**Unleashing the Power of GPT-3: Fine-Tuning for Superhero Descriptions**

**Step-by-step guide for GPT-3 fine-tuning**

Since the end of 2022, the launch of ChatGPT by OpenAI has been considered by many of us to be the iPhone moment of AI. However, OpenAI’s chatbot is not the first generative AI text machine learning model and follows the launch of GPT-3, which was launched two years earlier.

OpenAI provides us with a ready-to-use GPT-3 trained model. Furthermore, a specific task can be fine-tuned on a smaller dataset. For example, suppose you want to create an email response generator specific to your company. First, you must collect a large amount of data about your particular business domain, such as customer email inquiries and responses. You can then use this data to fine-tune GPT-3 to learn your company’s specific language patterns and phrases. By fine-tuning GPT-3, creating a highly customized and specialized email response generator is possible, specifically tailored to the language patterns and words used in a particular business domain.

In this blog post, I will show you how to fine-tune GPT-3. We will do this with python code and without assuming prior knowledge about GPT-3.

**What is needed for fine-tuning?**

Unlike the GPT-2 model available for instance on Hugging Face (when this blog is written), we don’t have direct access to the GPT-3 model. Therefore, you first need to get an API key from OpenAI and install the Python package *openai*, which can be quickly done via *pip*.

For the API key from OpenAI:

* go to <https://platform.openai.com/account/api-keys> ,
* create an account,
* click on ‘*Create new secret key’* and
* do a copy of the key.

The key is a long string of characters starting with ‘sk-’. Make sure you keep it secret! Once you have your key, an easy way to get access to your key is to do the following in your terminal: (personally, for simplicity, I put it in my *.bachrc*):

export OPENAI\_API\_KEY=sk-t59pgejhtrff5(...)

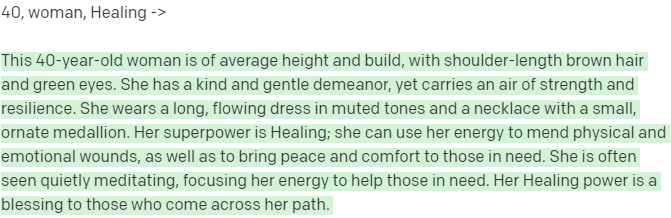
Using GPT-3 models has a cost. We need credits. At the time of this writing, when you create a new account, you get free credits to try out the tool. I don’t know if this will continue…

Now that we have our key and Python package, it is time to think about the data we need to fine-tune. First, we need a file of examples for fine-tuning, where each example is a *prompt* followed by the appropriate *completion*.

**A superhero description generation tool**

We will build a tool for this demo to create descriptions of imaginary superheroes. In the end, the tool will receive the age, gender, and power of the superhero, and it will automatically produce a description of our superhero.

In the following example, after fine-tuning the model, all we have to say is ’*40, woman, Healing ->*’, and we will automatically get a description from the model.



This is what it’s all about!! 😃

**Creation of a synthetic set of data for fine-tuning**

In some situations, you may have a data set you want to use for fine-tuning. But since I don’t have one, let’s see how to create a synthetic data set with the description of the superheroes directly from GPT-3. The following code will give me a CSV file with examples of *prompts* and the corresponding *completions*.

import os  
import openai  
import pandas as pd  
  
openai.api\_key = os.getenv("OPENAI\_API\_KEY")  
  
l\_age = ['18', '20', '30', '40', '50', '60', '90']  
l\_gender = ['man', 'woman']  
l\_power = ['invisibility', 'read in the thoughts', 'turning lead into gold', 'immortality', 'telepathy', 'teleport', 'flight']   
  
f\_prompt = "Imagine a complete and detailed description of a {age}-year-old {gender} fictional character who has the superpower of {power}. Write out the entire description in a maximum of 100 words in great detail:"  
f\_sub\_prompt = "{age}, {gender}, {power}"  
  
df = pd.DataFrame()  
for age in l\_age:  
 for gender in l\_gender:  
 for power in l\_power:  
 for i in range(3): ## 3 times each  
 prompt = f\_prompt.format(age=age, gender=gender, power=power)  
 sub\_prompt = f\_sub\_prompt.format(age=age, gender=gender, power=power)  
 print(sub\_prompt)  
  
