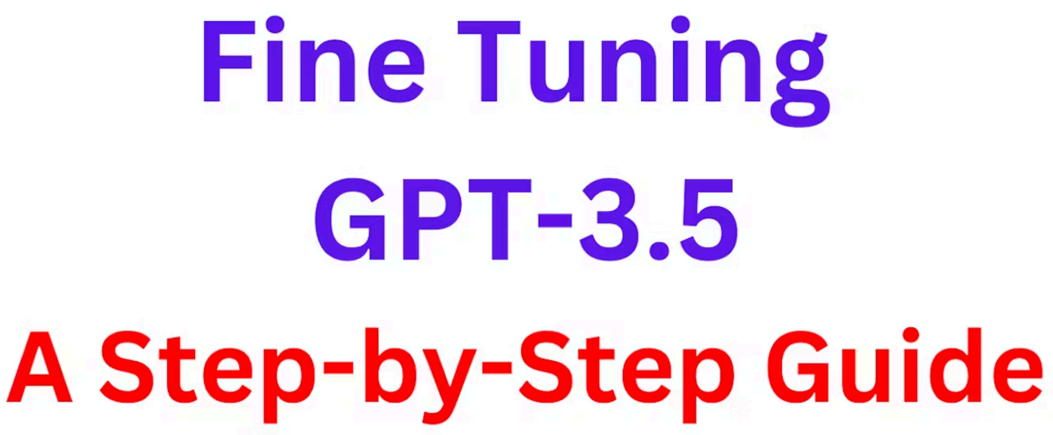
FutureSmart AI Blog

Follow



Fine-Tuning GPT-3.5: A Step-by-Step Guide

Pradip Nichite

Nov 13, 2023 · 4 min read

AN

2 3

Show more

Introduction

In the rapidly evolving world of AI and machine learning, fine-tuning pre-trained models like GPT-3.5 has become a pivotal step in achieving enhanced and specialized performance. This guide will walk you through the fine-tuning process of the GPT-3.5 model, explaining its benefits and providing a step-by-step tutorial with code.

Why Fine-Tune GPT-3.5?

Fine-tuning GPT-3.5 has several advantages:

1. **Improved Quality**: It leads to higher quality results compared to using the model with generic prompts.
2. **Customization**: Fine-tuning allows the model to adapt to specific use cases or domains, which might not be effectively covered in the standard model.

3. **Efficiency**: It [illegible] lex prompts by embedding [illegible] odel.

Preparing for Fine-Tuning

Before starting the fine-tuning process, it's crucial to prepare your dataset. This dataset should consist of examples relevant to the specific task or domain you're targeting.

A Step-by-Step Guide to Fine-Tuning GPT-3.5

Install OpenAI Library:

Begin by installing the OpenAI library in your Python environment.

COPY

```
!pip install -U openai
```

Prepare Your Dataset:

Load and format your dataset. This example uses a dataset for customer support queries:

COPY

```
import pandas as pd
df = pd.read_csv("your_dataset.csv")
df.head()
```

Format the Data:

Convert your data [illegible] This involves structuring your e[illegible]nulating a conversation.

COPY

```
def convert_to_gpt35_format(dataset):
    fine_tuning_data = []
    for _, row in dataset.iterrows():
        json_response = '{"Top Category": "' + row['Top Categ
        fine_tuning_data.append({
            "messages": [
                {"role": "user", "content": row['Support Quer
                {"role": "system", "content": json_response}
            ]
        })
    return fine_tuning_data
```

Creating Training and Validation Sets

After formatting the data, the next step is to split it into training and validation sets. This is crucial for training the model on a subset of data and then validating its performance on a different subset.

COPY

```
from sklearn.model_selection import train_test_split

# Stratified splitting. Assuming 'Top Category' can be used f
train_data, val_data = train_test_split(
    converted_data,
    test_size=0.2,
    stratify=dataset['Top Category'],
    random_sta
)
```

Creating JSONL Files

Fine-tuning with OpenAI requires the data to be in JSONL format. The code demonstrates how to convert the training and validation sets into this format and save them as files.

COPY

```
def write_to_jsonl(data, file_path):
    with open(file_path, 'w') as file:
        for entry in data:
            json.dump(entry, file)
            file.write('\n')

training_file_name = "train.jsonl"
validation_file_name = "val.jsonl"

write_to_jsonl(train_data, training_file_name)
write_to_jsonl(val_data, validation_file_name)
```

Uploading Data and Starting the Fine-Tuning Job

With the JSONL files ready, you upload them to OpenAI and initiate the fine-tuning process.

