Set up the Jupyter notebook environment

Install the Google Cloud Vertex AI, Cloud Storage and BigQuery SDKs.

[1]:
```
! pip3 install --upgrade google-cloud-aiplatform \
                         google-cloud-storage \
                         'google-cloud-bigquery[pandas]'
```

```
Requirement already satisfied: google-cloud-aiplatform in ./.local/lib/python3.10/site-packages (1.94.0)
Requirement already satisfied: google-cloud-storage in /opt/conda/lib/python3.10/site-packages (2.19.0)
Collecting google-cloud-storage
  Downloading google_cloud_storage-3.1.0-py2.py3-none-any.whl.metadata (12 kB)
Requirement already satisfied: google-cloud-bigquery[pandas] in ./.local/lib/python3.10/site-packages (3.33.0)
Requirement already satisfied: google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.*,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,<3.0.0,>=1.34.1 in /opt/conda/lib/python3.10/
site-packages (from google-api-core[grpc]!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.*,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,<3.0.0,>=1.34.1->google-cloud-aiplatform) (2.24.2)
Requirement already satisfied: google-auth<3.0.0,>=2.14.1 in /opt/conda/lib/python3.10/site-packages (from google-cloud-aiplatform) (2.38.0)
Requirement already satisfied: proto-plus<2.0.0,>=1.22.3 in /opt/conda/lib/python3.10/site-packages (from google-cloud-aiplatform) (1.26.1)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<7.0.0,>=3.20.2 in /opt/conda/lib/python3.10/site-packages (fr
om google-cloud-aiplatform) (3.20.3)
Requirement already satisfied: packaging>=14.3 in /opt/conda/lib/python3.10/site-packages (from google-cloud-aiplatform) (24.2)
Requirement already satisfied: google-cloud-resource-manager<3.0.0,>=1.3.3 in /opt/conda/lib/python3.10/site-packages (from google-cloud-aiplatform) (1.14.
2)
Requirement already satisfied: shapely<3.0.0 in /opt/conda/lib/python3.10/site-packages (from google-cloud-aiplatform) (2.0.7)
Requirement already satisfied: google-genai<2.0.0,>=1.0.0 in ./.local/lib/python3.10/site-packages (from google-cloud-aiplatform) (1.16.1)
Requirement already satisfied: pydantic<3 in /opt/conda/lib/python3.10/site-packages (from google-cloud-aiplatform) (2.11.0)
Requirement already satisfied: typing-extensions in /opt/conda/lib/python3.10/site-packages (from google-cloud-aiplatform) (4.13.0)
Requirement already satisfied: docstring-parser<1 in /opt/conda/lib/python3.10/site-packages (from google-cloud-aiplatform) (0.16)
Requirement already satisfied: google-cloud-core<3.0dev,>=2.3.0 in /opt/conda/lib/python3.10/site-packages (from google-cloud-storage) (2.4.3)
Requirement already satisfied: google-resumable-media>=2.7.2 in /opt/conda/lib/python3.10/site-packages (from google-cloud-storage) (2.7.2)
Requirement already satisfied: requests<3.0.0dev,>=2.18.0 in /opt/conda/lib/python3.10/site-packages (from google-cloud-storage) (2.32.3)
Requirement already satisfied: google-crc32c<2.0dev,>=1.0 in /opt/conda/lib/python3.10/site-packages (from google-cloud-storage) (1.7.1)
Requirement already satisfied: python-dateutil<3.0.0,>=2.8.2 in /opt/conda/lib/python3.10/site-packages (from google-cloud-bigquery[pandas]) (2.9.0.post0)
Requirement already satisfied: pandas>=1.3.0 in /opt/conda/lib/python3.10/site-packages (from google-cloud-bigquery[pandas]) (2.2.3)
Requirement already satisfied: pandas-gbq>=0.26.1 in /opt/conda/lib/python3.10/site-packages (from google-cloud-bigquery[pandas]) (0.28.0)
Requirement already satisfied: grpcio<2.0.0,>=1.47.0 in /opt/conda/lib/python3.10/site-packages (from google-cloud-bigquery[pandas]) (1.71.0)
Requirement already satisfied: pyarrow>=3.0.0 in /opt/conda/lib/python3.10/site-packages (from google-cloud-bigquery[pandas]) (19.0.1)
Requirement already satisfied: db-dtypes<2.0.0,>=1.0.4 in /opt/conda/lib/python3.10/site-packages (from google-cloud-bigquery[pandas]) (1.4.2)
Requirement already satisfied: numpy>=1.16.6 in /opt/conda/lib/python3.10/site-packages (from db-dtypes<2.0.0,>=1.0.4->google-cloud-bigquery[pandas]) (2.1.
3)
Requirement already satisfied: googleapis-common-protos<2.0.0,>=1.56.2 in /opt/conda/lib/python3.10/site-packages (from google-api-core!=2.0.*,!=2.1.*,!=2.
2.*,!=2.3.*,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,<3.0.0,>=1.34.1->google-api-core[grpc]!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.*,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,<3.0.0,>=
1.34.1->google-cloud-aiplatform) (1.69.2)
Requirement already satisfied: grpcio-status<2.0.dev0,>=1.33.2 in /opt/conda/lib/python3.10/site-packages (from google-api-core[grpc]!=2.0.*,!=2.1.*,!=2.2.
