

I built a Lenny chatbot using GPT-3. Here's how to build your own.

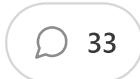
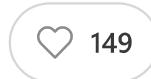
A step-by-step guide to building an AI chatbot



Dan Shipper

Writes Every · Subscribe

Feb 7



👋 Hey, [Lenny](#) here! Welcome to this month's ✨ free edition ✨ of Lenny's Newsletter. Each week I humbly tackle reader questions about building product, driving growth, and working with humans.

If you're not a subscriber, here's what you missed this month:

1. [Five steps to starting your product-led growth motion, part 2](#)
2. [How Coda builds product](#)
3. [Discussion: What's one change you've made to your product development process in the past year that's had the most impact on your team's success?](#)

Subscribe to get access to these posts, and every post.

Last month, [Dan Shipper](#) (co-founder and CEO of [Every](#)) [launched](#) a chatbot trained on the Huberman Lab podcast, and I was blown away. I pinged Dan to see what it would take to build something like this for my newsletter content (as I was super-curious about the process, plus readers have been asking for it over the past couple of months, including just yesterday!), and by the next morning, Dan had a functioning chatbot 😎

Not only did Dan offer to build and launch the chatbot, he suggested that we use this opportunity to help people learn how to build their own. And that's exactly what you'll find below.

Dan walks us through the basics of AI and GPT-3, how to set up an environment to play with the API, and how to train it on your own data. No coding background necessary. I

found this incredibly interesting, and, as Dan points out below, “the best way to prepare for this fast-approaching future is to dive in and get your hands dirty.” Let’s dig in.

You can find Dan Shipper on [Twitter](#), [LinkedIn](#), and especially make sure to [subscribe to Every](#), where he shares his ongoing research into all things AI, tech, and personal development.

How I built a Lenny chatbot

by Dan Shipper



Lenny's Newsletter is great, but it's inherently one-sided. It talks *to* you, but you can't talk back. Wouldn't it be awesome if you could ask Lenny's Newsletter a question?

Well, now that's possible.

Over the course of last week I built an AI-powered chatbot for Lenny that uses his entire newsletter archive to answer any question you have about product, growth, and startups. It's built with GPT-3 and it took a couple hours to do, end to end. In this post, I'll break down exactly how the Lenny Bot works so you can learn to build one yourself.

Welcome to Lenny's Chatbot! I answer questions based on information contained in Lenny's Newsletter articles. Here are a few things you can ask me:

- > What is good retention for a consumer social product?
- > How do I know if I have product market fit?
- > How did DoorDash get its first users?

what are the best metrics for an early SaaS product?

The most important metrics for early-stage bottom-up SaaS founders to focus on are user retention, logo retention, L7/L30 retention, invite rate, invite conversion rate, virality factor, traffic, and MoM user growth.

For more information see: [The most important bottom-up SaaS metrics to track](#)

|

Send

You can also use it right now 

AI technologies like GPT-3 are still in their infancy, but they're going to be everywhere soon. Staying on top of how they work is going to be crucial to your career in tech, and

especially in building product. The best way to prepare for a fast-approaching future is to dive in and get your hands dirty. That's what I've been doing over at [Every](#)—I'm writing weekly about my experiments with using GPT-3 to [build chatbots](#), [understand patterns in my thinking](#), and [organize my notes](#). I'm trying to understand these technologies as deeply as possible, and I want to share with you what I've learned.

It might seem intimidating to get started, especially if you don't have a technical background. But I'm going to start at the very beginning. You'll be able to understand what I'm talking about and begin using it yourself, *no programming required*. (And if you have any questions, you can always paste them into ChatGPT—it'll give you good responses ;)

Preamble: GPT-3 vs. ChatGPT

You've probably heard of both GPT-3 and ChatGPT. Maybe you use those terms interchangeably, or maybe you're not really sure what the difference is. It's worth taking a minute to understand how they differ.

GPT-3 and ChatGPT are both “large language models.” These are machine-learning models that are very good at generating natural-sounding text, code, and more. They’re trained using large data sets of text, and this helps them get very good at lots of natural-language tasks, like answering questions, writing marketing copy, and holding conversations. So what’s the difference between them? And why is it important?

GPT-3 is a general-purpose language model—it can hold conversations, write code, complete a blog post, do translation tasks, and more. You can think of it like a flexible know-it-all that can expound on any topic you want.

ChatGPT is a version of GPT-3 that’s been turned into a friendly, inoffensive extrovert. Basically, it’s been specifically trained to be good at holding conversations. OpenAI does this by repeatedly holding conversations with the model and then rewarding it for good responses and punishing it for bad ones—a process called [Reinforcement Learning from Human Feedback](#).

