely needed if the prompt
asks for a specific
action, requires external data lookup, or involves
calculations handled by the
functions. If the prompt is a general question or can be
answered directly, a
function call is likely NOT needed.

If you determine a function call IS required: Respond ONLY
with a JSON object in
the format {"name": "function_name", "parameters":
{"argument_name": "value"}}.
Ensure parameter values are concrete, not variables.

If you determine a function call IS NOT required: Respond
directly to the user's
prompt in plain text, providing the answer or information
requested. Do not
output any JSON.
```

Then you can create a new model with the following command:

```
ollama create llama3.2-modified -f model_file_to_modify
```

Use OpenAI provider

Alternatively, you can use `openai` as the provider name. This approach requires setting the `OPENAI_API_BASE=http://localhost:11434/v1` and `OPENAI_API_KEY=anything` env variables instead of `OLLAMA_API_BASE`. Note that the `API_BASE` value has `/v1` at the end.

```
root_agent = Agent(
    model=LiteLlm(model="openai/mistral-small3.1"),
    name="dice_agent",
    description=(
        "hello world agent that can roll a dice of 8 sides
and check prime"
        " numbers."
    ),
    instruction="""
      You roll dice and answer questions about the outcome of
the dice rolls.
    """,
    tools=[
        roll_die,
        check_prime,
    ],
)
```

```
export OPENAI_API_BASE=http://localhost:11434/v1
export OPENAI_API_KEY=anything
adk web
```

Debugging

You can see the request sent to the Ollama server by adding the following in your agent code just after imports.

```
import litellm
litellm._turn_on_debug()
```

Look for a line like the following:

```
Request Sent from LiteLLM:
curl -X POST \
http://localhost:11434/api/chat \
-d '{'model': 'mistral-small3.1', 'messages': [{'role':
'system', 'content': ...
```