*,!=2.3.*,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,<3.0.0,>=1.34.1->google-cloud-aiplatform) (1.49.0rc1)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /opt/conda/lib/python3.10/site-packages (from google-auth<3.0.0,>=2.14.1->google-cloud-aiplatform)
(5.5.2)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /opt/conda/lib/python3.10/site-packages (from google-auth<3.0.0,>=2.14.1->google-cloud-aiplatform)
(0.4.2)
Requirement already satisfied: rsa<5,>=3.1.4 in /opt/conda/lib/python3.10/site-packages (from google-auth<3.0.0,>=2.14.1->google-cloud-aiplatform) (4.9)
Requirement already satisfied: grpc-google-iam-v1<1.0.0,>=0.14.0 in /opt/conda/lib/python3.10/site-packages (from google-cloud-resource-manager<3.0.0,>=1.
3.3->google-cloud-aiplatform) (0.14.2)
Requirement already satisfied: anyio<5.0.0,>=4.8.0 in /opt/conda/lib/python3.10/site-packages (from google-genai<2.0.0,>=1.0.0->google-cloud-aiplatform)
(4.9.0)
Requirement already satisfied: httpx<1.0.0,>=0.28.1 in ./.local/lib/python3.10/site-packages (from google-genai<2.0.0,>=1.0.0->google-cloud-aiplatform) (0.
28.1)
Requirement already satisfied: websockets<15.1.0,>=13.0.0 in /opt/conda/lib/python3.10/site-packages (from google-genai<2.0.0,>=1.0.0->google-cloud-aiplatf
orm) (15.0.1)
Requirement already satisfied: pytz>=2020.1 in /opt/conda/lib/python3.10/site-packages (from pandas>=1.3.0->google-cloud-bigquery[pandas]) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /opt/conda/lib/python3.10/site-packages (from pandas>=1.3.0->google-cloud-bigquery[pandas]) (2025.2)
Requirement already satisfied: setuptools in /opt/conda/lib/python3.10/site-packages (from pandas-gbq>=0.26.1->google-cloud-bigquery[pandas]) (75.8.2)
Requirement already satisfied: pydata-google-auth>=1.5.0 in /opt/conda/lib/python3.10/site-packages (from pandas-gbq>=0.26.1->google-cloud-bigquery[panda
s]) (1.9.1)
Requirement already satisfied: google-auth-oauthlib>=0.7.0 in /opt/conda/lib/python3.10/site-packages (from pandas-gbq>=0.26.1->google-cloud-bigquery[panda
s]) (1.2.1)
Requirement already satisfied: annotated-types>=0.6.0 in /opt/conda/lib/python3.10/site-packages (from pydantic<3->google-cloud-aiplatform) (0.7.0)
Requirement already satisfied: pydantic-core==2.33.0 in /opt/conda/lib/python3.10/site-packages (from pydantic<3->google-cloud-aiplatform) (2.33.0)
Requirement already satisfied: typing-inspection>=0.4.0 in /opt/conda/lib/python3.10/site-packages (from pydantic<3->google-cloud-aiplatform) (0.4.0)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.10/site-packages (from python-dateutil<3.0.0,>=2.8.2->google-cloud-bigquery[pandas]) (1.1
7.0)
Requirement already satisfied: charset_normalizer<4,>=2 in /opt/conda/lib/python3.10/site-packages (from requests<3.0.0dev,>=2.18.0->google-cloud-storage)
(3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.10/site-packages (from requests<3.0.0dev,>=2.18.0->google-cloud-storage) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /opt/conda/lib/python3.10/site-packages (from requests<3.0.0dev,>=2.18.0->google-cloud-storage) (1.26.
20)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/python3.10/site-packages (from requests<3.0.0dev,>=2.18.0->google-cloud-storage) (2025.
1.31)
Requirement already satisfied: exceptiongroup>=1.0.2 in /opt/conda/lib/python3.10/site-packages (from anyio<5.0.0,>=4.8.0->google-genai<2.0.0,>=1.0.0->goog
le-cloud-aiplatform) (1.2.2)
Requirement already satisfied: sniffio>=1.1 in /opt/conda/lib/python3.10/site-packages (from anyio<5.0.0,>=4.8.0->google-genai<2.0.0,>=1.0.0->google-cloud-
aiplatform) (1.3.1)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /opt/conda/lib/python3.10/site-packages (from google-auth-oauthlib>=0.7.0->pandas-gbq>=0.26.1->g
oogle-cloud-bigquery[pandas]) (2.0.0)
Requirement already satisfied: httpcore==1.* in ./.local/lib/python3.10/site-packages (from httpx<1.0.0,>=0.28.1->google-genai<2.0.0,>=1.0.0->google-cloud-
aiplatform) (1.0.9)
Requirement already satisfied: h11>=0.16 in ./.local/lib/python3.10/site-packages (from httpcore==1.*->httpx<1.0.0,>=0.28.1->google-genai<2.0.0,>=1.0.0->go
ogle-cloud-aiplatform) (0.16.0)
Requirement already satisfied: pyasn1<0.7.0,>=0.6.1 in /opt/conda/lib/python3.10/site-packages (from pyasn1-modules>=0.2.1->google-auth<3.0.0,>=2.14.1->goo
gle-cloud-aiplatform) (0.6.1)
Requirement already satisfied: oauthlib>=3.0.0 in /opt/conda/lib/python3.10/site-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib>=0.7.0->pand
as-gbq>=0.26.1->google-cloud-bigquery[pandas]) (3.2.2)
```

