## BasePromptTemplate 类、方法与属性速查

（一）BasePromptTemplate 类 在LLM应用中，BasePromptTemplate 类为所有提示模板提供了基础，其主要功能是返回适当的提示。

（二）类名: BasePromptTemplate

（三）属性和方法:

|  |  |
| --- | --- |
| input\_variables: | 一个属性，存储模板预期的变量名列表。 |
| output\_parser: | 一个可选属性，用于解析调用此格式化提示的LLM输出。 |
| partial\_variables: | 一个默认为字典的属性，存储部分变量的映射。 |
| lc\_serializable: | 一个属性，如果对象可以序列化，则返回True。 |
| format\_prompt: | 一个抽象方法，用于根据给定的关键字参数创建聊天消息。 |
| validate\_variable\_names: | 一个方法，用于确保变量名不包含受限制的名称。 |
| partial: | 一个方法，返回提示模板的部分实例。 |
| \_merge\_partial\_and\_user\_variables: | 一个方法，用于合并部分和用户变量。 |
| format: | 一个抽象方法，用于格式化提示，接受任意关键字参数并返回一个格式化的字符串。 |

## BaseLLM类、方法与属性速查

（一）BaseLLM 类

在LLM框架中，BaseLLM 类为大型语言模型（LLM）提供了一个核心的接口，其定义了与模型的基本交互方式。

（二）类名: BaseLLM

（三）属性和方法:

|  |  |
| --- | --- |
| cache: | 用于确定是否缓存模型的结果。 |
| verbose: | 用于决定是否打印响应文本。 |
| callbacks: | 定义了在模型运行过程中的回调函数。 |
| tags: | 用于向运行追踪添加标签。 |
| \_generate: | 一个方法，用于在给定的提示和输入上运行LLM。 |
| \_agenerate: | 一个异步方法，同样用于在给定的提示和输入上运行LLM。 |
| generate\_prompt: | 用于将提示转换为字符串并在LLM上运行。 |
| agenerate\_prompt: | 一个异步方法，用于将提示转换为字符串并在LLM上运行。 |
| generate: | 用于在给定的提示和输入上运行LLM。 |
| agenerate: | 一个异步方法，用于在给定的提示和输入上运行LLM。 |
| call: | 用于在给定的提示和输入上运行LLM并返回字符串。 |
| \_call\_async: | 一个异步方法，同样用于在给定的提示和输入上运行LLM并返回字符串。 |
| predict: | 根据输入的文本进行预测。 |
| predict\_messages: | 根据输入的消息列表进行预测。 |
| apredict: | 一个异步方法，用于根据输入的文本进行预测。 |
| apredict\_messages: | 一个异步方法，用于根据输入的消息列表进行预测。 |
| \_identifying\_params: | 获取标识参数。 |
| \_llm\_type: | 返回LLM的类型。 |
| dict: | 返回LLM的字典表示。 |
| save: | 保存LLM。 |

## BaseChatModel类、方法与属性速查

（一）BaseChatModel 类

BaseChatModel 是基础的聊天模型接口。

（二）类名: BaseChatModel

（三）属性和方法:

|  |  |
| --- | --- |
| verbose: | 是否打印出响应文本。 |
| callbacks: | 回调。 |
| callback\_manager: | 可选的基础回调管理器。 |
| tags: | 添加到运行追踪的标签。 |
| raise\_deprecation: | 验证器，如果使用callback\_manager则发出弃用警告。 |
| \_combine\_llm\_outputs: | 方法，组合LLM输出。 |
| generate: | 方法，顶级调用。 |
| agenerate: | 异步方法，顶级调用。 |
| generate\_prompt: | 方法，生成提示。 |
| agenerate\_prompt: | 异步方法，生成提示。 |
| \_generate: | 抽象方法，顶级调用。 |
| \_agenerate: | 抽象方法，顶级调用。 |
| call: | 方法，调用BaseChatModel。 |
| \_call\_async: | 异步方法，调用BaseChatModel。 |
| call\_as\_llm: | 方法，作为LLM调用。 |
| predict: | 方法，预测文本。 |
| predict\_messages: | 方法，预测消息。 |
| apredict: | 异步方法，预测文本。 |
| apredict\_messages: | 异步方法，预测消息。 |
| \_identifying\_params: | 属性，获取识别参数。 |
| \_llm\_type: | 抽象属性，返回聊天模型的类型。 |
| dict: | 方法，返回LLM的字典表示。 |

