



## AutoGen

S  
🏠 Referenceretrieve\_utils

On this page

# retrieve\_utils

## UNSTRUCTURED\_FORMATS

These formats will be parsed by the 'unstructured' library, if installed.

### split\_text\_to\_chunks

```
def split_text_to_chunks(text: str,
                        max_tokens: int = 4000,
                        chunk_mode: str = "multi_lines",
                        must_break_at_empty_line: bool = True,
                        overlap: int = 10)
```

Split a long text into chunks of max\_tokens.

### extract\_text\_from\_pdf

```
def extract_text_from_pdf(file: str) -> str
```

Extract text from PDF files

### split\_files\_to\_chunks

```
def split_files_to_chunks(files: list,
                        max_tokens: int = 4000,
                        chunk_mode: str = "multi_lines",
                        must_break_at_empty_line: bool = True,
                        custom_text_split_function: Callable = None)
```

Split a list of files into chunks of max\_tokens.

### get\_files\_from\_dir

```
def get_files_from_dir(dir_path: Union[str, List[str]],
                    types: list = TEXT_FORMATS,
                    recursive: bool = True)
```

Return a list of all the files in a given directory, a url, a file path or a list of them.

### get\_file\_from\_url

```
def get_file_from_url(url: str, save_path: str = None)
```

Download a file from a URL.

### is\_url

```
def is_url(string: str)
```

Return True if the string is a valid URL.

### create\_vector\_db\_from\_dir

```
def create_vector_db_from_dir(dir_path: Union[str, List[str]],
                             max_tokens: int = 4000,
                             client: API = None,
                             db_path: str = "/tmp/chromadb.db",
                             collection_name: str = "all-my-documents",
                             get_or_create: bool = False,
                             chunk_mode: str = "multi_lines",
                             must_break_at_empty_line: bool = True,
                             embedding_model: str = "all-MiniLM-L6-v2",
                             embedding_function: Callable = None,
                             custom_text_split_function: Callable = None,
                             custom_text_types: List[str] = TEXT_FORMATS,
                             recursive: bool = True,
                             extra_docs: bool = False) -> API
```

Create a vector db from all the files in a given directory, the directory can also be a single file or a url to a single file. We support chromadb compatible APIs to create the vector db, this function is not required if you prepared your own vector db.

#### Arguments:

- `dir_path` *Union[str, List[str]]* - the path to the directory, file, url or a list of them.
- `max_tokens` *Optional, int* - the maximum number of tokens per chunk. Default is 4000.
- `client` *Optional, API* - the chromadb client. Default is None.
- `db_path` *Optional, str* - the path to the chromadb. Default is "/tmp/chromadb.db".
- `collection_name` *Optional, str* - the name of the collection. Default is "all-my-documents".
- `get_or_create` *Optional, bool* - Whether to get or create the collection. Default is False. If True, the collection will be returned if it already exists. Will raise ValueError if the collection already exists and `get_or_create` is False.
- `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 "all-MiniLM-L6-v2". Will be ignored if `embedding_function` is not None.
- `embedding_function` *Optional, Callable* - the embedding function to use. Default is None, SentenceTransformer with the given `embedding_model` will be used. If you want to use OpenAI, Cohere, HuggingFace or other embedding functions, you can pass it here, follow the examples in `max_tokens1`.
- `max_tokens2` *Optional, Callable* - a custom function to split a string into a list of strings. Default is None, will use the default function in `max_tokens3`.
- `max_tokens4` *Optional, List[str]* - a list of file types to be processed. Default is TEXT\_FORMATS.
- `max_tokens5` *Optional, bool* - whether to search documents recursively in the `dir_path`. Default is True.
- `max_tokens6` *Optional, bool* - whether to add more documents in the collection. Default is False

#### Returns:

- `max_tokens7` - the chromadb client.

#### query\_vector\_db

```
def query_vector_db(query_texts: List[str],
                    n_results: int = 10,
                    client: API = None,
                    db_path: str = "/tmp/chromadb.db",
                    collection_name: str = "all-my-documents",
                    search_string: str = "",
                    embedding_model: str = "all-MiniLM-L6-v2",
                    embedding_function: Callable = None) -> QueryResult
```

Query a vector db. We support chromadb compatible APIs, it's not required if you prepared your own vector db and query function.

#### Arguments:

- `query_texts` *List[str]* - the list of strings which will be used to query the vector db.
- `n_results` *Optional, int* - the number of results to return. Default is 10.
- `client` *Optional, API* - the chromadb compatible client. Default is None, a chromadb client will be used.
- `db_path` *Optional, str* - the path to the vector db. Default is "/tmp/chromadb.db".
- `collection_name` *Optional, str* - the name of the collection. Default is "all-my-documents".
- `search_string` *Optional, str* - the search string. Only docs that contain an exact match of this string will be retrieved. 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.
- `embedding_function` *Optional, Callable* - the embedding function to use. Default is None, SentenceTransformer with the given `embedding_model` will be used. If you want to use OpenAI, Cohere, HuggingFace or other embedding functions, you can pass it here, follow the examples in <https://docs.trychroma.com/embeddings>.

#### Returns:

- `n_results0` - the query result. The format is: class QueryResult(TypedDict):
- `n_results1` - List[IDs]

- `n_results2` - Optional[List[List[Embedding]]]
- `n_results3` - Optional[List[List[Document]]]
- `n_results4` - Optional[List[List[Metadata]]]
- `n_results5` - Optional[List[List[float]]]

 [Edit this page](#)

[Previous](#)

[« math\\_utils](#)

[Next](#)

[token\\_count\\_utils »](#)

Community

[Discord](#) 

[Twitter](#) 

Copyright © 2024 AutoGen Authors | [Privacy and Cookies](#)