Atlas Vector Search Quick Start

Select your language

Python

This quick start describes how to load sample documents that contain vector embeddings into an Atlas cluster or local Atlas deployment, create an Atlas Vector Search index on those embeddings, and then perform semantic search to return documents that are similar to your query.

Time required: 15 minutes

Work with a runnable version of this tutorial as a **Python notebook.**

Objectives

In this quick start, you will do the following steps:

1. Create an index definition for the `sample_mflix.embedded_movies` collection that indexes the `plot_embedding` field as the `vector` type. The `plot_embedding` field contains embeddings created using OpenAI's `text-embedding-ada-002` embedding model. The index definition specifies `1536` vector dimensions and measures similarity using `dotProduct`.
2. Run an Atlas Vector Search query that searches the sample `sample_mflix.embedded_movies` collection. The query uses the `$vectorSearch` stage to search the `plot_embedding` field, which contains embeddings created using OpenAI's `text-embedding-ada-002` embedding model. The query searches the `plot_embedding` field using vector embeddings for the string *time travel*. It considers up to `150` nearest neighbors, and returns `10` documents in the results.

To learn more, see **Learning Summary.**

Create a Vector Search Index

➤ To set the client you use to run the examples on this page, use the **Select your language** drop-down menu in the right navigation pane.

In this section, you create an Atlas Vector Search index on sample data that you load into an Atlas cluster or a deployment hosted on your local computer:

Atlas Cluster Local Deployment

1 **Set up your Atlas cluster.**

a. **Create a free Atlas account or sign in to an existing account.**

b. If you don't yet have an Atlas cluster, **create a free M0 cluster.** To learn more about creating an Atlas cluster, see **Create a Cluster.**

> **NOTE**
>
> If you are working with an existing cluster, you must have `Project Data Access Admin` or higher **access** to your Atlas project.
>
> If you create a new cluster, you have the necessary permissions by default.

You can create only one `M0` Free cluster per **project.**

c. In the left sidebar, click **Atlas Search**. Choose your cluster from the **Select data source** menu and click **Go to Atlas Search**.

d. If you haven't yet loaded the sample dataset onto your cluster, click **Load a Sample Dataset**. In the Load Sample Dataset dialog box, click **Load Sample Dataset** to confirm.

   If you already loaded the sample dataset, **check** that the `sample_mflix` database contains the `embedded_movies` collection. If it doesn't, **drop** the sample databases and **reload** the sample dataset.

Loading the **sample dataset** can take several minutes to complete.

2 Create a Vector Search index.

NOTE

You can use the `mongosh` command or driver helper methods to create Atlas Vector Search indexes on all Atlas cluster tiers. For a list of supported driver versions, see **Supported Clients.**

a. Add the PyMongo Driver as a dependency in your project:

Python

```
pip install pymongo
```

For more detailed installation instructions, see the **MongoDB Python Driver documentation.**

b. Define the index.

Create a file named `vector-index.py`. Copy and paste the following code into the file.

vector-index.py

Python

```
from pymongo.mongo_client import MongoC
from pymongo.operations import SearchIn
import time

# Connect to your Atlas deployment
uri = "<connectionString>"
client = MongoClient(uri)

# Access your database and collection
database = client["sample_mflix"]
collection = database["embedded_movies'

# Create your index model, then create
search_index_model = SearchIndexModel(
  definition={
    "fields": [
      {
        "type": "vector",
        "path": "plot_embedding",
        "numDimensions": 1536,
        "similarity": "dotProduct",
        "quantization": "scalar"
      }
    ]
  },
  name="vector_index",
  type="vectorSearch"
)

result = collection.create_search_index
print("New search index named " + resul

# Wait for initial sync to complete
print("Polling to check if the index is
predicate=None
if predicate is None:
  predicate = lambda index: index.get('

while True:
  indices = list(collection.list_search
  if len(indices) and predicate(indices
    break
  time.sleep(5)
print(result + " is ready for querying.

client.close()
```

This index definition:

  - Indexes the `plot_embedding` field as the `vector` **type.** This field contains **vector embeddings** that represent the summary of a movie's plot.
    - Specifies `1536` **vector dimensions.**
    - Measures **similarity** using `dotProduct` similarity.
    - Enables **automatic** `quantization` of the vectors.

  This code also includes a polling mechanism to check if the index is ready to use.

c. Specify the `<connection-string>`.

  Replace `<connection-string>` with the connection string for your Atlas cluster or local deployment, and then save the file.

  Your connection string should use the following format:

```
mongodb+srv://<db_username>:<db_passwo
```

NOTE

Ensure that your connection string includes your database user's credentials. To learn more about finding your connection string, see **Connect via Drivers.**

d. Run the following command to create the index.

Python

```
python vector-index.py
```

HIDE OUTPUT

```
New search index named vector_index is bui
Polling to check if the index is ready. Th
vector_index is ready for querying.
```

Run a Vector Search Query

In this section, you run a sample vector search query on your indexed embeddings.