## BaseCallbackManager类、方法与属性速查

BaseCallbackManager 类、方法与属性速查

（一）BaseCallbackManager 类

在LangChain框架中，BaseCallbackManager 类为LangChain的回调提供了基础的管理接口。

（二）类名: BaseCallbackManager

（三）属性和方法:

|  |  |
| --- | --- |
| handlers: | 一个包含BaseCallbackHandler的列表，用于存储回调处理器。 |
| inheritable\_handlers: | 可继承的回调处理器列表。 |
| parent\_run\_id: | 父运行的UUID标识符。 |
| tags: | 与回调管理器关联的标签列表。 |
| inheritable\_tags: | 可继承的标签列表。 |
| is\_async: | 属性，用于判断回调管理器是否为异步。 |
| add\_handler: | 方法，用于向回调管理器添加处理器。 |
| remove\_handler: | 方法，用于从回调管理器中移除处理器。 |
| set\_handlers: | 方法，设置为回调管理器的唯一处理器的处理器列表。 |
| set\_handler: | 方法，设置为回调管理器的唯一处理器。 |
| add\_tags: | 方法，用于添加标签到回调管理器。 |
| remove\_tags: | 方法，用于从回调管理器中移除标签。 |

## Embeddings类、方法与属性速查

（一）Embeddings 类

在LangChain框架中，Embeddings 类是嵌入模型的接口。

（二）类名:Embeddings

（三）属性和方法:

|  |  |
| --- | --- |
| embed\_documents: | 用于嵌入搜索文档。输入的数据格式是列表字符串，输出的数据格式是浮点数列表的列表。 |
| embed\_query: | 用于嵌入查询字符串。输入的数据格式是列表字符串，输出的数据格式是浮点数列表。 |

## Agent 类、方法与属性速查

（一）Agent 类

在LangChain框架中，Agent 类负责调用语言模型并决定行动。它是由一个LLMChain驱动的，其中LLMChain的提示必须包括一个名为"agent\_scratchpad"的变量，代理可以在其中放置其中间工作。

（二）类名: Agent

（三）属性和方法:

|  |  |
| --- | --- |
| llm\_chain: | LLMChain实例，描述代理如何与语言模型交互。 |
| output\_parser: | AgentOutputParser的实例，用于解析语言模型的输出。 |
| allowed\_tools: | 可选的工具列表。 |
| dict: | 方法，返回代理的字典表示。 |
| get\_allowed\_tools: | 方法，返回允许的工具列表。 |
| return\_values: | 属性，返回输出值列表。 |
| \_fix\_text: | 方法，修复文本。 |
| \_stop: | 属性，定义停止令牌列表。 |
| \_construct\_scratchpad: | 方法，构建代理继续其思考过程的草稿板。 |
| plan: | 方法，根据输入决定要做什么。 |
| aplan: | 方法，根据输入决定要做什么。 |
| get\_full\_inputs: | 方法，从中间步骤创建LLMChain的完整输入。 |
| input\_keys: | 属性，返回输入键的列表。 |
| validate\_prompt: | 方法，验证提示是否匹配格式。 |
| observation\_prefix: | 属性，定义观察和LLM调用的前缀。 |
| llm\_prefix: | 属性，定义观察和LLM调用的前缀。 |
| create\_prompt: | 方法，为该类创建一个提示。 |
| \_validate\_tools: | 方法，验证工具并获取默认的输出解析器。 |
| \_get\_default\_output\_parser: | 方法，验证工具并获取默认的输出解析器。 |
| from\_llm\_and\_tools: | 方法，从LLM和工具构造代理。 |
| return\_stopped\_response: | 方法，当代理由于最大迭代次数而停止时返回响应。 |
| tool\_run\_logging\_kwargs: | 方法，返回工具运行日志的关键字参数。 |

