Contents

#How To Build Your Own Custom ChatGPT With Custom Knowledge Base

Step-by-step guide on how to feed your ChatGPT bot with custom data sources Photo by Christian Wiediger on Unsplash

ChatGPT has become an integral tool that most people use daily to automate various tasks. If you have used ChatGPT for any length of time, you would have realized that it can provide wrong answers and has limited to zero context on some niche topics. This brings up the question of how we can tap into chatGPT to bridge the gap and allow ChatGPT to have more custom data.

A wealth of knowledge is distributed in various platforms we interact with daily, i.e., via confluence wiki pages at work, slack groups, company knowledge base, Reddit, Stack Overflow, books, newsletters, and google documents shared by colleagues. Keeping up with all these information sources is a full-time job in itself.

Wouldn't it be nice if you could selectively choose your data sources and feed that information into ChatGPT conversation with your data with ease? 1. Feeding Data via Prompt Engineering

Before we jump into how we can extend ChatGPT, let's look at how we could extend ChatGPT manually and what the issues are. The conventional approach to extending ChatGPT is via prompt engineering.

This is quite simple to do since ChatGPT is context aware. First, we need to interact with ChatGPT by appending the original document content before the actual questions.

I will ask you questions based on the following content:

- Start of Content- Your very long text to give ChatGPT context
- End of Content-

The issue with this approach is the model has a limited context; it can only accept approximately 4,097 tokens for GPT-3. You will soon run into a wall with this approach as it's also quite a manual, tedious process to always have to paste in the content.

Imagine having hundreds of PDF documents you wanted to inject into ChatGPT. You will soon run into paywall issues. You might be thinking GPT-4 is the successor to GPT-3. It was just launched on March 14, 2023, and it can process 25,000 words — about eight times as many as GPT-3 process images — and handle much more nuanced instructions than GPT-3.5. This still has the same fundamental problem of data input limitation. How do we go about bypassing some of these limitations? We can leverage a Python library called LlamaIndex.

2. Extending ChatGPT With LlamaIndex (GPT Index)

LlamaIndex, also known as the GPT Index, is a project that provides a central interface to connect your LLMs with external data. Yeah, you read that correctly. With LlamaIndex, we can build something that looks like the illustration below: Custom data sources feeding into ChatGPT

LlamaIndex connects your existing data sources and types with available data connectors, for example (APIs, PDFs, docs, SQL, etc.) It enables you to employ LLMs by offering indexes over your structured and unstructured data. These indices facilitate in-context learning by removing typical boilerplate and pain points: preserving context in an accessible manner for quick insertion.

Dealing with prompt restrictions — a 4,096 token limit for the GPT-3 Davinci and an 8,000 token limit for GPT-4 — when the context is too large becomes much more accessible and tackles the text-splitting issue by giving users a way to interact with the index. LlamaInde also abstracts the process of extracting relevant parts from the documents and feeding them to the prompt.

0.1 How To Add Custom Data Source

In this section, we'll use GPT "text-davinci-003" and LlamaIndex to create a Q&A chatbot based on pre-existing documents.

0.1.1 Prerequisites

Before we start, make sure you have access to the following:

- Python >= 3.7 installed on your machine
- An OpenAI API key, which can be found on the OpenAI website. You can use your Gmail account to single sign-on.
- A few Word documents uploaded to your Google Docs. LlamaIndex supports many different data sources. For this tutorial, we will demonstrate Google Docs.

0.1.2 How It Works

- Create a document data index with LlamaIndex.
- Use natural language to search the index.
- The pertinent pieces will be retrieved by LlamaIndex and passed to the GPT prompt. LlamaIndex will transform your original document data into a query-friendly vectorized index. It will utilize this index to find the most pertinent sections based on how closely the query and data match. The information will then be loaded into the prompt, which will be sent to GPT so that GPT has the background necessary to respond to your question.
- After that, you may ask ChatGPT, given the feed in context.

Create a new folder for your Python project, which you can call mychatbot, preferably using a virtual environment or conda environment.

We'll need to install the dependency libraries first. Here's how:

pip install openai pip install llama-index pip install google-auth-oauthlib

Next, we'll import the libraries in Python and set up your OpenAI API key in a new main.py file.

0.1.3 Import necessary packages

import os import pickle

from google.auth.transport.requests import Request

from google_auth_oauthlib.flow import InstalledAppFlow from llama_index import GPTSimpleVectorIndex, download_loader

os.environ['OPENAI_API_KEY'] = 'SET-YOUR-OPEN-AI-API-KEY'

In the above snippet, we are explicitly setting the environment variable for clarity, as the LlamaIndex package implicitly requires access to OpenAI. In a typical production environment, you can put your keys in environment variables, vault, or whatever secrets management service your infra can access.

