



Modules

Retrieval

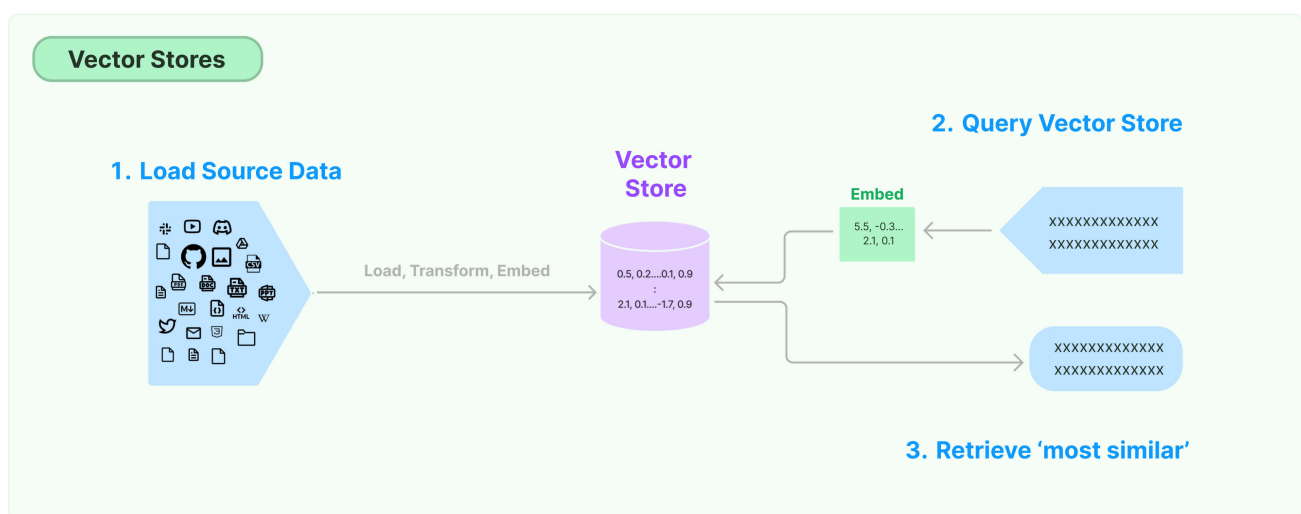
Vector stores

Vector stores

! INFO

Head to [Integrations](#) for documentation on built-in integrations with 3rd-party vector stores.

One of the most common ways to store and search over unstructured data is to embed it and store the resulting embedding vectors, and then at query time to embed the unstructured query and retrieve the embedding vectors that are 'most similar' to the embedded query. A vector store takes care of storing embedded data and performing vector search for you.



Get started

This walkthrough showcases basic functionality related to vector stores. A key part of working with vector stores is creating the vector to put in them, which is usually created via embeddings. Therefore, it is recommended that you familiarize yourself with the **text embedding model** interfaces before diving into this.

There are many great vector store options, here are a few that are free, open-source, and run entirely on your local machine. Review all integrations for many great hosted offerings.

Chroma **FAISS** **Lance**

This walkthrough uses the **chroma** vector database, which runs on your local machine as a library.

```
pip install chromadb
```

We want to use OpenAIEmbeddings so we have to get the OpenAI API Key.

```
import os
import getpass

os.environ['OPENAI_API_KEY'] =
getpass.getpass('OpenAI API Key:')
```

```
from langchain_community.document_loaders import
TextLoader
from langchain_openai import OpenAIEmbeddings
from langchain.text_splitter import
CharacterTextSplitter
from langchain_community.vectorstores import Ch

# Load the document, split it into chunks, embed
each chunk and load it into the vector store.
raw_documents =
TextLoader('../.../state_of_the_union.txt').load_documents()
text_splitter =
CharacterTextSplitter(chunk_size=1000,
chunk_overlap=0)
documents =
text_splitter.split_documents(raw_documents)
db = Chroma.from_documents(documents,
OpenAIEmbeddings())
```

Similarity search

```
query = "What did the president say about  
Ketanji Brown Jackson"  
docs = db.similarity_search(query)  
print(docs[0].page_content)
```

Tonight. I call on the Senate to: Pass the Freedom to Vote Act. Pass the John Lewis Voting Rights Act. And while you're at it, pass the Disclose Act so Americans can know who is funding our elections.

Tonight, I'd like to honor someone who has dedicated his life to serve this country: Justice Stephen Breyer—an Army veteran, Constitutional scholar, and retiring Justice of the United States Supreme Court. Justice Breyer, thank you for your service.

One of the most serious constitutional responsibilities a President has is nominating someone to serve on the United States Supreme Court.

And I did that 4 days ago, when I nominated Circuit Court of Appeals Judge Ketanji Brown Jackson. One of our nation's top legal minds, who will continue Justice Breyer's legacy of excellence.

