`BaseOpenAI` and `OpenAI` Documentation

Table of Contents

---

1. Overview

The `BaseOpenAI` and `OpenAI` classes are part of the LangChain library, designed to interact with OpenAI's large language models (LLMs). These classes provide a seamless interface for utilizing OpenAI's API to generate natural language text.

2. Class Architecture

Both `BaseOpenAI` and `OpenAI` classes inherit from `BaseLLM`, demonstrating an inheritance-based architecture. This architecture allows for easy extensibility and customization while adhering to the principles of object-oriented programming.

3. Purpose

The purpose of these classes is to simplify the interaction with OpenAI's LLMs. They encapsulate API calls, handle tokenization, and provide a high-level interface for generating text. By instantiating an object of the `OpenAI` class, developers can quickly leverage the power of OpenAI's models to generate text for various applications, such as chatbots, content generation, and more.

4. Class Attributes

Here are the key attributes and their descriptions for the `BaseOpenAI` and `OpenAI` classes:

Attribute	Description
`lc_secrets`	A dictionary of secrets required for LangChain, including the OpenAI API key.
`lc_attributes`	A dictionary of attributes relevant to LangChain.
`is_lc_serializable()`	A method indicating if the class is serializable for LangChain.
`model_name`	The name of the language model to use.
`temperature`	The sampling temperature for text generation.

| `max_tokens` | The maximum number of tokens to generate in a completion. |

| `top_p` | The total probability mass of tokens to consider at each step. |

| `frequency_penalty` | Penalizes repeated tokens according to frequency. |

| `presence_penalty` | Penalizes repeated tokens. |

| `n` | How many completions to generate for each prompt. |

| `best_of` | Generates `best_of` completions server-side and returns the "best." |

| `model_kwargs` | Holds any model parameters valid for `create` calls not explicitly specified.

|

| `openai_api_key` | The OpenAI API key used for authentication. |

| `openai_api_base` | The base URL for the OpenAI API. |

| `openai_organization` | The OpenAI organization name, if applicable. |

| `openai_proxy` | An explicit proxy URL for OpenAI requests. |

| `batch_size` | The batch size to use when passing multiple documents for generation. |

| `request_timeout` | The timeout for requests to the OpenAI completion API. |

| `logit_bias` | Adjustment to the probability of specific tokens being generated. |

| `max_retries` | The maximum number of retries to make when generating. |

| `streaming` | Whether to stream the results or not. |

| `allowed_special` | A set of special tokens that are allowed. |

| `disallowed_special` | A collection of special tokens that are not allowed. |

| `tiktoken_model_name` | The model name to pass to `tiktoken` for token counting. |

5. Methods

5.1 Construction

5.1.1 `__new__(cls, **data: Any) -> Union[OpenAIChat, BaseOpenAI]`

- Description: Initializes the OpenAI object.
- Arguments:
  - `cls` (class): The class instance.
  - `data` (dict): Additional data for initialization.
- Returns:
  - Union[OpenAIChat, BaseOpenAI]: An instance of the OpenAI class.

5.2 Configuration

5.2.1 `build_extra(cls, values: Dict[str, Any]) -> Dict[str, Any]`

- Description: Builds extra kwargs from additional params passed in.
- Arguments:
  - `cls` (class): The class instance.
  - `values` (dict): Values and parameters to build extra kwargs.
- Returns:
  - Dict[str, Any]: A dictionary of built extra kwargs.

5.2.2 `validate_environment(cls, values: Dict) -> Dict`

- Description: Validates that the API key and python package exist in the environment.
- Arguments:
  - `values` (dict): The class values and parameters.
- Returns:
  - Dict: A dictionary of validated values.

5.3 Tokenization

5.3.1 `get_sub_prompts(self, params: Dict[str, Any], prompts: List[str], stop: Optional[List[str]] = None) -> List[List[str]]`

- Description: Gets sub-prompts for LLM call.
- Arguments:
  - `params` (dict): Parameters for LLM call.
  - `prompts` (list): List of prompts.
  - `stop` (list, optional): List of stop words.
- Returns:
  - List[List[str]]: List of sub-prompts.

5.3.2 `get_token_ids(self, text: str) -> List[int]`

- Description: Gets token IDs using the `tiktoken` package.
- Arguments:
  - `text` (str): The text for which to calculate token IDs.
- Returns:
  - List[int]: A list of token IDs.

5.3.3 `modelname_to_contextsize(modelname: str) -> int`

- Description: Calculates the maximum number of tokens possible to generate for a model.
- Arguments:
  - `modelname` (str): The model name to determine the context size for.
- Returns:
  - int: The maximum context size.

5.3.4 `max_tokens_for_prompt(self, prompt: str) -> int`

- Description: Calculates the maximum number of tokens possible to generate for a prompt.

- Arguments:
  - `prompt` (str): The prompt for which to

 determine the maximum token limit.
- Returns:
  - int: The maximum token limit.

5.4 Generation

5.4.1 `generate(self, text: Union[str, List[str]], **kwargs) -> Union[str, List[str]]`
- Description: Generates text using the OpenAI API.
- Arguments:
  - `text` (str or list): The input text or list of inputs.
  - `**kwargs` (dict): Additional parameters for the generation process.
- Returns:
  - Union[str, List[str]]: The generated text or list of generated texts.

5.5 Asynchronous Generation

5.5.1 `generate_async(self, text: Union[str, List[str]], **kwargs) -> Union[str, List[str]]`
- Description: Generates text asynchronously using the OpenAI API.
- Arguments:
  - `text` (str or list): The input text or list of inputs.
  - `**kwargs` (dict): Additional parameters for the asynchronous generation process.
- Returns:
  - Union[str, List[str]]: The generated text or list of generated texts.

6. Usage Examples

6.1 Creating an OpenAI Object

```python
# Import the OpenAI class
from swarm_models import OpenAI

# Set your OpenAI API key
api_key = "YOUR_API_KEY"

# Create an OpenAI object
openai = OpenAI(api_key)
```

6.2 Generating Text

```python
# Generate text from a single prompt
prompt = "Translate the following English text to French: 'Hello, how are you?'"
generated_text = openai.generate(prompt, max_tokens=50)

# Generate text from multiple prompts
prompts = [
    "Translate this: 'Good morning' to Spanish.",
```

```
    "Summarize the following article:",
    article_text,
]
generated_texts = openai.generate(prompts, max_tokens=100)

# Generate text asynchronously
async_prompt = "Translate 'Thank you' into German."
async_result = openai.generate_async(async_prompt, max_tokens=30)

# Access the result of an asynchronous generation
async_result_text = async_result.get()
```

6.3 Advanced Configuration

```python
# Configure generation with advanced options
custom_options = {
    "temperature": 0.7,
    "max_tokens": 100,
    "top_p": 0.9,
    "frequency_penalty": 0.2,
    "presence_penalty": 0.4,
}
generated_text = openai.generate(prompt, **custom_options)
```

This documentation provides a comprehensive understanding of the `BaseOpenAI` and `OpenAI` classes, their attributes, methods, and usage examples. Developers can utilize these classes to interact with OpenAI's language models efficiently, enabling various natural language generation tasks.