

✦ Get unlimited access to the best of Medium for less than \$1/week. [Become a member](#)



A Step-by-Step Guide to Custom Fine-Tuning with ChatGPT's API using a Custom Dataset



R2consulting · [Follow](#)

4 min read · Feb 9, 2024



35



1



Introduction

Fine-tuning OpenAI's ChatGPT with a custom dataset allows you to tailor the model to specific tasks or industries. This step-by-step guide will walk you through the process of custom fine-tuning using ChatGPT's API and a custom dataset. We'll also cover how to convert your dataset into the required JSONL format. Finally, this article will illuminate some of the most important pros and cons associated with this technique.

Prerequisites:

1. OpenAI API key
2. Python installed on your machine
3. Basic understanding of Python programming

Step 1: Gather Your Custom Dataset

- Collect a dataset that is relevant to your task or domain. Ensure it is in a text format (e.g., CSV, TXT) and contains both input messages and corresponding model-generated responses.

Step 2: Install OpenAI Python Library and Set Up Environment Variable

```
!pip install openai
```

```
import openai
```

```
print(openai.__version__)
```

```
# Environment Variable
```

```
import os
```

```
os.environ['OPENAI_API_KEY'] = 'sk-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
```

Chat Completion Example code

Open in app ↗



Search



Write



M

```
messages=[  
  {"role": "system", "content": "You are a helpful assistant."},  
  {"role": "user", "content": "Describe US history?"}  
]  
)
```

Step 3: Prepare Your Data

- Split your dataset into two columns: “messages” and “model-generated”.
- Ensure each row contains a conversation snippet with the user’s message and the model-generated response.

Step 4: Convert Dataset to JSONL Format

- Write a Python script to convert your dataset into JSONL format. Here’s a simple example:

```
import json

def convert_to_jsonl(input_file, output_file):
    with open(input_file, 'r', encoding='utf-8') as infile:
        data = infile.readlines()

    jsonl_data = []

    for line in data:
        # Assuming your dataset is comma-separated
        user_message, model_response = line.strip().split(',')
        conversation = [{'role': 'system', 'content': 'You are a helpful assistant.'},
                        {'role': 'user', 'content': user_message},
                        {'role': 'assistant', 'content': model_response}]

        jsonl_data.append(json.dumps({"messages": conversation}))

    with open(output_file, 'w', encoding='utf-8') as outfile:
        outfile.write('\n'.join(jsonl_data))

convert_to_jsonl('your_dataset.csv', 'dataset.jsonl')
```

Step 5: Fine-Tune the Model

- Use the OpenAI API to fine-tune the model. Replace **YOUR_API_KEY** with your actual OpenAI API key.

```
import openai

openai.api_key = 'YOUR_API_KEY'

fine_tuning_prompt = """[Conversation]
"""

with open('dataset.jsonl', 'r', encoding='utf-8') as infile:
    conversations = infile.read()

response = openai.ChatCompletion.create(
    model="gpt-3.5-turbo",
    messages=fine_tuning_prompt + conversations,
    max_tokens=1024,
    n=1,
    stop=None
)

model_id = response['id']
print(f"Fine-tuned model ID: {model_id}")
```

Step 6: Use the Fine-Tuned Model

- You can now use the fine-tuned model for your specific task by referring to the generated `model_id`. Make API calls using this ID.

Pros and Cons of Using a Pre-trained Language Model (LLM) for Custom Fine-Tuning

Pros of using a Pre-trained Language Model for Custom Fine-Tuning –



1. Transfer Learning Benefits:

- Pre-trained LLMs have already learned rich language representations from vast amounts of diverse data. Fine-tuning allows you to leverage these general language capabilities for more specific tasks without starting from scratch.

2. Reduced Data Requirements:

- Fine-tuning a pre-trained model often requires less labeled data compared to training a model from scratch. This is especially beneficial when dealing with limited task-specific datasets.

3. Time and Resource Efficiency:

- Training a state-of-the-art language model from scratch is computationally expensive and time-consuming. Fine-tuning saves

resources by building upon existing knowledge, making it a more efficient process.

