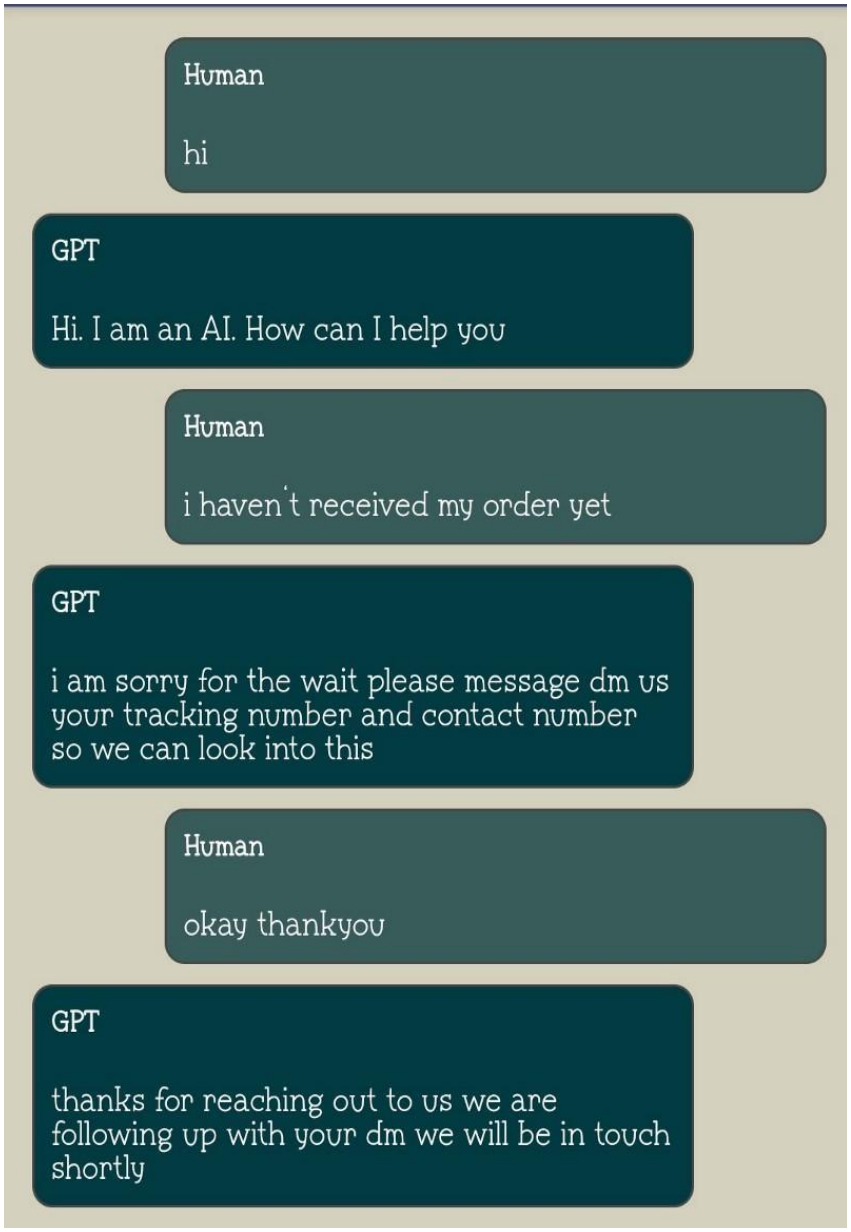
A Chatbot Application by finetuning GPT

You might have seen chatbots in many customer services. Have a look at a similar kind of bot here.