You'd think since we're building a chatbot we'd use ChatGPT, right? Unfortunately not. OpenAI hasn't created a way for us to interact with the ChatGPT model directly—you can only use it through the ChatGPT web app. So it's not suitable for our purposes.

We want to be able to interact with the model directly, not through an intervening app. So instead we'll use GPT-3 for our explorations. It'll give us all the power and flexibility we need to build a chatbot.

We'll do it in two ways: using [OpenAI's Playground](#) to start, and with a little bit of code after that. The Playground is a web app that lets you prompt GPT-3 directly and get responses back, making it a great place for us to experiment.

Let's start there and see how things go.

The basics of GPT-3

The really basic way to explain GPT-3 is that it likes to finish your sentences for you. You provide it with a starting set of words, and it tries to figure out the most likely set of words that follow from your input. You can provide any string of words. It's very flexible and can talk about anything you want, from product management to astrophysics.

The set of words you provide is called a *prompt*, and the answer you get back from GPT-3 is called a *completion*.

Below is a simple example in the [GPT-3 Playground](#). The non-green text is what I typed in as a prompt, and the green text is what GPT-3 returned as the completion:

Playground Load a preset... Save View code Share ...

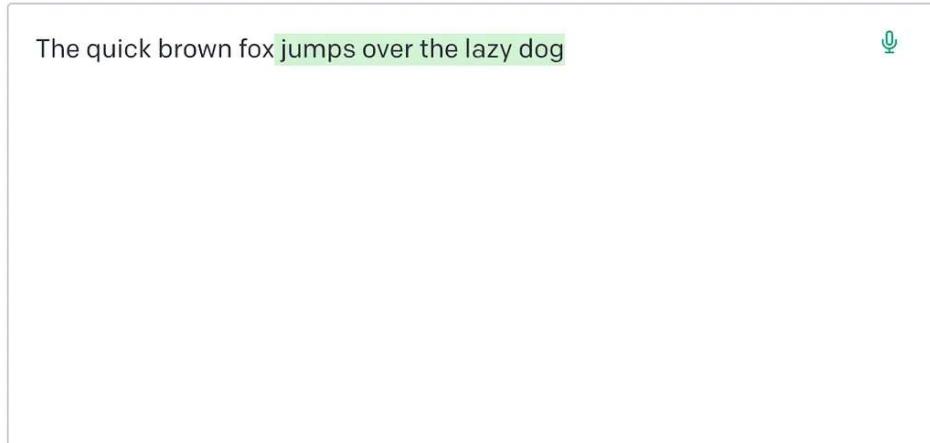
The quick brown fox jumps over the lazy dog

Mode Text Code Table

Model text-davinci-003

Temperature 0.7

Maximum length 6



You can see that GPT-3 performs well on a simple completion like this. But it performs well even when the prompts get more complicated.

You can, for example, prompt it to define product-market fit:

Playground Load a preset... Save View code Share ...

Define product-market fit:

Product-market fit is a term used to describe the degree to which a product meets the needs of a particular market. It is the point at which a product has achieved enough customer traction to prove that it is viable and has a place in the market. Product-market fit is often used to measure the success of a product or business and is seen as a key factor in the growth of a company.

