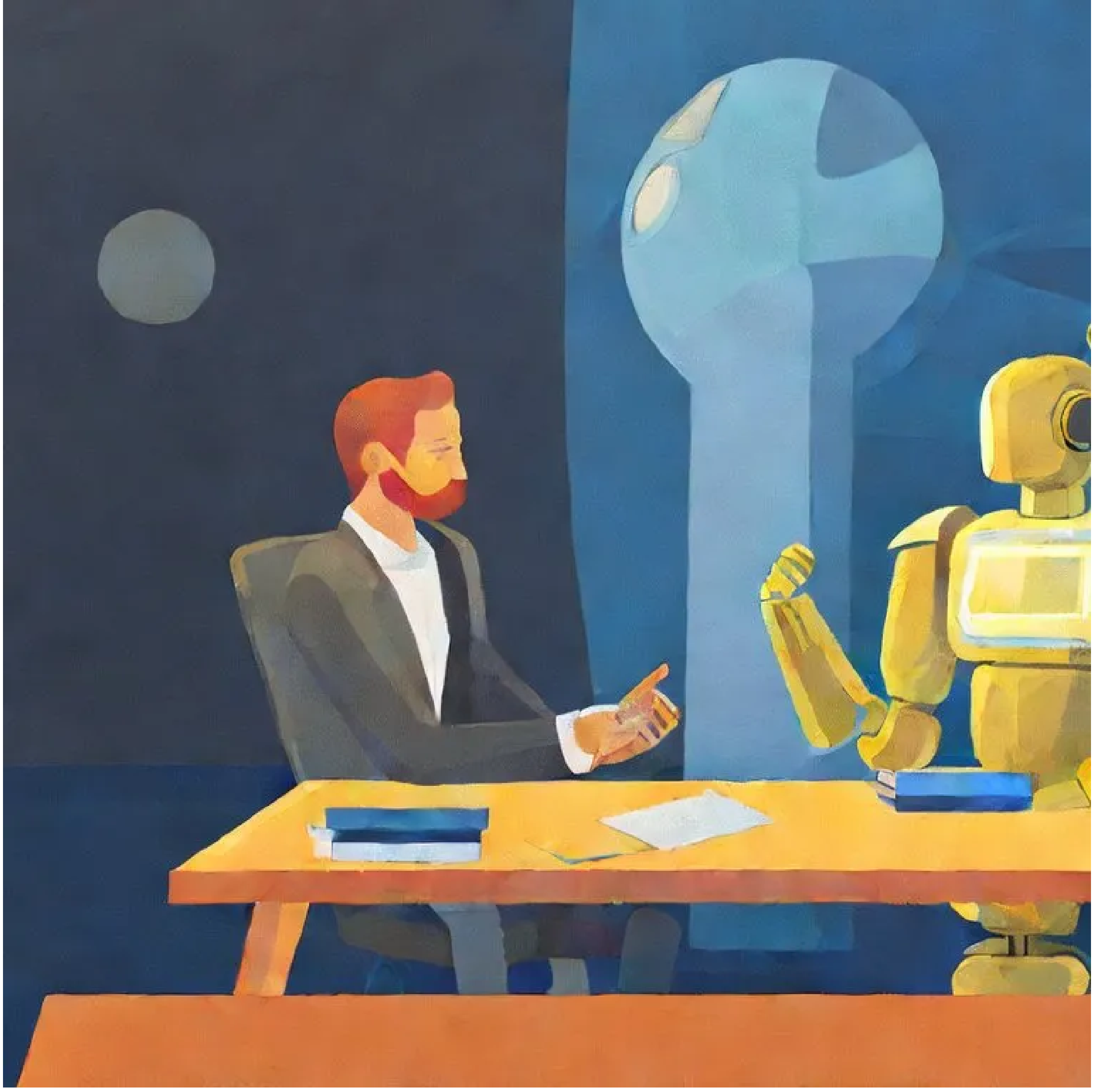
Published in Bootcamp

Guodong (Troy) Zhao Following
Mar 11 · 6 min read · ✦ Member-only · ▶ Listen

A step-by-step guide to building a chatbot based on your own documents with GPT

Chatting with ChatGPT is fun and informative — I've been chit-chatting with it for past time and exploring some new ideas to learn. But these are more casual use cases and the novelty can quickly wean off, especially when you realize that it can generate hallucinations.