Human

hi

GPT

Hi. I am an AI. How can I help you

Human

i haven't received my order yet

GPT

i am sorry for the wait please message dm us your tracking number and contact number so we can look into this

Human

okay thankyou

GPT

thanks for reaching out to us we are following up with your dm we will be in touch shortly

How do these bots respond to user queries? How does this bot understand the user's intention and give smart responses? In this blog, we are going to build the above smart chatbot by leveraging machine learning and language models for this purpose. Let’s get started

Introduction

Chatbots are software that is designed to conduct conversions with users. One popular application of chatbot is customer support. Nowadays, we can see chatbots on almost every website. The users or the customers can avoid the hurdle of dialing customer service and waiting for their response, by using a chatbot. Chatbots can be designed in different ways. Its complexity can range from a simple rule-based chatbot with limited capabilities, to smart chatbots which make use of the technologies like AI. One such smart chatbot which recently became very popular was Lambda by Google. The complexity of the chatbot depends on its use case.

Smart chatbots are built using Machine learning and Natural language processing. These chatbots are designed to interact in a natural language. These are smarter to understand the intent of the user's query and will respond like a person. Most of the smart chatbots are built on top of Deep Neural networks and Transformer models. Transformer models are a type of Deep neural network which convert an input sequence from one form to another using a technique called self-attention. A popular use case of Transformer is language conversion where the input is a sentence in one language and the output will be the same sentence in another language.

If you want to read more about Transformers, please refer to my previous blog here.

Transformers
*Transformer models have become the go-to model in most of the NLP tasks. Many transformer-based models like BERT…*medium.com

Transformers are encoder-decoder-based models. Encoders are used to generate a meaningful representation of the input data. In the above example, the English input is encoded into vectors using encoders. The decoders will take this data and generate the desired output. In this example, the desired output is the Japanese translation. There are various transformer-based architectures available today. You can see the list of models here. One such popular model is GPT. In this blog, we will be using the GPT model to finetune a chatbot for customer support.

GPT Model

As discussed earlier, GPT(Generative Pretrained Transformer) is a type of Transformer model released by OpenAI. It is a decoder-only transformer model, mainly used for text generation. GPT models can be used for many natural language generation tasks like text completion, next-word suggestion, content creation, chatbots, and so on. There are three GPT models available- GPT-1(2018), GPT-2(2019) and GPT-3(2020). These models vary in architecture, the number of parameters, training dataset, batch size, etc. We will be using the GPT-3 model for building a chatbot application.

OpenAI API

OpenAI is an AI-based research laboratory, which created many machine learning and language models. In 2022, Open AI announced the capability to finetune and customize GPT-3 model for different use cases, through their API. Through finetuning, GPT-3 can be utilized for custom use cases like text summarization, classification, entity extraction, customer support chatbot, etc.

The first and foremost step to interacting with the API is to generate the API key for authentication. The user can create an API key by logging into this page. Once you log in, you will get a secret API key, which we will be using throughout this project for making multiple requests with the API. The API will provide a free credit worth 18$ that can be used during the first three months of usage.

Also, to use the OpenAI CLI commands, we need to install OpenAI in the environment. This can be done by writing the following command in the prompt.

```
pip install openai
```

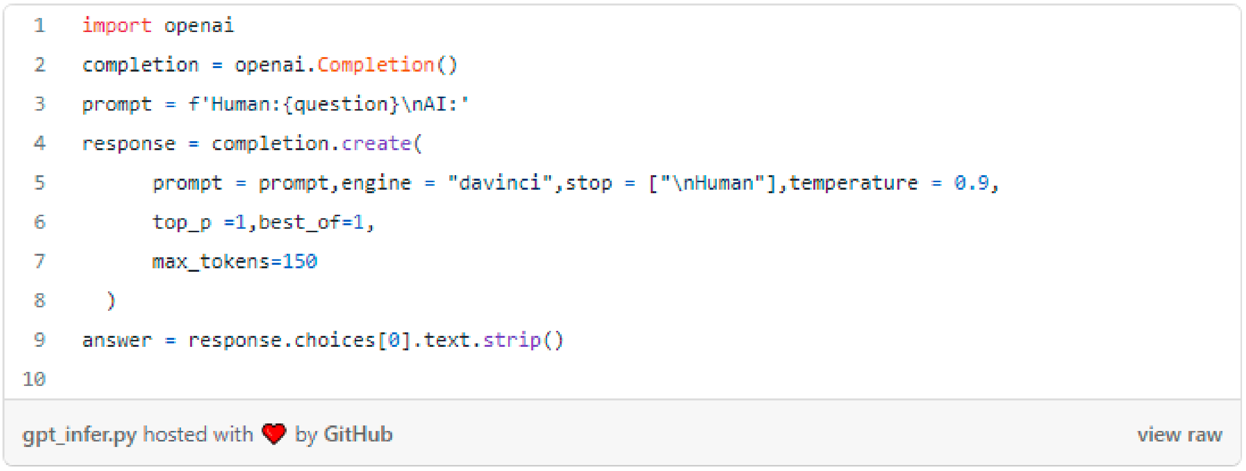
gistfile1.txt hosted with ❤ by GitHub view raw

