langchain_community.vectorstores.pathway.PathwayVectorClient

class langchain_community.vectorstores.pathway.**PathwayVectorClient**(*host: Optional[str] = None*, *port: Optional[int] = None*, *url: Optional[str] = None*) [source]

VectorStore connecting to Pathway Vector Store.

A client you can use to query Pathway Vector Store.

Please provide aither the *url*, or *host* and *port*.

Parameters

- **host** (-) – host on which Pathway Vector Store listens
- **port** (-) – port on which Pathway Vector Store listens
- **url** (-) – url at which Pathway Vector Store listens

Attributes

embeddings	Access the query embedding object if available.

Methods

__init__([host, port, url])	A client you can use to query Pathway Vector Store.
aadd_documents(documents, **kwargs)	Async run more documents through the embeddings and add to the vectorstore.
aadd_texts(texts[, metadatas])	Async run more texts through the embeddings and add to the vectorstore.
add_documents(documents, **kwargs)	Add or update documents in the vectorstore.
add_texts(texts[, metadatas])	Pathway is not suitable for this method.
adelete([ids])	Async delete by vector ID or other criteria.
afrom_documents(documents, embedding, **kwargs)	Async return VectorStore initialized from documents and embeddings.
afrom_texts(texts, embedding[, metadatas])	Async return VectorStore initialized from texts and embeddings.
aget_by_ids(ids, /)	Async get documents by their IDs.
amax_marginal_relevance_search(query[, k, ...])	Async return docs selected using the maximal marginal relevance.
amax_marginal_relevance_search_by_vector(...)	Async return docs selected using the maximal marginal relevance.
as_retriever(**kwargs)	Return VectorStoreRetriever initialized from this VectorStore.
asearch(query, search_type, **kwargs)	Async return docs most similar to query using a specified search type.
asimilarity_search(query[, k])	Async return docs most similar to query.
asimilarity_search_by_vector(embedding[, k])	Async return docs most similar to embedding vector.
asimilarity_search_with_relevance_scores(query)	Async return docs and relevance scores in the range [0, 1].
asimilarity_search_with_score(*args, **kwargs)	Async run similarity search with distance.
delete([ids])	Delete by vector ID or other criteria.
from_documents(documents, embedding, **kwargs)	Return VectorStore initialized from documents and embeddings.
from_texts(texts, embedding[, metadatas])	Return VectorStore initialized from texts and embeddings.
get_by_ids(ids, /)	Get documents by their IDs.
get_input_files([metadata_filter, ...])	List files indexed by the Vector Store.
get_vectorstore_statistics()	Fetch basic statistics about the Vector Store.
max_marginal_relevance_search(query[, k, ...])	Return docs selected using the maximal marginal relevance.
max_marginal_relevance_search_by_vector(...)	Return docs selected using the maximal marginal relevance.
search(query, search_type, **kwargs)	Return docs most similar to query using a specified search type.
similarity_search(query[, k])	Return docs most similar to query.
similarity_search_by_vector(embedding[, k])	Return docs most similar to embedding vector.
similarity_search_with_relevance_scores(query)	Return docs and relevance scores in the range [0, 1].
similarity_search_with_score(query[, k, ...])	Run similarity search with Pathway with distance.

A client you can use to query Pathway Vector Store.

Please provide aither the *url*, or *host* and *port*.

Parameters

- **host** (-) – host on which Pathway Vector Store listens
- **port** (-) – port on which Pathway Vector Store listens
- **url** (-) – url at which Pathway Vector Store listens

Return type

None

async **aadd_documents**(*documents: List[Document]*, ***kwargs: Any*) → List[str]

Async run more documents through the embeddings and add to the vectorstore.

Parameters

- **documents** (*List[Document]*) – Documents to add to the vectorstore.
- **kwargs** (*Any*) – Additional keyword arguments.

Returns

List of IDs of the added texts.

Raises

ValueError – If the number of IDs does not match the number of documents.

Return type

List[str]

async **aadd_texts**(*texts: Iterable[str]*, *metadatas: Optional[List[dict]] = None*, ***kwargs: Any*) → List[str]

Async run more texts through the embeddings and add to the vectorstore.