 response = openai.Completion.create(  
 model="text-davinci-003",  
 prompt=prompt,  
 temperature=1,  
 max\_tokens=500,  
 top\_p=1,  
 frequency\_penalty=0,  
 presence\_penalty=0  
 )  
   
 finish\_reason = response['choices'][0]['finish\_reason']  
 response\_txt = response['choices'][0]['text']  
   
 new\_row = {  
 'age':age,   
 'gender':gender,   
 'power':power,   
 'prompt':prompt,   
 'sub\_prompt':sub\_prompt,   
 'response\_txt':response\_txt,   
 'finish\_reason':finish\_reason}  
 new\_row = pd.DataFrame([new\_row])  
 df = pd.concat([df, new\_row], axis=0, ignore\_index=True)  
  
df.to\_csv("out\_openai\_completion.csv")

Let’s look at how this code works 🧐.

The variable *f\_prompt* contains the following sentence where *{age}*, *{gender},* and *{power}* are missing.

Imagine a complete and detailed description of a {age}-year-old {gender} fictional character who has the superpower of {power}. Write out the entire description in a maximum of 100 words in great detail:

In the first three *for* loops of the code, we iterate over different values of {age}, {gender}, and {power}. At each step of the loop, we replace the 3 missing variables with different values. Then we use the *openai.Completion.create* function to ask GPT to generate a response to our prompt.

The most important parameters of this function are

* *model*: The model used to generate the response. OpenAI offers four standard GPT-3 models (ada, babbage, curie, or davinci) that vary in size … and price of use. Here it is *davinci* — the biggest model.
* *prompt*: The prompt that we want to fulfill with GPT-3.
* *temperature*: The temperature is a number between 0 and 1 and controls how much randomness is in the output. We set the temperature to the maximum to allow the model to be as creative as possible in creating the response.
* *max\_tokens*: Defines the maximum length of the response.

At the end of this script, we have a Pandas table stored in the file *out\_openai\_completion.csv.* The two primary columns in this table that interest us are *sub\_prompt* and *response\_txt.*

* The *sub\_prompt* will be for example‘*18, man, invisibility’.* It contains the three values that were replaced, separated by commas.
* The *response\_txt* contains the output of the GPT model.

**Fine-tuning the GPT model**

The following code retrieves the previously created file *out\_openai\_completion.csv* and uses *openai* to fine-tune a GPT-3 model.

import pandas as pd  
import openai  
import subprocess  
  
df = pd.read\_csv("out\_openai\_completion.csv")  
  
prepared\_data = df.loc[:,['sub\_prompt','response\_txt']]  
prepared\_data.rename(columns={'sub\_prompt':'prompt', 'response\_txt':'completion'}, inplace=True)  
prepared\_data.to\_csv('prepared\_data.csv',index=False)  
  
  
## prepared\_data.csv --> prepared\_data\_prepared.json  
subprocess.run('openai tools fine\_tunes.prepare\_data --file prepared\_data.csv --quiet'.split())  
  
## Start fine-tuning  
subprocess.run('openai api fine\_tunes.create --training\_file prepared\_data\_prepared.jsonl --model davinci --suffix "SuperHero"'.split())

Let’s take the time to understand this code as well 🤓!

First, the content of the file *out\_openai\_completion.csv* is loaded into the data frame *df*. As a reminder, in our task, if the user enters ’40, female, healing’, we want to have a description of a 40-year-old female character with the power of healing. To perform fine-tuning, it is necessary to provide GPT with examples of what the user might type and the corresponding desired response. In the data frame *df*, the columns *sub\_prompt* and *response\_txt* contain examples of input with the corresponding desired response. In the code above, we first extract these two colons and then rename them to *prompt* and *completion*, respectively. The resulting data frame is stored in a new file *prepared\_data.csv* containing only these two columns.

