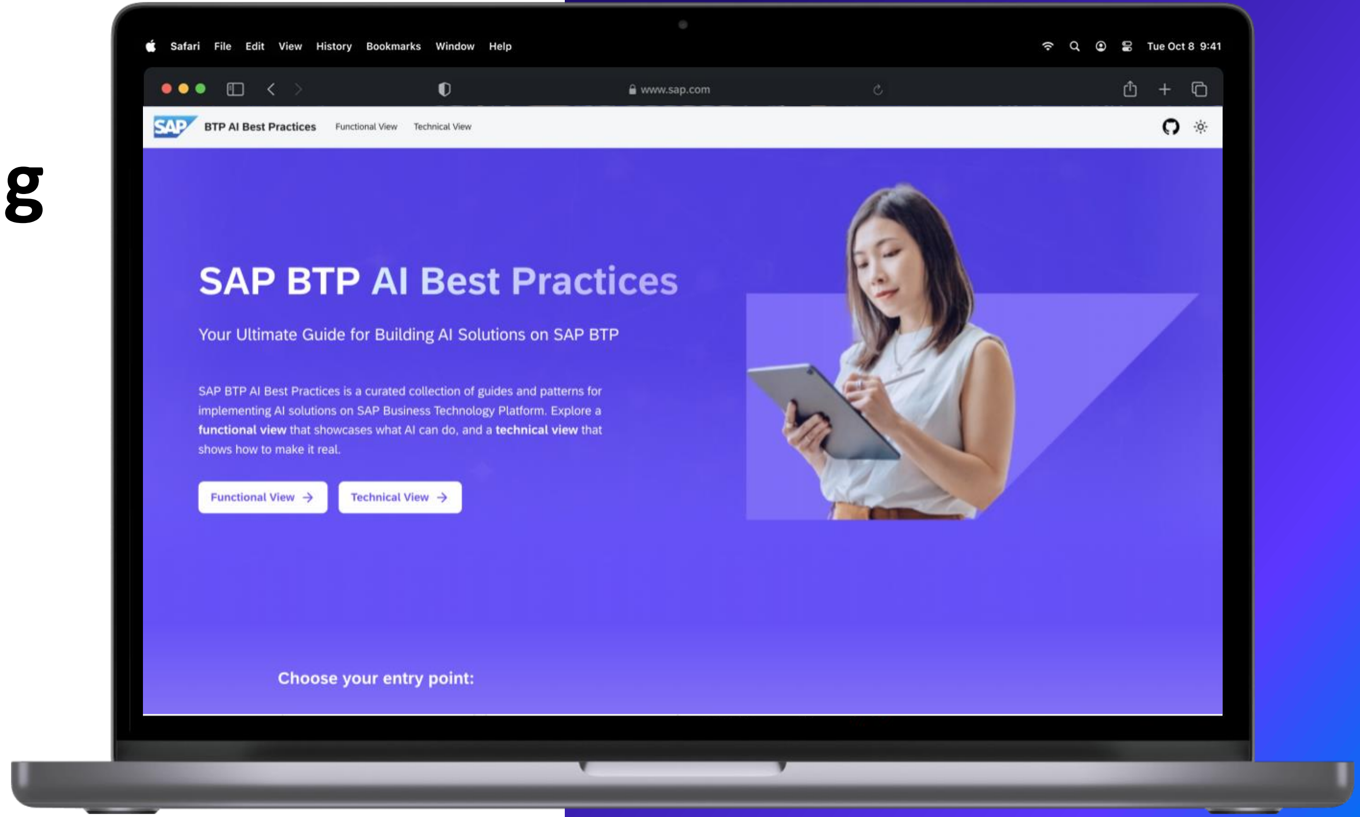
SAP BTP AI Best Practices

Vector-based RAG Embedding

A powerful approach to effectively improve LLM responses with augmented context.