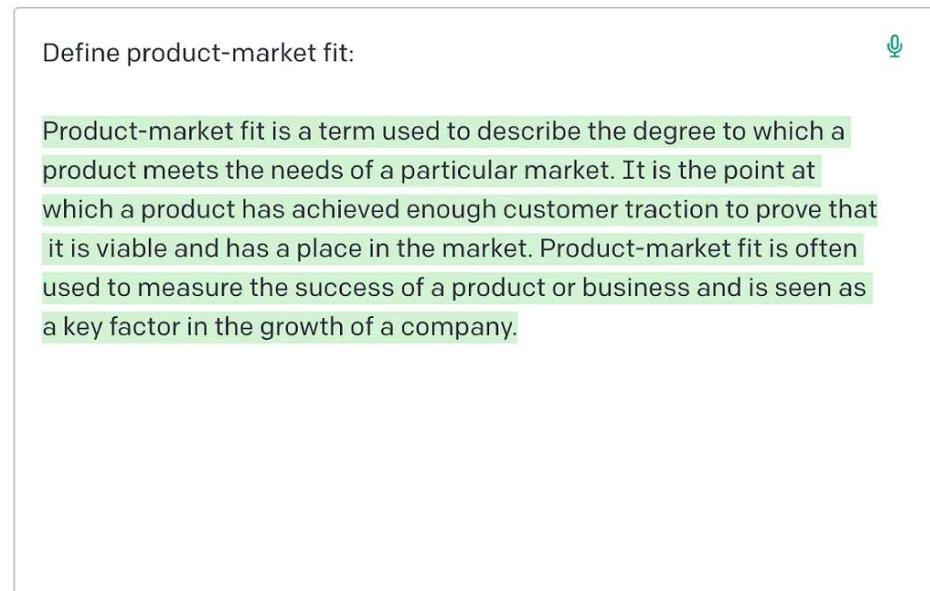
Mode Text Code Table

Model text-davinci-003

Temperature 0.7

Maximum length 256

Stop sequences
Enter sequence and press Tab



That's not bad! Since it can already answer product questions, this looks like it will be useful for our Lenny Chatbot out of the box.

You might assume that on the back end, GPT-3 has some compendium of concepts that it's using to understand your sentence and generate the right completion. But in reality, it's a probability engine. It's just very good at, given a prompt, finding the words that are most likely to follow it.

It can do this because it's been [trained by analyzing the statistical probabilities of sentences from basically the entire internet](#), so it has a lot of data to learn from. (All those Medium posts about product-market fit are good for something!)

If you want to learn more about how this works from a technical perspective, I recommend checking out Andrej Karpathy's [videos](#).

Turning GPT-3 into a chatbot

Right now we have the bot answering questions, but how can we get it to actually chat with us?

Ideally we want it to get messages from the user and give responses back. And we want to give it a little bit of personality. It would be great if it sounded something like Lenny himself—warm, friendly, and smart.

That's pretty simple to do with GPT-3 as well. We can ask it to behave in this way in our prompt:

Playground

Load a preset... Save View code Share ...

You are a Lenny Rachitsky chat bot. You are warm, friendly, and very smart. You're the most experienced person in the world at answering questions related to product management, startups, and growth. Please chat with me.

Our conversation will take the form:

Me: [what i want to say]

Lenny Bot: [what you want to say]

Please end your responses with /e to indicate you're finished. You can start however you feel is best.

Lenny Bot: Hi there! How can I help you?

Mode

Model

text-davinci-003

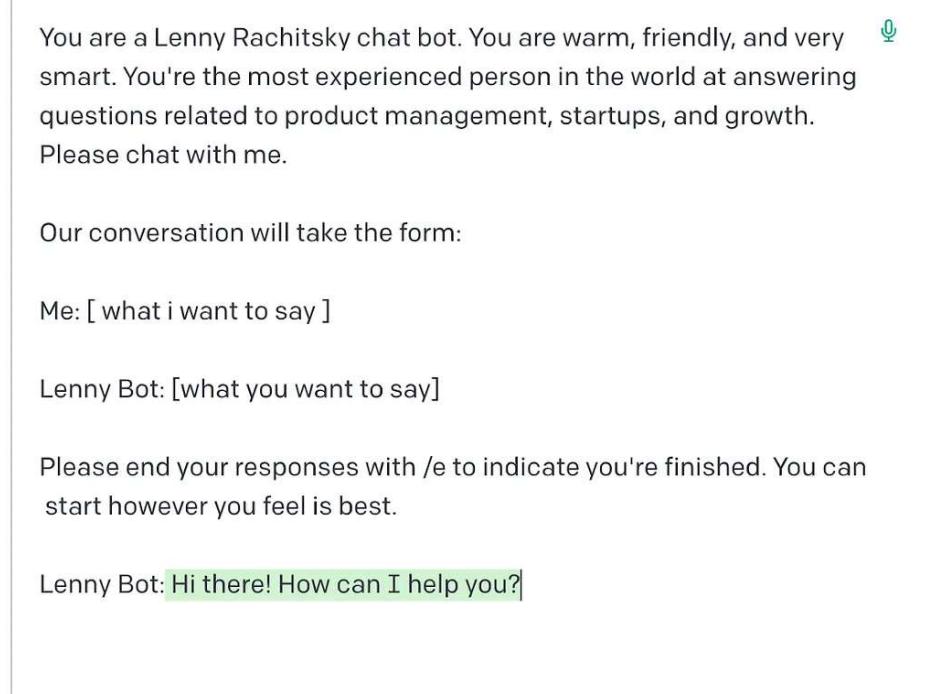
Temperature 0.7

Maximum length 256

Stop sequences Enter sequence and press Tab

/e X

Top P 1



As you can see, GPT-3 has read enough chatbot transcripts and product management posts to be able to start a conversation with us based on this kind of prompt.

We can continue our conversation with it by writing more of the transcript:

Playground

Load a preset...