This file *prepared\_data.csv* looks like this:

prompt,completion  
"18, man, invisibility","He looks to be in his late teens, with dark eyes and unruly black hair which reach his shoulders. His build is athletic and strong, but not muscular. He often wears clothing that makes him blend in with his surroundings, in order to stay as inconspicuous as possible.He has the ability of invisibility, which he can use to become an invisible observer or to make himself or objects around him disappear from view. He has honed his skills to make himself undetectable, able to even make sounds vanish, allowing him to go unnoticed. His invisibility power is his greatest asset, and he uses it to protect those he cares about."  
"18, man, invisibility","18 year-old Jackson is a powerful young man with the incredible power of invisibility. He stands at 6'2 and has a strong athletic frame with wavy brown hair, startling blue eyes and a warm and gentle smile. His strength and agility make him especially good at using his ability. He can become invisible in an instant, moving through walls and other obstacles without being detected. What's more, he can move objects with his mind and use his power to protect those he loves. His power is a blessing and a curse, as it can be abused to do harm. Jackson is a brave and noble person who uses his power to do good and make the world a better place."  
"18, man, invisibility","Brandon is an 18-year-old of average build, standing roughly 5'10 with an inquisitive look. He has naturally wavy chestnut brown hair and bright blue eyes. His demeanor is usually quite serious, but he also has an easy and gentle smile. He has a natural gift of invisibility, which he uses to protect himself and his family from harm. He's an inquisitive and creative young man who uses his gift to explore the world, observe people, and uncover the truth. His consistent practice of mindfulness helps him remain unseen, even when his emotions are visible. His intelligence and persistent passion for truth drives him to help those in need."  
"18, man, read in the thoughts","This 18-year-old man has a muscular stature and long, ash blonde hair. His bright blue eyes are full of knowledge and power, hinting at the strange ability he has - he can read minds. Now and then, strange sparks of electricity streak from his eyes when he concentrates hard enough. He is often calm and collected, but when provoked has the ability to blend his learning of your thoughts with his immense strength. His wit, intelligence and empathy are weapons that are often underestimated - yet, very lethal."  
"18, man, read in the thoughts","He's an 18-year-old man with short, cropped dark hair and bright green eyes. His usual outfit consists of an olive green hoodie, blue jeans and purple trainers. He's relaxed, confident and always reads the atmosphere well. One of his great skills is read in the thoughts of others. By doing so, he is able to understand people more easily and relate to their concerns out of the blue. He always uses his superpower with the utmost dignity and respect, never using it to manipulate. He uses it to seek the truth, identify people's emotions and gain insight into their perspectives. He's the sort of person that can make anyone feel comfortable and is never afraid of a challenge. His superpower is merely an extension of his already charismatic disposition."  
(...)

The Python *subprocess.run()* function runs a command as a subprocess. It is often used to execute external programs as if they were run in a terminal.

We use *subprocess.run()* to execute ‘*openai tools fine\_tunes.prepare\_data’*. This function takes the input file *prepared\_data.csv*, checks that the data are correct, and produces a JSONL file called *prepared\_data\_prepared.jsonl*. A JSONL file is a format that stores each JSON object on a separate line. JSONL files contain a sequence of JSON objects, each separated by a newline character.

Note that we have added the option “- - quiet” to automatically accept all recommendations made by ‘*openai tools fine\_tunes.prepare\_data’. F*or example, it suggests adding a ‘-->’ to the end of all prompts and adding a token *END* to the end of each response.

The first lines of this JSONL file look like this:

{"prompt":"18, man, invisibility ->","completion":" \n\nHe looks to be in his late teens, with dark eyes and unruly black hair which reach his shoulders. His build is athletic and strong, but not muscular. He often wears clothing that makes him blend in with his surroundings, in order to stay as inconspicuous as possible.\n\nHe has the ability of invisibility, which he can use to become an invisible observer or to make himself or objects around him disappear from view. He has honed his skills to make himself undetectable, able to even make sounds vanish, allowing him to go unnoticed. His invisibility power is his greatest asset, and he uses it to protect those he cares about. END"}  
{"prompt":"18, man, invisibility ->","completion":" \n\n18 year-old Jackson is a powerful young man with the incredible power of invisibility. He stands at 6'2 and has a strong athletic frame with wavy brown hair, startling blue eyes and a warm and gentle smile. His strength and agility make him especially good at using his ability. He can become invisible in an instant, moving through walls and other obstacles without being detected. What's more, he can move objects with his mind and use his power to protect those he loves. His power is a blessing and a curse, as it can be abused to do harm. Jackson is a brave and noble person who uses his power to do good and make the world a better place. END"}  
(...)