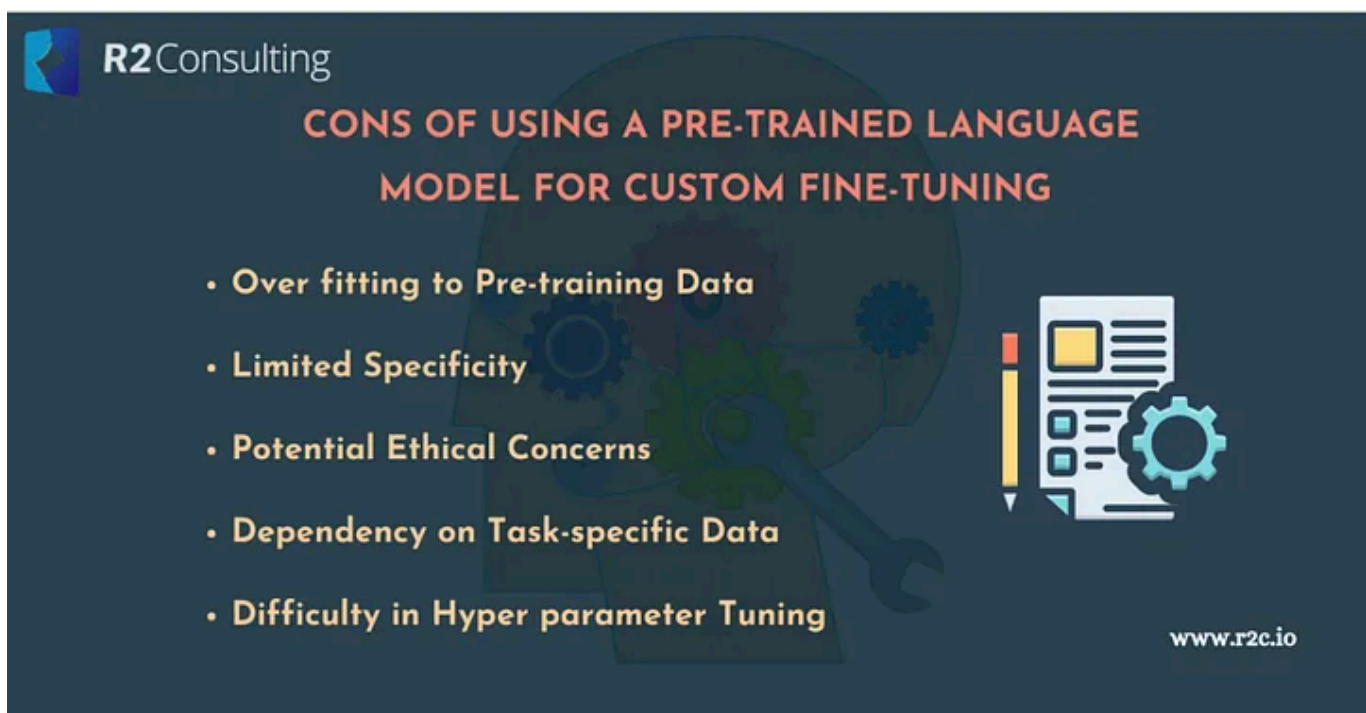
4. Domain Adaptability:

- Pre-trained models capture general linguistic patterns, making them adaptable to various domains and tasks. Fine-tuning allows customization for specific industries or applications without compromising the model's underlying language understanding.

5. Quality of Generated Content:

- Pre-trained models often produce coherent and contextually relevant responses. Fine-tuning helps enhance the quality of generated content by tailoring the model to understand and respond to task-specific nuances.

Cons of Using a Pre-trained Language Model for Custom Fine-Tuning –



R2Consulting

CONS OF USING A PRE-TRAINED LANGUAGE MODEL FOR CUSTOM FINE-TUNING

- Over fitting to Pre-training Data
- Limited Specificity
- Potential Ethical Concerns
- Dependency on Task-specific Data
- Difficulty in Hyper parameter Tuning

www.r2c.io

1. Over fitting to Pre-training Data:

- Fine-tuning on a specific dataset may result in the model being biased towards the characteristics of the pre-training data. This can be a limitation if the pre-training data doesn't align well with the target task or domain.

