

Vector Search Embedding in Databricks

PoC Workflow Overview

- **Goal:** Identify usage and benefit of storing embeddings on Databricks to serve an LLM model.
- **Key Steps:**
 - Create Delta table with source data.
 - Create Vector Search Endpoint (UI or SDK).
 - Create the Index (target columns, embedding model).
 - Query the Index (semantic search).
 - Serve the Index via RAG to a Language Model.

Databricks Setup: Vector Search & Index

Vector Search Prerequisites

- **Unity Catalog** enabled workspace.
- **Serverless compute** available.

Index Creation

- Can be done via **UI** or **Python SDK**.
- Requires: Primary ID, text columns, and an **Embedding Model**.
- *Note: Change Data Feed must be enabled on the source Delta table.*

[Databricks Documentation](#)

[Vector Search Modules](#)

How Vector Index Works: Core Concepts

1. Embedding

- Text strings are converted into an array of floats (vector).
- Models (e.g., GTE, BERT) process tokens to capture context.

2. Vector Similarity

- Distance computation between two embedding vectors.
- **Smaller distance = Similar semantic items** (used for effective querying).

Vector Similarity Visualized

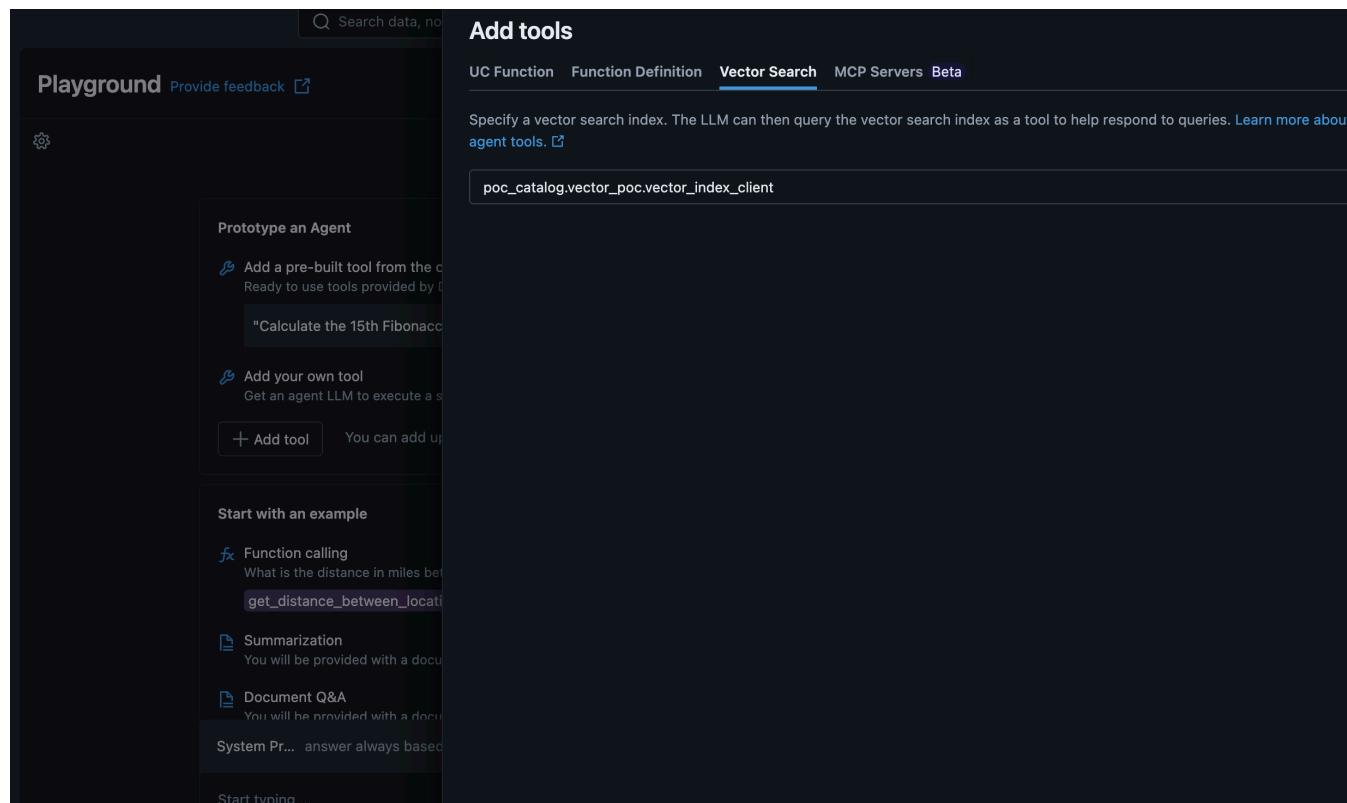
Semantic Clustering in 2D Space

- A vector database holds the text and its embedded vector.
- Semantically similar words are grouped together.

Databricks Agentbricks & RAG

Using the Index in AI Context

- Index can be used directly from Unity Catalog.
- Integrated as a tool for RAG in the Databricks Playground.



Model Serving and MLflow

Model Deployment

- Code generated in Playground (`driver.ipynb` , `agent.py`) can be used to **Register the Model**.
- Model is served via **Serving Endpoints** (UI or Python SDK).

The screenshot shows the Databricks workspace interface. On the left, a sidebar titled "Select an endpoint" lists several serving endpoints:

Name	Type
GPT-5.1 databricks-gpt-5-1	Pay-per-token
GPT OSS 120B databricks-gpt-oss-120b	Pay-per-token
GPT OSS 20B databricks-gpt-oss-20b	Pay-per-token
