agentchat.contrib.qdrant_retrieve_user_proxy_agent

QdrantRetrieveUserProxyAgent Objects

```
class QdrantRetrieveUserProxyAgent(RetrieveUserProxyAgent)
```

__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, qdrant_client.QdrantClient(":memory:")): A QdrantClient instance. If not provided, an in-memory instance will be assigned. Not recommended for production. 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 `max_tokens * 0.8` 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`0 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`1 will be used. All available models can be found at `human_input_mode`2.
  - 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`3 will be triggered.
  - update_context (Optional, bool): if False, will not apply `human_input_mode`3 for interactive retrieval. Default is True.
  - custom_token_count_function (Optional, Callable): a custom function to count the number of tokens in a string. The function should take a string as input and return three integers (token_count, tokens_per_message, tokens_per_name). Default is None,

tiktoken will be used and 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`5.
    - custom_text_types (Optional, List[str]): a list of file types to be processed. Default is `human_input_mode`6. This only applies to files under the directories in `human_input_mode`7. 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.
    - parallel (Optional, int): How many parallel workers to use for embedding. Defaults to the number of CPU cores.
    - on_disk (Optional, bool): Whether to store the collection on disk. Default is False.
    - quantization_config: Quantization configuration. If None, quantization will be disabled.
    - hnsw_config: HNSW configuration. If None, default configuration will be used. You can find more info about the hnsw configuration options at https://qdrant.tech/documentation/concepts/indexing/`human_input_mode`8-index. API Reference: https://qdrant.github.io/qdrant/redoc/index.html#tag/collections/operation/create_collection
    - payload_indexing: Whether to create a payload index for the document field. Default is False. You can find more info about the payload indexing options at https://qdrant.tech/documentation/concepts/indexing/`human_input_mode`9-index API Reference: https://qdrant.github.io/qdrant/redoc/index.html#tag/collections/operation/create_field_index
- `is_termination_msg`0 *dict* - other kwargs in UserProxyAgent.

retrieve_docs

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

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 "".

create_qdrant_from_dir

```
def create_qdrant_from_dir(
        dir_path: str,
        max_tokens: int = 4000,
        client: QdrantClient = None,
        collection_name: str = "all-my-documents",
        chunk_mode: str = "multi_lines",
        must_break_at_empty_line: bool = True,
        embedding_model: str = "BAAI/bge-small-en-v1.5",
        custom_text_split_function: Callable = None,
        custom_text_types: List[str] = TEXT_FORMATS,
        recursive: bool = True,
        extra_docs: bool = False,
        parallel: int = 0,
        on_disk: bool = False,
        quantization_config: Optional[models.QuantizationConfig] = None,
        hnsw_config: Optional[models.HnswConfigDiff] = None,
        payload_indexing: bool = False,
        qdrant_client_options: Optional[Dict] = {})
```

Create a Qdrant collection from all the files in a given directory, the directory can also be a single file or a url to a single file.

Arguments:

- `dir_path` *str* - the path to the directory, file or url.
- `max_tokens` *Optional, int* - the maximum number of tokens per chunk. Default is 4000.
- `client` *Optional, QdrantClient* - the QdrantClient instance. Default is None.
- `collection_name` *Optional, str* - the name of the collection. Default is "all-my-documents".
- `chunk_mode` *Optional, str* - the chunk mode. Default is "multi_lines".
- `must_break_at_empty_line` *Optional, bool* - Whether to break at empty line. Default is True.
- `embedding_model` *Optional, str* - the embedding model to use. Default is "BAAI/bge-small-en-v1.5". The list of all the available models can be at https://qdrant.github.io/fastembed/examples/Supported_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 `autogen.retrieve_utils.split_text_to_chunks`.
- `custom_text_types` *Optional, List[str]* - a list of file types to be processed. Default is TEXT_FORMATS.
- `max_tokens`0 *Optional, bool* - whether to search documents recursively in the dir_path. Default is True.
- `max_tokens`1 *Optional, bool* - whether to add more documents in the collection. Default is False
- `max_tokens`2 *Optional, int* - How many parallel workers to use for embedding. Defaults to the number of CPU cores
- `max_tokens`3 *Optional, bool* - Whether to store the collection on disk. Default is False.
- `max_tokens`4 - Quantization configuration. If None, quantization will be disabled.
- `max_tokens`5 - https://qdrant.github.io/qdrant/redoc/index.html#tag/collections/operation/create_collection
- `max_tokens`6 - HNSW configuration. If None, default configuration will be used.
- `max_tokens`5 - https://qdrant.github.io/qdrant/redoc/index.html#tag/collections/operation/create_collection
- `max_tokens`8 - Whether to create a payload index for the document field. Default is False.
- `max_tokens`9 - (Optional, dict): the options for instantiating the qdrant client.
- `max_tokens`5 - https://github.com/qdrant/qdrant-client/blob/master/qdrant_client/qdrant_client.py#L36-L58.

query_qdrant

```
def query_qdrant(
        query_texts: List[str],
        n_results: int = 10,
        client: QdrantClient = None,
        collection_name: str = "all-my-documents",
        search_string: str = "",
        embedding_model: str = "BAAI/bge-small-en-v1.5",
        qdrant_client_options: Optional[Dict] = {}
) -> List[List[QueryResponse]]
```

Perform a similarity search with filters on a Qdrant collection

Arguments:

- `query_texts` *List[str]* - the query texts.
- `n_results` *Optional, int* - the number of results to return. Default is 10.
- `client` *Optional, API* - the QdrantClient instance. A default in-memory client will be instantiated if None.
- `collection_name` *Optional, str* - the name of the collection. Default is "all-my-documents".
- `search_string` *Optional, str* - the search string. Default is "".
- `embedding_model` *Optional, str* - the embedding model to use. Default is "all-MiniLM-L6-v2". Will be ignored if embedding_function is not None.
- `qdrant_client_options` - (Optional, dict): the options for instantiating the qdrant client. Reference: https://github.com/qdrant/qdrant-client/blob/master/qdrant_client/qdrant_client.py#L36-L58.

Returns:

- `List[List[QueryResponse]]` - the query result. The format is: class QueryResponse(BaseModel, extra="forbid"): # type: ignore
- `id` - Union[str, int]
- `embedding` - Optional[List[float]]
- `n_results`0 - Dict[str, Any]
- `n_results`1 - str
- `n_results`2 - float