1 Construct your vector search query.

This query searches for documents that include text in the `plot` field that is semantically related to the term "time travel".

a. Create a file named `atlas-vector-search-quick-start.py`.

b. Copy and paste the following sample query into the `atlas-vector-search-quick-start.py` file:

Python

```
import pymongo

# connect to your Atlas cluster
client = pymongo.MongoClient("<connect

# define pipeline
pipeline = [
  {
    '$vectorSearch': {
      'index': 'vector_index',
      'path': 'plot_embedding',
      'queryVector': [-0.0016261312, -0
      'numCandidates': 150,
      'limit': 10
    }
  }, {
    '$project': {
      '_id': 0,
      'plot': 1,
      'title': 1,
      'score': {
        '$meta': 'vectorSearchScore'
      }
    }
  }
]
```

```

# run pipeline
result = client["sample_mflix"]["embedd

# print results
for i in result:
    print(i)

```

This query uses the `$vectorSearch` stage to:

- Compare vector embeddings of the search term against vector embeddings of movie plots in the `plot_embedding` field of the `sample_mflix.embedded_movies` collection.
- Consider up to the 150 most similar movie plots and return the top 10 results.

It uses the `$project` stage to:

- Only include the movie `plot` and `title` fields in the results.

- Add a `score` field to show the relevance of each result to the search term.

To learn more about this pipeline stage, see **Run Vector Search Queries.**

2 Specify the `<connection-string>`.

Replace `<connection-string>` with the connection string for your Atlas cluster or local deployment, and then save the file.

Your connection string should use the following format:

```
mongodb+srv://<db_username>:<db_password>@
```

NOTE

Ensure that your connection string includes your database user's credentials. To learn more about finding your connection string, see **Connect via Drivers.**

3 Run your query.

Run the following command to query your collection:

Python

```
python atlas-vector-search-quick-start.py
```

HIDE OUTPUT

```
{'plot': 'A reporter, learning of time trav
{'plot': 'At the age of 21, Tim discovers h
{'plot': 'Hoping to alter the events of the
{'plot': "After using his mother's newly bu
{'plot': 'An officer for a security agency
{'plot': 'A time-travel experiment in which
{'plot': "Agent J travels in time to M.I.B.
{'plot': 'Bound by a shared destiny, a teen
{'plot': 'With the help of his uncle, a man
{'plot': 'A dimension-traveling wizard gets
```

Learning Summary

This quick start focused on retrieving documents from your Atlas cluster that contain text that is semantically related to a provided query. However, you can create a vector search index on embeddings that represent any type of data that you might write to an Atlas cluster, such as images or videos.

Sample Data

This quick start uses the `sample_mflix.embedded_movies` collection which contains details about movies. In each document in the collection, the `plot_embedding` field contains a vector embedding that represents the string in the `plot` field. For more information on the schema of the documents in the collection, see **Sample Mflix Dataset.**

By storing your source data and its corresponding vector embeddings in the same document, you can leverage both fields for complex queries or **hybrid search.** You can even store vector embeddings generated from different **embedding models** in the same document to streamline your workflow as you test the performance of different vector embedding models for your specific use case.

Vector Embeddings

The vector embeddings in the `sample_mflix.embedded_movies` collection and in the example query were created using the OpenAI `text-embedding-ada-002` embedding model. Your choice of embedding model informs the vector dimensions and vector similarity function you use in your vector search index. You can use any **embedding model** you like, and it is worth experimenting with different models as accuracy can vary from model to model depending on your specific use case.

To learn how to create vector embeddings of your own data, see **How to Create Vector Embeddings.**

Vector Index Definition

An **index** is a data structure that holds a subset of data from a collection's documents that improves database performance for specific queries. A **vector search index** points to the fields that contain your vector embeddings and includes the dimensions of your vectors as well as the function used to measure similarity between vectors of queries and vectors stored in the database.

Because the `text-embedding-ada-002` embedding model used in this quick start converts data into vector embeddings with 1536 dimensions and supports the `cosine` function, this vector search index specifies the same number of vector dimensions and similarity function.

Vector Search Query

The query you ran in this quick start is an **aggregation pipeline,** in which the `$vectorSearch` stage performs an **Approximate Nearest Neighbor (ANN)** search followed by a `$project` stage that refines the results. To see all the options for a vector search query, including using **Exact Nearest Neighbor (ENN)** or how to narrow the scope of your vector search with the `filter` option, see **Run Vector Search Queries.**

Next Steps

- To learn how to create embeddings from data and load them into Atlas, see **Create Embeddings.**
- To learn how to implement retrieval-augmented generation (RAG), see **Retrieval-Augmented Generation (RAG) with Atlas Vector Search.**
- To integrate Atlas Vector Search with popular AI frameworks and services, see **Integrate Vector Search with AI Technologies.**
- To build production ready AI chatbots using Atlas Vector Search, see the **MongoDB Chatbot Framework.**

- To learn how implement RAG without the need for API keys or credits, see **Build a Local RAG Implementation with Atlas Vector Search.**

Atlas Status

Customer Support

Manage Cookies

Deployment Options

MongoDB Atlas

Enterprise Advanced

Community Edition