How might we use it in a more productive way? With the recent release of the GPT 3.5 series API by OpenAI, we can do much more than just chit-chatting. One very productive use case for businesses and your personal use is QA (Question Answering) — **you ask the bot in natural language about your own documents/data, and it can quickly answer you by retrieving info from the documents and generating a response** [1]. You can use it for customer support, synthesizing user research, your personal knowledge management, and more!



Ask a bot for document-related questions. Image generated with Stable Diffusion.

In this article, I will explore how to build your own Q&A chatbot based on your own data, including why some approaches won't work, and a step-by-step guide for building a document Q&A chatbot in an efficient way with llama-index and GPT API.

(If you only want to know how to build the Q&A chatbot, you can jump directly to the section "Building document Q&A chatbot step-by-step")

Exploring different approaches

My day job is as a product manager — reading customer feedback and internal documents takes a big chunk of my life. When ChatGPT came out, I immediately thought of the idea of using it as an assistant to help me synthesize customer feedback or find related old product documents about the feature I'm working on.

I first thought of fine-tuning the GPT model with my own data to achieve the goal. But fine-tuning costs quite some money and requires a big dataset with examples. It's also impossible to fine-tune every time when there is a change to the document. An even more crucial point is that fine-tuning simply

CANNOT let the model “know” all the information in the documents, but rather it teaches the model a new skill. Therefore, for (multi-)document QA, fine-tuning is not the way to go.

The second approach that comes into my mind is prompt engineering by providing the context in the prompts. For example, instead of asking the question directly, I can append the original document content before the actual question. But the GPT model has a limited attention span — it can only take in a few thousand words in the prompt (about 4000 tokens or 3000 words). It’s impossible to give it all the context in the prompt, provided that we have thousands of customer feedback emails and hundreds of product documents. It’s also costly if you pass in a long context to the API because the pricing is based on the number of tokens you use.

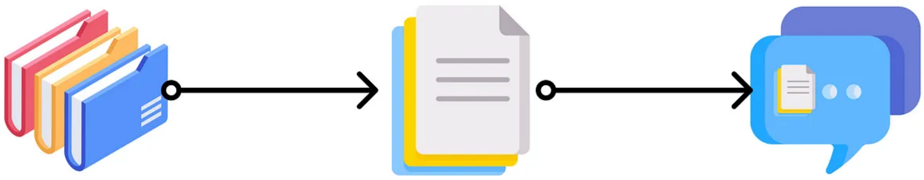
```
I will ask you questions based on the following context:
– Start of Context –

YOUR DOCUMENT CONTENT

– End of Context–
My question is: “What features do users want to see in the app?”
```

(If you want to learn more about fine-tuning and prompt engineering for GPT, you can read the article: https://medium.com/design-bootcamp/3-ways-to-tailor-foundation-language-models-like-gpt-for-your-business-e68530a763bd)

Because the prompt has limitations on the number of input tokens, I came up with the idea of first using an algorithm to search the documents and pick out the relevant excerpts and then passing only these relevant contexts to the GPT model with my questions. While I was researching this idea, I came across a library called gpt-index (now renamed to LlamaIndex), which does exactly what I wanted to do and it is straightforward to use [2].



Extract relevant parts from the documents and then feed them to the prompt. Icons from https://www.flaticon.com/

In the next section, I'll give a step-by-step tutorial on using LlamaIndex and GPT to build a Q&A chatbot on your own data.

Building document Q&A chatbot step-by-step

In this section, we will build a Q&A chatbot based on existing documents with LlamaIndex and GPT (text-davinci-003), so that you can ask questions about your document and get an answer from the chatbot, all in natural language.

Prerequisites

Before we start the tutorial, we need to prepare a few things:

- Your OpenAI API Key, which can be found at https://platform.openai.com/account/api-keys.
- A database of your documents. LlamaIndex supports many different data sources like Notion, Google Docs, Asana, etc [3]. For this tutorial, we'll just use a simple text file for demonstration.
- A local Python environment or an online Google Colab notebook.

Workflow

The workflow is straightforward and takes only a few steps:

1. Build an index of your document data with LlamaIndex
2. Query the index with natural language
3. LlamaIndex will retrieve the relevant parts and pass them to the GPT prompt
4. Ask GPT with the relevant context and construct a response

What LlamaIndex does is convert your original document data into a vectorized index, which is very efficient to query. It will use this index to find the most relevant parts based on the similarity of the query and the data. Then, it will plug in what's retrieved into the prompt it will send to GPT so that GPT has the context for answering your question.