Parameters

- **texts** (*Iterable[str]*) – Iterable of strings to add to the vectorstore.
- **metadatas** (*Optional[List[dict]]*) – Optional list of metadatas associated with the texts. Default is None.
- ****kwargs** (*Any*) – vectorstore specific parameters.

Returns

List of ids from adding the texts into the vectorstore.

Raises

- **ValueError** – If the number of metadatas does not match the number of texts.
- **ValueError** – If the number of ids does not match the number of texts.

Return type

List[str]

add_documents(*documents: List[Document]*, ***kwargs: Any*) → List[str]

Add or update documents in the vectorstore.

Parameters

- **documents** (*List[Document]*) – Documents to add to the vectorstore.
- **kwargs** (*Any*) – Additional keyword arguments. if kwargs contains ids and documents contain ids, the ids in the kwargs will receive precedence.

Returns

Raises

ValueError – If the number of ids does not match the number of documents.

Return type

List[str]

add_texts(*texts: Iterable[str]*, *metadatas: Optional[List[dict]] = None*, ***kwargs: Any*) → List[str] [source]

Pathway is not suitable for this method.

Parameters

- **texts** (*Iterable[str]*) –
- **metadatas** (*Optional[List[dict]]*) –
- **kwargs** (*Any*) –

Return type

List[str]

async **adelete**(*ids: Optional[List[str]] = None*, ***kwargs: Any*) → Optional[bool]

Async delete by vector ID or other criteria.

Parameters

- **ids** (*Optional[List[str]]*) – List of ids to delete. If None, delete all. Default is None.
- ****kwargs** (*Any*) – Other keyword arguments that subclasses might use.

Returns

True if deletion is successful, False otherwise, None if not implemented.

Return type

Optional[bool]

async classmethod **afrom_documents**(*documents: List[Document]*, *embedding: Embeddings*, ***kwargs: Any*) → VST

Async return VectorStore initialized from documents and embeddings.

Parameters

- **documents** (*List[Document]*) – List of Documents to add to the vectorstore.
- **embedding** (*Embeddings*) – Embedding function to use.
- **kwargs** (*Any*) – Additional keyword arguments.

Returns

VectorStore initialized from documents and embeddings.

Return type

VectorStore

async classmethod **afrom_texts**(*texts: List[str]*, *embedding: Embeddings*, *metadatas: Optional[List[dict]] = None*, ***kwargs: Any*) → VST

Async return VectorStore initialized from texts and embeddings.

Parameters

- **texts** (*List[str]*) – Texts to add to the vectorstore.
- **embedding** (*Embeddings*) – Embedding function to use.
- **metadatas** (*Optional[List[dict]]*) – Optional list of metadatas associated with the texts. Default is None.

Returns

VectorStore initialized from texts and embeddings.

Return type

VectorStore

async **aget_by_ids**(*ids: Sequence[str]*, */*) → List[Document]

Async get documents by their IDs.

The returned documents are expected to have the ID field set to the ID of the document in the vector store.

Fewer documents may be returned than requested if some IDs are not found or if there are duplicated IDs.

Users should not assume that the order of the returned documents matches the order of the input IDs. Instead, users should rely on the ID field of the returned documents.

This method should **NOT** raise exceptions if no documents are found for some IDs.

Parameters

ids (*Sequence[str]*) – List of ids to retrieve.

Returns

List of Documents.

Return type

List[Document]

New in version 0.2.11.

async **amax_marginal_relevance_search**(*query: str*, *k: int = 4*, *fetch_k: int = 20*, *lambda_mult: float = 0.5*, ***kwargs: Any*) → List[Document]

Async return docs selected using the maximal marginal relevance.

Maximal marginal relevance optimizes for similarity to query AND diversity among selected documents.

Parameters

- **query** (*str*) – Text to look up documents similar to.
- **k** (*int*) – Number of Documents to return. Defaults to 4.
- **fetch_k** (*int*) – Number of Documents to fetch to pass to MMR algorithm. Default is 20.
- **lambda_mult** (*float*) – Number between 0 and 1 that determines the degree of diversity among the results with 0 corresponding to maximum diversity and 1 to minimum diversity. Defaults to 0.5.
- **kwargs** (*Any*) –

Returns

List of Documents selected by maximal marginal relevance.