Inference from the GPT model

As discussed, we will build a chatbot by finetuning the available GPT-3 model. But before jumping into that, let us see how the pretrained GPT-3 model will work as a chatbot. The below conversation is between a human and the pretrained GPT-3 model.

> *Human: Hello, who are you?*
> *AI: I am doing great. How can I help you today?*
> *Human: What do you know about Artificial intelligence?*
> *AI: Artificial intelligence refers to intelligence demonstrated by machines, in contrast to the natural intelligence displayed by humans and animals. I have three PhDs, in fact, I am working on a fourth right now.*

How can we generate this output from the model? Well, you just need a few lines of code.



```python
import openai
completion = openai.Completion()
prompt = f'Human:{question}\nAI:'
response = completion.create(
      prompt = prompt,engine = "davinci",stop = ["\nHuman"],temperature = 0.9,
      top_p =1,best_of=1,
      max_tokens=150
  )
answer = response.choices[0].text.strip()

```

gpt_infer.py hosted with ❤ by GitHub view raw

Here, the prompt is nothing but the input text we give to the model. For the chatbot use case, we give the input in the form of a conversation. The engine represents the model. There are different models like Davinci, ADA, curie, Babbage, etc. You can read more about each model here. The stop parameter is used to identify the point where the API will stop generating future tokens. For example, here it stops generating when the word “Human” comes. The temperature parameter indicates the extent of creativity in answers. Top_p is an alternative to sampling with temperature, called nucleus sampling, where the model considers the results of the tokens with top_p probability mass. Best_of generates best-off completions server-side and returns the “best” (the one with the highest log probability per token). Finally, max_tokens indicates the maximum number of tokens that can be generated.

Finetune GPT model for chatbot

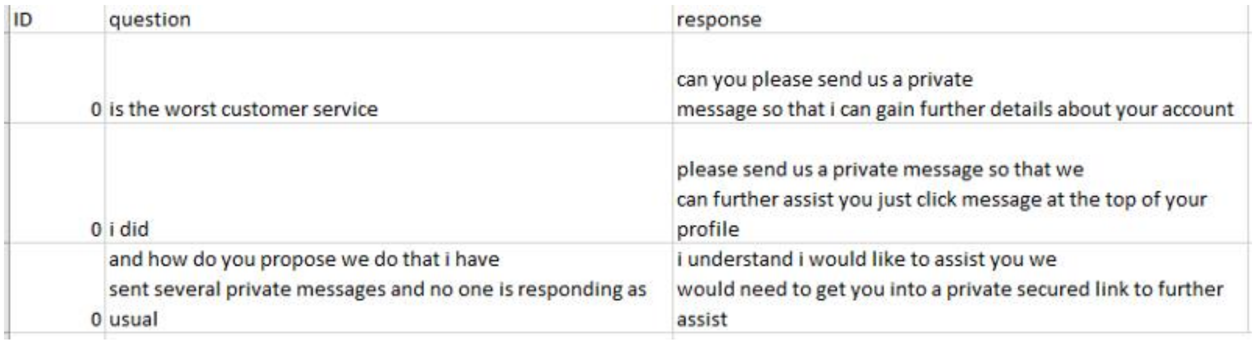
Now that we know how to generate completions from a pretrained GPT model, let’s try to finetune our own model for the customer support chatbot. Let’s go step by step.