Restart kernel

[2]:
```
import IPython

app = IPython.Application.instance()
app.kernel.do_shutdown(True)
```

[2]:
```
{'status': 'ok', 'restart': True}
```

Setup the environment values for your project

Setting up variables

[6]:
```
PROJECT = !gcloud config get-value project
```

```
PROJECT_ID = PROJECT[0]
REGION = "us-west1"
```

Import and initialize the Vertex AI Python SDK.

[7]:
```
import vertexai
vertexai.init(project = PROJECT_ID,
              location = REGION)
```

Prepare the data in BigQuery

The dataset used for this lab is the StackOverflow dataset. This public dataset is hosted in Google BigQuery and is included in BigQuery's 1TB/mo of free tier processing. This means that each user receives 1TB of free BigQuery processing every month, which can be used to run queries on this public dataset.

Stack Overflow is the largest online community for programmers to learn, share their knowledge, and advance their careers. Updated on a quarterly basis, this BigQuery dataset includes an archive of Stack Overflow content, including posts, votes, tags, and badges. This dataset is updated to mirror the Stack Overflow content on the Internet Archive, and is also available through the Stack Exchange Data Explorer.

The BigQuery table is too large to fit into memory, so you need to write a generator called `query_bigquery_chunks` to yield chunks of the dataframe for processing. Additionally, an extra column `title_with_body` is added, which is a concatenation of the question title and body that will be used for creating embeddings.

Import the libraries and initialize the BigQuery client.

[8]:
```
import math
from typing import Any, Generator

import pandas as pd
from google.cloud import bigquery

client = bigquery.Client(project=PROJECT_ID)
```

Define the BigQuery query for the remote dataset.

[9]:
```
QUERY_TEMPLATE = """
        SELECT distinct q.id, q.title, q.body
        FROM (SELECT * FROM `bigquery-public-data.stackoverflow.posts_questions` where Score>0 ORDER BY View_Count desc) AS q
        LIMIT {limit} OFFSET {offset};
        """
```

Create a function to access the BigQuery data in chunks

[10]:
```
def query_bigquery_chunks(
    max_rows: int, rows_per_chunk: int, start_chunk: int = 0
) -> Generator[pd.DataFrame, Any, None]:
    for offset in range(start_chunk, max_rows, rows_per_chunk):
        query = QUERY_TEMPLATE.format(limit=rows_per_chunk, offset=offset)
        query_job = client.query(query)
        rows = query_job.result()
        df = rows.to_dataframe()
        df["title_with_body"] = df.title + "\n" + df.body
        yield df
```

```
Get a dataframe of 1000 rows for demonstration purposes
```