Return type

List[Document]

async **amax_marginal_relevance_search_by_vector**(*embedding: List[float]*, *k: int = 4*, *fetch_k: int = 20*, *lambda_mult: float = 0.5*, ***kwargs: Any*) → List[Document]

Async return docs selected using the maximal marginal relevance.

Maximal marginal relevance optimizes for similarity to query AND diversity among selected documents.

Parameters

- **embedding** (*List[float]*) – Embedding to look up documents similar to.

- **fetch_k** (*int*) – Number of Documents to fetch to pass to MMR algorithm. Default is 20.
- **lambda_mult** (*float*) – Number between 0 and 1 that determines the degree of diversity among the results with 0 corresponding to maximum diversity and 1 to minimum diversity. Defaults to 0.5.
- ****kwargs** (*Any*) – Arguments to pass to the search method.

Returns

List of Documents selected by maximal marginal relevance.

Return type

List[Document]

`as_retriever`(***kwargs: Any*) → VectorStoreRetriever

Return VectorStoreRetriever initialized from this VectorStore.

Parameters

****kwargs** (*Any*) –

Keyword arguments to pass to the search function. Can include: search_type (Optional[str]): Defines the type of search that the Retriever should perform. Can be “similarity” (default), “mmr”, or “similarity_score_threshold”.

search_kwargs (Optional[Dict]): Keyword arguments to pass to the search function. Can include things like:

k: Amount of documents to return (Default: 4) score_threshold: Minimum relevance threshold

for similarity_score_threshold

fetch_k: Amount of documents to pass to MMR algorithm

(Default: 20)

lambda_mult: Diversity of results returned by MMR;

1 for minimum diversity and 0 for maximum. (Default: 0.5)

filter: Filter by document metadata

Returns

Retriever class for VectorStore.

Return type

VectorStoreRetriever

Examples:

```
# Useful if your dataset has many similar documents
docsearch.as_retriever(
    search_type="mmr",
    search_kwargs={'k': 6, 'lambda_mult': 0.25}
)

# Fetch more documents for the MMR algorithm to consider
# But only return the top 5
docsearch.as_retriever(
    search_type="mmr",
    search_kwargs={'k': 5, 'fetch_k': 50}
)

# Only retrieve documents that have a relevance score
# Above a certain threshold
docsearch.as_retriever(
    search_type="similarity_score_threshold",
    search_kwargs={'score_threshold': 0.8}
)

# Only get the single most similar document from the dataset
docsearch.as_retriever(search_kwargs={'k': 1})

# Use a filter to only retrieve documents from a specific paper
docsearch.as_retriever(
    search_kwargs={'filter': {'paper_title':'GPT-4 Technical Report'}}
)
```

async **asearch**(*query: str*, *search_type: str*, ***kwargs: Any*) → List[Document]

Async return docs most similar to query using a specified search type.

Parameters

- **query** (*str*) – Input text.
- **search_type** (*str*) – Type of search to perform. Can be “similarity”, “mmr”, or “similarity_score_threshold”.
- ****kwargs** (*Any*) – Arguments to pass to the search method.

Returns

List of Documents most similar to the query.

Raises

ValueError – If search_type is not one of “similarity”, “mmr”, or “similarity_score_threshold”.

Return type

List[Document]

async **asimilarity_search**(*query: str*, *k: int = 4*, ***kwargs: Any*) → List[Document]

Async return docs most similar to query.

Parameters

- **query** (*str*) – Input text.
- **k** (*int*) – Number of Documents to return. Defaults to 4.
- ****kwargs** (*Any*) – Arguments to pass to the search method.

Returns

List of Documents most similar to the query.

Return type

List[Document]

async **asimilarity_search_by_vector**(*embedding: List[float]*, *k: int = 4*, ***kwargs: Any*) → List[Document]

Async return docs most similar to embedding vector.

