


 Search
 Write

 R

Use Google Vertex AI Search & Conversation to Build RAG Chatbot

Euclidean AI · [Follow](#)

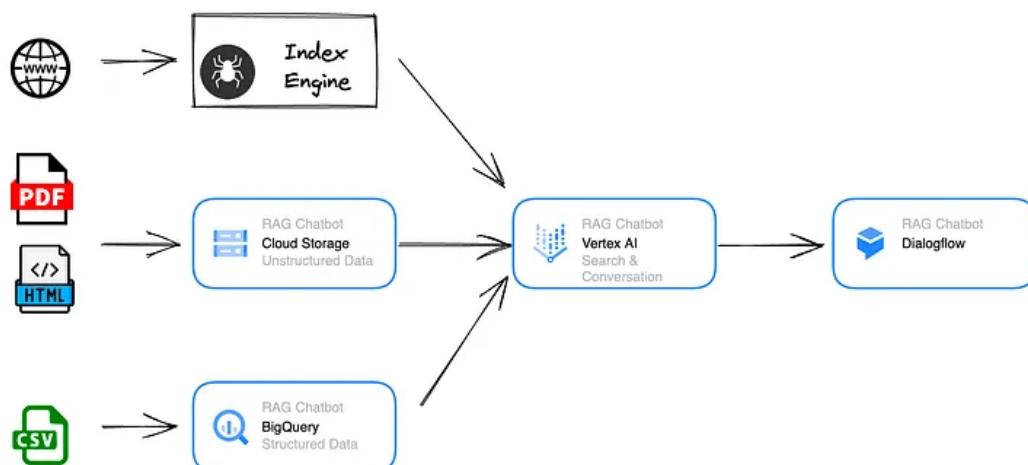
Published in Towards AI · 6 min read · Oct 25

16

1



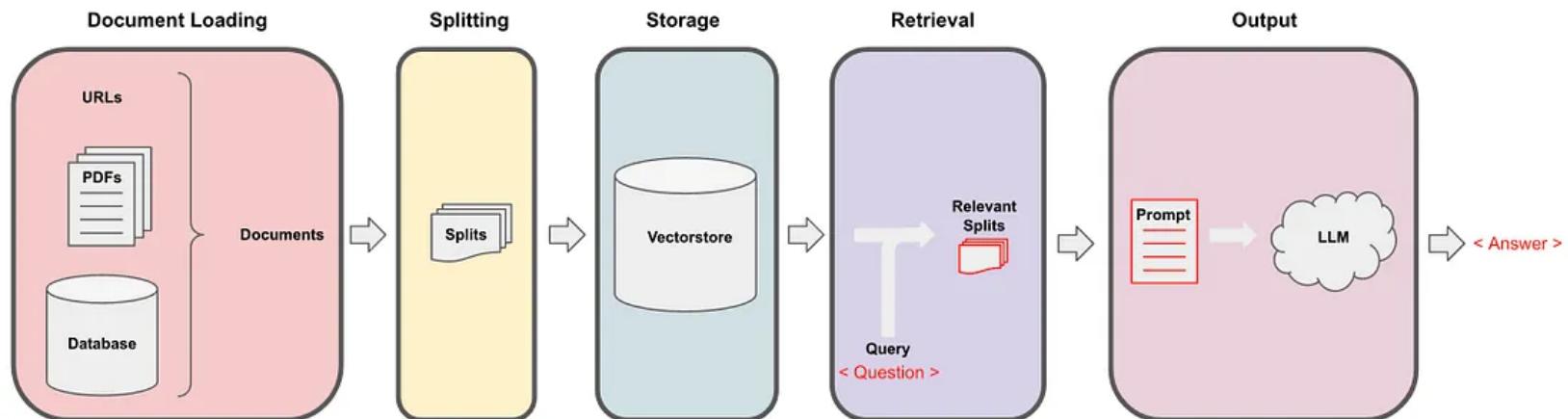
...



Source: Author's diagram

Google has recently released their managed RAG (Retrieval Augmented Generator) service, Vertex AI Search & Conversation, to GA (General Availability). This service, previously known as Google Enterprise Search, brings even more competition to the already thriving LLM (Large Language Model) chatbot market.

