

Google Cloud Skills Boost for Partners

[Main menu](#)

Develop Advanced Enterprise Search and Conversation Applications

Course - 8 hours 

Vector Search

Streaming Updates into Vertex AI Vector Search

Improving RAG Solutions

Google Cloud databases for embeddings

Storing and Querying

Embeddings with AlloyDB for PostgreSQL

Using BigQuery

Embeddings in a RAG Architecture

Challenge Lab

Generative AI

Search: Challenge

Course > Develop Advanced Enterprise Search and Conversation Applications >

Quick tip: Review the prerequisites before you run the lab

[Start Lab](#)

01:00:00

Generative AI Search: Challenge Lab

 Lab  1 hour  No cost  Intermediate

 Rate Lab

 This lab may incorporate AI tools to support your learning.

Lab instructions and tasks	-/100
Challenge Lab Overview	
Objective	
Setup	
Challenge Scenario	
Your challenge	
Task 1. Create a website search app	
Task 2. Create a structured and unstructured data search app	
Task 3. Create a generative Conversational Agents	
Task 4. Enhance customer engagement and optimize performance	
Task 5. Generate	

Challenge Lab Overview

This lab will challenge you to perform actions and automation across products. Instead of following step-by-step instructions, you are given a common business scenario and a set of tasks – you figure out how to complete them on your own! An automated scoring system (shown on this page) provides feedback on whether you have completed your

[Previous](#)[Next >](#)

When you take a Challenge Lab, you will not be taught Google Cloud concepts. You will need to use your skills to assess how to build the solution to the challenge presented. This lab is only recommended for students who have those skills. Are you up for the challenge?

Objective

The objective of this challenge lab is to showcase your proficiency in integrating Search in applications with AI Applications, while also demonstrating your ability to leverage the Vertex AI Text-Embeddings API. Specifically, you will focus on generating embeddings for text sentences and calculating similarity using the cosine similarity function. In this lab, you will design and implement a comprehensive solution that combines powerful search capabilities, a virtual assistant, and text embeddings for semantic search and similarity calculations to enhance the user experience.

Setup

Qwiklabs setup

For each lab, you get a new Google Cloud project and set of resources for a fixed time at no cost.

1. Make sure you signed into Qwiklabs using an **incognito window**.
2. Note the lab's access time (for example, **02:00:00**) and make sure you can finish in that time block.

There is no pause feature. You can restart if needed, but you have to start at the beginning.

4. Note your lab credentials. You will use them to sign in to the Google Cloud

[Open Google Console](#)

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](#).

Username

google2876526_student@qwiklabs.n 

Password

TG959yrKDX

GCP Project ID
qwiklabs-gcp-0855e773352d3560

New to labs? View our introductory video!

6. Click **Use another account** and copy/paste credentials for **this lab** into the prompts.

If you use other credentials, you'll get errors or **incur charges**.

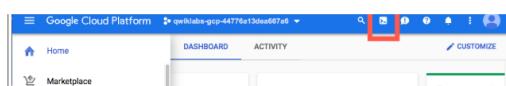
7. Accept the terms and skip the recovery resource page.

Do not click **End Lab** unless you are finished with the lab or want to restart it. This clears your work and removes the project.

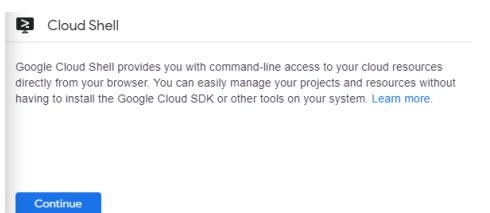
Activate Cloud Shell

Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud. Cloud Shell provides command-line access to your Google Cloud resources.

In the Cloud Console, in the top right toolbar, click the **Activate Cloud Shell** button.



Click **Continue**.



It takes a few moments to provision and connect to the environment. When you are connected, you are already authenticated, and the project is set to your *PROJECT_ID*. For example:



`gcloud` is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab-completion.