- **embedding** (*List[float]*) – Embedding to look up documents similar to.
- **k** (*int*) – Number of Documents to return. Defaults to 4.
- ****kwargs** (*Any*) – Arguments to pass to the search method.

Returns

List of Documents most similar to the query vector.

Return type

List[Document]

async **asimilarity_search_with_relevance_scores**(*query: str*, *k: int = 4*, ***kwargs: Any*) → List[Tuple[Document, float]]

Async return docs and relevance scores in the range [0, 1].

0 is dissimilar, 1 is most similar.

Parameters

- **query** (*str*) – Input text.
- **k** (*int*) – Number of Documents to return. Defaults to 4.
- ****kwargs** (*Any*) –
  kwargs to be passed to similarity search. Should include: score_threshold: Optional, a floating point value between 0 to 1 to filter the resulting set of retrieved docs

Returns

List of Tuples of (doc, similarity_score)

Return type

List[Tuple[Document, float]]

async **asimilarity_search_with_score**(**args: Any*, ***kwargs: Any*) → List[Tuple[Document, float]]

Async run similarity search with distance.

Parameters

- ***args** (*Any*) – Arguments to pass to the search method.
- ****kwargs** (*Any*) – Arguments to pass to the search method.

Returns

List of Tuples of (doc, similarity_score).

Return type

List[Tuple[Document, float]]

delete(*ids: Optional[List[str]] = None*, ***kwargs: Any*) → Optional[bool]

Delete by vector ID or other criteria.

Parameters

- **ids** (*Optional[List[str]]*) – List of ids to delete. If None, delete all. Default is None.
- ****kwargs** (*Any*) – Other keyword arguments that subclasses might use.

Returns

True if deletion is successful, False otherwise, None if not implemented.

Return type

Optional[bool]

Return VectorStore initialized from documents and embeddings.

Parameters

- **documents** (*List[Document]*) – List of Documents to add to the vectorstore.
- **embedding** (*Embeddings*) – Embedding function to use.
- **kwargs** (*Any*) – Additional keyword arguments.

Returns

VectorStore initialized from documents and embeddings.

Return type

VectorStore

classmethod **from_texts**(*texts: List[str]*, *embedding: Embeddings*, *metadatas: Optional[List[dict]] = None*, ***kwargs: Any*) → PathwayVectorClient [source]

Return VectorStore initialized from texts and embeddings.

Parameters

- **texts** (*List[str]*) – Texts to add to the vectorstore.
- **embedding** (*Embeddings*) – Embedding function to use.
- **metadatas** (*Optional[List[dict]]*) – Optional list of metadatas associated with the texts. Default is None.
- **kwargs** (*Any*) – Additional keyword arguments.

Returns

VectorStore initialized from texts and embeddings.

Return type

VectorStore

get_by_ids(*ids: Sequence[str]*, */*) → List[Document]

Get documents by their IDs.

The returned documents are expected to have the ID field set to the ID of the document in the vector store.

Fewer documents may be returned than requested if some IDs are not found or if there are duplicated IDs.

Users should not assume that the order of the returned documents matches the order of the input IDs. Instead, users should rely on the ID field of the returned documents.

This method should **NOT** raise exceptions if no documents are found for some IDs.

Parameters

ids (*Sequence[str]*) – List of ids to retrieve.

Returns

List of Documents.

Return type

List[Document]

New in version 0.2.11.

get_input_files(*metadata_filter: Optional[str] = None*, *filepath_globpattern: Optional[str] = None*) → list [source]

List files indexed by the Vector Store.

Parameters

- **filepath_globpattern** (*Optional[str]*) –

Return type

list

get_vectorstore_statistics() → dict [source]

Fetch basic statistics about the Vector Store.

Return type

dict

max_marginal_relevance_search(*query: str*, *k: int = 4*, *fetch_k: int = 20*, *lambda_mult: float = 0.5*, ***kwargs: Any*) → List[Document]

Return docs selected using the maximal marginal relevance.

Maximal marginal relevance optimizes for similarity to query AND diversity among selected documents.