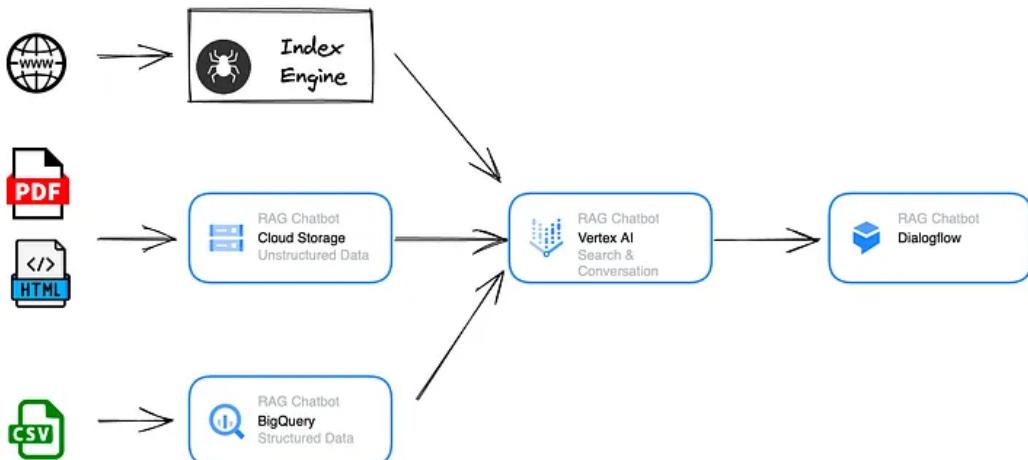
This article assumes that you have some knowledge on the RAG framework for building LLM-powered chatbots. The diagram below from Langchain provides a good introduction to RAG:



Source: https://python.langchain.com/docs/use_cases/question_answering/

In this article, we are mainly focused on setting up an LLM Chatbot with Google Vertex AI Search & Conversation (we will use Vertex AI Search in the rest of the article).

Solution Diagram



Source: Author's diagram

Datastore

Vertex AI Search currently supports html, pdf, and csv format source data.

Data store type	Supported formats	Examples	Input mechanism
Web	html, pdf	www.example.com/*	collected from google search index
Unstructured	html, pdf	mydoc.pdf, private.html	uploaded through Cloud Storage bucket or BigQuery
Structured	csv	"why?","why not"	uploaded through Cloud Storage bucket or BigQuery

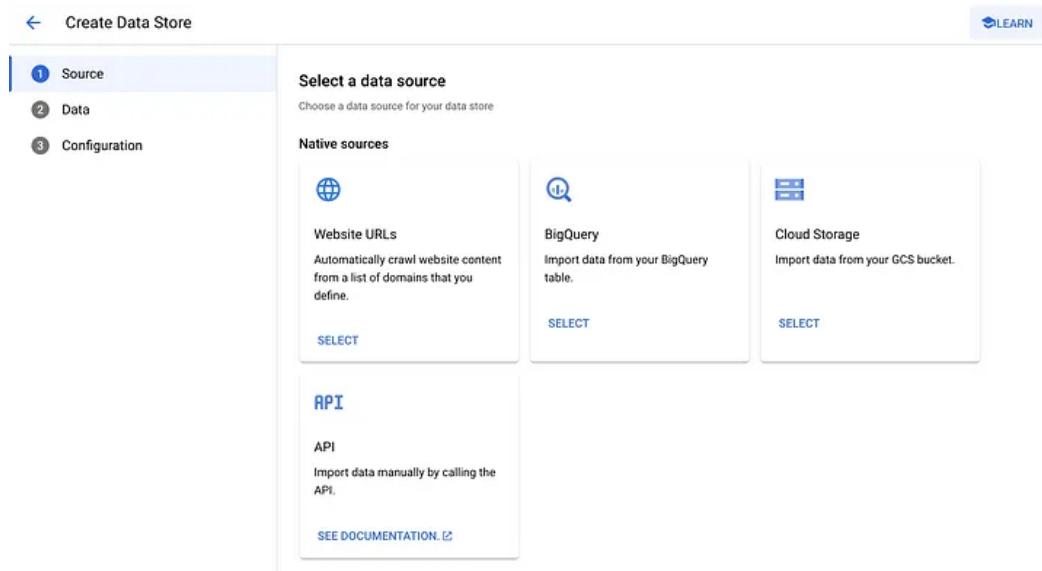
Source: <https://cloud.google.com/generative-ai-app-builder/docs/agent-data-store>