You can list the active account name with this command:



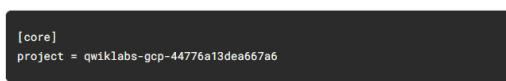
(Example output)



(Output)



(Example output)



Challenge Scenario



Here is a company overview as provided on the Cymbol Shops's website.

Cymbol Shops is an American retail chain headquartered in Minneapolis that sells homeware, electronics, and clothing.

Founded in 1974, Cymbol Shops started out as Cymbol Air, selling AC systems manufactured by Cymbol Group in Minnesota and neighboring states. The company quickly expanded into domestic merchandise in order to satisfy the need for quality products at an affordable price in the midst of a recession.

Cymbol Shops' broad product assortment was once a benefit - capturing planned purchases and last minute splurges. In recent years, however, the company has

keep pace and maintain relevance. Today, Cymbol Shops operates 714 stores across North America and reported \$15 billion in revenue in 2019. They currently employ 80,400 employees across the United States and Canada.

Cymbol Shops is a digitally transforming legacy retailer.

It is inspired by clients like: Bed Bath & Beyond, Best Buy, Home Depot, Nordstrom.

Your challenge

You work as an engineer at Cymbol Shops, a retail chain company renowned for its innovative solutions. Cymbol Shops is embarking on a significant project that involves developing a robust search functionality and virtual assistant for a new application. Your task is to design a Proof of Concept (POC) solution that seamlessly integrates AI Applications, and focuses on generating embeddings for text sentences using the

Cymbol Shops' new application demands a powerful and efficient search feature, aiming to catalog and retrieve various data types, including text documents, images, and structured data using AI Applications. Additionally, the company aims to create a virtual assistant using AI Applications to assist customers with queries about products and devices available in the Google Store. The virtual assistant will be built and configured using AI Applications. Moreover, the solution should leverage text embeddings generated by the Vertex AI Text-Embeddings API for semantic search and similarity calculations.

Therefore in this lab, you should:

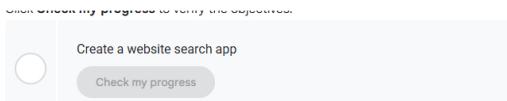
1. Create a website search app named `App Name 1`.
2. Create a structured data search app `App Name 2`.
3. Create an unstructured data search app `App Name 3`.
4. Create a generative conversational agents named `Agent Name`.
5. Enhance customer engagement and optimize performance.
6. Generate embeddings for text sentences.

Task 1. Create a website search app

In this task, demonstrate how AI Applications can be utilized to create a web search application, showcasing its effectiveness in searching and retrieving content from websites. This app should be capable of crawling web content, indexing it, and providing users with the ability to search for information.

Therefore in this task:

1. Create an app named **App Name 1** of type **Search** and set the location of the app to **Global**.
 2. Configure this app to use a datastore named **Datastore Name 1** that indexes data from the following website: cloud.google.com/*.
-
3. Finally, preview the app and verify if it is working as intended.



Task 2. Create a structured and unstructured data search app

In this task, showcase how AI Applications can be employed to create a structured data search application, emphasizing its capabilities in handling and searching structured information. The application should allow users to search for specific data points, filter results, and navigate through structured information efficiently.

Therefore in this task:

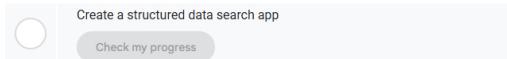
2. Configure this app to use a datastore named **Datastore Name 2** that ingests data from the following Cloud Storage bucket: cloud-samples-data/gen-app-builder/search/kaggle_movies.

Hint: This Cloud Storage bucket contains Newline Delimited JSON-formatted dataset of movies made available by [Kaggle](#).