BTP AI Services Center of Excellence

12.05.2025

Steps

1. Overview
2. Pre-requisites
3. Key Choices and Guidelines
4. Implementation

Retrieval Augmented Generation

Bring your organizational context to LLMs

Retrieve

Fetch relevant Info from knowledge base

Augment

Provide additional Info-chunks to enrich context to LLM

Generate

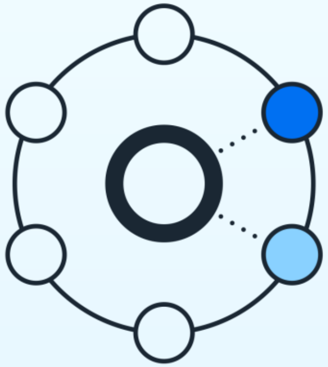
Content generation based on provided context

Expected Outcome

Helps LLMs to provide more accurate and contextually relevant answers
Allows scalable, low-latency semantic search based advanced GenAI applications

Key Benefits of vector-based RAG

Why use vector-based RAG?



Semantic Search & Understanding

Embeddings capture the meaning of text rather than just exact keywords

Improved Retrieval Quality

Embeddings enable high-quality retrieval from large knowledge bases

Scalability and Efficiency

Vector databases are optimized for fast, scalable similarity search using embeddings

Use cases

Vector Embedding approach is very useful in various scenarios

- **Semantic Search and Retrieval**

  Leverage vector embeddings to enhance semantic search capabilities, enabling users to find relevant information quickly.

- **Contextual Analysis:**

  Use vector embeddings for contextual analysis, allowing for a deeper understanding of relationships between different data points within the database.

- **Informed Decision Making:**

  Utilize vector embeddings to build intelligent data applications, unlocking insights and facilitating more informed decision-making.

- **Optimized Large Language Models (LLMs):**

  Enhance the output of LLMs by utilizing vector embeddings to optimize and add context to the generated content.

Key Concepts

- **Vector Embeddings:**

  Vector embeddings are vectors generated by embedding models to map objects like text or images into a high-dimensional space, preserving semantic similarity.

- **Similarity Measures:**

  These are mathematical functions that evaluate the similarity between vectors. Common measures include L2 distance and cosine similarity, which assess the distance and directional similarity between vectors, respectively.

