Practice: AI Vector Search

Practice Target

In this practice, you will implement a simple example of using the AI Vector Search feature in Oracle database 23ai.

Note: Vector Search in Oracle database 23ai is a large topic and this practice demonstrates on the basics of this feature. The documentation (Oracle AI Vector Search User's Guide) of this feature can be accessed from this link.

Practice Overview

In high level, you will perform the basic steps of using Oracle AI Vector Search feature in Oracle database 23ai.

Using AI Vector Search

1. Open SQL Developer and connect as **SYS** to the pdb.

2. Create a new user for our example

```
CREATE USER vector_user IDENTIFIED BY oracle
DEFAULT TABLESPACE users TEMPORARY TABLESPACE TEMP QUOTA UNLIMITED ON users;
```

3. Grant necessary privileges to the user

```
GRANT CREATE SESSION, CREATE TABLE, CREATE PROCEDURE, CREATE SEQUENCE, CREATE MINING
MODEL TO ector_user;

GRANT READ, WRITE ON DIRECTORY DATA_PUMP_DIR TO vector_user;
```

4. In SQL Developer, connect as the new user vector_user

We need an embedding model to convert text into numerical vectors. Oracle Database 23ai allows you to import ONNX models directly.

5. Download `all_MiniLM_L12_v2_augmented.zip` from the lecture downloadable resources to the sharing folder.

6. Connect as oracle in Putty to the vm, create a folder in the vm, and unzip the model file into it.

```
mkdir -p /home/oracle/models cd
/home/oracle/models
 unzip /media/sf_staging/all_MiniLM_L12_v2_augmented.zip -d
.
```

7. Move the file `all_MiniLM_L12_v2.onnx` into the Data Pump directory

```
mv ./all_MiniLM_L12_v2.onnx
/opt/oracle/admin/FREE/dpdump/2E711CD7D1133106E0630100007F5070
```

8. Load the ONNX model into the database:

```
BEGIN
  DBMS_VECTOR.LOAD_ONNX_MODEL(
    model_name => 'MINILM_L12_V2', -- A name for your model within the DB
    directory => 'DATA_PUMP_DIR',  -- directory where the model file is saved
file_name => 'all_MiniLM_L12_v2.onnx',
    metadata => JSON('{"function" : "embedding", "embeddingOutput" : "embedding",
"input": {"input": ["DATA"]}}') -- Corrected parameter name to 'metadata'
  );
END;
/
```

9. Verify the model is loaded

```
SELECT model_name, algorithm, mining_function FROM user_mining_models WHERE
model_name = 'MINILM_L12_V2';
```

10. Create a table to store some product descriptions and their vector embeddings.

```
CREATE TABLE products (
    product_id   NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY,
product_name VARCHAR2(255),     description  VARCHAR2(1000),
    description_vector VECTOR(384, FLOAT32), -- 384 dimensions for all-MiniLM-L12-v2,
FLOAT32 is common
    CONSTRAINT pk_products PRIMARY KEY (product_id) );
```

Note: The dimension count (e.g., 384 for MiniLM-L12-v2) and data type (FLOAT32) should match the output of your chosen embedding model.

