Using Pinecone to Retrieve Top 3 Most Similar Product Records

1. Objective

- Learn to initialize and interact with the Pinecone vector database using the official Python client.
- Set up a Pinecone index and upsert sample vectors representing product data.
- Perform similarity search to retrieve the top three most similar records for a given input query.

2. Problem Statement

- In e-commerce and recommendation systems, efficiently retrieving the most similar products based on descriptions or features is crucial.
- This exercise guides you to use Pinecone's vector search to store product embeddings and find the nearest neighbors for a new query embedding.

3. Inputs / Shared Artifacts

You will work with the following sample product dataset embedded within your script:

```
products = [
    {"id": "prod1", "title": "Red T-Shirt", "description": "Comfortable
cotton t-shirt in bright red"},
    {"id": "prod2", "title": "Blue Jeans", "description": "Stylish
denim jeans with relaxed fit"},
    {"id": "prod3", "title": "Black Leather Jacket", "description":
"Genuine leather jacket with classic style"},
```

```
    {"id": "prod4", "title": "White Sneakers", "description":
"Comfortable sneakers perfect for daily wear"},
    {"id": "prod5", "title": "Green Hoodie", "description": "Warm
hoodie made of organic cotton"},
]
```

4. Expected Outcome

- Pinecone index is created and populated with the sample product embeddings.
- Given a new input embedding (e.g., representing a search query), the program returns the top three most similar products by cosine similarity.
- Proper handling of Pinecone client initialization, index creation, upserting vectors, querying, and closing the connection.

5. Concepts Covered

- Initializing Pinecone client with API key.
- Creating and configuring a Pinecone index.
- Upserting vectors into the index.
- Performing similarity search queries.
- Retrieving and interpreting top-k results.
- Managing resources and error handling with Pinecone.

6. Example: Step-by-Step Instructions with Code

```
# pip install pinecone openai
from pinecone import Pinecone, ServerlessSpec
from openai import AzureOpenAI
import os

# Step 1: Initialize Pinecone client and embedding model
os.environ["AZURE_OPENAI_ENDPOINT"] = ""
os.environ["AZURE_OPENAI_API_KEY"] = ""
os.environ["AZURE_DEPLOYMENT_NAME"] = "text-embedding-3-small"

# get pinecone api key from https://www.pinecone.io/
os.environ["PINECONE_API_KEY"] = ""
```

```
client = AzureOpenAI(
    api_version="2024-07-01-preview",
    azure_endpoint=os.getenv("AZURE_OPENAI_ENDPOINT"),
    api_key=os.getenv("AZURE_OPENAI_API_KEY"),
)

pc = Pinecone(
    api_key=os.getenv("PINECONE_API_KEY")
)  # Replace with your actual Pinecone API key

# Step 2: Create or connect to an index

index_name = "product-similarity-index"

if index_name not in [index["name"] for index in pc.list_indexes()]:
    pc.create_index(
        name=index_name,
        dimension=1536,
        spec=ServerlessSpec(cloud="aws", region="us-east-1"),
    )  # dimension matches text-embedding-3-small output size

index = pc.Index(index_name)

# Step 3: Upsert sample product vectors into the index

products = [
    {"id": "prod1", "title": "Red T-Shirt"},
    {"id": "prod2", "title": "Blue Jeans"},
    {"id": "prod3", "title": "Black Leather Jacket"},
    {"id": "prod4", "title": "White Sneakers"},
    {"id": "prod5", "title": "Green Hoodie"},
]

def get_embedding(text):
    response = client.embeddings.create(
        input=text, model=os.getenv("AZURE_DEPLOYMENT_NAME")
    )
    return response.data[0].embedding

vectors = []
for p in products:
    embedding = get_embedding(p["title"])
    vectors.append((p["id"], embedding))

index.upsert(vectors)

# Step 4: Prepare input query embedding

query = "clothing item for summer"
query_embedding = get_embedding(query)

# Step 5: Query Pinecone index for top 3 most similar vectors
```

```
top_k = 3
results = index.query(vector=query_embedding, top_k=top_k,
include_metadata=False)
print("All result",results)

# Step 6: Display results

print(f"Top {top_k} similar products for the query: '{query}'\n")

for match in results.matches:
    product_id = match.id
    score = match.score
    # Find product details
    product = next(p for p in products if p["id"] == product_id)
    print(f"- {product['title']} (Similarity score: {score:.4f})")
```

7. Final Submission Checklist

- Submit your Python script that:
  - Initializes Pinecone client and index.
  - Upserts sample data vectors.
  - Queries the index with a sample input embedding.
  - Prints the top 3 most similar product titles and similarity scores.
- Include your sample dataset embedded in code.
- Use a dummy input list (auto input). Do not use manual input (input() built-in).
- Provide console output demonstrating similarity search results in logs file or README.
- (Optional) Write a short explanation of your approach and any challenges faced.