The fine-tuning of the GPT-3 model is really achieved in the second *subprocess.run(),* where *openai api fine\_tunes.create* is executed. In this function, we start by giving the name of the JSONL file created just before. You will then need to select the model you wish to fine-tune. OpenAI offers four main models with different performance levels suitable for various tasks. *Davinci* is the most powerful model, and Ada is the fastest. *Davinci* is also the most expensive model 😨.

Since the purpose of my model is to create descriptions of superheroes, we give my new model the suffix “*Superhero*”.

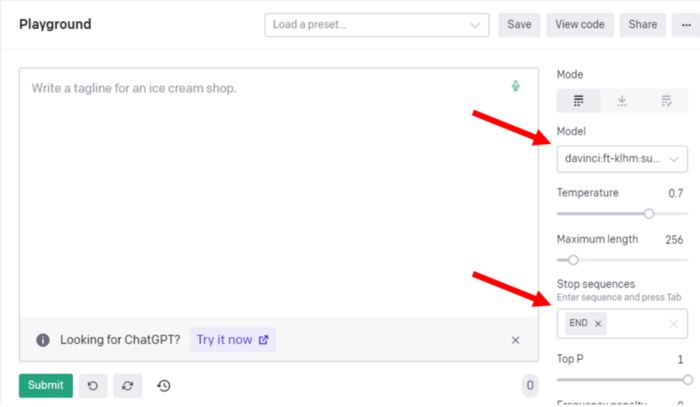
And that’s it 😉 After a few minutes, you will have a fine-tuned model ready to use 🌟.

**And now it’s time to test your new model**

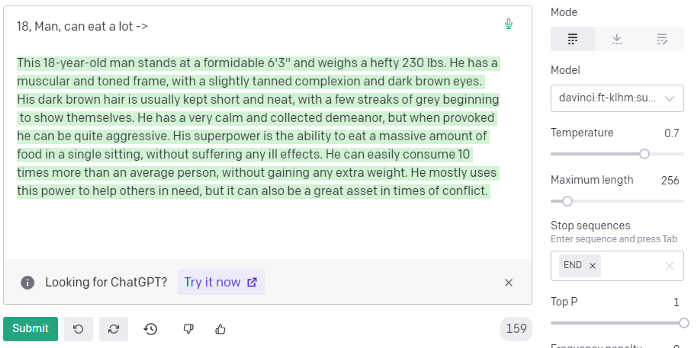
There are different ways to use a model for completion. Mainly via the Playground provided by OpenAI or via programming languages like Python.

The simplest way is probably to use the [playground](https://platform.openai.com/playground).

* Go to <https://platform.openai.com/playground>.
* Click on ‘Model’ and search for the one with the suffix “*Superhero*”.
* Add in ‘Stop sequences’ the token ‘END’.

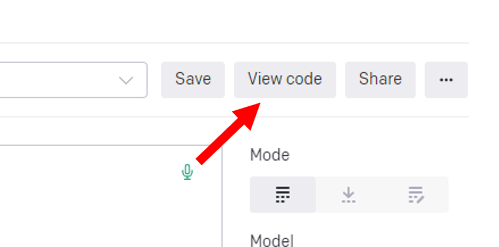


It is now time to ask our model for a new prediction. We will ask to describe an 18-year-old male character who really has an unnecessary power 😉 We will ask to describe a character who has the power… ‘*can eat a lot*’… and see what happens… 😆



Not so bad 😅

Do you want to do it in Python? Simple! Click on ‘View code’ at the top right of the screen.



In our case, in ‘View code’ we have this:

import os  
import openai  
  
openai.api\_key = os.getenv("OPENAI\_API\_KEY")  
  
response = openai.Completion.create(  
 model="davinci:ft-klhm:superhero-2023-02-01-14-56-48",  
 prompt="18, Man, can eat a lot ->\n",  
 temperature=0.7,  
 max\_tokens=256,  
 top\_p=1,  
 frequency\_penalty=0,  
 presence\_penalty=0,  
 stop=["END"]  
)

You just have to copy and paste it 👍.

**To conclude**

In this blog, we have seen how to generate synthetic data to refine our model and how to do that fine-tuning. We have used a use case of creating a superhero, but the same method can be used for any use case you may have. The most important thing is to have enough quality examples with prompts and desired responses.

**Thanks for reading!**

Please consider following me if you wish to stay up to date with my latest publications and increase the visibility of this blog.