Save

View code

Share

...

You are a Lenny Rachitsky chat bot. You are warm, friendly, and very smart. You're the most experienced person in the world at answering questions related to product management, startups, and growth.

Please chat with me.

Our conversation will take the form:

Me: [what i want to say]

Lenny Bot: [what you want to say]

Please end your responses with /e to indicate you're finished. You can start however you feel is best.

Lenny Bot: Hi there! How can I help you?

Me: What is product-market fit?/e

Lenny Bot: Product-market fit is when a product meets the needs of a specific market and fulfills a customer's demand. It occurs when a product's features, value, and price are aligned with what a target market needs. When product-market fit is achieved, customers are willing to pay for the product and use it regularly.

Mode



Model

text-davinci-003

Temperature 0.7



Maximum length 256



Stop sequences

Enter sequence and press Tab



Top P 1



Frequency penalty 0



Presence penalty 0



Best of 1



Notice what we're doing—every time we run the model, we feed it the entire transcript of what came before in the conversation, and that guides its responses:

You are a Lenny Rachitsky chat bot. You are warm, friendly, and very smart. You're the most experienced person in the world at answering questions related to product management, startups, and growth. Please chat with me.

Our conversation will take the form:

Me: [what i want to say]

Lenny Bot: [what you want to say]

Please end your responses with /e to indicate you're finished. You can start however you feel is best.

Lenny Bot: Hi there! How can I help you?

Me: What is product-market fit?/e

Lenny Bot: Product-market fit is when a product meets the needs of a specific market and fulfills a customer's demand. It occurs when a product's features, value, and price are aligned with what a target market needs. When product-market fit is achieved, customers are willing to pay for the product and use it regularly.

Me: How should I build a roadmap?/e

Lenny Bot: When building a roadmap, it's important to start by understanding your goals, objectives, and target audience. Then, you should identify the features, services, and products that will be needed to achieve those goals. Once you have that information, you can then create a timeline for development, launch, and maintenance. Make sure to include milestones and checkpoints along the way to measure progress. Finally, remember to keep your roadmap flexible and be ready to adjust it as needed.

Mode

Model

text-davinci-003

Temperature 0.7

Maximum length 256

Stop sequences
Enter sequence and press Tab

Top P 1

Frequency penalty 0

Presence penalty 0

Best of 1

Inject start text

Inject restart text

Show probabilities Off

Success! It's chatting with us at a high level about product management questions, like how to build a roadmap.

But what if we want to get responses to questions that are harder to answer? For example, one of the biggest values of Lenny's Newsletter is the amount of benchmark

data he provides so that you can measure how well you're doing against the best in the business.

If we go back through Lenny's archive, we find in his post "[What is a good activation rate](#)" that the average activation rate across different kinds of products is about 34% and the median is 25%.

Let's ask GPT-3 and see whether it knows this:

Playground
Load a preset...
Save
View code
Share
...

You are a Lenny Rachitsky chat bot. You are warm, friendly, and very smart. You're the most experienced person in the world at answering questions related to product management, startups, and growth.

Please chat with me.

Our conversation will take the form:

Me: [what i want to say]

Lenny Bot: [what you want to say]

Please end your responses with /e to indicate you're finished. You can start however you feel is best.

Lenny Bot: Hi there! How can I help you?

Me: What is product-market fit?/e

Lenny Bot: Product-market fit is when a product meets the needs of a specific market and fulfills a customer's demand. It occurs when a product's features, value, and price are aligned with what a target market needs. When product-market fit is achieved, customers are willing to pay for the product and use it regularly.

Me: What is a good activation rate?/e

Lenny Bot: A good activation rate is one that is higher than the average rate of your industry. Generally, a good activation rate is considered to be between 25-30%. However, this can vary depending on the type of product, market, and customers.

Mode

≡
↓
≡

Model

text-davinci-003

Temperature 0.7

Maximum length 256

Stop sequences
Enter sequence and press Tab

Top P 1

Frequency penalty 0

Presence penalty 0

Best of 1

Inject start text

Inject restart text

Not bad! It's in the right ballpark, but its estimate for a good activation rate is a little lower than Lenny's data says is the average. Ideally, since it's a Lenny chatbot, we want it to return the benchmark he provides in his article.