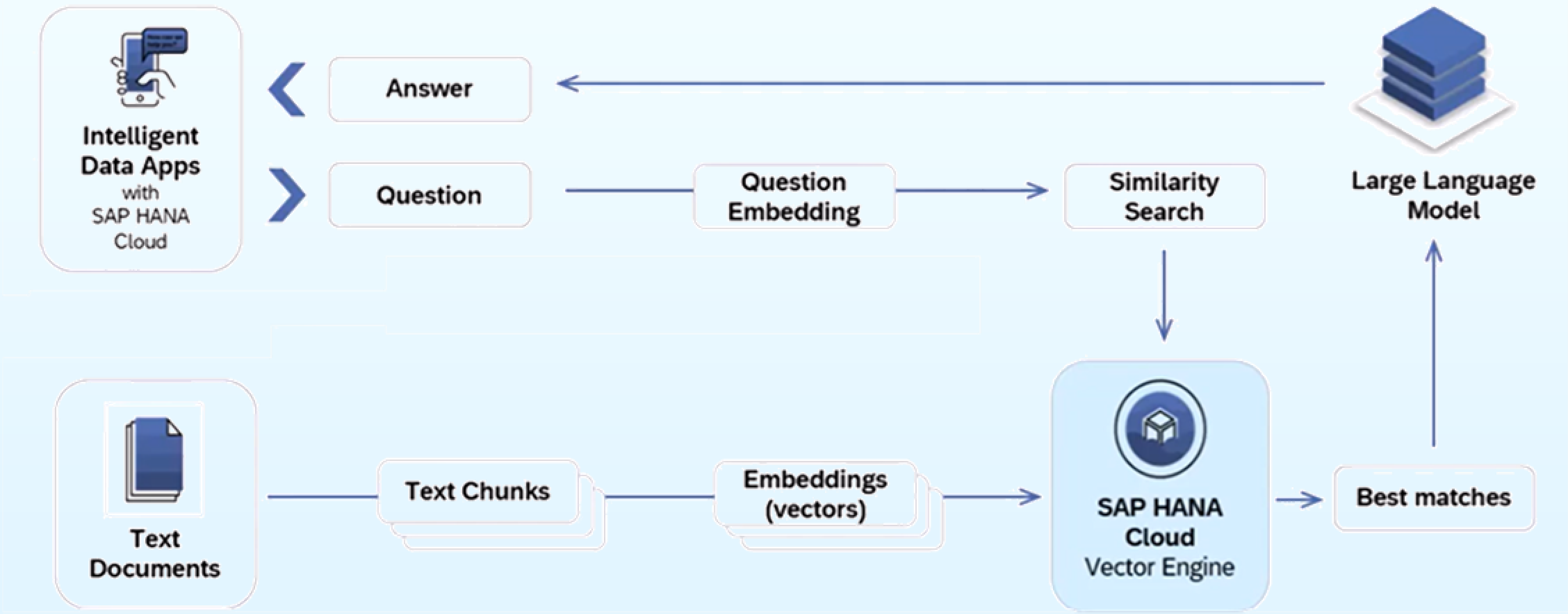
- **REAL_VECTOR Data Type:**

  In SAP HANA Cloud, vectors are stored using the REAL_VECTOR data type, which supports operations like creation, serialization, and similarity searches but has limitations such as no defined order and restrictions on arithmetic operations.

- **Chunking:**

  Chunking in Retrieval-Augmented Generation (RAG) refers to the process of dividing large text corpora into smaller, manageable pieces known as chunks. Please refer Chunking strategies for more details.

RAG Process Flow



Pre-requisites

Business

- SAP AI Core with the "Extended" tier on SAP BTP (Pricing Information)
- SAP HANA Cloud on SAP BTP (Pricing Information)
- SAP AI Launchpad (Pricing Information)

Technical

- SAP Business Technology Platform subaccount (Setup Guide)
- SAP AI Core (Setup Guide)
- SAP HANA Cloud - Vector Engine (Setup Guide)
- SAP AI Launchpad (Setup Guide)

SAP Business Technology Platform (SAP BTP)

- SAP Business Technology Platform (BTP) is an integrated suite of cloud services, databases, AI, and development tools that enable businesses to build, extend, and integrate SAP and non-SAP applications efficiently.

SAP HANA Cloud

- SAP HANA Cloud is a database as a service that powers mission-critical applications and real-time analytics with one solution at petabyte scale. Use relational, property graph, spatial, vector, and semi-structured data along with embedded machine learning to power intelligent data applications.