[11]:
```
df = next(query_bigquery_chunks(max_rows=1000, rows_per_chunk=1000))

# Examine the data
df.head()
```

[11]:

	id	title	body	title_with_body
0	9257858	Get mutual subscriptions between two users	I have a table as follows \n\n<pre><code...	Get mutual subscriptions between two users\n<p...
1	9243623	UITableView not selecting properly	I am working on an iPhone app, where I have...	UITableView not selecting properly\n I am wo...
2	9459273	.Ajax method works on local machine, but not o...	I have an .Ajax method that calls a method ...	.Ajax method works on local machine, but not o...
3	4569123	Content is not allowed in Prolog SAXParserExce...	I am trying to call a web service but facin...	Content is not allowed in Prolog SAXParserExce...
4	4308934	How to delete last character from a string usi...	How to delete last character from a string ...	How to delete last character from a string usi...

Create text embeddings from BigQuery data

```
Load the `Vertex AI Embeddings` for Text model.
```

[12]:
```
from typing import List, Optional
from vertexai.preview.language_models import TextEmbeddingModel

model = TextEmbeddingModel.from_pretrained("text-embedding-004")
```

Define an embedding method that uses the model.

[13]:
```
def encode_texts_to_embeddings(sentences: List[str]) -> List[Optional[List[float]]]:
    try:
        embeddings = model.get_embeddings(sentences)
        return [embedding.values for embedding in embeddings]
    except Exception:
        return [None for _ in range(len(sentences))]
```

According to the documentation, each request can handle up to 5 text instances. So we will need to split the BigQuery question results in batches of 5 before sending to the embedding API.

Create a `generate_batches` to split results in batches of 5 to be sent to the embeddings API.

[14]:
```
import functools
import time
from concurrent.futures import ThreadPoolExecutor
from typing import Generator, List, Tuple

import numpy as np
from tqdm.auto import tqdm


# Generator function to yield batches of sentences
def generate_batches(
    sentences: List[str], batch_size: int
) -> Generator[List[str], None, None]:
```

```
    for i in range(0, len(sentences), batch_size):
        yield sentences[i : i + batch_size]
```

Encapsulate the process of generating batches and calling the embeddings API in a method called `encode_text_to_embedding_batched`. This method also handles `rate-limiting` using `time.sleep`. For production use cases, you would want a more sophisticated rate-limiting mechanism that takes retries into account.

[15]:
```
def encode_text_to_embedding_batched(
    sentences: List[str], api_calls_per_second: int = 10, batch_size: int = 5
) -> Tuple[List[bool], np.ndarray]:

    embeddings_list: List[List[float]] = []

    # Prepare the batches using a generator
    batches = generate_batches(sentences, batch_size)

    seconds_per_job = 1 / api_calls_per_second

    with ThreadPoolExecutor() as executor:
        futures = []
        for batch in tqdm(
            batches, total=math.ceil(len(sentences) / batch_size), position=0
        ):
            futures.append(
                executor.submit(functools.partial(encode_texts_to_embeddings), batch)
            )
            time.sleep(seconds_per_job)

        for future in futures:
            embeddings_list.extend(future.result())

    is_successful = [
        embedding is not None for sentence, embedding in zip(sentences, embeddings_list)
    ]
    embeddings_list_successful = np.squeeze(
        np.stack([embedding for embedding in embeddings_list if embedding is not None])
    )
    return is_successful, embeddings_list_successful
```

Test the encoding function by encoding a subset of data and see if the embeddings and distance metrics make sense

[16]:
```
# Encode a subset of questions for validation
questions = df.title.tolist()[:500]
is_successful, question_embeddings = encode_text_to_embedding_batched(
    sentences=df.title.tolist()[:500]
)

# Filter for successfully embedded sentences
questions = np.array(questions)[is_successful]
```

```
100%|██████████| 100/100 [00:10<00:00,  9.85it/s]
```

Save the dimension size for later usage when creating the Vertex AI Vector Search index.

[17]:
```
DIMENSIONS = len(question_embeddings[0])

print(DIMENSIONS)
```

```
768
```

Sort questions in order of similarity. According to the embedding documentation, the similarity of embeddings is calculated using the dot-product, with np.dot. Once you have the similarity score, sort the results and print them for inspection. 1 means very similar, 0 means very different.