## AgentExecutor 类、方法与属性速查

（一）AgentExecutor 类

AgentExecutor 类封装了一个使用工具的代理。此类负责驱动代理，使其在工具集合上运行，并根据代理的建议采取行动。

（二）类名: AgentExecutor

（三）属性和方法:

|  |  |
| --- | --- |
| agent: | 代理，可以是单行动或多行动代理。 |
| tools: | 代理可以使用的工具序列。 |
| return\_intermediate\_steps: | 布尔值，决定是否返回中间步骤。 |
| max\_iterations: | 最大迭代次数。 |
| max\_execution\_time: | 最大执行时间。 |
| early\_stopping\_method: | 早期停止方法（例如，当达到最大迭代次数或时间限制时）。 |
| handle\_parsing\_errors: | 如何处理解析错误。 |
| from\_agent\_and\_tools: | 创建代理执行者的类方法。 |
| validate\_tools: | 验证工具与代理兼容的方法。 |
| validate\_return\_direct\_tool: | 验证工具与代理兼容的方法。 |
| save: | 保存代理执行者的方法（此方法会引发错误，因为AgentExecutor不支持保存）。 |
| save\_agent: | 保存底层代理的方法。 |
| input\_keys: | 返回输入键的属性。 |
| output\_keys: | 返回输出键的属性。 |
| lookup\_tool: | 方法，按名称查找工具。 |
| \_should\_continue: | 方法，确定是否应继续迭代。 |
| \_return: | 方法，返回代理的最终输出。 |
| \_areturn: | 异步方法，返回代理的最终输出。 |
| \_take\_next\_step: | 方法，代理在思考-行动-观察循环中采取单一步骤。 |
| \_atake\_next\_step: | 异步方法，代理在思考-行动-观察循环中采取单一步骤。 |
| \_call: | 方法，运行文本并获取代理响应。 |
| \_acall: | 异步方法，运行文本并获取代理响应。 |
| \_get\_tool\_return: | 方法，检查工具是否是返回工具。 |

## Chain 类、方法与属性速查

（一）Chain 类

Chain 是所有链应实现的基础接口。

（二）类名: Chain

（三）属性和方法:

|  |  |
| --- | --- |
| memory: | 可选的基础内存。 |
| callbacks: | 回调。 |
| callback\_manager: | 可选的基础回调管理器。 |
| verbose: | 用于决定是否打印响应文本的布尔值。 |
| tags: | 可选的标签列表。 |
| \_chain\_type: | 属性，需要子类实现。用于说明链的类型。 |
| raise\_deprecation: | 验证器，如果使用callback\_manager则发出弃用警告。 |
| set\_verbose: | 如果verbose为None，则设置它。 |
| input\_keys: | 抽象属性，此链期望的输入键。 |
| output\_keys: | 抽象属性，此链期望的输出键。 |
| \_validate\_inputs: | 方法，检查所有输入是否存在。 |
| \_validate\_outputs: | 方法，检查所有输出是否存在。 |
| \_call: | 抽象方法，运行此链的逻辑并返回输出。 |
| \_acall: | 异步方法，运行此链的逻辑并返回输出。 |
| call: | 方法，运行此链的逻辑，并根据需要添加到输出。 |
| acall: | 异步方法，运行此链的逻辑，并根据需要添加到输出。 |
| prep\_outputs: | 方法，验证和准备输出。 |
| prep\_inputs: | 方法，验证和准备输入。 |
| apply: | 方法，对列表中的所有输入调用链。 |
| \_run\_output\_key: | 属性，只有一个输出键时才支持运行。 |
| run: | 方法，以文本输入、文本输出或多个变量、文本输出的形式运行链。 |
| arun: | 异步方法，以文本输入、文本输出或多个变量、文本输出的形式运行链。 |
| dict: | 方法，返回链的字典表示。 |
| save: | 方法，保存链。 |

