```
from swarm_models.openai_function_caller import OpenAIFunctionCaller
from pydantic import BaseModel, Field


# Pydantic is a data validation library that provides data validation and parsing using Python type hints.
# It is used here to define the data structure for making API calls to retrieve weather information.
class SentimentAnalysisCard(BaseModel):
    text: str = Field(
        ...,
        description="The text to be analyzed for sentiment rating",
    )
    rating: str = Field(
        ...,
        description="The sentiment rating of the text from 0.0 to 1.0",
    )


# The WeatherAPI class is a Pydantic BaseModel that represents the data structure
# for making API calls to retrieve weather information. It has two attributes: city and date.

# Example usage:
# Initialize the function caller
model = OpenAIFunctionCaller(
    system_prompt="You're a sentiment Analysis Agent, you're purpose is to rate the sentiment of text",
```

```
    max_tokens=100,
    temperature=0.5,
    base_model=SentimentAnalysisCard,
    parallel_tool_calls=False,
)



# The OpenAIFunctionCaller class is used to interact with the OpenAI API and make function calls.
# Here, we initialize an instance of the OpenAIFunctionCaller class with the following parameters:
# - system_prompt: A prompt that sets the context for the conversation with the API.
# - max_tokens: The maximum number of tokens to generate in the API response.
# - temperature: A parameter that controls the randomness of the generated text.
# - base_model: The base model to use for the API calls, in this case, the WeatherAPI class.
out = model.run("This agent created the code incorrectly it sucked.")
print(out)
```