Let's construct a function to help us authenticate against our Google account to discover Google Docs.

def authorize_gdocs(): google_oauth2_scopes = ["https://www.googleapis.com/auth/documents.readonly"] cred = None if os.path.exists("token.pickle"): with open("token.pickle", 'rb') as token: cred = pickle.load(token) if not cred or not cred.valid: if cred and cred.expired and cred.refresh_token: cred.refresh(Request()) else: flow = InstalledAppFlow.from_client_secrets_file("credentials.json", google_oauth2_scopes) cred = flow.run_local_server(port=0) with open("token.pickle", 'wb') as token: pickle.dump(cred, token)

To enable the Google Docs API and fetch the credentials in the Google Console, you can follow these steps:

- Go to the Google Cloud Console website (console.cloud.google.com).

- Create a new project if you haven't already. You can do this by clicking on the "Select a project" dropdown menu in the top navigation bar and selecting "New Project." Follow the prompts to give your project a name and select the organization you want to associate it with.
- Once your project is created, please select it from the dropdown menu in the top navigation bar.
- Go to the "APIs & Services" section from the left-hand menu and click on the "+ ENABLE APIS AND SERVICES" button at the top of the page.
- Search for "Google Docs API" in the search bar and select it from the results list.
- Click the "Enable" button to enable the API for your project.
- Click on the OAuth consent screen menu and create and give your app a name, e.g., "mychatbot," then enter the support email, save, and add scopes.

You must also add test users since this Google app will not be approved yet. This can be your own email.

You will then need to set up credentials for your project to use the API. To do this, go to the "Credentials" section from the left-hand menu and click "Create Credentials." Select "OAuth client ID" and follow the prompts to set up your credentials.

Once your credentials are set up, you can download the JSON file and store it in the root of your application, as illustrated below: Example folder structure with google credentials in root

Once you have set up your credentials, you can access the Google Docs API from your Python project.

Go to your Google Docs, open up a few of them, and get the unique id that can be seen in your browser URL bar, as illustrated below: Gdoc ID

Copy out the gdoc IDs and paste them into your code below. You can have N number of gdocs that you can index so ChatGPT has context access to your custom knowledge base. We will use the GoogleDocsReader plugin from the LlamaIndex library to load your documents.

::: # function to authorize or download latest credentials authorize_gdocs()

1 initialize LlamaIndex google doc reader

GoogleDocsReader = download_loader('GoogleDocsReader')

2 list of google docs we want to index

gdoc_ids = ['1ofZ96nWEZYCJsteRfqik_xNQTGFHtnc-7cYrf0dMPKQ']

loader = GoogleDocsReader()

3 load gdocs and index them

documents = loader.load_data(document_ids=gdoc_ids) index = GPTSimpleVectorIndex(documents) :::

LlamaIndex has a variety of data connectors covering services such as Notion, Obsidian, Reddit, Slack, etc. You can find the compressive list of available data connectors here.

If you wish to save and load the index on the fly, you can use the following function calls. This will speed up the process of fetching from pre-saved indexes instead of making API calls to external sources.

::: # Save your index to a index.json file index.save_to_disk('index.json') # Load the index from your saved index.json file index = GPTSimpleVectorIndex.load_from_disk('index.json') :::

Querying the index and getting a response can be achieved by running the following code below. Code can easily be extended into a rest API that connects to a UI where you can interact with your custom data sources via the GPT interface.

::: # Querying the index while True: prompt = input("Type prompt...") response = index.query(prompt) print(response) :::

Given we have a Google Doc with details about me, information that's readily available if you publicly search on google.

We will interact directly with vanilla ChatGPT first to see what output it generates without injecting a custom data source.

That was a little disappointing! Let's try again.

