agentchat.contrib.compressible_agent

CompressibleAgent Objects

```
class CompressibleAgent(ConversableAgent)
```

(Experimental) CompressibleAgent agent. While this agent retains all the default functionalities of the `AssistantAgent`, it also provides the added feature of compression when activated through the `compress_config` setting.

`compress_config` is set to False by default, making this agent equivalent to the `AssistantAgent`. This agent does not work well in a GroupChat: The compressed messages will not be sent to all the agents in the group. The default system message is the same as AssistantAgent. `human_input_mode` is default to "NEVER" and `code_execution_config` is default to False. This agent doesn't execute code or function call by default.

__init__

```
def __init__(name: str,
             system_message: Optional[str] = DEFAULT_SYSTEM_MESSAGE,
             is_termination_msg: Optional[Callable[[Dict], bool]] = None,
             max_consecutive_auto_reply: Optional[int] = None,
             human_input_mode: Optional[str] = "NEVER",
             function_map: Optional[Dict[str, Callable]] = None,
             code_execution_config: Optional[Union[Dict, bool]] = False,
             llm_config: Optional[Union[Dict, bool]] = None,
             default_auto_reply: Optional[Union[str, Dict, None]] = "",
             compress_config: Optional[Dict] = False,
             description: Optional[str] = None,
             **kwargs)
```

Arguments:

- `name` *str* - agent name.
- `system_message` *str* - system message for the ChatCompletion inference. Please override this attribute if you want to reprogram the agent.
- `llm_config` *dict* - llm inference configuration. Please refer to OpenAIWrapper.create for available options.
- `is_termination_msg` *function* - a function that takes a message in the form of a dictionary and returns a boolean value indicating if this received message is a termination message. The dict can contain the following keys: "content", "role", "name", "function_call".
- `max_consecutive_auto_reply` *int* - the maximum number of consecutive auto replies. default to None (no limit provided, class attribute MAX_CONSECUTIVE_AUTO_REPLY will be used as the limit in this case). The limit only plays a role when human_input_mode is not "ALWAYS".
- `compress_config` *dict or True/False* - config for compression before oai_reply. Default to False. You should contain the following keys:
  - "mode" (Optional, str, default to "TERMINATE"): Choose from ["COMPRESS", "TERMINATE", "CUSTOMIZED"].
- `"TERMINATE"` - terminate the conversation ONLY when token count exceeds the max limit of current model. `trigger_count` is NOT used in this mode.
- `"COMPRESS"` - compress the messages when the token count exceeds the limit.
- `"CUSTOMIZED"` - pass in a customized function to compress the messages.
  - "compress_function" (Optional, callable, default to None): Must be provided when mode is "CUSTOMIZED". The function should takes a list of messages and returns a tuple of (is_compress_success: bool, compressed_messages: List[Dict]).
  - "trigger_count" (Optional, float, int, default to 0.7): the threshold to trigger compression. If a float between (0, 1], it is the percentage of token used. if a int, it is the number of tokens used.
  - "async" (Optional, bool, default to False): whether to compress asynchronously.
  - "broadcast" (Optional, bool, default to True): whether to update the compressed message history to sender.
  - "verbose" (Optional, bool, default to False): Whether to print the content before and after compression. Used when mode="COMPRESS".
  - "leave_last_n" (Optional, int, default to 0): If provided, the last n messages will not be compressed. Used when mode="COMPRESS".
- `system_message0` *str* - a short description of the agent. This description is used by other agents (e.g. the GroupChatManager) to decide when to call upon this agent. (Default: system_message)

- `system_message`1 *dict* - Please refer to other kwargs in ConversableAgent.

generate_reply

```
def generate_reply(
        messages: Optional[List[Dict]] = None,
        sender: Optional[Agent] = None,
        exclude: Optional[List[Callable]] = None) -> Union[str, Dict, None]
```

Adding to line 202:

```
    if messages is not None and messages != self._oai_messages[sender]:
        messages = self._oai_messages[sender]
```

on_oai_token_limit

```
def on_oai_token_limit(
        messages: Optional[List[Dict]] = None,
        sender: Optional[Agent] = None,
        config: Optional[Any] = None) -> Tuple[bool, Union[str, Dict, None]]
```

(Experimental) Compress previous messages when a threshold of tokens is reached.

TODO: async compress TODO: maintain a list for old oai messages (messages before compression)

compress_messages

```
def compress_messages(
        messages: Optional[List[Dict]] = None,
        config: Optional[Any] = None
) -> Tuple[bool, Union[str, Dict, None, List]]
```

Compress a list of messages into one message.

The first message (the initial prompt) will not be compressed. The rest of the messages will be compressed into one message, the model is asked to distinguish the role of each message: USER, ASSISTANT, FUNCTION_CALL, FUNCTION_RETURN. Check out the compress_sys_msg.

TODO: model used in compression agent is different from assistant agent: For example, if original model used by is gpt-4; we start compressing at 70% of usage, 70% of 8092 = 5664; and we use gpt 3.5 here max_toke = 4096, it will raise error. choosinng model automatically?

Edit this page

Previous
« agent_builder

Next
gpt_assistant_agent »