Vertex AI Search currently supports html, pdf, and csv format source data. Source data is collected from Google's search index (the same index used for Google Search). This is a major advantage compared to other providers that require separate scraping mechanisms to extract website information.

For unstructured data (currently, only pdf and html are supported), files first need to be uploaded to a Cloud Storage bucket. This storage bucket can be in any available region.

⚠ The datastore on Vertex AI Search is different from Cloud Storage. It is similar to the “vector databases” often referred by other providers.

Each app on Vertex AI Search currently will have its own datastore(s). It's possible to have multiple datastores under a single App.



Source: Author's screenshot from GCP environment

For demonstration purposes, we will use a student handbook sample pdf available online (<https://www.bcci.edu.au/images/pdf/student-handbook.pdf>). Note: we are not affiliated with this organization (it's just a sample pdf we can find online...)

Step 1 — Prepare the pdf files:

Split the single pdf handbook into multiple pages using Python. This only takes seconds.

```
from PyPDF2 import PdfWriter, PdfReader  
  
inputpdf = PdfReader(open("student-handbook.pdf", "rb"))
```

```

for i in range(len(inputpdf.pages)):
    output = PdfWriter()
    output.add_page(inputpdf.pages[i])
    with open("./split_pdfs_student_handbook/document-page%s.pdf" % i, "wb") as
        outputStream

```

Step 2 – Upload to GCS Bucket:

We will need to upload the pdfs to a google Cloud Storage. You can either do it through the Google Console or Use GCP SDK with any language of your choice.

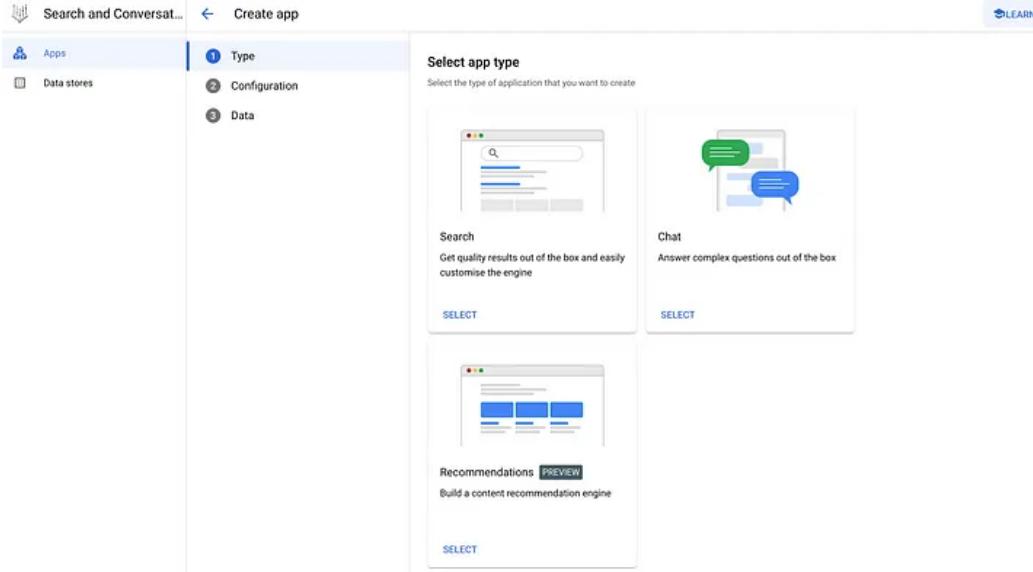
If only a single pdf document is required, ‘clickops’ will suffice for this demonstration.