BaseLoader 类、方法与属性速查

（一）BaseLoader 类

BaseLoader 是一个用于加载文档的接口。

（二）类名: BaseLoader

（三）属性和方法:

|  |  |
| --- | --- |
| load: | 抽象方法，加载数据到文档对象中。子类应将此方法实现为返回 list(self.lazy\_load())，此方法返回一个在内存中实体化的列表。 |
| load\_and\_split: | 方法，加载文档并分割成块。如果没有提供文本分割器，它将使用 RecursiveCharacterTextSplitter。 |
| lazy\_load: | 方法:为文档内容提供懒加载。请注意，这个方法会在所有现有子类中实现之后升级为一个抽象方法 |

BaseChatMemory 类、方法与属性速查

（一）BaseChatMemory 类

BaseChatMemory 是一个继承于 BaseMemory 的基础聊天内存抽象基类。

（二）类名: BaseChatMemory

（三）属性和方法:

|  |  |
| --- | --- |
| chat\_memory: | 用于存储聊天消息历史的属性，默认为ChatMessageHistory  实例。 |
| output\_key: | 输出键，用于确定哪个输出应该被保存到聊天历史中，默认为None。 |
| input\_key: | 输入键，用于确定哪个输入应该被保存到聊天历史中，默认为None。 |
| return\_messages: | 布尔值，决定是否返回消息，默认为False。 |
| \_get\_input\_output: | 私有方法，从输入和输出中获取对应的输入和输出字符串。 |
| save\_context: | 方法，将此次对话的上下文保存到缓冲区。 |
| clear: | 方法，清除内存内容。 |

StructuredOutputParser 类、方法与属性速查

（一）StructuredOutputParser 类

StructuredOutputParser 是继承于 BaseOutputParser 的结构化输出解析器类。

（二）类名: StructuredOutputParser

（三）属性和方法:

|  |  |
| --- | --- |
| response\_schemas: | 用于存储响应模式的列表。 |
| from\_response\_schemas: | 类方法，从响应模式列表创建StructuredOutputParser实例。 |
| get\_format\_instructions: | 方法，生成格式化的说明字符串。 |
| parse: | 方法，从给定文本中解析结构化输出。 |
| \_type: | 属性方法，返回字符串"structured"。 |

ArxivRetriever类、方法与属性速查

（一）ArxivRetriever 类

ArxivRetriever 是一个结合了 BaseRetriever 和 ArxivAPIWrapper 的检索器类。本质上，它是 ArxivAPIWrapper 的一个包装器。

（二）类名: ArxivRetriever

（三）概述:

该类有效地包装了 ArxivAPIWrapper。它将 load() 方法包装为 get\_relevant\_documents() 方法。此外，该类使用所有 ArxivAPIWrapper 的参数，不做任何更改。

（四）方法:

|  |  |
| --- | --- |
| get\_relevant\_documents: | 方法，获取与给定查询相关的文档。 |
| aget\_relevant\_documents: | 异步方法，但目前尚未实现。 |

BaseTool类、方法与属性速查

（一）BaseTool 类

BaseTool 提供了 LangChain 工具必须实现的接口。

（二）类名: BaseTool

（三）概述:

该类为所有 LangChain 工具提供了一个基本的接口。它定义了工具如何运行、如何解析输入和如何处理错误。

（四）属性和方法:

|  |  |
| --- | --- |
| \_parse\_input: | 将工具输入转换为pydantic模型。 |
| \_run: | 使用工具。 |
| \_arun: | 异步使用工具。 |
| run: | 运行工具。 |
| arun: | 异步运行工具。 |
| \_\_call\_\_: | 使工具可调用。 |
| name: | 工具的独特名称。 |
| description: | 描述如何/何时/为什么使用工具。 |
| args\_schema: | Pydantic模型类，用于验证和解析工具的输入参数。 |
| return\_direct: | 是否直接返回工具的输出。 |
| verbose: | 是否记录工具的进度。 |
| callbacks: | 在工具执行期间要调用的回调。 |
| callback\_manager: | 已弃用。请使用callbacks代替。 |
| handle\_tool\_error: | 处理抛出的ToolException的内容。 |