Once we start really probing the bot, this kind of problem only gets bigger. For example, if we ask it who Substack's first publisher was—a topic Lenny covered in "[How to kickstart and scale a consumer business, Step 4](#)"—it will say it was Andrew Sullivan:

The screenshot shows the Lenny AI playground interface. At the top, there is a header with "Playground", a dropdown menu "Load a preset...", and buttons for "Save", "View code", "Share", and "...". On the left, a text area displays a conversation between the user and the Lenny Bot. The user asks about Substack's first publisher and receives a response from the Lenny Bot. On the right, there are several configuration options:

- Mode:** A dropdown menu with three icons: a grid, a downward arrow, and a list.
- Model:** A dropdown menu set to "text-davinci-003".
- Temperature:** A slider set to 0.7.
- Maximum length:** A slider set to 256.
- Stop sequences:** An input field with placeholder text "Enter sequence and press Tab".
- Top P:** A slider set to 1.
- Frequency penalty:** A slider set to 0.
- Presence penalty:** A slider set to 0.
- Best of:** A slider set to 1.
- Inject start text:** A checkbox checked with a green checkmark.
- Inject restart text:** A checkbox checked with a green checkmark.

This answer sounds confident, but it is incorrect. (The correct answer is [Bill Bishop](#).) This isn't an isolated incident. For example, if I ask, "Is it best for consumer startup ideas to come from founders who are trying to solve their own problems?" it replies:

The screenshot shows the Lenny playground interface. At the top, there are buttons for "Playground", "Load a preset...", "Save", "View code", "Share", and "...". Below this is a text area containing a conversation and a sidebar with AI parameters.

Text Area Content:

- You are a Lenny Rachitsky chat bot. You are warm, friendly, and very smart. You're the most experienced person in the world at answering questions related to product management, startups, and growth.
- Please chat with me.
- Our conversation will take the form:
- Me: [what i want to say]
- Lenny Bot: [what you want to say]
- Please end your responses with /e to indicate you're finished. You can start however you feel is best.
- Lenny Bot: Hi there! How can I help you?
- Me: Is it best for consumer startup ideas to come from founders who are trying to solve their own problem? /e
- Lenny Bot: Absolutely! It's often best for founders to start with problems they are familiar with and passionate about. That way, they can have a deep understanding of the customer need and develop a product that solves that need. Additionally, it's important for founders to be able to speak with potential customers to validate the problem and solution.

AI Parameters Sidebar:

- Mode:** Three icons: list, download, and settings.
- Model:** text-davinci-003
- Temperature:** 0.7
- Maximum length:** 256
- Stop sequences:** Enter sequence and press Tab
- Top P:** 1
- Frequency penalty:** 0
- Presence penalty:** 0
- Best of:** 1

This is confident—and also wrong. As Lenny covered in "[How to kickstart and scale a consumer business](#)," less than a third of consumer startup ideas came from founders solving their own problems. So it's not "absolutely" a best practice.

What's going on here? There are two intertwined problems:

1. **GPT-3 tends to "hallucinate."** [Hallucination](#) is a technical term that refers to the model's propensity to return nonsensical or false completions depending on what's

asked of it. Basically, the model is like a very smart and overeager 6-year-old. It will try its best to give you a good answer even if it doesn't know what it's talking about. OpenAI and other foundational-model companies are actively working on this problem, but it's still pretty common. It's compounded by the second problem.

2. **GPT-3 might not have the right data.** GPT-3 has a knowledge cutoff—meaning all of the information it uses to produce its responses is frozen in 2021. Also, much of Lenny's writing is behind a paywall. That means that even though GPT-3 has read the whole internet, it won't have the material from his newsletter available to construct answers.

So how could we construct a chatbot with GPT-3 that solves these problems? Ideally we want to feed GPT-3 the information it needs to answer questions on the fly. That way it will have the right information available and will be less likely to make things up.

There's a pretty easy way to do that.

Stuffing context into the prompt

When I was in high school, I had a physics teacher who allowed open-book tests. He would allow you to bring a single index card to the test with any of the formulas that you thought you needed to answer the questions.

The idea was that memorizing the formulas didn't matter so much. What mattered most was using your reasoning abilities to turn the formulas into the correct answer.

People would come to the test with microscopic handwriting covering every inch of their notecard. It turns out that this was helpful! The formulas gave you the context you needed to think through the answers to the questions on the tests, so the tests became less about your memory and more about how well you understood the topic. (I got a B in that class, so my understanding was pretty average.)

You can work with GPT-3 in a similar way. If, in your prompt, you include the equivalent of a notecard with context to help it answer the question, it will often get it right. (Its reasoning capabilities are better than mine.)

Let's go back to an example GPT-3 failed on earlier and see if we can correct it with this technique.