Qwen3 Next Instruct Beta databricks-qwen3-next-80b-a3b-instruct	Pay-per-token
Llama 4 Maverick databricks-llama-4-maverick	Pay-per-token
Gemma 3 12B databricks-gemma-3-12b	Pay-per-token
Meta Llama 3.3 70B Instruct databricks-meta-llama-3-3-70b-instruct	Pay-per-token
✓ rag_agent_v1 Custom Agent	Custom Agent

A modal window is open for the "rag_agent_v1" endpoint, displaying its details:

- Name:** rag_agent_v1
- Type:** Ready • Agent-enabled • Custom Agent
- Agent capabilities:** poc_catalog.vector_poc.vector_index_client
- Model:** rag_agent_v1 (Databricks)
- View endpoint**

MLflow Functionality

End-to-End ML Lifecycle Management

- **Experiment Tracking:** Log parameters, metrics, artifacts.
- **Model Registry:** Centralized store for versioning and management.
- **Model Deployment:** Deploy to batch, streaming, or real-time endpoints.
- Seamlessly integrated with Databricks workflows.

Local Integration: RAG System

Local LLM Setup

- Target Model: **Google Flan-T5-Base** (250M params).
- Use `huggingface_hub` to download and cache the model locally.

Access Requirements

- **Databricks Token** (for accessing Vector Search API).
- Token and model cache path stored in `.env` file.

Local Integration: Backend Query

Running the Backend Server

The command to spin up the server is:

```
uvicorn src.backend.main:app --reload --host 0.0.0.0 --port 8000
```

Testing (Untrained/No Context)

The cURL command is:

```
curl -X POST 'http://127.0.0.1:8000/query' \
-H 'Content-Type: application/json' \
-H 'X-Api-Key: default_key' \
-d '{
    "query": "What is a delta lake",
    "top_k": 1,
    "query_context": false
}'
```

Local Integration: RAG Success

Testing (With Databricks Vector Context)

- Enables retrieval of context from the vector index.
- Confirms successful connection and RAG pipeline function.

The RAG-enabled cURL command is:

```
curl -X POST 'http://127.0.0.1:8000/query' \
-H 'Content-Type: application/json' \
-H 'X-Api-Key: default_key' \
-d '{
    "query": "What is a delta lake",
    "top_k": 1,
    "query_context": true
}'
```