COPY

```
from openai import OpenAI
client = OpenAI(api_key="your_open_ai_key")

# Upload Training and Validation Files
```

```
training_file = client.files.create(
    file=ope                                  se="fine-tune"
)
validation_file = client.files.create(
    file=open(validation_file_name, "rb"), purpose="fine-tune
)

# Create Fine-Tuning Job
suffix_name = "yt_tutorial"
response = client.fine_tuning.jobs.create(
    training_file=training_file.id,
    validation_file=validation_file.id,
    model="gpt-3.5-turbo",
    suffix=suffix_name,
)
```

Testing the Fine-Tuned Model

Once fine-tuned, it's essential to test the model's performance. The provided code includes a function to format test queries, a prediction function using the fine-tuned model, and a method to store predictions.

COPY

```
from sklearn.metrics import accuracy_score, precision_score,

def format_test(row):
    formatted_message = [{"role": "user", "content": row['Sup
    return formatted_message

def predict(test_messages, fine_tuned_model_id):
    response = client.chat.completions.create(
        model=fine_tuned_model_id, messages=test_messages, te
    )
    return response.choices[0].message.content

def store_predictions(test_df, fine_tuned_model_id):
    test_df['Prediction'] = None
    for index, row in test_df.iterrows():
        test_message = format_test(row)
        prediction_result = predict(test_message, fine_tuned_
        test_df.at[index, 'Prediction'] = prediction_result

    test_df.to_csv("predictions.csv")
```

Observations

With just 100 examples, the model shows promising results, particularly in identifying top categories. This experiment highlights the importance of starting with a small dataset and progressively adding more data for refinement.

Conclusion

This detailed guide, enriched with code snippets and explanations, illustrates the entire process of fine-tuning the GPT-3.5 model. It's a testament to the power and flexibility of AI models in adapting to specific needs and domains, providing enhanced and more relevant responses.

If you're curious about the latest in AI technology, I invite you to visit my project, AI Demos, at https://www.aidemos.com/. It's a rich resource offering a wide array of video demos showcasing the most

advanced AI tools te and illuminate
the diverse possi

For even more in-depth exploration, be sure to visit my YouTube channel at https://www.youtube.com/@aidemos.futuresmart. Here, you'll find a wealth of content that delves into the exciting future of AI and its various applications.

Code and Dataset:

https://github.com/PradipNichite/Youtube-Tutorials/tree/main/GPT3.5%20Finetuning

fine tuning openai gpt3.5

Written by

Pradip Nichite

🚀 I'm a Top Rated Plus NLP freelancer on Upwork with over $100K in earnings and a 100% Job Success rate. This journey began in 2022 after years of enriching experience in the field of Data Science.

📚 Starting my career in 2013 as a Software Developer focusing on backend and API development, I soon pursued my interest in Data Science by earning my M.Tech in IT from IIIT Bangalore, specializing in Data Science (2016 - 2018).

💼 Upon graduation, I carved out a path in the industry as a Data Scientist at MiQ (2018 - 2020) and later ascended to the role of Lead Data Scientist at Oracle (2020 - 2022).

🌐 Inspired by my fr art AI in September 2022. using the latest models and techniques in NLP.

🎥 In addition, I run AI Demos, a platform aimed at educating people about the latest AI tools through engaging video demonstrations.

🧰 My technical toolbox encompasses: 🔧 Languages: Python, JavaScript, SQL. 🧪 ML Libraries: PyTorch, Transformers, LangChain. 🔍 Specialties: Semantic Search, Sentence Transformers, Vector Databases. 🖥️ Web Frameworks: FastAPI, Streamlit, Anvil. ☁️ Other: AWS, AWS RDS, MySQL.

🚀 In the fast-evolving landscape of AI, FutureSmart AI and I stand at the forefront, delivering cutting-edge, custom NLP solutions to clients across various industries.

Follow

Published on

FutureSmart AI Blog

FutureSmart AI provides custom Natural Language Processing (NLP) solutions.

Follow

MORE ARTICLES

Avikumar talaviya

Sakalya Mitra

My Internship Journey: From RAGs to Project Management

Introduction I always wanted to work on internship before I finish my graduation. I had applied mult...

Master RAG with LangChain: A Practical Guide

Introduction In the realm of Natural Language Processing (NLP), the quest for more human-like text c...

Venkata Vinay Vijjapu

Guide to langsmith

Introduction Large language models (LLMs) are the talk of the town, with their potential application...