Let’s drop the pdfs to a Cloud Storage bucket.

Name	Size	Type	Created	Storage class	Last modified	Public access	⋮
document-page0.pdf	36.2 KB	application/pdf	21 Oct 2023, 18:36:06	Standard	21 Oct 2023, 18:36:06	Not public	⋮
document-page1.pdf	24.6 KB	application/pdf	21 Oct 2023, 18:36:06	Standard	21 Oct 2023, 18:36:09	Not public	⋮
document-page10.pdf	37.2 KB	application/pdf	21 Oct 2023, 18:36:19	Standard	21 Oct 2023, 18:36:19	Not public	⋮
document-page11.pdf	89.5 KB	application/pdf	21 Oct 2023, 18:36:20	Standard	21 Oct 2023, 18:36:20	Not public	⋮
document-page12.pdf	89 KB	application/pdf	21 Oct 2023, 18:36:22	Standard	21 Oct 2023, 18:36:22	Not public	⋮
document-page13.pdf	106.2 KB	application/pdf	21 Oct 2023, 18:36:23	Standard	21 Oct 2023, 18:36:23	Not public	⋮
document-page14.pdf	36.6 KB	application/pdf	21 Oct 2023, 18:36:23	Standard	21 Oct 2023, 18:36:23	Not public	⋮
document-page15.pdf	89.4 KB	application/pdf	21 Oct 2023, 18:36:26	Standard	21 Oct 2023, 18:36:26	Not public	⋮
document-page16.pdf	90.2 KB	application/pdf	21 Oct 2023, 18:36:26	Standard	21 Oct 2023, 18:36:26	Not public	⋮
document-page17.pdf	91.4 KB	application/pdf	21 Oct 2023, 18:36:26	Standard	21 Oct 2023, 18:36:26	Not public	⋮
document-page18.pdf	89.1 KB	application/pdf	21 Oct 2023, 18:36:29	Standard	21 Oct 2023, 18:36:29	Not public	⋮

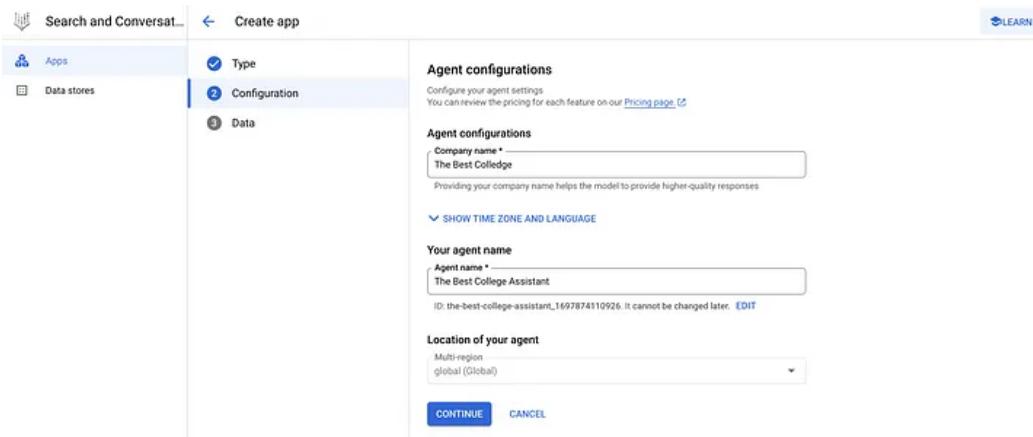
Source: Author’s screenshot from GCP environemnt

Step 3 – Set up App and Datastore:



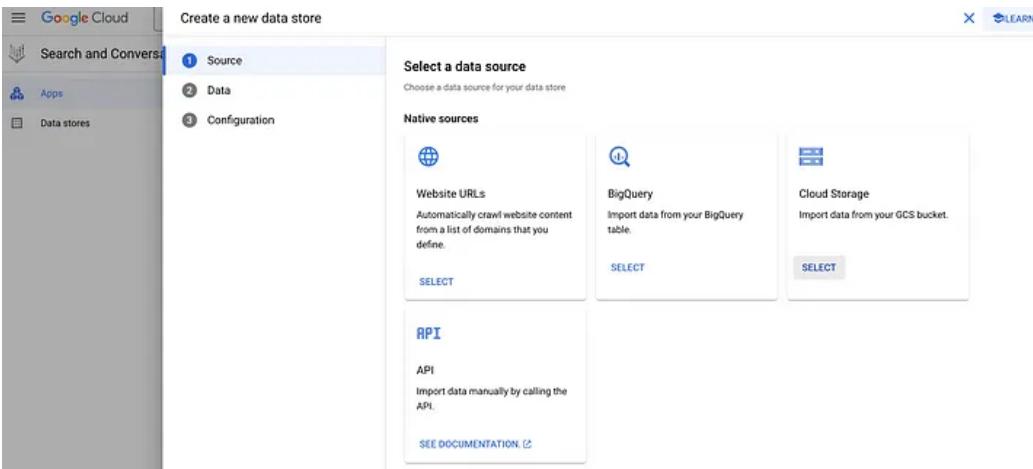
Source: Author's screenshot from GCP environment

In the GCP console, find ‘Search and Conversation’ and click on ‘Create App’. Select the Chat app type. Configure the app by naming the company and agent. Note: the agent is only available in the Global region.



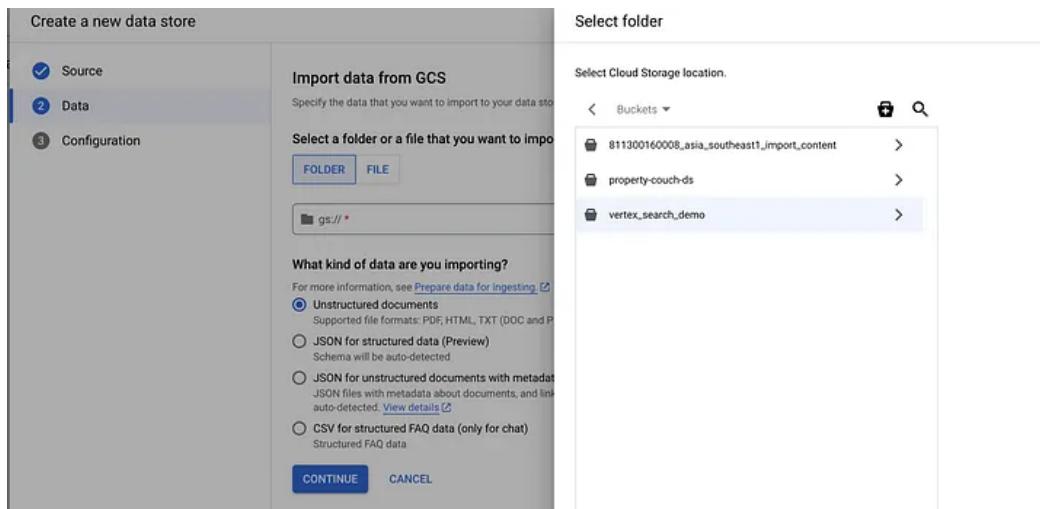
Source: Author's screenshot from GCP environment

Next, create a data store. Select ‘Cloud Storage’ and choose the bucket created in step 2.



Then, it will ask you to create a data store. Again, this is actually a vector datastore that stores the embeddings for semantic search and LLM to work.

Select ‘Cloud Storage’. Then, select the cloud storage that we created in the last step.



Source: Author's screenshot from GCP environment

After this, click on ‘Continue’. Then, the embedding will start. This process will take anywhere from a few minutes to an hour, depending on the amount of data. Also, GCP has **SDK libraries with async clients available, which can expedite the whole process**. Otherwise, you can always use GCP console. It uses synchronous API calls by default.

You can check the import (embedding) progress under the activity tab.