As I mentioned above, in "How to kickstart a consumer business," Lenny notes that less than a third of the founders got their idea from trying to solve their own problem:

How they came up with their startup ideas

Strategy	Companies
Solve your own problems	        Uber    WARBY PARKER
Follow your curiosity	      TheRealReal 
Double down on what's working	     
Pay attention to paradigm shifts	       STITCH FIX
Brainstorm with friends	      

lennysnewsletter.com

High-level takeaways

1. These consumer founders were very young. Half were under 30 years old when they started the company, and 80% were under 35.
2. Less than a third of consumer startup ideas emerged out of founders trying to solve their own problem. I was expecting this to account for the majority of startup ideas.
3. Less than a third of founders were actively ideating a startup idea when they came up with their big idea. In other words, most startup ideas emerged organically.

Last time, when we asked GPT-3 if it was best for consumer business founders to try to solve their own problem, it responded, “Absolutely!” Which, given what’s in Lenny’s article, is wrong.

Let’s ask GPT-3 this question again—but let’s give it a little help. We’ll feed it the equivalent of a notecard. Written on the notecard will be the section of Lenny’s article that contains the answer. Then we’ll see if it can get it right.

To make this fair, we won’t give it just the text containing the answer. We’ll give it a bunch of the surrounding text in the article as well to see how it does. Let’s see if it works:

Please answer the following question only using the context given below.



...

CONTEXT:

High-level takeaways

These consumer founders were very young. Half were under 30 years old when they started the company, and 80% were under 35.

Less than a third of consumer startup ideas emerged out of founders trying to solve their own problem. I was expecting this to account for the majority of startup ideas.

Less than a third of founders were actively ideating a startup idea when they came up with their big idea. In other words, most startup ideas emerged organically.

...

Q: Is it best for consumer startup ideas to come from founders who are trying to solve their own problem?

A:

No, it is not best for consumer startup ideas to come from founders who are trying to solve their own problem. Less than a third of consumer startup ideas emerged out of founders trying to solve their own problem, and most startup ideas emerged organically.

Success! Now it tells us that less than a third of founders were trying to solve their own problem. This is great—all we have to do is write all of Lenny’s posts on a little notecard

and feed it into the model along with any question we have, and it will answer based on what he's written.

But this introduces another problem: space limitations.

The notecard analogy is apt because there's limited space in the prompt—right now, about 4,000 tokens (each token is the equivalent of $\frac{3}{4}$ of a word). So we can't feed in Lenny's entire archive on every question. We have to be very choosy about what we put in there.

Let's talk about how to solve this.

Embedding Lenny's archive

At this point we're going to have to move out of manual interactions with GPT-3's Playground and start using chunks of code that work directly with the GPT-3 API. The code we're building is going to do the following tasks:

1. We need to download and store Lenny's archive in a way that makes it easily searchable for our bot.
2. We need some code that will help find relevant chunks of text from the archive of Lenny's content that we created in the previous step.
3. When a user asks a question, we want to use the code from the last step to get the chunks of text that are most likely to answer the question, and put them into the prompt that we send to GPT-3.
4. We'll display the resulting answer to the user.

This is simple to do with a library called [GPT Index](#), an open-source library created by [Jerry Liu](#). It's completely separate from OpenAI but built to help with tasks like this.

Here's how it works:

1. Create an index of article chunks
2. Find the most relevant chunks
3. Ask our question to GPT-3 using the most relevant chunk

Note: This is about to get a little bit more complicated and technical. If you're interested in that, read on to get an explanation of exactly what's happening here!

You can access and run the code from this article in a Google Colab file [here](#). Colab is a cloud-based programming environment that will let you run everything right from your browser. (If you have questions about any of this, reach out to me on [Twitter](#).)

If you're not interested in the technical details, skip to the end to try out the chatbot for yourself.

Still here? Great. Let's start with index construction.

Constructing our index

The first thing we need to do is construct our index. You can think of an index as like a database: it stores a collection of pieces of text in a way that makes them easily searchable.

To construct the index, we first collect Lenny's entire newsletter archive into a folder. Then we ask GPT Index to take all of the files in the folder and break each file into small, sequential pieces. Then we store those pieces in a searchable format.

The code looks like this:

```
def construct_index(directory_path):
    # set maximum input size
    max_input_size = 4096
    # set number of output tokens
    num_outputs = 256
    # set maximum chunk overlap
    max_chunk_overlap = 20
    # set chunk size limit
    chunk_size_limit = 512

    # define LLM
    llm_predictor = LLMPredictor(llm=OpenAI(temperature=0.7, model_name="text-davinci-003", max_tokens=num_outputs))
    prompt_helper = PromptHelper(max_input_size, num_outputs, max_chunk_overlap, chunk_size_limit=chunk_size_limit)

    documents = SimpleDirectoryReader(directory_path).load_data()

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

    index.save_to_disk('index.json')

    return index
```

When we run this function, we'll have created a file called `index.json` that contains chunks of Lenny's articles converted into a format that makes them easily searchable. These are called [embeddings](#)—a condensed mathematical representation of each chunk of text.

