swarms.agents

1. Introduction

`AbstractAgent` is an abstract class that serves as a foundation for implementing AI agents. An agent is an entity that can communicate with other agents and perform actions. The `AbstractAgent` class allows for customization in the implementation of the `receive` method, enabling different agents to define unique actions for receiving and processing messages.

`AbstractAgent` provides capabilities for managing tools and accessing memory, and has methods for running, chatting, and stepping through communication with other agents.

2. Class Definition

```python
class AbstractAgent:
    """An abstract class for AI agent.

    An agent can communicate with other agents and perform actions.
    Different agents can differ in what actions they perform in the `receive` method.

    Agents are full and completed:

    Agents = llm + tools + memory
    """
```

```
def __init__(self, name: str):
    """
    Args:
        name (str): name of the agent.
    """
    self._name = name

@property
def name(self):
    """Get the name of the agent."""
    return self._name

def tools(self, tools):
    """init tools"""

def memory(self, memory_store):
    """init memory"""

def reset(self):
    """(Abstract method) Reset the agent."""

def run(self, task: str):
    """Run the agent once"""

def _arun(self, taks: str):
    """Run Async run"""
```

```
    def chat(self, messages: List[Dict]):
        """Chat with the agent"""

    def _achat(self, messages: List[Dict]):
        """Asynchronous Chat"""

    def step(self, message: str):
        """Step through the agent"""

    def _astep(self, message: str):
        """Asynchronous step"""
```

3. Functionality and Usage

The `AbstractAgent` class represents a generic AI agent and provides a set of methods to interact with it.

To create an instance of an agent, the `name` of the agent should be specified.

Core Methods

1. `reset`

The `reset` method allows the agent to be reset to its initial state.

```python
agent.reset()
```

2. `run`

The `run` method allows the agent to perform a specific task.

```python
agent.run("some_task")
```

3. `chat`

The `chat` method enables communication with the agent through a series of messages.

```python
messages = [{"id": 1, "text": "Hello, agent!"}, {"id": 2, "text": "How are you?"}]
agent.chat(messages)
```

4. `step`

The `step` method allows the agent to process a single message.

```python
agent.step("Hello, agent!")
```

Asynchronous Methods

The class also provides asynchronous variants of the core methods.

Additional Functionality

Additional functionalities for agent initialization and management of tools and memory are also provided.

```python
agent.tools(some_tools)
agent.memory(some_memory_store)
```

4. Additional Information and Tips

When implementing a new agent using the `AbstractAgent` class, ensure that the `receive` method is overridden to define the specific behavior of the agent upon receiving messages.

5. References and Resources

For further exploration and understanding of AI agents and agent communication, refer to the

relevant literature and research on this topic.