The image shows the 'student-handbook' data store details page in the GCP console. On the left, there's a sidebar with 'Available data stores' and a 'Preview' button. The main area shows the data store ID 'student-handbook_1697874664957', type 'Unstructured data', region 'global', number of documents '0', and last document import '21 Oct 2023, 18:51:18'. Below this is a 'VIEW DETAILS' button. At the bottom, there are 'DOCUMENTS' and 'ACTIVITY' tabs. The 'ACTIVITY' tab is selected, showing 'Activity 0' and '+ IMPORT DATA'. A 'Filter' section allows filtering by status, details, items succeeded, operation name, and last updated. A table shows one entry: 'Import in progress' with 'No errors' and '0' items succeeded, operation name 'import-documents-1093328139688841803', and last updated '21 Oct 2023, 18:51:18'.

Source: Author's screenshot from GCP environment

Once the import is completed, head over to ‘Preview’ on the left, it will take you to a Dialogflow CX Console.

Step 4 – Dialogflow:

The screenshot shows the Dialogflow console interface. On the left, the 'Start Page' configuration is displayed under 'Default Start Flow'. It includes sections for 'Routes' (Default Welcome Intent, Talk to agent), 'Event handlers' (sys.no-match-default, sys.no-input-default), and 'Data stores' (Edit data stores). A note states: 'Data store updates take some time to propagate and might not be immediately available.' On the right, the 'Data stores' panel is open, showing three entries: 'Data store' (Website), 'student-handbook' (Unstructured documents), and 'FAQ documents'. Below the data stores is a 'Fulfillment' section with a note about displaying FAQ questions and answers. At the bottom, there's a 'Parameter presets' section.

Source: Author's screenshot from GCP environment

On the Dialogflow Console, click on ‘Start Page’. It will open the data store settings on the right. You can select multiple data stores (website, pdfs, etc. at the same time). In this case, leave it to be the student handbook that we created in Step 3.

Find ‘Generator’ under data store settings.

The screenshot shows the 'Generator' settings within the 'Data stores' panel of the Dialogflow Start Page configuration. The 'Display name' is set to 'student assistant'. The 'Text prompt' field contains the following text: 'You are a virtual assistant who works at The Best College. You will provide general advice to the student based on the student handbook. Always respond politely and with constructive language.' Below this, there's a 'Model configuration' section with a note about customizing the model, a 'Model' dropdown set to 'text-bison (latest)', and a 'Temperature' slider.

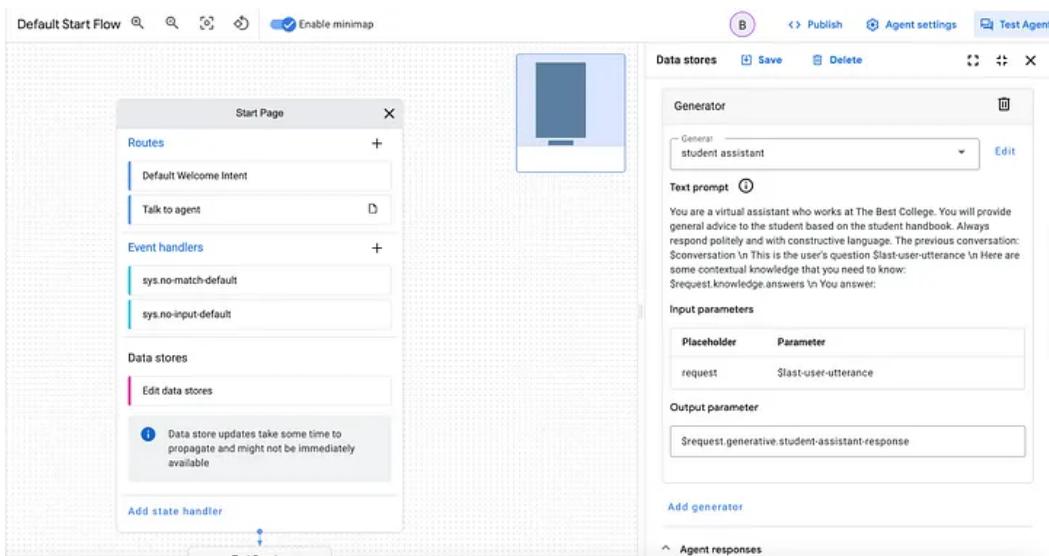
Source: Author's screenshot from GCP environment