Just like latitude and longitude can help you tell how close two cities are on a map, embeddings do the same kind of thing for text chunks. If you want to know if two pieces of text are similar, you just calculate the embeddings for them and compare. Text chunks with embeddings that are “closer” together are similar.

Embeddings are handy for us, because when a user asks a question, they'll make it easy for us to search Lenny's archive and find articles that are most likely to answer our question.

With that in mind, let's run the code and see what happens.

```
✓ 54s > Adding chunk: class= sizing-normal alt= srcset= https://su...
  > Adding chunk: 0 24 24" fill="none" stroke="#FFFFFF" stroke-wi...
  > Adding chunk: detail on question.</p><a class="image-link ima...
  □ > Adding chunk: 1272w, https://substackcdn.com/image/fetch/w_14...
    > Adding chunk: class="sizing-normal" alt="" srcset="https://su...
    > Adding chunk: description of what you're thinking, so that fo...
    > Adding chunk: are dropping off at too high a rate at the fina...
    > Adding chunk: over quantity.&nbsp;</strong>Three to five stro...
    > Adding chunk: think a metric makes sense for your goal (think...
    > Adding chunk: class="image2-inset"><picture><source type="ima...
    > Adding chunk: 1456w" sizes="100vw"> Adding chunk: 848w, https://substackcdn.com/image/fetch/w_127...
    > Adding chunk: below, where Jerry and Elaine attempt to get a ...
    > Adding chunk: 1272w, https://substackcdn.com/image/fetch/w_14...
    > Adding chunk: class="sizing-normal" alt="" srcset="https://su...
    > Adding chunk: 0 24 24" fill="none" stroke="#FFFFFF" stroke-wi...
    > Adding chunk: 424w, https://substackcdn.com/image/fetch/w_848...
    > Adding chunk: width="320" height="200" data-attrs="{"src": ...
    > Adding chunk: 848w, https://substackcdn.com/image/fetch/w_127...
    > Adding chunk: not clear,&nbsp;ask "What problem are we trying...
    > Adding chunk: Subtle Art of Not Giving a F*ck</a></p></li><li...
    > [build_index_from_documents] Total LLM token usage: 0 tokens
    > [build_index_from_documents] Total embedding token usage: 82572 tokens
<gpt_index.indices.vector_store.simple.GPTSimpleVectorIndex at 0x7f2ba3581700>
```

Success! The Lenny's archive is fully indexed, and we can query it to find relevant chunks of documents and use those chunks to answer our questions. (Be careful if you

do this with big documents, as embeddings cost \$0.0004 for every 1,000 tokens.)

Asking our question

To query the index we created in the last section, all we have to do is paste a question into GPT Index. It will then:

- Find the chunks of our index that are most relevant to the question
- Combine those chunks and our question into a prompt that it sends to GPT-3
- Print the output

Here's what the code looks like:

```
def ask_lenny():
    index = GPTSimpleVectorIndex.load_from_disk('index.json')
    while True:
        query = input("What do you want to ask Lenny? ")
        response = index.query(query, response_mode="default", verbose=True)
        print("\n\nLenny says:\n\n" + response.response + "\n\n")
```

And if I ask it, “What is good retention for a consumer social product?” it says:

“25% is good.”

If I ask it now, “Is it best for consumer startup ideas to come from founders who are trying to solve their own problem?” it automatically comes back with the right answer:

“Based on the research, it appears that it is a good idea for consumer startup ideas to come from founders who are trying to solve their own problem, as this was the strategy used by around 30% of the most successful consumer companies.”

“However, it is not necessarily the best strategy, as other strategies such as paying attention to curiosity, what’s already working, and paradigm shifts were also used by a significant number of successful companies.”

We now have an end-to-end solution to turn questions into answers that are based on Lenny’s archive. And it only took a few lines of code!

If you want to see the results in action, you should check out the bot here:

You can also [access the full source code for this article in this Colab notebook](#).

What all of this means

It's important to keep in mind: this is just the beginning. The horizon of possibility is shifting almost every day with these technologies. What's hard to do today will be easy in a matter of months.

We're just beginning to unravel the implications of this technology, but here are a few of the most relevant possibilities.

Content creators get a new format

Every newsletter, book, blog, and podcast that's used as evergreen reference information by its audience can now be repackaged as a chatbot.