2. Limited Specificity:

- While pre-trained models offer broad language understanding, they may lack specificity for certain niche tasks. Fine-tuning helps, but the model might not excel in highly specialized domains without extensive fine-tuning.

3. Potential Ethical Concerns:

- Pre-trained models inherit biases present in their training data, and fine-tuning may not completely eliminate these biases. It's crucial to be aware of and address ethical considerations related to bias, fairness, and potential misuse of the model.

4. Dependency on Task-specific Data:

- Fine-tuning still requires task-specific data for optimal performance. If the dataset used for fine-tuning is too small or not representative of the target task, the model may not generalize well.

5. Difficulty in Hyper parameter Tuning:

- The pre-trained model comes with its set of hyperparameters, and finding the right balance during fine-tuning can be challenging. Improper tuning may lead to suboptimal performance.

Final Thoughts

While using a pre-trained LLM for custom fine-tuning offers numerous advantages, it's essential to carefully consider the characteristics of the pre-training data, potential biases, and the specificity required for the target task or domain. Fine-tuning should be approached with a thoughtful understanding of the trade-offs involved.

Custom fine-tuning with ChatGPT's API opens up new possibilities for tailoring the model to specific needs. Follow the six steps above and experiment with different datasets and parameters to achieve optimal results for your use case.

ChatGPT

Fine Tuning

API

Llm

OpenAI

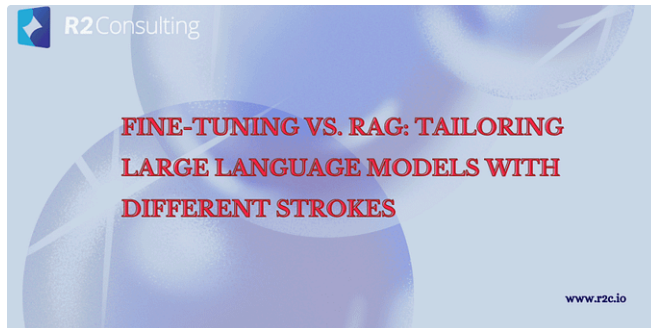


Written by R2consulting

Follow

2 Followers

More from R2consulting



R2consulting

Fine-tuning vs. RAG: Tailoring Large Language Models with...

Introduction

4 min read · Feb 9, 2024

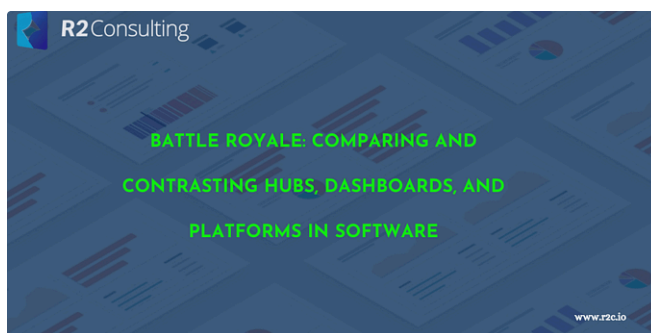


R2consulting

Understanding the Differences Between SDKs, Plugins, and...

Introduction

4 min read · Nov 3, 2023

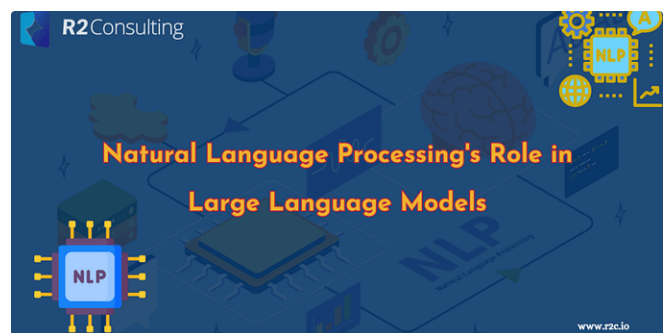


R2consulting

Battle Royale: Comparing and Contrasting Hubs, Dashboards,...