::: INFO:google_auth_oauthlib.flow:"GET /?state=oz9XY8CE3LaLLsTxIz4sDgrHha4fEJ&code=4/0AWtgzh4LlIfmCMEa0t36
GiZ___Wg-hBBg&scope=https://www.googleapis.com/auth/documents.readonly HTTP/1.1" 200 65
INFO:googleapiclient.discovery_cache:file_cache is only supported with oauth2client<4.0.0 INFO:root:>
[build_index_from_documents] Total LLM token usage: 0 tokens INFO:root:> [build_index_from_documents]
Total embedding token usage: 175 tokens Type prompt... who is timothy mugayi hint he is a writer on medium

INFO:root:> [query] Total LLM token usage: 300 tokens INFO:root:> [query] Total embedding token usage: 14
tokens Timothy Mugayi is an Engineering Manager at OVO (PT Visionet Internasional), a subsidiary of GRAB.
He is also an avid writer on medium.com who writes on technical topics covering python and freelancing side
hustling for programmers. Timothy has been coding for over 15 years, building enterprise solutions for large
cooperations. During his free time, he enjoys mentoring and coaching. last_token_usage=300 Type prompt...

Type prompt... Given you know who timothy mugayi is write an interesting introduction about him

Timothy Mugayi is an experienced and accomplished professional with a wealth of knowledge in engineering,
coding, and mentoring. He is currently an Engineering Manager at OVO, a subsidiary of GRAB, and has been
coding for over 15 years, building enterprise solutions for large cooperations. In his free time, Timothy enjoys
writing on technical topics such as Python and freelancing side hustling for programmers on medium.com, as
well as mentoring and coaching. With his impressive background and expertise, Timothy is a valuable asset to
any organization. last_token_usage=330 :::

It can now infer answers using a new custom data source, accurately producing the following output.

We can take things further.

::: Type prompt... Write a cover letter for timothy mugayi for an upwork python project to build a custom
ChatGPT bot with access to external data sources INFO:root:> [query] Total LLM token usage: 436 tokens
INFO:root:> [query] Total embedding token usage: 30 tokens

Dear [Hiring Manager],

I am writing to apply for the Python project to build a custom ChatGPT bot with access to external data
sources. With over 15 years of experience in coding and building enterprise solutions for large corporations, I am
confident that I am the ideal candidate for this position.

I am currently an Engineering Manager at OVO (PT Visionet Internasional), a subsidiary of GRAB. I have
extensive experience in Python and have been writing on technical topics covering Python and freelancing side
hustling for programmers on medium.com. I am also an avid mentor and coach, and I believe that my experience
and skillset make me the perfect candidate for this project.

I am confident that I can deliver a high-quality product that meets the requirements of the project. I am also
available to discuss the project further and answer any questions you may have.

Thank you for your time and consideration.

Sincerely, Timothy Mugayi last_token_usage=436 Type prompt... :::

LlamaIndex will internally accept your prompt, search the index for pertinent chunks, and then pass both your
prompt and the pertinent chunks to the ChatGPT model. The procedures above demonstrate a fundamental
first use of LlamaIndex and GPT for answering questions. Yet, there is much more you can do. You are only
limited by your creativity when configuring LlamaIndex to utilize a different large language model (LLM), using
a different type of index for various activities, or updating old indices with a new index programmatically.

Here is an example of changing the LLM model explicitly. This time we tap into another Python package that
comes bundled with LlamaIndex called langchain.

::: from langchain import OpenAI from llama_index import LLMPredictor, GPTSimpleVectorIndex,
PromptHelper

...

4 define anoter LLM explicitly

llm_predictor = LLMPredictor(llm=OpenAI(temperature=0, model_name="text-davinci-003"))

5 define prompt configuraiton

6 set maximum input size

max_input_size = 4096 # set number of output tokens num_output = 256 # set maximum chunk overlap max_chunk_overlap = 20 prompt_helper = PromptHelper(max_input_size, num_output, max_chunk_overlap)

index = GPTSimpleVectorIndex(documents, llm_predictor=llm_predictor, prompt_helper=prompt_helper) :::

If you want to keep tabs on your OpenAI free or paid credits, you can navigate to the OpenAI dashboard and check how much credit is left.

Creating an index, inserting into an index, and querying an index will use tokens. Hence, it's always important to ensure you output token usage for tracking purposes when building your custom bots.

::: last_token_usage = index.llm_predictor.last_token_usage

print(f"last_token_usage={last_token_usage}") :::

Final Thoughts

ChatGPT combined with LlamaIndex can help to build a customized ChatGPT chatbot that can infer knowledge based on its own document sources. While ChatGPT and other LLM are pretty powerful, extending the LLM model provides a much more refined experience and opens up the possibility of building a conversational-style chatbot that can be used to build real business use cases like customer support assistance or even spam classifiers. Given we can feed real-time data, we can evaluate some of the limitations of ChatGPT models being trained up to a certain period.

For the complete source code, you can refer to this GitHub repo.

`If you are looking to build custom ChatGPT bots that understand your domain, drop a message in the comm`