11. Insert some sample product data and use the `VECTOR_EMBEDDING` function to generate their embeddings directly within the `INSERT` statement.

```
INSERT INTO products (product_name, description, description_vector) VALUES (
'Smartwatch X1',
    'A cutting-edge smartwatch with advanced health monitoring, GPS, and long battery
life. Perfect for fitness enthusiasts.',
    VECTOR_EMBEDDING(MINILM_L12_V2 USING 'A cutting-edge smartwatch with advanced
health monitoring, GPS, and long battery life. Perfect for fitness enthusiasts.' AS
DATA)
);
INSERT INTO products (product_name, description, description_vector) VALUES (
    'Noise-Cancelling Headphones',
    'Immersive audio experience with industry-leading noise cancellation. Ideal for
travel and focused work.',
    VECTOR_EMBEDDING(MINILM_L12_V2   USING   'Immersive   audio   experience   with
industryleading noise cancellation. Ideal for travel and focused work.' AS DATA) );
```

```
INSERT INTO products (product_name, description, description_vector) VALUES (
    'Ergonomic Office Chair',
    'Designed for maximum comfort and support during long working hours. Features
adjustable lumbar support and breathable mesh.',
    VECTOR_EMBEDDING(MINILM_L12_V2 USING 'Designed for maximum comfort and support
during long working hours. Features adjustable lumbar support and breathable mesh.'
AS DATA)
);
INSERT INTO products (product_name, description, description_vector) VALUES (
    'Portable Bluetooth Speaker',
    'Compact and powerful speaker with rich bass and clear highs. Waterproof design
for outdoor adventures.',
    VECTOR_EMBEDDING(MINILM_L12_V2 USING 'Compact and powerful speaker with rich bass
and clear highs. Waterproof design for outdoor adventures.' AS DATA) );
INSERT INTO products (product_name, description, description_vector) VALUES (
    'High-Performance Gaming PC',
    'Unleash your gaming potential with this powerful PC. Equipped with the latest
graphics card and liquid cooling.',
    VECTOR_EMBEDDING(MINILM_L12_V2 USING 'Unleash your gaming potential with this
powerful PC. Equipped with the latest graphics card and liquid cooling.' AS DATA)
);
INSERT INTO products (product_name, description, description_vector) VALUES (
'Fitness Tracker Band',
    'Track your daily activity, heart rate, and sleep patterns. A simple and
effective tool for a healthier lifestyle.',
    VECTOR_EMBEDDING(MINILM_L12_V2 USING 'Track your daily activity, heart rate, and
sleep patterns. A simple and effective tool for a healthier lifestyle.' AS DATA) );

COMMIT;
```

12. Verify data insertion (optional: you can see the vector data, but it's a long array of numbers):

```
SELECT product_id, product_name, description FROM products;
-- This will show the raw vector which is very long.
SELECT product_id, product_name, description, description_vector FROM products;
```

Now, let's perform some semantic similarity searches using the `VECTOR_DISTANCE` function. We'll compare a query string's embedding with the stored product embeddings. Cosine similarity is a common metric for text embeddings, where a smaller distance indicates higher similarity.

13. Define a variable for the query text

```
VARIABLE query_text VARCHAR2(255);
```

14. Example 1: Find products similar to "health and wellness gadgets"

```
EXEC :query_text := 'health and wellness gadgets'; SELECT
    product_name,
description,
    VECTOR_DISTANCE(description_vector, VECTOR_EMBEDDING(MINILM_L12_V2 USING
:query_text AS DATA), COSINE) AS similarity_score FROM
    products ORDER
BY
    similarity_score ASC -- For COSINE distance, smaller is more similar FETCH
FIRST 3 ROWS ONLY;
```

15. Example 2: Find products similar to "audio devices".

```
EXEC :query_text := 'audio devices'; SELECT
    product_name,
description,
    VECTOR_DISTANCE(description_vector, VECTOR_EMBEDDING(MINILM_L12_V2 USING
:query_text AS DATA), COSINE) AS similarity_score FROM
    products ORDER
BY
    similarity_score ASC FETCH
FIRST 3 ROWS ONLY;
```

As you can see, the similarity scores correctly identify products that are semantically related to the query, even if they don't contain the exact keywords.

16. As a cleanup, perform the following:

```
DROP TABLE products purge ;

-- disconnect vector_user from SQL Developer
DROP USER vector_user CASCADE ;

-- in Putty:
rm
/opt/oracle/admin/FREE/dpdump/2E711CD7D1133106E0630100007F5070/all_MiniLM_L12_v2.onnx
rm -rf /home/oracle/models
```

Summary

This practice provided a walkthrough of using Vector Search in Oracle Database 23ai. It includes the following steps:

- Loading an embedding model ○ Creating a table with a `VECTOR` column ○ Generating and inserting vector embeddings ○ Performing similarity searches
- Creating a Vector Index (Optional but recommended)