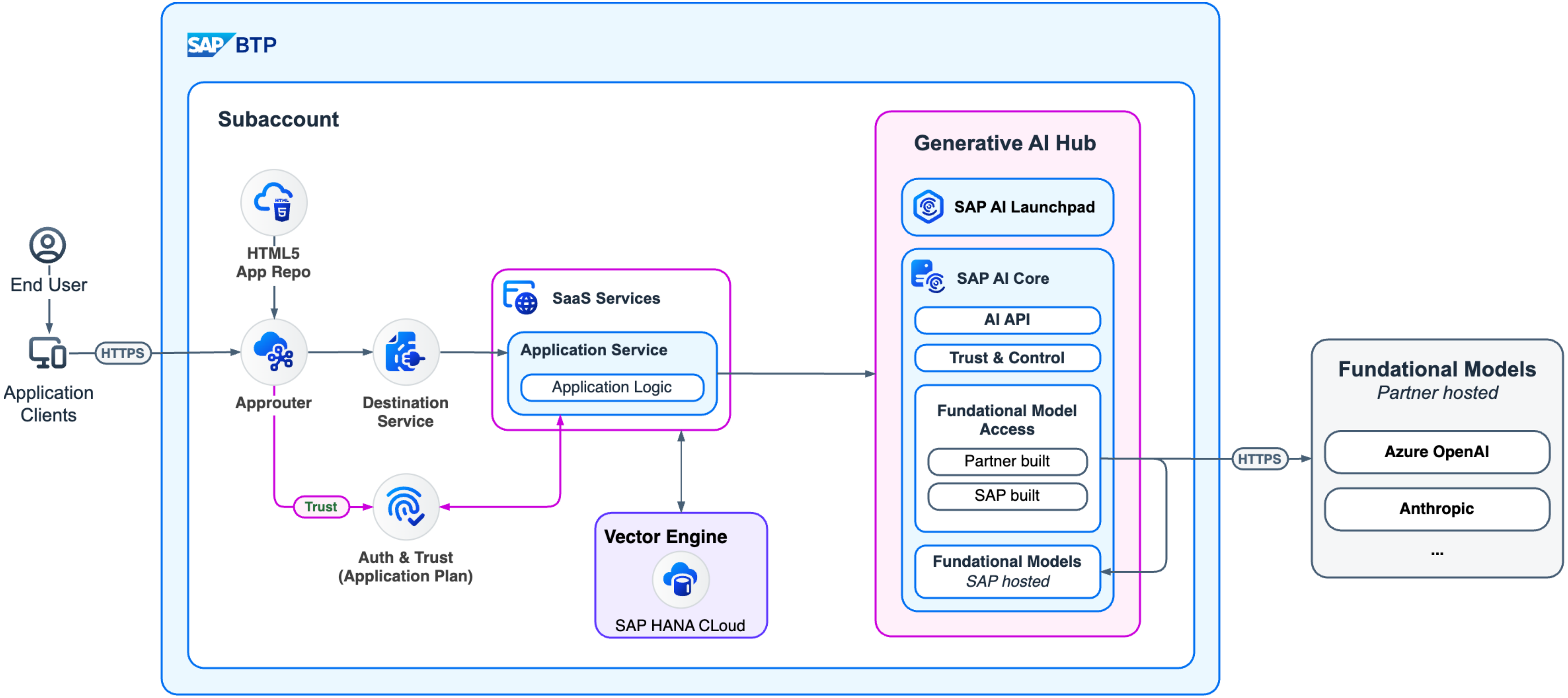
SAP AI Core

- SAP AI Core is a managed AI runtime that enables scalable execution of AI models and pipelines, integrating seamlessly with SAP applications and data on SAP BTP that supports full lifecycle management of AI scenarios.

SAP AI Launchpad

- SAP AI Launchpad is a multitenant SaaS application in SAP BTP. Customers can use SAP AI Launchpad to manage AI use cases (scenarios) across multiple instances of AI runtimes.

High-level reference architecture



Key Choices and Guidelines

Decisions that impact the quality and performance of your Vector Embeddings

Key Choices and Guidelines

1

Decisions that impact the quality and performance of your Vector Embeddings

Choose the right embedding model

- Embedding mode is a key choice and impact the overall effectiveness and performance of the RAG pipeline. You may follow below guidelines to choose model for your application.
  - General-purpose RAG: Use high-performing general models like OpenAI or Cohere.
  - Domain-specific RAG: Choose models fine-tuned on your domain (e.g., legal, finance, healthcare).
  - Consider model size vs. inference speed and cost.
  - Use multilingual models if your documents span multiple languages.

- Some available embedding models are as below-
  - OpenAI API Embedding models
  - Access to Open Source LLM Model API's (e.g. LLAMA2, PHI-2, MISTRAL, GROK, etc.)
  - SAP GenAI Hub (Enterprise grade access to a curated set of models e.g., text-embedding-3-small, text-embedding-3-large, text-embedding-ada-002 from OpenAI, and amazon–titan-embed-text from Azure).

Key Choices and Guidelines

Decisions that impact the quality and performance of your Vector Embeddings

Decide on Chunking Strategy

Chunking is how you break your documents into smaller text units. To perform chunking effectively you may follow below guidelines-

- Prefer semantic chunking (e.g., sentences, paragraphs) over arbitrary splits.
- Maintain context by optionally overlapping chunks (e.g., sliding window).
- Include metadata (e.g., title, tags, document ID) with each chunk to enable post-filtering.
- Some additional chunking methods can also be used. e.g., recursive character-based chunking (preserve the context better), context-enriched chunking (easier retrieval)

Key Choices and Guidelines

Decisions that impact the quality and performance of your Vector Embeddings

Normalize Input Text

Its always better to pre-process the input text before inputting it for embedding creations. You may perform below normalization:

- Clean the text: remove boilerplate, HTML, excessive whitespace, special characters. Also Lowercase (depending on model behaviour) and Standardize symbols, punctuation, and encoding.
- Keep a balance, don’t strip away useful context like section headers or lists.