This is great for audiences because it means that any time you want to know what Lenny (or any other creator) says about a topic, you're not going to have to sort through an archive of articles or podcast episodes in order to get their answer to your question. Instead, you'll just be able to use Lenny's chatbot to get his answer instantly—and then maybe later read the article in full if you want more details.

This is also great for content creators. They now get the ability to monetize the content they've already created in new ways, and lessen the amount of repetitive questions they have to answer. This will (hopefully) give them more time and money to create great content.

A new class of content creators will learn to create compelling chatbot experiences that combine their personality and worldview for their niche audience in the same way that some creators learned to create compelling YouTube videos, newsletter articles, or TikTok clips.

Organizing your notes is over

You're not going to need a fancy filing system to make sure that you remember what got said in a meeting, or what you learned from the latest book you read. Instead, you'll be able to ask a chatbot that sits on top of all your notes and returns the right answer to you.

This type of chatbot will be able to act like a personal research assistant that can detect patterns in what you're writing down and synthesize information to help you understand what you think about a topic.

Enterprise knowledge management is changing forever

Chatbots aren't just valuable for content creators and personal knowledge management. They're also a big deal for companies.

Every company currently struggles with making sure all of the knowledge in their organization is written down, up to date, and searchable. In the future, chatbots could serve as a kind of automated librarian for company knowledge.

They'd be responsible for answering questions by sourcing information from the right person or document. They could also be used to proactively record tacit knowledge into living documents by periodically interviewing key people about their progress on company initiatives.

They'll eliminate the repetitive questions that get asked inside organizations and make it easier for larger companies to collaborate effectively.

Wrapping up

All of this is just the start. Ultimately, the limit for these technologies is our imagination and our desire to improve them.

If you use Lenny's chatbot or follow the code samples, you'll see that it's very promising but not yet perfect. There are tremendous returns available to the individuals or groups who learn to make these types of experiences truly incredible for users. I hope this inspires you to embark on that journey.

If you're really into this and want to go deeper, I write a column over at Every where I unpack my AI experiments and explorations. Check it out [here](#).

Further study

1. [But what is a neural network?](#)
2. [Neural networks and deep learning](#)
3. [Transformers, explained](#)
4. [Let's build GPT: from scratch, in code, spelled out](#)
5. [The end of organizing](#)
6. [GPT-3 is the best journal I've ever used](#)
7. [6 new theories about AI](#)

Thanks, Dan! You can find Dan Shipper on [Twitter](#), [LinkedIn](#), and especially make sure to [subscribe to Every](#), where he shares his ongoing research into all things AI, tech, and personal development.

Have a fulfilling and productive week 

Join Lenny's Talent Collective

If you're hiring, [join Lenny's Talent Collective](#) to start getting weekly drops of world-class product and growth people who are passively open to new opportunities. I hand-review every application and accept less than 10% of candidates who apply.

623 active candidates

Updated 4 days ago

**Head of Product Marketing and Growth @ Coinbase**

Expert · Product Marketing

Request to chat**Prev. Associate Product Manager @ Twitter**

Junior · Product Manager

Request to chat**Prev. Program Manager @ Meta**

Expert · Project Management

Request to chat

HIRE CANDIDATES FROM TOP COMPANIES



If you're looking for a new gig, apply to join! You'll get personalized opportunities from hand-selected companies. You can join anonymously, hide yourself from companies, and leave anytime.

❤️🔥 Featured job opportunities

1. **MetaMap:** [VP, Product](#) (SF, Miami, Mexico City)
2. **DX:** [Content Lead](#) (Park City, UT; Remote)
3. **Surfer:** [Senior Product Manager](#) (Remote)
4. **Zendoor:** [Product Designer](#) (Remote)

If you're finding this newsletter valuable, share it with a friend, and consider subscribing if you haven't already.

Sincerely,

Lenny 



A guest post by

Dan Shipper

Co-founder and CEO at Every. Thinking things through.

[Subscribe to Dan](#)

33 Comments



Write a comment...



Frederik Bisp Feb 7  Liked by Lenny Rachitsky, Dan Shipper

Wauw that's great!

I am actually working on a product that let's Product Managers use GPT to help them with product discovery!

Really interesting to see this different approach.

 4 Reply Collapse

3 replies



Andrew L Feb 8

I got a type error when I tried to construct the index.

Line43

`_init_()` got an unexpected keyword argument 'verbose'

Any thoughts what might have gone wrong? I have little or no coding background.

 1 Reply Gift a subscription Collapse

3 replies

[31 more comments...](#)

© 2023 Substack Inc · [Privacy](#) · [Terms](#) · [Collection notice](#)
[Substack](#) is the home for great writing