[18]:
```
import random

question_index = random.randint(0, 99)

print(f"Query question = {questions[question_index]}")

# Get similarity scores for each embedding by using dot-product.
scores = np.dot(question_embeddings[question_index], question_embeddings.T)

# Print top 20 matches
for index, (question, score) in enumerate(
    sorted(zip(questions, scores), key=lambda x: x[1], reverse=True)[:20]
):
    print(f"\t{index}: {question}: {score}")
```

```
Query question = DI: Register-Resolve-Release when using child containers
        0: DI: Register-Resolve-Release when using child containers: 0.9999978110458627
        1: C# not releasing memory after task complete: 0.5136752460923943
        2: Embed xml into dll: 0.4803991893181705
        3: Load whole *ui file in an frame/widget of another *.ui file: 0.4545462072696377
        4: Using a 'using alias = class' with generic types?: 0.44325575692733843
        5: Returning values from MyBatis <insert> mapped methods: 0.4405694235804478
        6: How inefficient is passing Collections.unmodifiable* an instance which is already wrapped with Collections.unmodifiable*?: 0.4169665087870481
        7: Is a jbyteArray a jobject (i.e.: as a parameter to DeleteLocalRef)?: 0.3929884675899803
        8: Regex.Replace and String immutability: 0.3880198360882109
        9: PHP internationalization with intl: 0.3860386004774648
        10: Render List returned via JQuery: 0.3849918594436284
        11: Where to Store Encryption Keys MVC Application: 0.38287309031907313
        12: Autodiscover widgets for Django apps registered in settins.py: 0.381602263327151
        13: How to call an ASP.NET WebMethod in a UserControl (.ascx): 0.38152237600273614
        14: Using one css/js file: 0.3794957218423135
        15: Send file with skype4com in C#?: 0.37857736328437996
        16: Get single row result with Doctrine NativeQuery: 0.3757350959589837
        17: Create new element with doctrine entity manager?: 0.3747012171479602
        18: What is the best way to resolve a tree conflict with TortoiseSVN if I want to keep my local changes?: 0.3740287643546872
        19: pass request parameter value to class function and return value in classic asp: 0.37375727870085207
```

Save the embeddings in JSONL format. The data must be formatted in JSONL format, which means each embedding dictionary is written as an individual JSON object on its own line.

[19]:
```
import tempfile
from pathlib import Path

# Create temporary file to write embeddings to
embeddings_file_path = Path(tempfile.mkdtemp())

print(f"Embeddings directory: {embeddings_file_path}")
```

```
Embeddings directory: /var/tmp/tmpzx18w_a0
```

Write embeddings in batches to prevent out-of-memory errors. Notice we are only using 5000 questions so that the embedding creation process and indexing is faster. The dataset contains more than 50,000 questions. This step will take around 5 minutes.

[24]:
```
import gc
import json

BQ_NUM_ROWS = 5000
BQ_CHUNK_SIZE = 1000
BQ_NUM_CHUNKS = math.ceil(BQ_NUM_ROWS / BQ_CHUNK_SIZE)

START_CHUNK = 0

# Create a rate limit of 300 requests per minute. Adjust this depending on your quota.
API_CALLS_PER_SECOND = 300 / 60
# According to the docs, each request can process 5 instances per request
ITEMS_PER_REQUEST = 5

# Loop through each generated dataframe, convert
for i, df in tqdm(
    enumerate(
        query_bigquery_chunks(
            max_rows=BQ_NUM_ROWS, rows_per_chunk=BQ_CHUNK_SIZE, start_chunk=START_CHUNK
        )
    ),
    total=BQ_NUM_CHUNKS - START_CHUNK,
    position=-1,
    desc="Chunk of rows from BigQuery",
):
    # Create a unique output file for each chunk
    chunk_path = embeddings_file_path.joinpath(
        f"{embeddings_file_path.stem}_{i+START_CHUNK}.json"
    )
    with open(chunk_path, "a") as f:
        id_chunk = df.id

        # Convert batch to embeddings
        is_successful, question_chunk_embeddings = encode_text_to_embedding_batched(
            sentences=df.title_with_body.to_list(),
            api_calls_per_second=API_CALLS_PER_SECOND,
            batch_size=ITEMS_PER_REQUEST,
        )

        # Append to file
        embeddings_formatted = [
            json.dumps(
                {
                    "id": str(id),
                    "embedding": [str(value) for value in embedding],
                }
            )
            + "\n"
            for id, embedding in zip(id_chunk[is_successful], question_chunk_embeddings)
        ]
        f.writelines(embeddings_formatted)

        # Delete the DataFrame and any other large data structures
        del df
        gc.collect()
```

```
Chunk of rows from BigQuery: 100% 5/5 [03:34<00:00, 43.13s/it]
100% 200/200 [00:40<00:00, 4.97it/s]
100% 200/200 [00:40<00:00, 4.96it/s]
100% 200/200 [00:40<00:00, 4.96it/s]
100% 200/200 [00:40<00:00, 4.96it/s]
100% 200/200 [00:40<00:00, 4.97it/s]
```