Add the following prompt:

The screenshot shows the generated text prompt in the 'Text prompt' field of the 'Model configuration' section. The text is as follows:

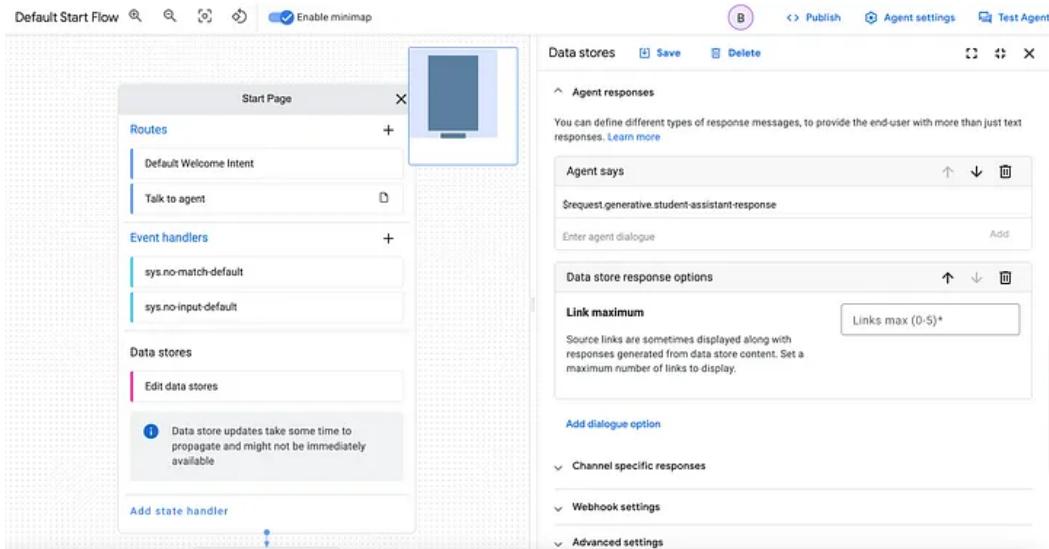
```
You are a virtual assistant who works at The Best College. You will provide gene
The previous conversation: $conversation \n
This is the user's question $last-user-utterance \n
Here are some contextual knowledge that you need to know: $request.knowledge.ans
You answer:
```

\$conversation, \$last-user-utterance, \$request.knowledge.answers are Dialogflow parameters.



Source: Author's screenshot from GCP environment

In the ‘Input Parameter’, type in \$last-user-utterance. This will pass the user’s question to the generator. In the ‘Output Parameter’, type in \$request.generative.student-assistant-response. Always remember to hit ‘Save’ button on the top!



Source: Author's screenshot from GCP environment

Under Agent says, don’t forget to type in \$request.generative.student-assistant-response. This will make sure the output from the generator gets picked up in the agent chat.

Step 5 – Testing & Integration:

To test this bot, you can either click on the ‘Test Agent’ simulator in the top right corner or use the Manage Tab on the Left to pick an integration. Personally, I prefer using ‘Dialogflow Messenger’ to ‘Test Agent’, as it shows me the final response format.

The screenshot shows the GCP console's 'Integrations' section under the 'Manage' tab. On the left sidebar, 'Integrations' is selected. The main area displays a grid of integration cards. The 'Text Based' section includes cards for Dialogflow Messenger, Messenger, LINE, Google Chat, Slack, MMD Smart, Twilio (Text Messaging), Discord, Spark (Webex), Telegram, Viber, and Azure Bot Service/Skype. Each card has a 'Connect' button below it.

Source: Author's screenshot from GCP environment

Once you click Dialogflow Messenger, choose ‘enable the unauthenticated API’. This is just for testing. In the real case scenario, depending on your authentication requirements, you can choose whether you want to enable this.

This screenshot shows the configuration dialog for connecting Dialogflow Messenger. It includes fields for selecting an environment (set to 'Draft'), choosing an API type ('Unauthenticated API (anonymous access)' is selected), choosing a UI style ('Pop-out' is selected), and a 'Restrict domain access' dropdown. At the bottom are 'Enable the unauthenticated API' and 'Done' buttons.