Parameters

- **query** (*str*) – Text to look up documents similar to.
- **k** (*int*) – Number of Documents to return. Defaults to 4.
- **fetch_k** (*int*) – Number of Documents to fetch to pass to MMR algorithm. Default is 20.
- **lambda_mult** (*float*) – Number between 0 and 1 that determines the degree of diversity among the results with 0 corresponding to maximum diversity and 1 to minimum diversity. Defaults to 0.5.
- ****kwargs** (*Any*) – Arguments to pass to the search method.

Returns

List of Documents selected by maximal marginal relevance.

Return type

List[Document]

max_marginal_relevance_search_by_vector(*embedding: List[float]*, *k: int = 4*, *fetch_k: int = 20*, *lambda_mult: float = 0.5*, ***kwargs: Any*) → List[Document]

Return docs selected using the maximal marginal relevance.

Maximal marginal relevance optimizes for similarity to query AND diversity among selected documents.

Parameters

- **embedding** (*List[float]*) – Embedding to look up documents similar to.
- **k** (*int*) – Number of Documents to return. Defaults to 4.
- **fetch_k** (*int*) – Number of Documents to fetch to pass to MMR algorithm. Default is 20.
- **lambda_mult** (*float*) – Number between 0 and 1 that determines the degree of diversity among the results with 0 corresponding to maximum diversity and 1 to minimum diversity. Defaults to 0.5.
- ****kwargs** (*Any*) – Arguments to pass to the search method.

Returns

List of Documents selected by maximal marginal relevance.

Return type

List[Document]

search(*query: str*, *search_type: str*, ***kwargs: Any*) → List[Document]

Return docs most similar to query using a specified search type.

- **query** (*str*) – Input text
- **search_type** (*str*) – Type of search to perform. Can be “similarity”, “mmr”, or “similarity_score_threshold”.
- ****kwargs** (*Any*) – Arguments to pass to the search method.

Returns

List of Documents most similar to the query.

Raises

ValueError – If search_type is not one of “similarity”, “mmr”, or “similarity_score_threshold”.

Return type

List[Document]

`similarity_search`(*query: str*, *k: int = 4*, ***kwargs: Any*) → List[Document] [source]

Return docs most similar to query.

Parameters

- **query** (*str*) – Input text.
- **k** (*int*) – Number of Documents to return. Defaults to 4.
- ****kwargs** (*Any*) – Arguments to pass to the search method.

Returns

List of Documents most similar to the query.

Return type

List[*Document*]

`similarity_search_by_vector`(*embedding: List[float]*, *k: int = 4*, ***kwargs: Any*) → List[Document]

Return docs most similar to embedding vector.

Parameters

- **embedding** (*List[float]*) – Embedding to look up documents similar to.
- **k** (*int*) – Number of Documents to return. Defaults to 4.
- ****kwargs** (*Any*) – Arguments to pass to the search method.

Returns

List of Documents most similar to the query vector.

Return type

List[Document]

`similarity_search_with_relevance_scores`(*query: str*, *k: int = 4*, ***kwargs: Any*) → List[Tuple[Document, float]]

Return docs and relevance scores in the range [0, 1].

0 is dissimilar, 1 is most similar.

Parameters

- **query** (*str*) – Input text.
- **k** (*int*) – Number of Documents to return. Defaults to 4.
- ****kwargs** (*Any*) –
  kwargs to be passed to similarity search. Should include: score_threshold: Optional, a floating point value between 0 to 1 to filter the resulting set of retrieved docs.

Returns

List of Tuples of (doc, similarity_score).

List[Tuple[Document, float]]

---

similarity_search_with_score(*query: str*, *k: int = 4*, *metadata_filter: Optional[str] = None*) → List[Tuple[Document, float]]

[source]

Run similarity search with Pathway with distance.

Parameters

- **query** (-) – Query text to search for.
- **k** (-) – Number of results to return. Defaults to 4.
- **metadata_filter** (-) – Filter by metadata. Filtering query should be in JMESPath format. Defaults to None.

Returns

List of documents most similar to the query text and cosine distance in float for each. Lower score represents more similarity.

Return type

List[Tuple[Document, float]]

Examples using PathwayVectorClient

- Pathway