Upload embeddings to Cloud Storage

Upload the text-embeddings to Cloud Storage, so that Vertex AI Vector Search can access them later.

Define a bucket where you will store your embeddings.

[25]:
```
BUCKET_URI = f"gs://{PROJECT_ID}-unique"
```

Create your Cloud Storage bucket.

[26]:
```
! gsutil mb -l {REGION} -p {PROJECT_ID} {BUCKET_URI}
```

```
Creating gs://qwiklabs-gcp-02-9c2eecad5f04-unique/...
```

Upload the training data to a Google Cloud Storage bucket.

[27]:
```
remote_folder = f"{BUCKET_URI}/{embeddings_file_path.stem}/"
! gsutil -m cp -r {embeddings_file_path}/* {remote_folder}
```

```
Copying file:///var/tmp/tmpzx18w_a0/tmpzx18w_a0_0.json [Content-Type=application/json]...
Copying file:///var/tmp/tmpzx18w_a0/tmpzx18w_a0_1.json [Content-Type=application/json]...
Copying file:///var/tmp/tmpzx18w_a0/tmpzx18w_a0_2.json [Content-Type=application/json]...
Copying file:///var/tmp/tmpzx18w_a0/tmpzx18w_a0_3.json [Content-Type=application/json]...
Copying file:///var/tmp/tmpzx18w_a0/tmpzx18w_a0_4.json [Content-Type=application/json]...
- [5/5 files][ 27.9 MiB/ 27.9 MiB] 100% Done
Operation completed over 5 objects/27.9 MiB.
```

Create an Index in Vertex AI Vector Search for your embeddings

Setup your index name and description.

[28]:
```
DISPLAY_NAME = "stack_overflow"
DESCRIPTION = "question titles and bodies from stackoverflow"
```

Create the index. Notice that the index reads the embeddings from the Cloud Storage bucket. The indexing process can take from 45 minutes up to 60 minutes. Wait for completion, and then proceed. You can open a different Google Cloud Console page, navigate to Vertex AI Vector search, and see how the index is being created.

[29]:
```
from google.cloud import aiplatform

aiplatform.init(project=PROJECT_ID, location=REGION, staging_bucket=BUCKET_URI)

DIMENSIONS = 768

tree_ah_index = aiplatform.MatchingEngineIndex.create_tree_ah_index(
```

```
    display_name=DISPLAY_NAME,
    contents_delta_uri=remote_folder,
    dimensions=DIMENSIONS,
    approximate_neighbors_count=150,
    distance_measure_type="DOT_PRODUCT_DISTANCE",
    leaf_node_embedding_count=500,
    leaf_nodes_to_search_percent=80,
    description=DESCRIPTION,
)
```

```
Creating MatchingEngineIndex
Create MatchingEngineIndex backing LRO: projects/545248027730/locations/us-west1/indexes/6648183863208050688/operations/3743299981147111424
MatchingEngineIndex created. Resource name: projects/545248027730/locations/us-west1/indexes/6648183863208050688
To use this MatchingEngineIndex in another session:
index = aiplatform.MatchingEngineIndex('projects/545248027730/locations/us-west1/indexes/6648183863208050688')
```

```
Reference the index name to make sure it got created successfully
```

[30]:
```
INDEX_RESOURCE_NAME = tree_ah_index.resource_name
INDEX_RESOURCE_NAME
```

[30]:
```
'projects/545248027730/locations/us-west1/indexes/6648183863208050688'
```

Using the resource name, you can retrieve an existing MatchingEngineIndex.