Table Of contents

4 min read · Oct 13, 2023



R2consulting

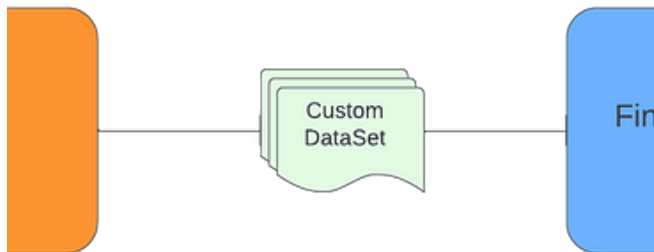
Natural Language Processing's Role in Large Language Models

Natural Language Processing (NLP) has undergone a remarkable transformation in...

4 min read · Dec 21, 2023

[See all from R2consulting](#)

Recommended from Medium



Suman Das

Fine Tune Large Language Model (LLM) on a Custom Dataset with...

The field of natural language processing has been revolutionized by large language...

15 min read · Jan 25, 2024



1.2K



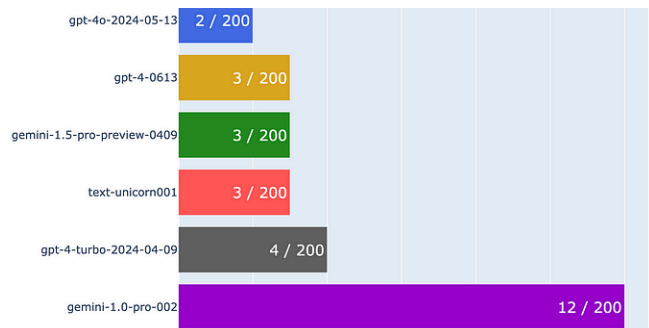
15



1.2K



22



Lars Wiik

GPT-4o vs. GPT-4 vs. Gemini 1.5 ★ —Performance Analysis

Measuring English Language Understanding of OpenAI's New Flagship Model

5 min read · May 14, 2024

Lists



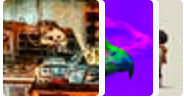
ChatGPT prompts

47 stories · 1578 saves



ChatGPT

21 stories · 642 saves



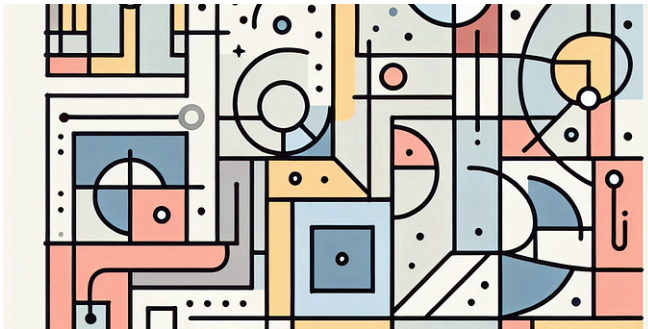
What is ChatGPT?

9 stories · 357 saves



The New Chatbots: ChatGPT, Bard, and Beyond

12 stories · 381 saves



Gareth Cull

Fine Tuning AI Models—A Practical Guide for Beginners

Get started fine tuning OpenAI models such as gpt-3.5-turbo in this easy to understand...

9 min read · Jan 10, 2024



4



1



Mohit Dulani

Fine tune any LLM using your custom dataset

In this article I will share with you a fine tuning template using which you can train any mod...

4 min read · Feb 19, 2024



142



Practicing DatScy

Fine-tuning with OpenAI

I finally was able to test fine-tuning using OpenAI!! Fine-tuning using OpenAI's gpt...



Yvann in Better Programming

Build a Chatbot on Your CSV Data With LangChain and OpenAI

Chat with your CSV file with a memory chatbot 🤖 — Made with Langchain 🦜️...

9 min read · Dec 4, 2023



5 min read · Jun 2, 2023



See more recommendations