3. Preview the **Datastore Name 2** app and configure the search results to only display the following:

Key	Value
Title	title
Thumbnail	poster_path
Text 1	tagline
Text 2	revenue

4. Finally, preview the app and verify if it is working as intended.
-



In this sub task, demonstrate how AI Applications can be harnessed to create an unstructured data search application, illustrating its proficiency in searching and presenting unstructured content.

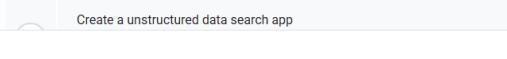
Therefore, in this task,

5. Create an app named **App Name 3** of type **Search** and set the location of the app to **Global**.

6. Configure this app to use a datastore named **Datastore Name 3** that ingests unstructured data from the following Cloud Storage bucket: cloud-samples-data/gen-app-builder/search/alphabet-investor-pdfs.

7. Finally, preview the app and verify if it is working as intended.

Click **Check my progress** to verify the objectives.



Task 3. Create a generative Conversational Agents

In this task, you should create a Data Store Agent using AI Applications. This agent should be able to comprehend unstructured data. Your objective is to configure and deploy a virtual agent that can provide assistance to customers, with a particular focus on handling unstructured information.

Therefore, in this task,

1. Enable necessary APIs: Dialogflow and AI Applications.
2. Create a Conversational agent with the following configurations and Select Build your own on Get started with Conversational Agents pop-up.

Company Name	Company Name
Agent Name	Agent Name
Location	Global

3. To Create Data store, select + Data store.
4. Configure this application to use a Datastore Name 4 that ingests unstructured data from the folder present in the following Cloud Storage bucket: cloud-samples-data/gen-app-builder/search/alphabet-investor-pdfs

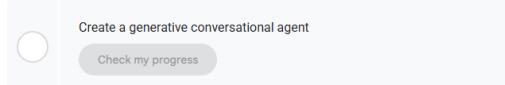
Note: It can take up to 10 minutes for your documents to be available and ready for use by your agent while your newly added data is being indexed.

5. Test the virtual agent by starting an interactive session with the chatbot to see how it responds to various questions that a customer might ask.
 - a. Type a greeting to your agent such as Hello.
 - b. Ask the agent the some questions such as.

- Where does Google see the most potential opportunity for long term growth?
- Where is Google allocating the most amount of its capital?

Note: If your agent is giving you responses from the default negative intent (e.g., "I'm sorry, I didn't get that. Can you rephrase your question?"), then be aware that it can take some time for the agent to be ready for use while your domains, URLs, or documents are being indexed.

Click **Check my progress** to verify the objectives.



and optimize performance

In this task, you will set up both voice and chat interactions for your virtual agent, created with AI Applications. The objective is to create versatile communication channels that allow customers to engage with your agent through voice commands and text-based chat, delivering a flexible and user-friendly experience. You will further delve into the conversation history and analytics of your Data Store Agent to view and interpret data related to customer interactions, agent responses, and overall performance. Understanding these analytics is essential for optimizing your virtual agent's capabilities and enhancing customer service.

Therefore in this task;

1. Back to Conversational Agents console > Integrations, Integrate a phone gateway into your bot named Agent Name for the United States. Name this phone gateway as Phone Gateway Name.

Note: This bot makes use of the Speech-to-Text and Text-to-Speech capabilities in Google Cloud.

2. Embed a chat widget on a website so customers can chat with it in addition to making a phone call to connect with it.

3. Finally, enable interaction logging for conversation history and analytics to view statistics related to agent requests and responses, and determine where the problems and friction points are in your virtual agent.

Click **Check my progress** to verify the objectives.

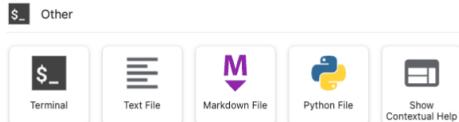


Task 5. Generate embedding and calculate similarity

Open Python notebook and install libraries

2. On the left navigation menu select **Workbench > Instance > generative-ai-jupyterlab > Open Jupyterlab**.

3. Once your Jupyter notebook opens select **Terminal**.



4. Use `gsutil` to download the `genai_search_challenge_notebook.ipynb` IPython notebook file from the Cloud Storage bucket `gs://partner-genai-bucket/genai024/`.