Key Choices and Guidelines

Decisions that impact the quality and performance of your Vector Embeddings

Choosing Vector Indexing

Vector indexes accelerate query times by reducing the computational load required to find similar vectors.

- Ensure your vector columns are indexed.
- One of the best option is to use Hierarchical Navigable Small World (HNSW) index, this indexing is available in SAP HANA.

Key Choices and Guidelines

Decisions that impact the quality and performance of your Vector Embeddings

Quality Assurance & Evaluation

Evaluating your embedding representation is important as it ensure the quality of the retrieved documents.

- Sample chunks and test nearest-neighbor searches manually.
- Use similarity scoring tools to visualize cluster quality.
- Run pilot queries to see if embeddings return intuitive, relevant results.
- Log retrieval outcomes and get user feedback early.

Key Choices and Guidelines

Decisions that impact the quality and performance of your Vector Embeddings

Embedding Update Strategy

Knowledge bases get new updates which needs to be considered so that results are always up-to-date.

- Automate re-embedding when documents are added/changed.
- Use versioning to track changes in content or embedding models.
- Store old embeddings if you need backward compatibility or A/B testing.

Key Choices and Guidelines

Decisions that impact the quality and performance of your Vector Embeddings

Multi-language and Tokenization Considerations

If you are having data or user queries in more than one language, you should consider below recommendations.

- Ensure your embedding model supports the language(s) in your data.
- Avoid creating mixed-language chunks—keep chunks in a single language.

Implementation

Programming Model reference to implement vector embeddings

Python

SDK

- SAP Generative AI hub SDK (For building apps)
- SAP AI Core SDK and AI API Client SDK (AI Core lifecycle)
- HANA ML SDK

Reference Code

- End-to-end Embedding Creation
- RAG with HANA VectorDB Example
- GenAI- Vector DB Example

Learning Journeys

- Predictive AI with SAP AI Core
- RAG with HANA Vector Engine

JavaScript/TypeScript

SDK

- SAP Cloud SDK for AI

Reference Code

- SAP Cloud SDK for AI - Sample Code

Learning Journeys

- There are currently no learning journeys using the official SDK.

CAP App

SDK

- SAP Cloud SDK for AI (Recommended)
- CAP LLM Plugin

Reference Code

- SAP Cloud SDK for AI - Sample Code

Learning Journeys

Recommended

- There are currently no learning journeys using the official SDK.

Other

- GenAI Mail Insights: Develop a CAP application using GenAI and Retrieval Augmented Generation (RAG).

Code Sample

Python

```python
from langchain_community.document_loaders import TextLoader
from langchain_community.vectorstores.hanavector import HanaDB
from langchain_core.documents import Document
from langchain_openai import OpenAIEmbeddings
from langchain_text_splitters import CharacterTextSplitter

# Load the sample document "state_of_the_union.txt" and create chunks from it.
text_documents = TextLoader("/workspaces/integration-api-recommend-service/_temp/AICore.txt").load()
text_splitter = CharacterTextSplitter(chunk_size=500, chunk_overlap=0)
text_chunks = text_splitter.split_documents(text_documents)
print(f"Number of document chunks: {len(text_chunks)}")

# Initialize embeddings model
EMBEDDINGS_MODEL = "text-embedding-ada-002"

from gen_ai_hub.proxy.langchain.openai import OpenAIEmbeddings
embeddings = OpenAIEmbeddings(proxy_model_name=EMBEDDINGS_MODEL)

# Initialize HANA Vector Store instance
db = HanaDB(
    embedding=embeddings, connection=connection, table_name="AICORE_VECTOR_TABLE"
)
# Delete already existing documents from the table
db.delete(filter={})

# add the loaded document chunks
db.add_documents(text_chunks)
```

Contributors


Gupta, Chirag


Behera, Bhagabat Prasad


Marques, Luis


CIGAINA, MARCO

RF
Robledo, Francisco


Antonio, Dan

HI
Humphries, Ian

Thank you