[31]:
```
tree_ah_index = aiplatform.MatchingEngineIndex(index_name=INDEX_RESOURCE_NAME)
```

Create an IndexEndpoint so that it can be accessed via an API

[32]:
```
my_index_endpoint = aiplatform.MatchingEngineIndexEndpoint.create(
    display_name=DISPLAY_NAME,
    description=DISPLAY_NAME,
    public_endpoint_enabled=True,
)
```

```
Creating MatchingEngineIndexEndpoint
Create MatchingEngineIndexEndpoint backing LRO: projects/545248027730/locations/us-west1/indexEndpoints/2908542507002363904/operations/8625201977216729088
MatchingEngineIndexEndpoint created. Resource name: projects/545248027730/locations/us-west1/indexEndpoints/2908542507002363904
To use this MatchingEngineIndexEndpoint in another session:
index_endpoint = aiplatform.MatchingEngineIndexEndpoint('projects/545248027730/locations/us-west1/indexEndpoints/2908542507002363904')
```

Deploy your index to the created endpoint. This can take up to 15 minutes

[33]:
```
DEPLOYED_INDEX_ID = "deployed_index_id_unique"

DEPLOYED_INDEX_ID


my_index_endpoint = my_index_endpoint.deploy_index(
    index=tree_ah_index, deployed_index_id=DEPLOYED_INDEX_ID
)

my_index_endpoint.deployed_indexes
```

```
Deploying index MatchingEngineIndexEndpoint index_endpoint: projects/545248027730/locations/us-west1/indexEndpoints/2908542507002363904
Deploy index MatchingEngineIndexEndpoint index_endpoint backing LRO: projects/545248027730/locations/us-west1/indexEndpoints/2908542507002363904/operation
s/2257112104114847744
MatchingEngineIndexEndpoint index_endpoint Deployed index. Resource name: projects/545248027730/locations/us-west1/indexEndpoints/2908542507002363904
```

[33]:
```
[id: "deployed_index_id_unique"
index: "projects/545248027730/locations/us-west1/indexes/6648183863208050688"
create_time {
  seconds: 1748425073
  nanos: 774808000
}
index_sync_time {
  seconds: 1748426978
  nanos: 935213000
}
automatic_resources {
  min_replica_count: 2
  max_replica_count: 2
}
deployment_group: "default"
]
```

Verify number of declared items matches the number of embeddings.

Each IndexEndpoint can have multiple indexes deployed to it.

For each index, you can retrieve the number of deployed vectors using the index_endpoint._gca_resource.index_stats.vectors_count.

The numbers may not match exactly due to potential rate-limiting failures incurred when using the embedding service.

[34]:
```
number_of_vectors = sum(
    aiplatform.MatchingEngineIndex(
        deployed_index.index
    )._gca_resource.index_stats.vectors_count
    for deployed_index in my_index_endpoint.deployed_indexes
)

print(f"Expected: {BQ_NUM_ROWS}, Actual: {number_of_vectors}")
```

```
Expected: 5000, Actual: 1165
```

Create online queries

After you build your indexes, you may query against the deployed index to find nearest neighbors.

Note: For the `DOT_PRODUCT_DISTANCE` distance type, the "distance" property returned with each MatchNeighbor actually refers to the similarity.

Create an embedding for a test question.

[35]:
```
test_embeddings = encode_texts_to_embeddings(sentences=["Install GPU for Tensorflow"])
```

Test the query to retrieve the similar embeddings.

[36]:
```
NUM_NEIGHBOURS = 10

response = my_index_endpoint.find_neighbors(
    deployed_index_id=DEPLOYED_INDEX_ID,
    queries=test_embeddings,
    num_neighbors=NUM_NEIGHBOURS,
```

```
)

response
```