GoogleSerperAPIWrapper类、方法与属性速查

（一）GoogleSerperAPIWrapper 类

GoogleSerperAPIWrapper 是一个围绕 Serper.dev Google 搜索 API 的包装器。

（二）类名: GoogleSerperAPIWrapper

（三）概述:

该类为 Serper.dev Google 搜索 API 提供了一个接口。用户可以使用环境变量 SERPER\_API\_KEY 或通过构造函数的 serper\_api\_key 参数提供 API 密钥。

（四）属性和方法:

|  |  |
| --- | --- |
| results: | 通过GoogleSearch运行查询。 |
| run: | 通过GoogleSearch运行查询并解析结果。 |
| aresults: | 异步地通过GoogleSearch运行查询。 |
| arun: | 异步地通过GoogleSearch运行查询并解析结果。 |
| \_parse\_snippets: | 从搜索结果中解析摘录。 |
| \_parse\_results: | 解析搜索结果。 |
| \_google\_serper\_api\_results: | 获取GoogleSerperAPI的结果。 |
| \_async\_google\_serper\_search\_results: | 异步获取GoogleSerperAPI的结果。 |
| k: | 返回的搜索结果的最大数量。 |
| gl: | 地理位置代码。 |
| hl: | 语言代码。 |
| type: | 搜索类型，可选值为"news","search","places"和"images"。 |
| result\_key\_for\_type: | 用于查找搜索结果的键。 |
| tbs: | 时间范围限制。 |
| serper\_api\_key: | API密钥。 |
| aiosession: | aiohttp的ClientSession对象。 |

VectorStore 类、属性与方法速查

（一）VectorStore 类

在LangChain框架中，VectorStore 类是向量存储的接口。

（二）类名: VectorStore

（三）属性和方法:

|  |  |
| --- | --- |
| add\_texts: | 通过嵌入运行更多的文本并添加到向量存储。它接受一个可迭代的文本，一个可选的元数据列表和特定于向量存储的参数。返回添加到向量存储的文本的ID列表。 |
| aadd\_texts: | 异步版本的 add\_texts，具有相同的功能和参数。 |
| add\_documents: | 通过嵌入运行更多的文档并添加到向量存储。输入是一个文档列表，返回的是已添加文本的ID列表。 |
| search: | 使用指定的搜索类型返回与查询最相似的文档。 |
| asearch: | 异步版本的 search 方法。 |
| similarity\_search\_with\_relevance\_scores: | 返回范围为 [0, 1] 的文档和相关性分数，其中0表示不相似，1表示最相似。 |
| similarity\_search | 返回与查询最相似的文档。 |

|  |  |
| --- | --- |
| asimilarity\_search\_with\_relevance\_scores: | 异步版本的similarity\_search\_with\_relevance\_scores方法。 |
| asimilarity\_search: | 异步版本的similarity\_search方法。 |
| similarity\_search\_by\_vector: | 根据给定的嵌入向量返回与之最相似的文档。 |
| asimilarity\_search\_by\_vector: | 异步版本similarity\_search\_by\_vector方法。 |
| max\_marginal\_relevance\_search: | 使用最大边际相关性返回文档。 |
| max\_marginal\_relevance\_search\_by\_vector: | 根据给定的嵌入向量，使用最大边际相关性返回文档。 |
| amax\_marginal\_relevance\_search\_by\_vector: | 异步版本的max\_marginal\_relevance\_search\_by\_vector方法。 |
| from\_documents: | 返回从文档和嵌入初始化的VectorStore。 |
| afrom\_documents: | 异步版本的from\_documents方法。 |
| from\_texts: | 返回从文本和嵌入初始化VectorStore。 |
| afrom\_texts: | 异步版本的from\_texts方法。 |
| as\_retriever: | 返回VectorStoreRetriever。 |