MajorityVoting Module Documentation

The `MajorityVoting` module provides a mechanism for performing majority voting among a group of agents. Majority voting is a decision rule that selects the option which has the majority of votes. This is particularly useful in systems where multiple agents provide responses to a query, and the most common response needs to be identified as the final output.

Key Concepts

- **Majority Voting**: A method to determine the most common response from a set of answers.
- **Agents**: Entities (e.g., models, algorithms) that provide responses to tasks or queries.
- **Output Parser**: A function that processes the responses from the agents before performing the majority voting.

Function Definitions

Function: `majority_voting`

Performs majority voting on a list of answers and returns the most common answer.

Parameters

Parameter	Type	Description
`answers`	`List[str]`	A list of answers from different agents.

Returns

Return Value	Type	Description
`answer`	`str`	The most common answer in the list. If the list is empty, returns "I don't know".

Class Definitions

Class: `MajorityVoting`

Class representing a majority voting system for agents.

Parameters

Parameter	Type	Description
`agents`	`List[Agent]`	A list of agents to be used in the majority voting system.
`output_parser`	`Callable`	A function used to parse the output of the agents. If not provided, the default `majority_voting` function is used.
`autosave`	`bool`	A boolean indicating whether to autosave the conversation to a file. Default is `False`.
`verbose`	`bool`	A boolean indicating whether to enable verbose logging. Default is `False`.

Method: `__init__`

Initializes the `MajorityVoting` system.

Parameters

Parameter	Type	Description
`agents`	`List[Agent]`	A list of agents to be used in the majority voting system.
`output_parser`	`Callable`	A function used to parse the output of the agents. Default is the `majority_voting` function.
`autosave`	`bool`	A boolean indicating whether to autosave the conversation to a file. Default is `False`.
`verbose`	`bool`	A boolean indicating whether to enable verbose logging. Default is `False`.
`args`	`tuple`	Additional positional arguments.
`kwargs`	`dict`	Additional keyword arguments.

Method: `run`

Runs the majority voting system and returns the majority vote.

Parameters

Parameter	Type	Description
`task`	`str`	The task to be performed by the agents.
`args`	`tuple`	Variable length argument list.

| `kwargs` | `dict` | Arbitrary keyword arguments. |

Returns

Return Value	Type	Description
`results`	`List[Any]`	The majority vote.

Usage Examples

Example 1: Basic Majority Voting

```python
from swarms.structs.agent import Agent
from swarms.structs.majority_voting import MajorityVoting

# Initialize agents
agents = [
    Agent(
        agent_name="Devin",
        system_prompt=(
            "Autonomous agent that can interact with humans and other"
            " agents. Be Helpful and Kind. Use the tools provided to"
            " assist the user. Return all code in markdown format."
        ),
        llm=llm,
```

```
    max_loops="auto",
    autosave=True,
    dashboard=False,
    streaming_on=True,
    verbose=True,
    stopping_token="<DONE>",
    interactive=True,
    tools=[terminal, browser, file_editor, create_file],
    code_interpreter=True,
),
Agent(
    agent_name="Codex",
    system_prompt=(
        "An AI coding assistant capable of writing and understanding"
        " code snippets in various programming languages."
    ),
    llm=llm,
    max_loops="auto",
    autosave=True,
    dashboard=False,
    streaming_on=True,
    verbose=True,
    stopping_token="<DONE>",
    interactive=True,
    tools=[terminal, browser, file_editor, create_file],
    code_interpreter=True,
```

```
    ),
    Agent(
        agent_name="Tabnine",
        system_prompt=(
            "A code completion AI that provides suggestions for code"
            " completion and code improvements."
        ),
        llm=llm,
        max_loops="auto",
        autosave=True,
        dashboard=False,
        streaming_on=True,
        verbose=True,
        stopping_token="<DONE>",
        interactive=True,
        tools=[terminal, browser, file_editor, create_file],
        code_interpreter=True,
    ),
]

# Create MajorityVoting instance
majority_voting = MajorityVoting(agents)

# Run the majority voting system
result = majority_voting.run("What is the capital of France?")
print(result)  # Output: 'Paris'
```

```

### Example 2: Running a Task with Detailed Outputs

```python
from swarms.structs.agent import Agent
from swarms.structs.majority_voting import MajorityVoting

# Initialize agents
agents = [
    Agent(
        agent_name="Devin",
        system_prompt=(
            "Autonomous agent that can interact with humans and other"
            " agents. Be Helpful and Kind. Use the tools provided to"
            " assist the user. Return all code in markdown format."
        ),
        llm=llm,
        max_loops="auto",
        autosave=True,
        dashboard=False,
        streaming_on=True,
        verbose=True,
        stopping_token="<DONE>",
        interactive=True,
        tools=[terminal, browser, file_editor, create_file],
```
```

```
    code_interpreter=True,
),
Agent(
    agent_name="Codex",
    system_prompt=(
        "An AI coding assistant capable of writing and understanding"
        " code snippets in various programming languages."
    ),
    llm=llm,
    max_loops="auto",
    autosave=True,
    dashboard=False,
    streaming_on=True,
    verbose=True,
    stopping_token="<DONE>",
    interactive=True,
    tools=[terminal, browser, file_editor, create_file],
    code_interpreter=True,
),
Agent(
    agent_name="Tabnine",
    system_prompt=(
        "A code completion AI that provides suggestions for code"
        " completion and code improvements."
    ),
    llm=llm,
```

```
        max_loops="auto",
        autosave=True,
        dashboard=False,
        streaming_on=True,
        verbose=True,
        stopping_token="<DONE>",
        interactive=True,
        tools=[terminal, browser, file_editor, create_file],
        code_interpreter=True,
    ),
]

# Create MajorityVoting instance
majority_voting = MajorityVoting(agents)

# Run the majority voting system with a different task
result = majority_voting.run("Create a new file for a plan to take over the world.")
print(result)
```