agentchat.contrib.retrieve_user_proxy_agent

RetrieveUserProxyAgent Objects

```
class RetrieveUserProxyAgent(UserProxyAgent)
```

__init__

```
def __init__(name="RetrieveChatAgent",
             human_input_mode: Optional[str] = "ALWAYS",
             is_termination_msg: Optional[Callable[[Dict], bool]] = None,
             retrieve_config: Optional[Dict] = None,
             **kwargs)
```

Arguments:

- `name` *str* - name of the agent.
- `human_input_mode` *str* - whether to ask for human inputs every time a message is received. Possible values are "ALWAYS", "TERMINATE", "NEVER". (1) When "ALWAYS", the agent prompts for human input every time a message is received. Under this mode, the conversation stops when the human input is "exit", or when is_termination_msg is True and there is no human input. (2) When "TERMINATE", the agent only prompts for human input only when a termination message is received or the number of auto reply reaches the max_consecutive_auto_reply. (3) When "NEVER", the agent will never prompt for human input. Under this mode, the conversation stops when the number of auto reply reaches the max_consecutive_auto_reply or when is_termination_msg is True.
- `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".
- `retrieve_config` *dict or None* - config for the retrieve agent. To use default config, set to None. Otherwise, set to a dictionary with the following keys:
    - task (Optional, str): the task of the retrieve chat. Possible values are "code", "qa" and "default". System prompt will be different for different tasks. The default value is `default`, which supports both code and qa.
    - client (Optional, chromadb.Client): the chromadb client. If key not provided, a default client `chromadb.Client()` will be used. If you want to use other vector db, extend this class and override the `retrieve_docs` function.
    - docs_path (Optional, Union[str, List[str]]): the path to the docs directory. It can also be the path to a single file, the url to a single file or a list of directories, files and urls. Default is None, which works only if the collection is already created.
    - extra_docs (Optional, bool): when true, allows adding documents with unique IDs without overwriting existing ones; when false, it replaces existing documents using default IDs, risking collection overwrite., when set to true it enables the system to assign unique IDs starting from "length+i" for new document chunks, preventing the replacement of existing documents and facilitating the addition of more content to the collection.. By default, "extra_docs" is set to false, starting document IDs from zero. This poses a risk as new documents might overwrite existing ones, potentially causing unintended loss or alteration of data in the collection.
    - collection_name (Optional, str): the name of the collection. If key not provided, a default name `autogen-docs` will be used.
    - model (Optional, str): the model to use for the retrieve chat. If key not provided, a default model `gpt-4` will be used.
    - chunk_token_size (Optional, int): the chunk token size for the retrieve chat. If key not provided, a default size `max_tokens * 0.4` will be used.
    - context_max_tokens (Optional, int): the context max token size for the retrieve chat. If key not provided, a default size `human_input_mode`0 will be used.
    - chunk_mode (Optional, str): the chunk mode for the retrieve chat. Possible values are "multi_lines" and "one_line". If key not provided, a default mode `human_input_mode`1 will be used.
    - must_break_at_empty_line (Optional, bool): chunk will only break at empty line if True. Default is True. If chunk_mode is "one_line", this parameter will be ignored.
    - embedding_model (Optional, str): the embedding model to use for the retrieve chat. If key not provided, a default model `human_input_mode`2 will be used. All available models can be found at `human_input_mode`3. The default model is a fast model. If you want to use a high performance model, `human_input_mode`4 is recommended.
    - embedding_function (Optional, Callable): the embedding function for creating the vector db. Default is None, SentenceTransformer with the given `human_input_mode`5 will be used. If you want to use OpenAI, Cohere, HuggingFace or

other embedding functions, you can pass it here, follow the examples in `human_input_mode`6.
    - customized_prompt (Optional, str): the customized prompt for the retrieve chat. Default is None.
    - customized_answer_prefix (Optional, str): the customized answer prefix for the retrieve chat. Default is "". If not "" and the customized_answer_prefix is not in the answer, `human_input_mode`7 will be triggered.
    - update_context (Optional, bool): if False, will not apply `human_input_mode`7 for interactive retrieval. Default is True.
    - get_or_create (Optional, bool): if True, will create/return a collection for the retrieve chat. This is the same as that used in chromadb. Default is False. Will raise ValueError if the collection already exists and get_or_create is False. Will be set to True if docs_path is None.
    - custom_token_count_function (Optional, Callable): a custom function to count the number of tokens in a string. The function should take (text:str, model:str) as input and return the token_count(int). the retrieve_config["model"] will be passed in the function. Default is autogen.token_count_utils.count_token that uses tiktoken, which may not be accurate for non-OpenAI models.
    - custom_text_split_function (Optional, Callable): a custom function to split a string into a list of strings. Default is None, will use the default function in `human_input_mode`9.
    - custom_text_types (Optional, List[str]): a list of file types to be processed. Default is `is_termination_msg`0. This only applies to files under the directories in `is_termination_msg`1. Explicitly included files and urls will be chunked regardless of their types.
    - recursive (Optional, bool): whether to search documents recursively in the docs_path. Default is True.

- `is_termination_msg`2 *dict* - other kwargs in UserProxyAgent.

  Example of overriding retrieve_docs: If you have set up a customized vector db, and it's not compatible with chromadb, you can easily plug in it with below code. `is_termination_msg`3

retrieve_docs

```
def retrieve_docs(problem: str, n_results: int = 20, search_string: str = "")
```

Retrieve docs based on the given problem and assign the results to the class property `_results`. In case you want to customize the retrieval process, such as using a different vector db whose APIs are not compatible with chromadb or filter results with metadata, you can override this function. Just keep the current parameters and add your own parameters with default values, and keep the results in below type.

Type of the results: Dict[str, List[List[Any]]], should have keys "ids" and "documents", "ids" for the ids of the retrieved docs and "documents" for the contents of the retrieved docs. Any other keys are optional. Refer to `chromadb.api.types.QueryResult` as an example. ids: List[string] documents: List[List[string]]

Arguments:

- `problem` *str* - the problem to be solved.
- `n_results` *int* - the number of results to be retrieved. Default is 20.
- `search_string` *str* - only docs that contain an exact match of this string will be retrieved. Default is "".

generate_init_message

```
def generate_init_message(problem: str,
                          n_results: int = 20,
                          search_string: str = "")
```

Generate an initial message with the given problem and prompt.

Arguments:

- `problem` *str* - the problem to be solved.
- `n_results` *int* - the number of results to be retrieved.
- `search_string` *str* - only docs containing this string will be retrieved.

Returns:

- `str` - the generated prompt ready to be sent to the assistant agent.

Previous
« retrieve_assistant_agent

Next
text_analyzer_agent »

Discord

Twitter