Similarity search by vector

It is also possible to do a search for documents similar to a given embedding vector using

`similarity_search_by_vector` which accepts an embedding vector as a parameter instead of a string.

```
embedding_vector =  
OpenAIEmbeddings().embed_query(query)  
docs =  
db.similarity_search_by_vector(embedding_vector)  
print(docs[0].page_content)
```

The query is the same, and so the result is also the same.

Tonight. I call on the Senate to: Pass the Freedom to Vote Act. Pass the John Lewis Voting Rights Act. And while you're at it, pass the Disclose Act so Americans can know who is funding our elections.

Tonight, I'd like to honor someone who has dedicated his life to serve this country: Justice Stephen Breyer—an Army veteran, Constitutional scholar, and retiring Justice of the United States Supreme Court. Justice Breyer, thank you for your service.

One of the most serious constitutional responsibilities a President has is nominating someone to serve on the United States Supreme Court.

And I did that 4 days ago, when I nominated Circuit Court of Appeals Judge Ketanji Brown Jackson. One of our nation's top legal minds, who will continue Justice Breyer's legacy of excellence.

Asynchronous operations

Vector stores are usually run as a separate service that requires some IO operations, and therefore they might be called asynchronously. That gives performance benefits as you don't waste time waiting for responses from external services. That might also be important if you work with an asynchronous framework, such as [FastAPI](#).

LangChain supports async operation on vector stores. All the methods might be called using their async counterparts, with the prefix `a`, meaning `async`.

`Qdrant` is a vector store, which supports all the async operations, thus it will be used in this walkthrough.

```
pip install qdrant-client
```

```
from langchain_community.vectorstores import  
Qdrant
```

Create a vector store asynchronously

```
db = await Qdrant.afrom_documents(documents,  
embeddings, "http://localhost:6333")
```

Similarity search

```
query = "What did the president say about  
Ketanji Brown Jackson"  
docs = await db.asimilarity_search(query)  
print(docs[0].page_content)
```

Tonight. I call on the Senate to: Pass the Freedom to Vote Act. Pass the John Lewis Voting Rights Act. And while you're at it, pass the Disclose Act so Americans can know who is funding our elections.

Tonight, I'd like to honor someone who has dedicated his life to serve this country:

Justice Stephen Breyer—an Army veteran, Constitutional scholar, and retiring Justice of the United States Supreme Court. Justice Breyer, thank you for your service.

One of the most serious constitutional responsibilities a President has is nominating someone to serve on the United States Supreme Court.

And I did that 4 days ago, when I nominated Circuit Court of Appeals Judge Ketanji Brown Jackson. One of our nation's top legal minds, who will continue Justice Breyer's legacy of excellence.

Similarity search by vector

```
embedding_vector = embeddings.embed_query(query)
docs = await
db.asimilarity_search_by_vector(embedding_vecto
```

Maximum marginal relevance search (MMR)

Maximal marginal relevance optimizes for similarity to query **and** diversity among selected documents. It is also supported

in async API.

```
query = "What did the president say about  
Ketanji Brown Jackson"  
found_docs = await  
qdrant.amax_marginal_relevance_search(query,  
k=2, fetch_k=10)  
for i, doc in enumerate(found_docs):  
    print(f"{i + 1}.", doc.page_content,  
"\n")
```

1. Tonight. I call on the Senate to: Pass the Freedom to Vote Act. Pass the John Lewis Voting Rights Act. And while you're at it, pass the Disclose Act so Americans can know who is funding our elections.

Tonight, I'd like to honor someone who has dedicated his life to serve this country: Justice Stephen Breyer—an Army veteran, Constitutional scholar, and retiring Justice of the United States Supreme Court. Justice Breyer, thank you for your service.

One of the most serious constitutional responsibilities a President has is nominating someone to serve on the United States Supreme Court.

And I did that 4 days ago, when I nominated Circuit Court of Appeals Judge Ketanji Brown Jackson. One of our nation's top legal minds, who will continue Justice Breyer's legacy of excellence.

2. We can't change how divided we've been. But we can change how we move forward—on COVID-19 and other issues we must face together.

I recently visited the New York City Police Department days after the funerals of Officer Wilbert Mora and his partner, Officer Jason Rivera.

They were responding to a 9-1-1 call when a man shot and killed them with a stolen gun.

Officer Mora was 27 years old.

Officer Rivera was 22.

Both Dominican Americans who'd grown up on the same streets they later chose to patrol as police officers.

I spoke with their families and told them that we are forever in debt for their sacrifice, and we will carry on their mission to restore the trust and safety every

community deserves.

I've worked on these issues a long time.

I know what works: Investing in crime prevention and community police officers who'll walk the beat, who'll know the neighborhood, and who can restore trust and safety.