Setting Up

We'll need to install the libraries first. Simply run the following command in your terminal or on Google Colab notebook. These commands will install both LlamaIndex and OpenAI.

```
!pip install llama-index
!pip install openai
```

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

```
# Import necessary packages
from llama_index import GPTSimpleVectorIndex, Document, SimpleDirectoryReader
import os

os.environ['OPENAI_API_KEY'] = 'sk-YOUR-API-KEY'
```

Constructing the index and saving it

After we installed the required libraries and import them, we will need to construct an index of your document.

To load your document, you can use the SimpleDirectoryReader method provided by LllamaIndex or you can load it from strings.

```
# Loading from a directory
documents = SimpleDirectoryReader('your_directory').load_data()

# Loading from strings, assuming you saved your data to strings text1, text2, ..
text_list = [text1, text2, ...]
```

LlamaIndex also provides a variety of data connectors, including Notion, Asana, Google Drive, Obsidian, etc. You can find the available data connectors at https://llamahub.ai/.

After loading the documents, we can then construct the index simply with

```
# Construct a simple vector index
index = GPTSimpleVectorIndex(documents)
```

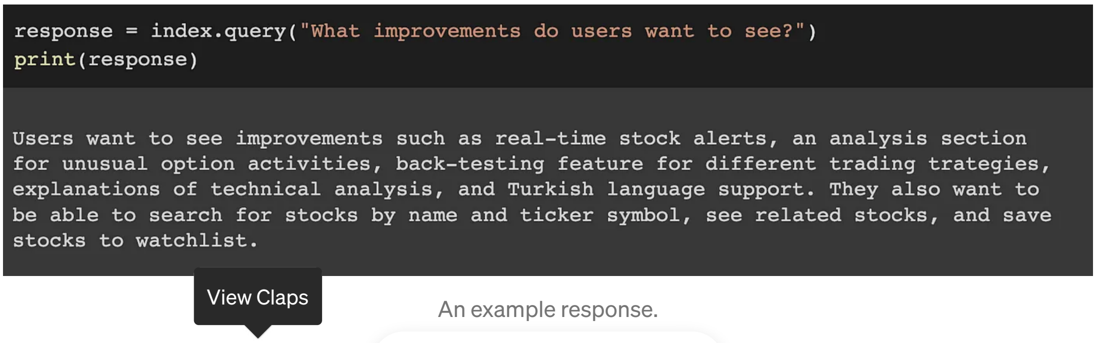
If you want to save the index and load it for future use, you can use the following methods

```
# 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

Querying the index is simple

```
# Querying the index
response = index.query("What features do users want to see in the app?")
print(response)
```



```
response = index.query("What improvements do users want to see?")
print(response)
```

```
Users want to see improvements such as real-time stock alerts, an analysis section
for unusual option activities, back-testing feature for different trading trategies,
explanations of technical analysis, and Turkish language support. They also want to
be able to search for stocks by name and ticker symbol, see related stocks, and save
stocks to watchlist.
```

An example response.

And voilà! You will get your answer printed. Under the hood, LlamaIndex will take your prompt, search for relevant chunks in the index, and pass your prompt and the relevant chunks to GPT.

Some notes for advanced usage

The steps above show only a very simple starter usage for question answering with LlamaIndex and GPT. But you can do much more than that. In fact, you can configure LlamaIndex to use a different large language model (LLM), use a different type of index for different tasks, update existing indices with a new index, etc. If you're interested, you can read their doc at https://gpt-index.readthedocs.io/en/latest/index.html.

Some final words

In this post, we've seen how to use GPT in combination with LlamaIndex to build a document question-answering chatbot. While GPT (and other LLM) is powerful in itself, its powers can be much amplified if we combine it with other tools, data, or processes.

What would you use a document question-answering chatbot for?

(For more advanced usages of GPT, you can read: Advanced Skills in GPT for your product/business beyond simple chats; for solving the problems while implementing GPT/LLM-backed apps like message history/memory/caching/scaling, you can read Cookbook for solving common problems in building GPT/LLM apps)

References:

[1] *What Is Question Answering? — Hugging Face*. 5 Dec. 2022, https://huggingface.co/tasks/question-answering.

[2] Liu, Jerry. *LlamaIndex*. Nov. 2022. GitHub, https://github.com/jerryjliu/gpt_index.

AI Technology ChatGPT Artificial Intelligence Product Management