5. Click on the refresh button on top of the file browser to see the IPython Notebook you just downloaded and open it.

File Edit View Run Kernel Git Tabs Settings Help

+		+	↑	↶	↶	☰	✖
Filter files by name							🔍
📁 /							
Name		Last Modified					
📁 generative-ai		6 minutes ago					
📄 genai_search_challenge_notebook.ipynb		seconds ago					
📄 notebook_template.ipynb		7 minutes ago					

Click **Check my progress** to verify the objective.

Download Jupyter notebook file

Check my progress

6. Select the first cell of the notebook and either hit the play button on the notebook toolbar or press `Shift + Enter` on your keyboard to execute the cell.

Generate embeddings

7. Use the notebook toolbar to press the play icon or enter `Shift + Enter` to run the all of the cells of the notebook up until the cell with the following statement:

8. The next cell exports your `PROJECT_ID` AND `REGION`. Substitute the values, as shown in **Connection Details Panel** on the left hand side.

Output:

```
PROJECT = !gcloud config get-value project
PROJECT_ID = PROJECT[0]
REGION = 'GCP_REGION'
```

The block should look like this:

```
REGION = "Lab GCP Region"

# Set up the Cloud Logging client
from google.cloud import logging as cloud_logging
import logging

client = cloud_logging.Client()
handler = cloud_logging.handlers.CloudLoggingHandler(client)
logging.getLogger().setLevel(logging.INFO)
logging.getLogger().addHandler(handler)
```

9. In the next cell, add statements to load a pre-trained text embedding model named `text-embedding-004`.

10. In the next cell, obtain text embeddings for the following sentences: I love

11. In the next cell, write a code to provide information about the length and initial values of the vector representation of the first input text sentence obtained from the text embedding model.

Your code must extract the numerical values (vector) from the first element of the

lines of code already provided in the cell.

Example:

```
logging.info(f"Length of the vector: {len(vector)}")
logging.info(f"First 10 values of the vector: {vector[:10]}")
```

This is a common practice to inspect the characteristics of the obtained embeddings.

The output should look similar to the following.

```
Length = 768
[ 0.4948215733100267, -0.48243948975158817, -0.4362820588115994, 0.49134962733717, 0.4955770812349988, 0.49967738239276384384, 0.46738973520649, 0.4172081092514797, -0.4601179363750452]
```

Calculate the similarity from embeddings

12. To calculate the similarity we will use the sklearn libraries, specifically the `cosine_similarity` function. Run the next cell to import them.
13. In the next cell, write a code to generate individual embeddings for the following sentences:

b. Exploring the wonders of deep learning.

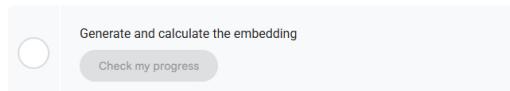
14. Subsequently, define vectors `vec_1` and `vec_2` that hold the numerical values that represent the semantic content of the respective sentences in the embedding space.
15. Finally, in the next cell, write a code that prints the similarity between these vectors. The cell already has necessary lines of code that logs the value.

The output should look similar to the following

Output:

```
Cosine Similarity between vec_1 and vec_2: 0.49482157331002674
```

Click **Check my progress** to verify the objective.



Congratulations!

You've successfully integrated Search in applications with AI Applications, and demonstrated proficiency in generating embeddings for text sentences while calculating similarity.

Manual Last Updated January 13, 2025

Lab Last Tested January 13, 2025

Copyright 2023 Google LLC All rights reserved. Google and the Google logo are trademarks of Google LLC. All other company and product names may be trademarks of the respective companies with which they are associated.