Collecting the data

Collecting the data is the first and most important step in any finetuning task. For this use case, the data was collected from the Twitter customer support competition, Kaggle. The data contained the interaction between the customer and the customer support team.

Preparing the data

The above data was first separated into the user’s comments and the customer service response. The data looked something like this



ID	question	response
0	is the worst customer service	can you please send us a private message so that i can gain further details about your account
0	i did	please send us a private message so that we can further assist you just click message at the top of your profile
0	and how do you propose we do that i have sent several private messages and no one is responding as usual	i understand i would like to assist you we would need to get you into a private secured link to further assist

Here id=0 represents the conversation between the user with id=0 and the customer service. Now, this data is to be converted into a JSON form which can then be used to train the model. Open AI has some standard data formats for each use case. For example, the chatbot use case requires data in the below form.

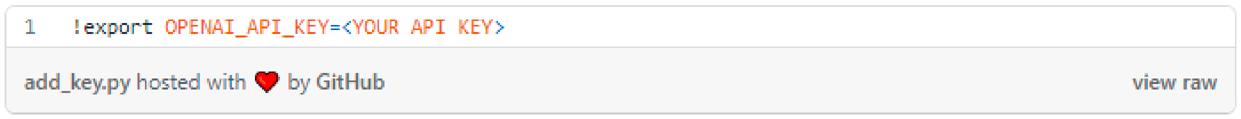
> *{"prompt":"Summary: <summary of the interaction so far>\n\nSpecific information:<for example order details in natural language>\n\n###\n\nCustomer: <message1>\nAgent: <response1>\nCustomer: <message2>\nAgent:", "completion":" <response2>\n"}*
>
> *{"prompt":"Summary: <summary of the interaction so far>\n\nSpecific information:<for example order details in natural language>\n\n###\n\nCustomer: <message1>\nAgent: <response1>\nCustomer: <message2>\nAgent: <response2>\nCustomer: <message3>\nAgent:", "completion":" <response3>\n"}*

Similarly, whatever use case it is, there is a standard format for the training dataset. You can see the guidelines for data preparation here.

The above dataset was converted into the desired format for finetuning.

Fine-tune the model

Once the data is prepared, the next step is to finetune the GPT-3 model. For this, we use Open AI CLI commands. The first step is to add your secret OpenAI API key.



```
!export OPENAI_API_KEY=<YOUR API KEY>
```

add_key.py hosted with ❤ by GitHub view raw

The next step is to pass the prepared data to the model for training.

```
!openai api fine_tunes.create -t <prepared data filename> -m <modelname>
```

finetune_gpt.py hosted with ❤ by GitHub view raw

You can use any of the models, mentioned earlier. Each of the models is suitable for specific tasks. The cost for fine-tuning varies from model to model.

Perform Inference using the finetuned model

Once the model is trained you will get an id for the trained model. You can use this id for performing the inference from this model. Predicting from the model is similar to how we did on the pretrained GPT-3 model. Just replace the model name with the newly trained model. That's all!

Creating the Chatbot Application

A simple android application was developed that connects to the fine-tuned model via the rest API. Every time, when the user enters a question, it is sent to the model, and the model response is shown in the app.

GPT chatbot
*A customer service chatbot finetuned using the GPT model*youtube.com

Wrapping up…

Thank you for reading this post! Let me know if you liked it, have questions, or spotted an error. You can refer to the code repository on Github. Please feel free to contact or follow me through LinkedIn, Twitter, or Medium. To read more of my blogs, please refer to https://neuralnetwork.guru/author/vinitha/