Source: Author's screenshot from GCP environment

The screenshot shows the Dialogflow Messenger configuration page in the GCP console. A modal window is open, displaying the generated JavaScript code for embedding the messenger on a website. The code includes properties like project ID, agent ID, language code, and chat title. Below the code, it says "Unauthenticated API for Dialogflow Messenger is enabled". At the bottom of the modal are buttons for "Disable the unauthenticated API", "Try it now", "New session", and "Done". To the right of the modal, there's a sidebar with tabs for "Simulator", "Test agent", "Environment", "Flow", and "Page". The "Page" tab is selected, showing a "Default Start" page with contact information for "BCC Institute". The "Flow" tab shows a "Default Start" flow. At the bottom right, there's a message input field with "Ask something..." and a send button.

Source: Author's screenshot from GCP environment

Now, you can type in the questions you have. The answers generated by RAG will be based on the student handbook pdf. It also comes with a reference link at the bottom. If you click on it, it takes you to the particular pdf page that contains the information.

Artificial Intelligence

Llm

Chatbots

Google

Machine Learning



Written by Euclidean AI

3 Followers · Writer for Towards AI

Purpose-built AI Solutions (www.euclideanai.com)

Follow



More from Euclidean AI and Towards AI



 Euclidean AI in Towards AI

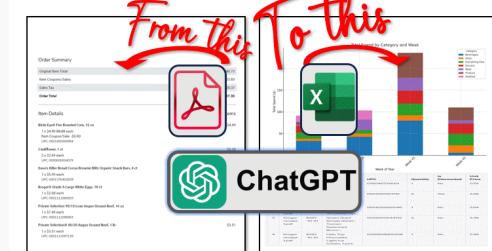
Building a Gen AI CV screener at DataRobot and AWS Hackathon...

This article describes a solution for a generative AI resume screener that got us 3rd...

6 min read · Nov 5

 5 



 David Leibowitz in Towards AI

ChatGPT As an OCR For PDFs: Your ETL Engine for Data Analysis

Coding in English at the speed of thought

 · 14 min read · Nov 3

 944 



 Kris Ograbek in Towards AI

AutoGen is Mindblowing: 4 Features that Make AutoGen the...

How to Build Your Custom AI Assistant Teams in Minutes.

 · 8 min read · Oct 14

 1.1K 



 Ignacio de Gregorio in Towards AI

France's New AI Champion Scares Silicon Valley

Mistral's New Model is a Box of Pleasant Surprises

 · 8 min read · Oct 28

 2.1K 

[See all from Euclidean AI](#)

[See all from Towards AI](#)

Recommended from Medium



 Madhukar Kumar in madhukarkumar

What does OpenAI's announcement mean for Retrieval...

In case you missed it , OpenAI made a series of announcements today. I plan to delve into...

3 min read · Nov 6

 191  6  



 Ozgur Guler

How to improve RAG performance ? —Advanced RAG Patterns—Part2

In the realm of experimental Large Language Models (LLMs), creating a captivating LLM...

11 min read · Oct 18

 40  1  

Lists



Natural Language Processing

842 stories · 392 saves



Predictive Modeling w/ Python

20 stories · 590 saves



AI Regulation

6 stories · 184 saves



Practical Guides to Machine Learning

10 stories · 670 saves



 Jerry Liu in LlamaIndex Blog

Multi-Modal RAG

(co-authored by Haotian Zhang, Laurie Voss, and Jerry Liu @ LlamaIndex)

6 min read · 3 days ago

 230  4  



 Geronimo

Finetuning Llama 2 and Mistral

A beginner's guide to finetuning SOTA LLMs with QLoRA

17 min read · Nov 5

 175  2  



 Baptiste Wlodarczyk in Google Cloud - Community

Generative AI Pricing : OpenAI vs Google Cloud

A case study of pricing models from leading Generative AI Platforms

9 min read · Jul 13

 244

 2



•••

 107

 1



•••

[See more recommendations](#)