[36]:
```
[[MatchNeighbor(id='51235428', distance=0.598986804485321, sparse_distance=None, feature_vector=[], crowding_tag='0', restricts=[], numeric_restricts=[], sparse_embedding_values=[], sparse_embedding_dimensions=[]),
  MatchNeighbor(id='35513574', distance=0.5083289742469788, sparse_distance=None, feature_vector=[], crowding_tag='0', restricts=[], numeric_restricts=[], sparse_embedding_values=[], sparse_embedding_dimensions=[]),
  MatchNeighbor(id='50833570', distance=0.47362852096557617, sparse_distance=None, feature_vector=[], crowding_tag='0', restricts=[], numeric_restricts=[], sparse_embedding_values=[], sparse_embedding_dimensions=[]),
  MatchNeighbor(id='44636416', distance=0.4730268716812134, sparse_distance=None, feature_vector=[], crowding_tag='0', restricts=[], numeric_restricts=[], sparse_embedding_values=[], sparse_embedding_dimensions=[]),
  MatchNeighbor(id='61777922', distance=0.4651716947555542, sparse_distance=None, feature_vector=[], crowding_tag='0', restricts=[], numeric_restricts=[], sparse_embedding_values=[], sparse_embedding_dimensions=[]),
  MatchNeighbor(id='36587036', distance=0.4651235342025757, sparse_distance=None, feature_vector=[], crowding_tag='0', restricts=[], numeric_restricts=[], sparse_embedding_values=[], sparse_embedding_dimensions=[]),
  MatchNeighbor(id='55560819', distance=0.45997512340545654, sparse_distance=None, feature_vector=[], crowding_tag='0', restricts=[], numeric_restricts=[], sparse_embedding_values=[], sparse_embedding_dimensions=[]),
  MatchNeighbor(id='69295934', distance=0.45184260606765747, sparse_distance=None, feature_vector=[], crowding_tag='0', restricts=[], numeric_restricts=[], sparse_embedding_values=[], sparse_embedding_dimensions=[]),
  MatchNeighbor(id='53651531', distance=0.45116370916366577, sparse_distance=None, feature_vector=[], crowding_tag='0', restricts=[], numeric_restricts=[], sparse_embedding_values=[], sparse_embedding_dimensions=[]),
  MatchNeighbor(id='15314791', distance=0.4485892057418823, sparse_distance=None, feature_vector=[], crowding_tag='0', restricts=[], numeric_restricts=[], sparse_embedding_values=[], sparse_embedding_dimensions=[])]]
```

Verify that the retrieved results are relevant by checking the StackOverflow links

[37]:
```
for match_index, neighbor in enumerate(response[0]):
    print(f"https://stackoverflow.com/questions/{neighbor.id}")
```

```
https://stackoverflow.com/questions/51235428
https://stackoverflow.com/questions/35513574
https://stackoverflow.com/questions/50833570
https://stackoverflow.com/questions/44636416
https://stackoverflow.com/questions/61777922
https://stackoverflow.com/questions/36587036
https://stackoverflow.com/questions/55560819
https://stackoverflow.com/questions/69295934
https://stackoverflow.com/questions/53651531
https://stackoverflow.com/questions/15314791
```

Clean up the Google Cloud environment

To clean up all Google Cloud resources used in this project, you can delete the Google Cloud project you used for the lab. You can also manually delete resources that you created by running the following code.

[38]:
```
import os

delete_bucket = False

# Force undeployment of indexes and delete endpoint
my_index_endpoint.delete(force=True)

# Delete indexes
tree_ah_index.delete()

if delete_bucket or os.getenv("IS_TESTING"):
    ! gsutil rm -rf {BUCKET_URI}
```

```
Undeploying MatchingEngineIndexEndpoint index_endpoint: projects/545248027730/locations/us-west1/indexEndpoints/2908542507002363904
Undeploy MatchingEngineIndexEndpoint index_endpoint backing LRO: projects/545248027730/locations/us-west1/indexEndpoints/2908542507002363904/operations/8073511022863843328
MatchingEngineIndexEndpoint index_endpoint undeployed. Resource name: projects/545248027730/locations/us-west1/indexEndpoints/2908542507002363904
Deleting MatchingEngineIndexEndpoint : projects/545248027730/locations/us-west1/indexEndpoints/2908542507002363904
MatchingEngineIndexEndpoint deleted. . Resource name: projects/545248027730/locations/us-west1/indexEndpoints/2908542507002363904
Deleting MatchingEngineIndexEndpoint resource: projects/545248027730/locations/us-west1/indexEndpoints/2908542507002363904
Delete MatchingEngineIndexEndpoint backing LRO: projects/545248027730/locations/us-west1/indexEndpoints/2908542507002363904/operations/561225369433145344
MatchingEngineIndexEndpoint resource projects/545248027730/locations/us-west1/indexEndpoints/2908542507002363904 deleted.
Deleting MatchingEngineIndex : projects/545248027730/locations/us-west1/indexes/6648183863208050688
MatchingEngineIndex deleted. . Resource name: projects/545248027730/locations/us-west1/indexes/6648183863208050688
Deleting MatchingEngineIndex resource: projects/545248027730/locations/us-west1/indexes/6648183863208050688
Delete MatchingEngineIndex backing LRO: projects/545248027730/locations/us-west1/indexes/6648183863208050688/operations/579521242919337984
MatchingEngineIndex resource projects/545248027730/locations/us-west1/indexes/6648183863208050688 deleted.
```

[ ]: