Quick tip: Review the prerequisites before you run the lab

End Lab 01:00:00

Provisioning lab resources

Estimated time remaining 2 minutes

Caution: When you are in the console, do not deviate from the lab instructions. Doing so may cause your account to be blocked. Learn more.

Understanding embeddings with Vertex AI Text-Embeddings API

Lab 1 hour No cost Intermediate

★★★★★ Rate Lab

This lab may incorporate AI tools to support your learning.

Previous Next

Overview

a pivotal role in many tasks involving the identification of similar items, like Google searches, online shopping recommendations, and personalized music suggestions.

You can use text embeddings for tasks like classification, outlier detection, text clustering and semantic search. You can combine semantic search with the text generation capabilities of an LLM to build a question-answering systems using Google Cloud's Vertex AI.

What you will learn

- The properties of word and sentence embeddings.
- How embeddings can be used to measure the semantic similarity between two pieces of text.
- How to visualize embeddings in a 2-dimensional space.

Setup and requirements

Before you click the Start Lab button

Read these instructions. Labs are timed and you cannot pause them. The timer, which starts when you click **Start Lab**, shows how long Google Cloud resources will be made available to you.

This Qwiklabs hands-on lab lets you do the lab activities yourself in a real cloud environment, not in a simulation or demo environment. It does so by giving you new, temporary credentials that you use to sign in and access Google Cloud for the duration of the lab.

What you need

To complete this lab, you need:

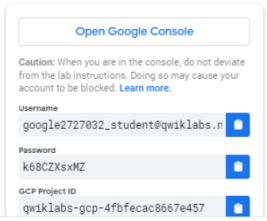
Note: If you already have your own personal Google Cloud account or project, do not use it for this lab.

Note: If you are using a Pixelbook, open an Incognito window to run this lab.

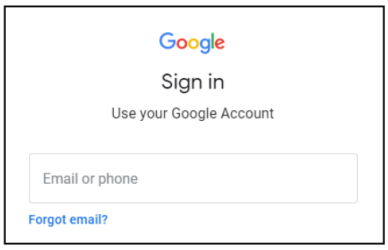
How to start your lab and sign in to the Google Cloud Console

1. Click the **Start Lab** button. If you need to pay for the lab, a pop-up opens for you to select your payment method. On the left is a panel populated with the temporary

credentials that you must use for this lab.

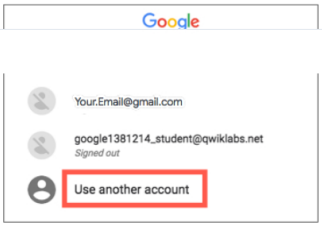


2. Copy the username, and then click **Open Google Console**. The lab spins up resources, and then opens another tab that shows the **Sign in** page.



Tip: Open the tabs in separate windows, side-by-side.

If you see the **Choose an account** page, click **Use Another Account**.



3. In the **Sign in** page, paste the username that you copied from the Connection Details panel. Then copy and paste the password.

Important: You must use the credentials from the Connection Details panel. Do not use your Qwiklabs credentials. If you have your own Google Cloud account, do not use it for this lab (avoids incurring charges).

4. Click through the subsequent pages:

- Accept the terms and conditions.
- Do not add recovery options or two-factor authentication (because this is a temporary account).
- Do not sign up for free trials.

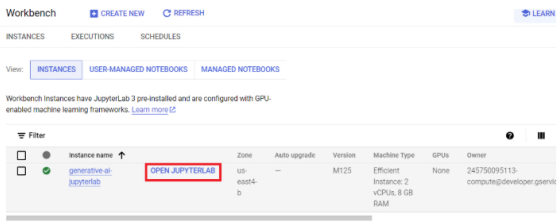
Note: You can view the menu with a list of Google Cloud Products and Services by clicking the **Navigation menu** at the top-left.

Task 1. Open a Jupyter notebook in Vertex AI Workbench

1. In your Google Cloud project, navigate to Vertex AI Workbench. In the top search bar of the Google Cloud console, enter **Vertex AI Workbench**, and click on the first result.

2. Under **Instances**, click on **Open Jupyterlab** for **generative-ai-jupyterlab** notebook.

The JupyterLab will run in a new tab.



3. On the **Launcher**, under **Notebook**, click on **Python 3** to open a new python notebook.

Task 2. Generate your first embeddings

1. In the first cell run the following command to install the Google Cloud Vertex AI SDKs. To run the command, execute **SHIFT+ENTER**.

```
!pip install google-cloud-aiplatform
```

2. Import and initialize the Vertex AI Python SDK.

```
import vertexai
vertexai.init()
```

Ignore any warnings regarding Tensorflow.

3. Import and load the embeddings model.

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

4. Generate a word embedding.

```
embedding = embedding_model.get_embeddings(['life'])
```

This command returns a list with a single text embedding object. That list contains 768 items, in the form of numbers between -1 and 1, called dimensions. These dimensions are used to compare similarity between embeddings.

5. Explore the embedding generated by printing the lenght and the first 10 values.

```
vector = embedding[0].values
print(f"Length = {len(vector)}")
print(vector[:10])
```

6. Generate a sentence embedding.

```
embedding = embedding_model.get_embeddings(['What is the meaning of life?'])
```

7. Print the values for this embedding.

```
vector = embedding[0].values
print(f"Length = {len(vector)}")
print(vector[:10])
```

Notice that it also has 768 dimensions and the list or vector, is composed of numbers between -1 and 1.

Task 3. Calculate the similarity from embeddings

To calculate the similarity we will use the sklearn libraries, specifically the cosine_similarity function.

1. Import the cosine_similarity function in the sklearn libraries.

```
from sklearn.metrics.pairwise import cosine_similarity
```

2. Create 3 embeddings for 3 different sentences, so that we can compare them to each other.

```
emb_1 = embedding_model.get_embeddings(['What is the meaning of
life?'])
emb_2 = embedding_model.get_embeddings(['How does one spend their
time well on Earth?'])
emb_3 = embedding_model.get_embeddings(['Would you like a
salad?'])

vec_1 = [emb_1[0].values]
vec_2 = [emb_2[0].values]
vec_3 = [emb_3[0].values]
```

Note: the reason we wrap the embeddings (a Python list) in another list is because the `cosine_similarity` function expects either a 2D numpy array or a list of lists.

3. Print the similarity between these vectors.

```
print(cosine_similarity(vec_1,vec_2))
print(cosine_similarity(vec_2,vec_3))
print(cosine_similarity(vec_1,vec_3))
```

Output

```
[[0.65503744]]
[[0.52001556]]
[[0.54139322]]
```

Notice that this numbers are in a very similar range. Cosine similarity can go between 0 and 1. But because we have a very high number of dimensions the numbers are going to be close. However, even if they look close, they are still very far apart from each other relatively speaking.

Task 4. Visualizing Embedddings

In this task, we will create embeddings for several sentences. Then we will reduce the dimensions from 768 to only 2, so that we can visualize the vectors in plot. We will see graphically that similar vectors are represented closer to one another.

1. Set up 7 sentences for our example.

```
in_1 = "Missing flamingo discovered at swimming pool"
in_2 = "Sea otter spotted on surfboard by beach"
in_3 = "Baby panda enjoys boat ride"
in_4 = "Breakfast themed food truck beloved by all!"
in_5 = "New curry restaurant aims to please!"

input_text_lst_news = [in_1, in_2, in_3, in_4, in_5, in_6, in_7]
```

2. Get embeddings for all pieces of text.

```
embeddings = []
for input_text in input_text_lst_news:
    emb = embedding_model.get_embeddings(
        [input_text])[0].values
    embeddings.append(emb)
```

3. Store them in a 2D NumPy array, with one row for each embedding.

```
import numpy as np
embeddings_array = np.array(embeddings)
print("Shape: " + str(embeddings_array.shape))
print(embeddings_array)
```

The shape is 7 rows of 768 dimensions

4. Reduce embeddings from 768 to 2 dimensions so that we can visualize them. We'll use principal component analysis (PCA).

We'll use principal component analysis (PCA).

```
from sklearn.decomposition import PCA

# Perform PCA for 2D visualization
PCA_model = PCA(n_components = 2)
PCA_model.fit(embeddings_array)
new_values = PCA_model.transform(embeddings_array)
```

5. Show the shape of the new array.

```
print("Shape: " + str(new_values.shape))
print(new_values)
```

The shape is now 7 rows of 2 dimensions.

6. Install the visualization libraries.

7. Visualize the output.

```
import seaborn as sns
import pandas as pd

data = pd.DataFrame({ 'x':new_values[:,0], 'y':new_values[:,1],
'sentences': input_text_lst_news})

# Create a visualization
sns.relplot(
    data,
    x='x',
    y='y',
    kind='scatter',
    hue='sentences'
)
```

Notice that similar sentences are plotted closer to each other. For instance the sentences about Python developers and programming languages are both plotted to the

Congratulations

You have now completed the lab! In this lab, you created embeddings using the Vertex AI Text Embeddings API. Then you explored how similar embeddings might be, and plotted those embeddings to have a graphical representation.

Next steps

Congratulations

You have now completed the lab! In this lab, you created embeddings using the Vertex AI Text Embeddings API. Then you explored how similar embeddings might be, and plotted those embeddings to have a graphical representation.

Next steps

- Check out the Generative AI on Vertex AI documentation.
- Learn more about Generative AI on the Google Cloud Tech YouTube channel.

Google Cloud Training & Certification